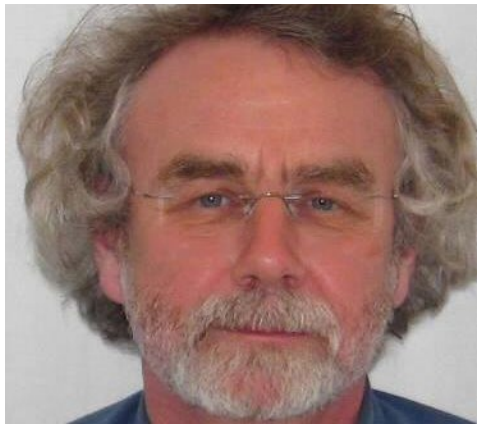
Adil RASHEED
Frank WESTAD

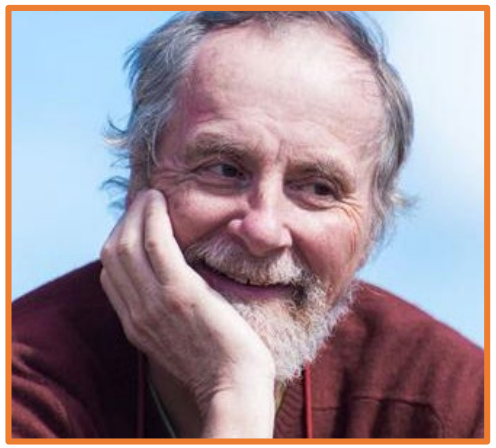
Lecture 4: PCA

Bigdatacybernetics

- Instructors
  - Harald MARTENS
  - Øivind RIIS
  - Kristin TØNDEL
  - Damiano VARAGNOLO
  - Frank Ove WESTAD
  - Adil RASHEED

Homework: Who is who ?





Recap

Physics-based modeling

Data-driven Modeling

Bigdata Cybernetics

Convinced ?

Why data analytics when first principle models are available in abundance?

Why to take a course in Multivariate Data Analysis when so many courses on ML are available?

What makes this course different from other courses on similar topics?

Some taste of Bigdatacybernetics

Get to know each other and encourage to prepare our own data

Install Unscrambler, Python, Matlab

Formal details about the course

In-class performance analytics

Home work (Status)

Install Unscrambler, python and matlab

Try the python notebook on github for Lecture 1 to convince the need for interpretable data analytics

Gone through the recommended videos from the flipped classroom

Prepare some data from your own research field to work with

Content

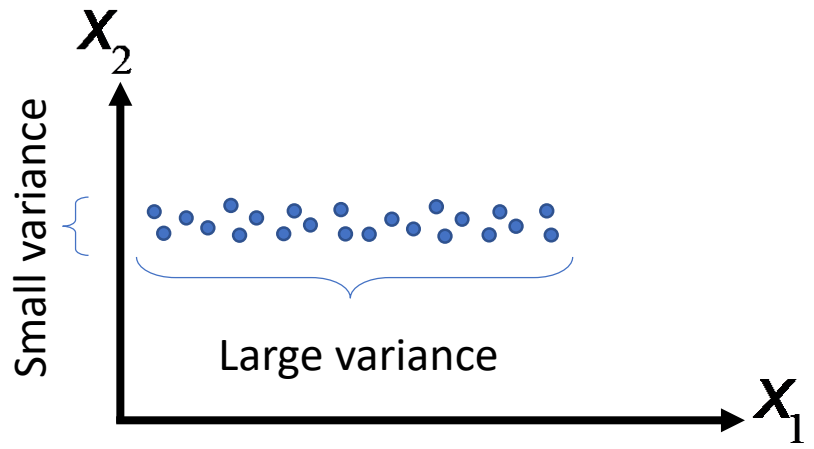
Preliminaries	Deriving PCs	Commonly used nomenclature in PCA	Ways to interpret PCA model	PCA Applications	Application of PCA to timeseries data (Unscrambler)
• Dimensionality Reduction • Projections • Eigen values and eigen vectors • Geometric interpretation of matrix multiplication of vectors	• Eigenvalue decomposition • Singular Value Decomposition • Non-linear Iterative PArtial Least Squares • Autoencoder appraoch	• Scores • Loading • Residuals	• Choosing the number of components • Getting insight through the Score, Loading and residual plots	• Clustering • Noise Reduction • Developing Insight • Outlier Detection • Dimensionality Reduction • Model order reduction / Reduced Order Model	• Ozone timeseries

Preliminaries

- Dimensionality Reduction
- Geometric interpretation of matrix multiplication of vectors
- Physical interpretation of eigen values

Simple dimensionality reduction

$$X = \begin{pmatrix} x_1 & x_2 \\ \text{Outdoor temp 1} & \text{Indoor temp 1} \end{pmatrix}$$

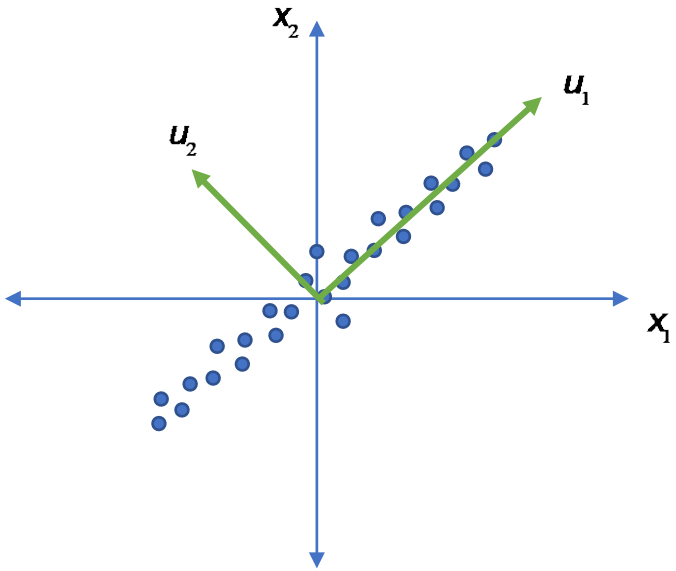


$$X' = \begin{pmatrix} x_1 \\ \text{Outdoor temp 1} \end{pmatrix}$$

Preserving direction with maximum variance

2D → 1D

Mean centering and dimensionality reduction

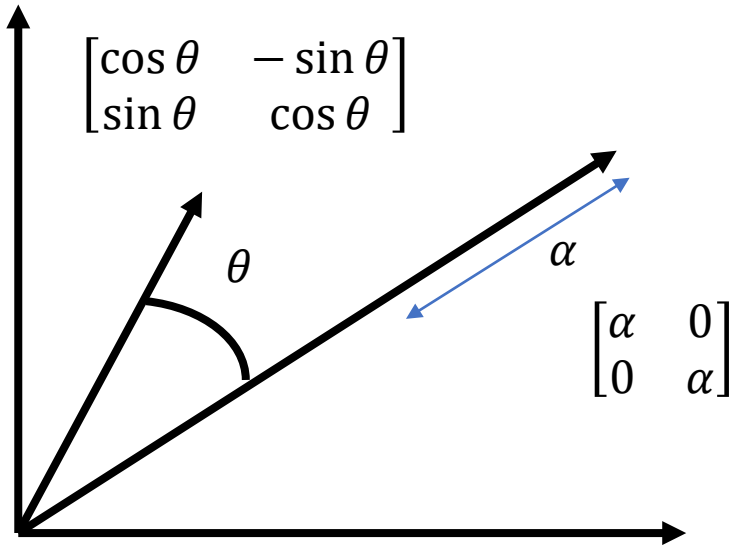


1. Find u_1 and u_2 such that $u_1 \perp u_2$ in such a way that the spread on u_2 << spread on u_1
2. Project all the points from $x_1 - x_2$ to $u_1 - u_2$
3. Drop the component which has much smaller variance

Geometric meaning of multiplication by a matrix

$$X = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \qquad A = \begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix}$$

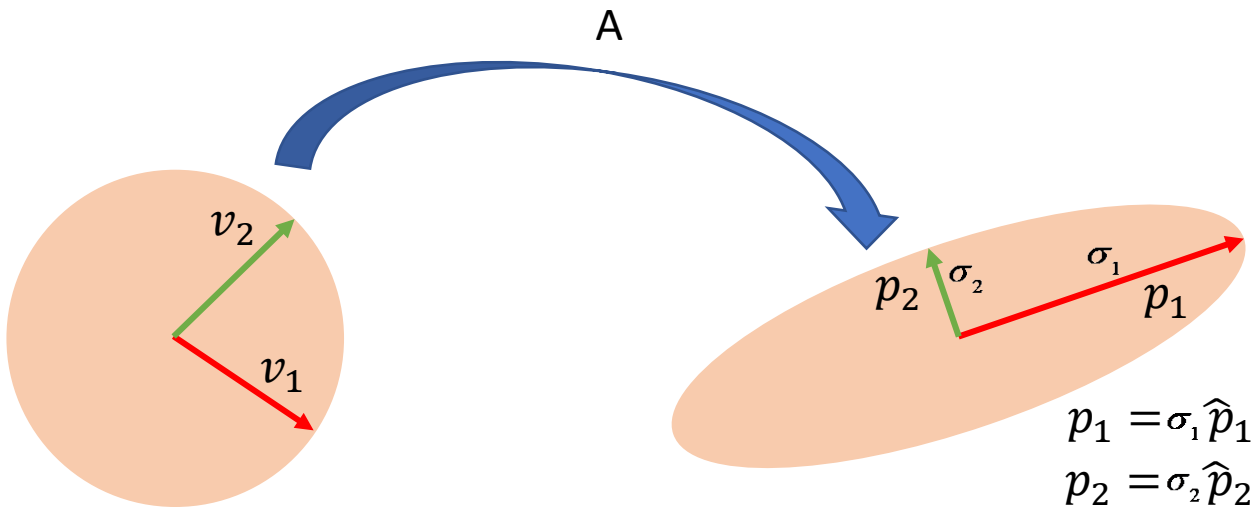
$$y = AX = \begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$$



Any matrix multiplication will rotate, stretch or do both to a vector

Linear transformation

What happens to a circle under this transformation ?



$$(v_1, v_2) \Rightarrow (p_1, p_2)$$

In n dimension a hypersphere transforms to a hyperellipse

$$(v_1, v_2, v_3, v_4 \dots\dots) \Rightarrow (p_1, p_2, p_3, p_4 \dots\dots)$$

$$AV = \Sigma P$$

Extension to n-dimension

$u_1, u_2 \ldots$ principal axis

$\sigma_1\ \sigma_2 \ldots$ singular values

$A\vec{V}_1 = \sigma_1 \hat{u}_1$ eigen value problem $A\vec{V} = \lambda\vec{V}$

V_1 produces $\hat{u}_1$

intimate relationship between this and eig value prob

$A\vec{V}_j = \sigma_j \vec{u}_j \quad j = 1, 2, \ldots, n$

$$\begin{bmatrix} & \\ A & \\ & \end{bmatrix}_{m \times n} \begin{bmatrix} \vec{V}_1 & \vec{V}_2 & \cdots & \vec{V}_n \end{bmatrix}_{(n \times n)} = \begin{bmatrix} \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_n \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_3 & \\ & & & \end{bmatrix}$$

$$A\hat{V} = \hat{U}\hat{\Sigma}$$

$\hat{V}$ orthogonal co-ordinates

all the rotations picked up by $\hat{U}$

"unitary transformations"

$\hat{U}$ orthonormal basis

special property $V^{-1} = V^*$ complex conjugate transform

$\hat{U}^{-1} = \hat{U}^*$

finding the inverse is easy

LHS $\quad A\hat{V}\hat{V}^{-1} = AI = A$

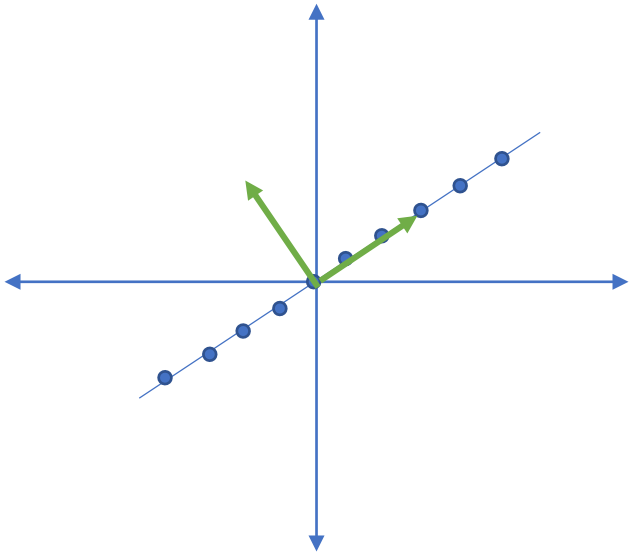
RHS $\quad \hat{U}\hat{\Sigma}\hat{V}^{-1} = (\hat{U}\hat{\Sigma}V^*)$ Reduced SVD

$$A = \hat{U}\hat{\Sigma}V^*$$

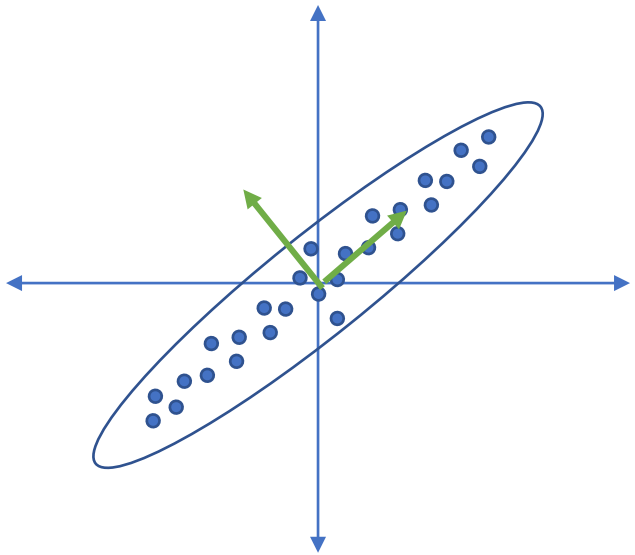
Rotation — stretching

one matrix decomposed into 3.

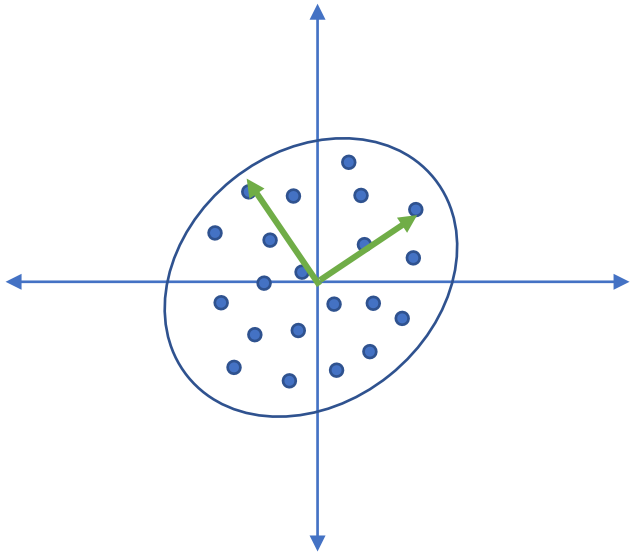
Geometric interpretation of eigenvalues



$$\frac{\sigma_1}{\sigma_1+\sigma_2}=1$$



$$\frac{\sigma_1}{\sigma_1+\sigma_2}=\frac{3}{4}$$



$$\frac{\sigma_1}{\sigma_1+\sigma_2}=\frac{3}{5}$$

% of the variance explained

Eigen Value Decomposition

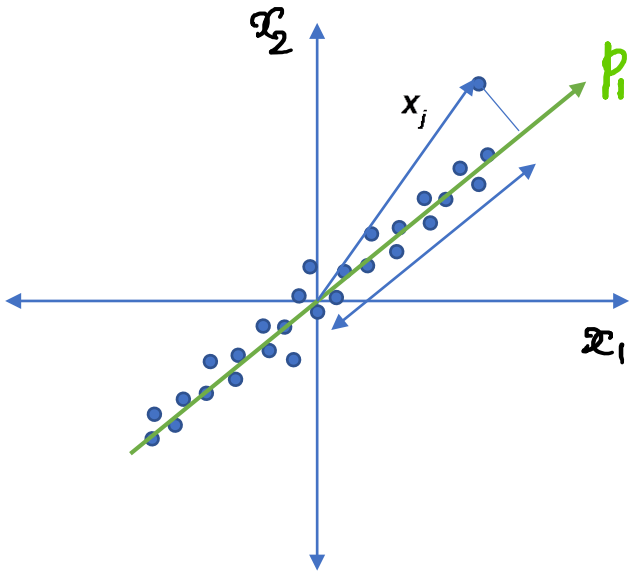
Singular Value Decomposition

Non Linear Iterative PLS

Autoencoder approach

Deriving the PCs

Maximizing the variance



Projection along p_1 is $x_j p_1$

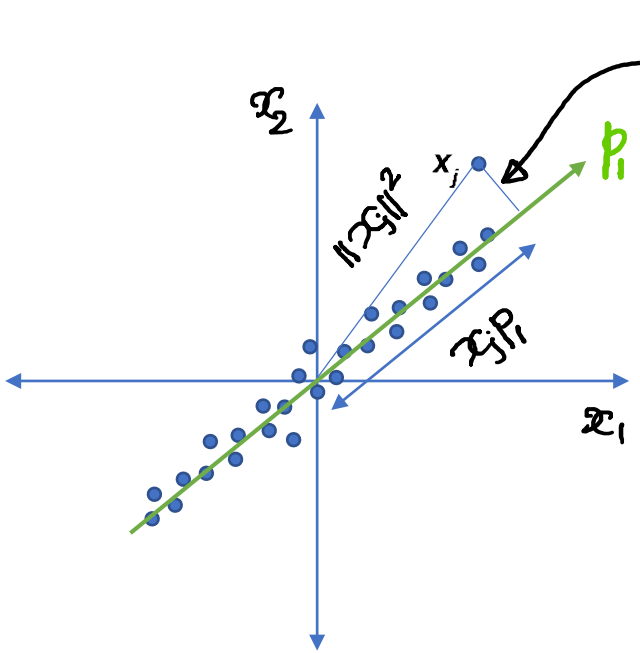
Find p_1 such that $\sum_{i=1}^{n} (x_j p_1 - \bar{x}_j p_1)^2$ is maximum

$\bar{x}_j p_1 = 0$

$\Rightarrow$ maximize $\sum_{i=1}^{n} (x_j p_1)^2$

With the constraint $p_1^T p_1 = 1$

Minimizing the projection error



$$\sqrt{\|x_j\|^2 - (x_j p_1)^2}$$

minimize $\sum_{j=1}^{n} \left(\|x_j\|^2 - (x_j p_1)^2\right)$

$$p_1^T p_1 = 1$$

$\Rightarrow$ maximize $\sum_{j=1}^{n} (x_j p_1)^2$ with $p_1^T p_1 = 1$

Eigenvalue decomposition approach

ALGORITHM

Recall that the latent variable directions (the loading vectors) were oriented so that the variance of the scores in that direction were maximal. We can cast this as an optimization problem. For the first component:

$$max(\phi) = \mathbf{t}_1'\mathbf{t}_1 = \mathbf{p}_1'\mathbf{X}'\mathbf{X}\mathbf{p}_1$$

such that

$$\mathbf{p}_1'\mathbf{p}_1 = 1$$

.

This is equivalent to

$$max(\phi) = \mathbf{p}_1'\mathbf{X}'\mathbf{X}\mathbf{p}_1 - \lambda(\mathbf{p}_1'\mathbf{p}_1 - 1)$$

because we can move the constraint into the objective function with a Lagrange multiplier, λ. The maximum value must occur when the partial derivatives with respect to $\mathbf{p}_1$, our search variable, are zero:

$$\frac{\partial\phi}{\partial p_1} = \frac{\partial(\mathbf{p}_1'\mathbf{X}'\mathbf{X}\mathbf{p}_1 - \lambda(\mathbf{p}_1'\mathbf{p}_1 - 1))}{\partial\mathbf{p}_1} = 0$$
$$2\mathbf{X}'\mathbf{X}\mathbf{p}_1 - 2\lambda_1\mathbf{p}_1 = 0$$
$$(\mathbf{X}'\mathbf{X} - \lambda_1\mathbf{I})\mathbf{p}_1 = 0$$
$$\mathbf{X}'\mathbf{X}\mathbf{p}_1 = \lambda_1\mathbf{p}_1$$

which is just the eigenvalue equation, indicating that $\mathbf{p}_1$ is the eigenvector of $\mathbf{X}'\mathbf{X}$ and λ_1 is the eigenvalue. One can show that $\lambda_1 = \mathbf{t}_1'\mathbf{t}_1$, which is proportional to the variance of the first component. In a similar manner we can calculate the second eigenvalue, but this time we add the additional constraint that $\mathbf{p}_1 \perp \mathbf{p}_2$. Writing out this objective function and taking partial derivatives leads to showing that

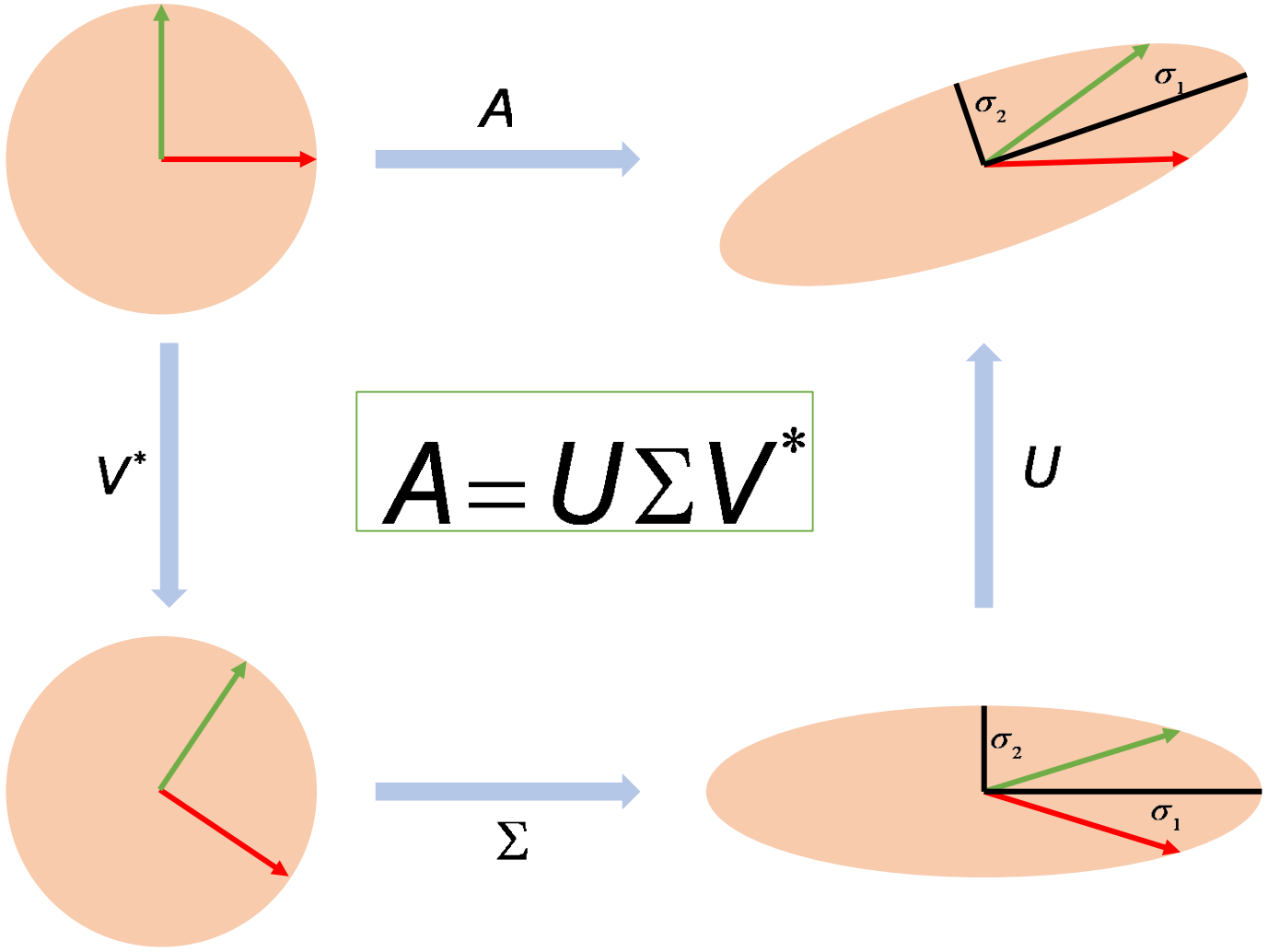
$$\mathbf{X}'\mathbf{X}\mathbf{p}_2 = \lambda_2\mathbf{p}_2$$

Eigenvalue decomposition approach

- The loadings are the eigenvectors of $\mathbf{X}'\mathbf{X}$
- Sorting the eigenvalues in order from largest to smallest gives the order of the corresponding eigenvectors, the loadings.
- We know from the theory of eigenvalues that if there are distinct eigenvalues, then their eigenvectors are linearly independent (orthogonal).
- We also know the eigenvalues of $\mathbf{X}'\mathbf{X}$ must be real values and positive; this matches with the interpretation that the eigenvalues are proportional to the variance of each score vector.
- Also, the sum of the eigenvalues must add up to sum of the diagonal entries of $\mathbf{X}'\mathbf{X}$, which represents of the total variance of the X matrix, if all eigenvectors are extracted. So plotting the eigenvalues is equivalent to showing the proportion of variance explained in X by each component. This is not necessarily a good way to judge the number of components to use, but it is a rough guide: use a Pareto plot of the eigenvalues (though in the context of eigenvalue problems, this plot is called a scree plot).

SVD

$$A = U\Sigma V^*$$



Computing

$$A = U\Sigma V^*$$

$$A^T A = (U\Sigma V^*)^T (U\Sigma V^*)$$
$$A^T A = (V\Sigma U^*)(U\Sigma V^*)$$
$$A^T A = (V\Sigma^2 V^*)$$

$$AA^T = (U\Sigma V^*)(U\Sigma V^*)^T$$
$$AA^T = (U\Sigma V^*)(V\Sigma U^*)$$
$$AA^T = U\Sigma^2 U^*$$

$$U^* = U^{-1}$$

$$V^* = V^{-1}$$

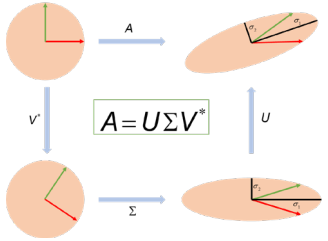
$$A^T AV = V\Sigma^2$$

$$AA^T U = U\Sigma^2$$

Solve the eigen value problem to get U, Σ, V^*

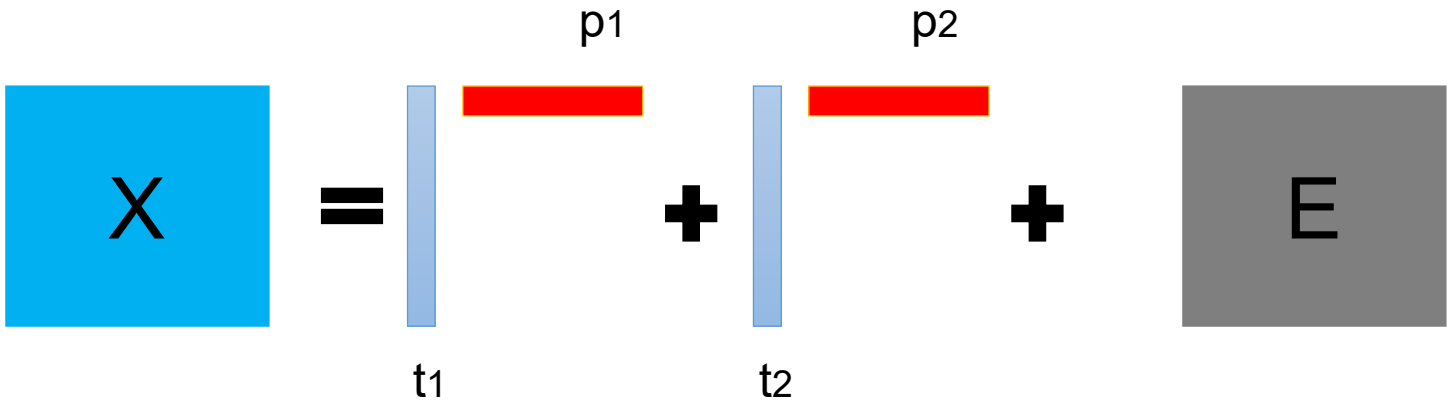
Since $A^T A$ is a symmetric matrix the eigenvectors are guarenteed to be orthogonal

PCA



$$X_{m\times n} = \underbrace{U_{m\times m}}_{\text{Rotation}}\ \underbrace{\Sigma_{m\times n}}_{\text{Stretching}}\ \underbrace{V_{n\times n}}_{\text{Rotation}} = U_{m\times r}\ \Sigma_{r\times r}\ V_{r\times n} + E_{m\times n} = T_{m\times r}\ P^T_{r\times n} + E_{r\times r}$$

Dimensionality reduction



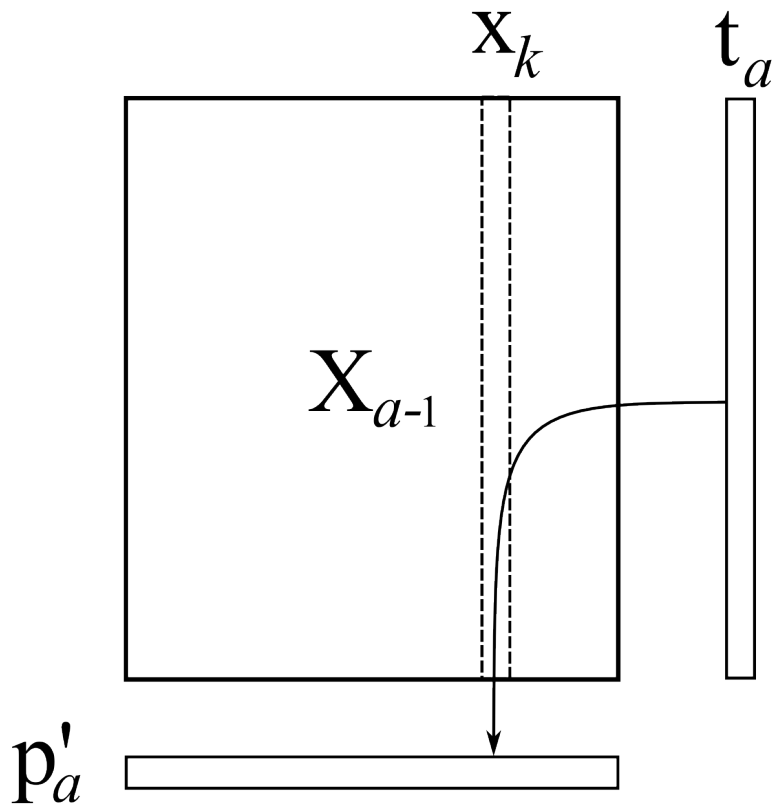
Vector form: $X = t_1 p_1^T + t_2 p_2^T + E$

Matrix form: $X = TP^T + E$

PCA by means of Singular Value Decomposition (SVD)

- Let X be a matrix of size (nObj x nVar)
- SVD on **X**: [u,s,v] = svd(**X**);
  - S: Singular values (square root of Eigenvalues)
  - Scores T = u*s
  - Loadings P = v
- In case of matrices which are “long thin” or “short fat”:
  - Perform SVD on the covariance matrix of the smallest dimension; $X^TX/(nObj-1)$ or $XX^T/(nObj-1)$
  - 1. If nVar < nObj: [u,s,v] = svd($X^TX/(nObj-1)$)
  - Both u and v hold the loadings P
  - 2. If nObj < nVar: [u,s,v] = svd($XX^T/(nObj-1)$)
    - v = X’v
    - P = norm(v)
- T = XP ($X = TP^T$); $P^TP = I$
- Eigenvalues = diag(s)
- Explained variance = cumsum(Eigenvalues)/sum(Eigenvalues)

Non-linear Iterative Partial Least Squares algorithm



ALGORITHM

Steps to compute PCA using NIPALS algorithm

- Step 1: Initialize an arbitrary column vector $\mathbf{t}_a$ either randomly or by just copying any column of X.
- Step 2: Take very column of X, X_k and regress it onto the $\mathbf{t}_a$ vector and store the regression coefficeints as $\mathbf{p}_{ka}$. (Note: This simply means performing an ordinary least squares regression ($y = mx$) with $x = t_a$ and $y = X_k$ with $m = (\mathbf{x}'\mathbf{x})^{-1}\mathbf{x}'\mathbf{y}$). In the current notation we get

  $$p_{ka} = \frac{\mathbf{t}_a'\mathbf{X}_k}{\mathbf{t}_a'\mathbf{t}_a}$$

  Repeat it for each of the columns of X to get the entire vector $\mathbf{p}_k$. This is shown in the illustration above where each column from X is regressed, one at a time, on $\mathbf{t}_a$, to calculate the loading entry, p_{ka} In practice we don't do this one column at time; we can regress all columns in X in go:

  $$\mathbf{p}_a' = \frac{1}{\mathbf{t}_a'\mathbf{t}_a}.\mathbf{t}_a'\mathbf{X}_a$$

  where $\mathbf{t}_a$ is an $N \times 1$ column vector, and $\mathbf{X}_a$ us an $N \times K$ matrix.
- The loading vector $\mathbf{p}_a'$ won't have unit length (magnitude) yet. So we simply rescale it to have magnitude of 1.0:

  $$\mathbf{p}_a' = \frac{1}{\sqrt{\mathbf{p}_a'\mathbf{p}_a}.\mathbf{p}_a'}$$

- Step 4: Regress every row in X onto this normalized loadings vector. As illustrated below, in our linear regression the rows in X are our y-variable each time, while the loadings vector is our x-variable. The regression coefficient becomes the score value for that i^{th} row:

  $$p_{i,a} = \frac{\mathbf{x}_i'\mathbf{p}_a}{\mathbf{p}_a'\mathbf{p}_a}$$

  where x_i' is a $K \times 1$ column vector. We can combine these N separate least-squares models and calculate them in one go to get the entire vector,

  $$\mathbf{t}_a' = \frac{1}{\mathbf{p}_a'\mathbf{p}_a}\mathbf{X}\mathbf{p}_a'$$

  where p_a is a $K \times 1$ column vector.
- Step 5: Continue looping over steps 2,3,4 until the change in vector t_a is below a chosen tolerance
- Step 6: On convergence, the score vector and the loading vectors, $\mathbf{t}_a$ and $\mathbf{p}_a$ are stored as the a^{th} column in matrix $\mathbf{T}$ and $\mathbf{P}$. We then deflate the $\mathbf{X}$ matrix. This crucial step removes the variability captured in this component (t_a and p_a) from $\mathbf{X}$:

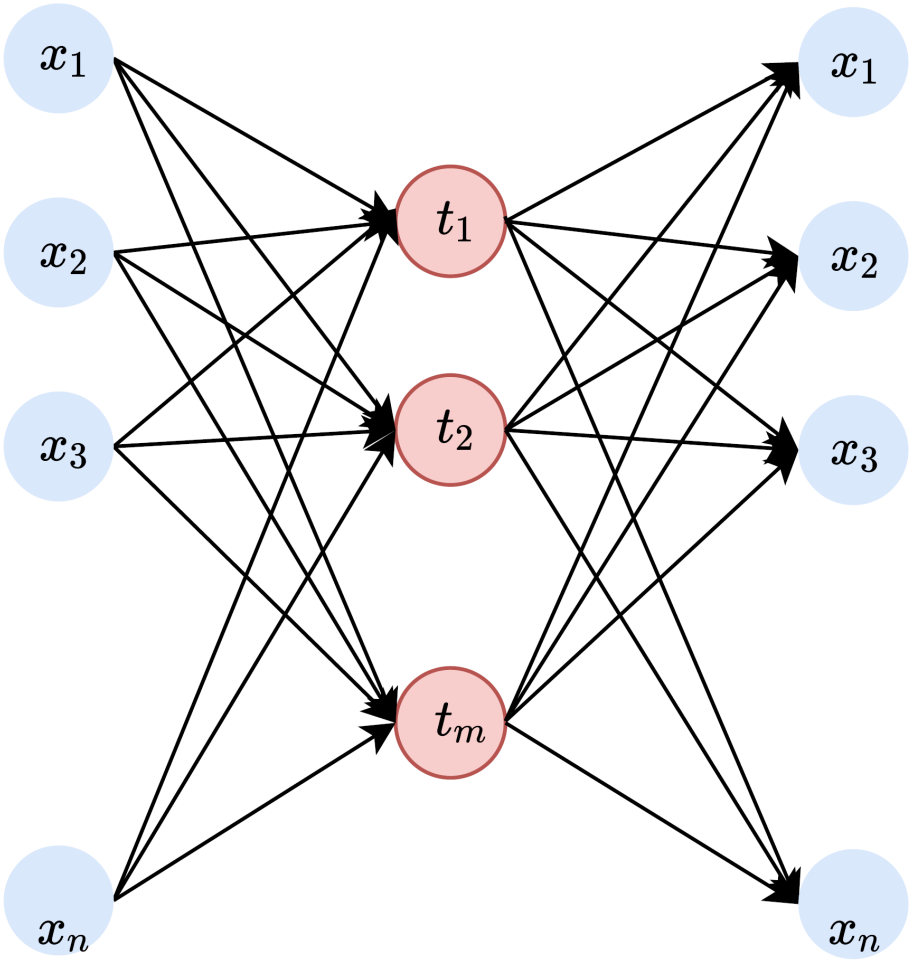
  $$E_a = X_a - t_a p_a'$$
  $$X_{a+1} = E_a$$

For the first component, X_a is just the preprocessed raw data. So we can see that the second component is actually calculated on the residuals E_1, obtained after extracting the first component. This is called deflation, and nicely shows why each component is orthogonal to the others. Each subsequent component is only seeing variation remaining after removing all the others; there is no possibility that two components can explain the same type of variability. After deflation we go back to step 1 and repeat the entire process for the next component.

Advantages of NIPALS algorithm

- The NIPALS algorithm computes one component at a time. The first component computed is equivalent to the t1 and p1 vectors that would have been found from an eigenvalue or singular value decomposition.
- The algorithm can handle missing data in X.
- The algorithm always converges, but the convergence can sometimes be slow.
- It is also known as the Power algorithm to calculate eigenvectors and eigenvalues.
- It works well for very large data sets.
- It is used by most software packages, especially those that handle missing data.
- Of interest: it is well known that Google used this algorithm for the early versions of their search engine, called PageRank148.

Autoencoder approach



Github

https://github.com/adil-rasheed/TK8117/blob/master/Lecture4/PCA-Algorithms.ipynb

Score

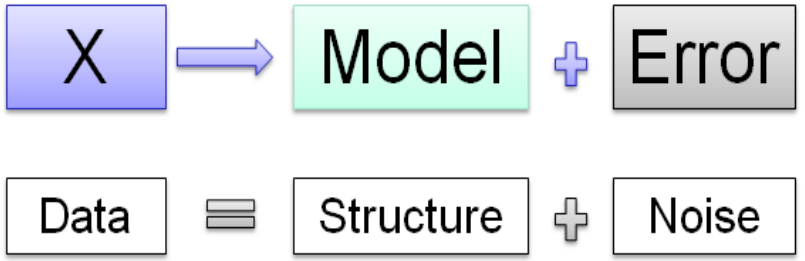
Loading

Residuals

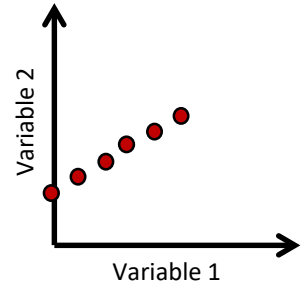
Nomenclature of PCA

Principal Component Analysis

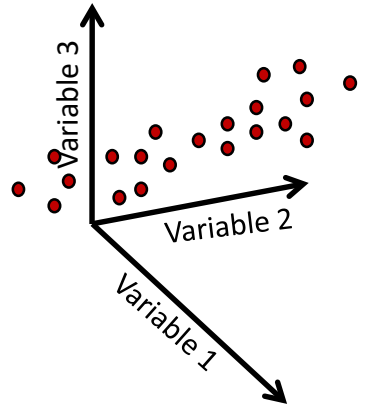
- A method to reduce dimensionality
  - Replace original variables with latent variables
  - Linear combination of the original ones
  - Do not necessarily represent physical factors
  - Useful if variables are correlated
- Information extracted and noise removed
- Many application areas
  - Exploratory data analysis
  - Classification and identification
  - Variable reduction
  - Process monitoring
  - Visual analysis of variance



2D



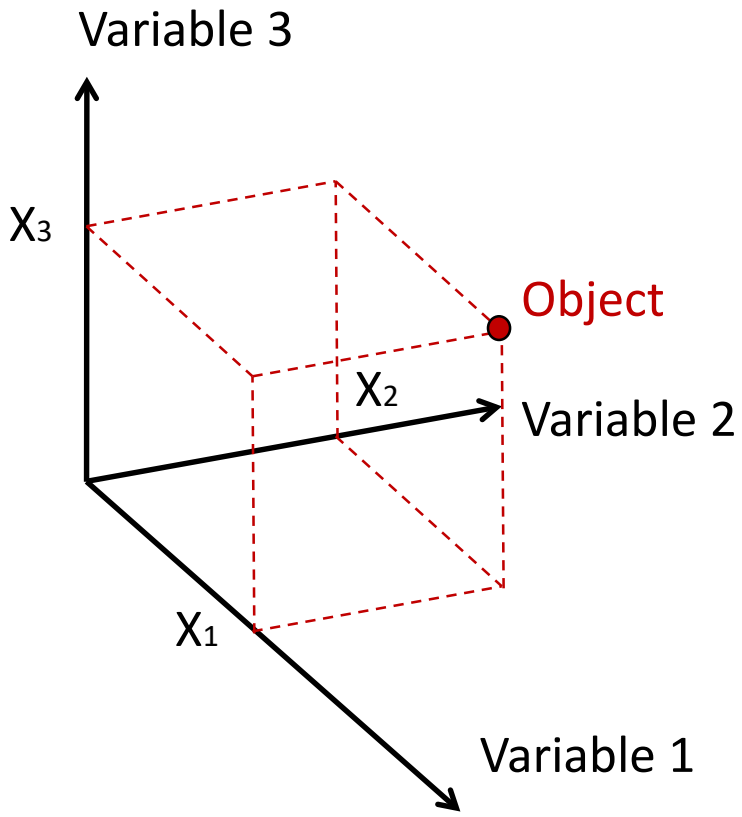
3D



The principles of projection (1/4)

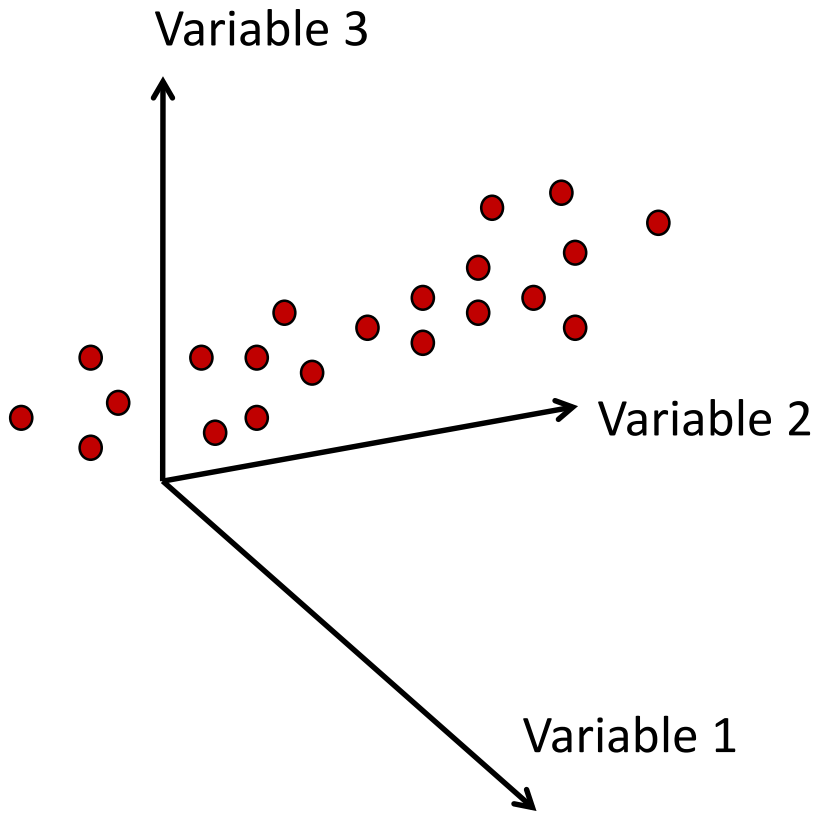
- Each variable defines an axis. A coordinate system can be made using all variables, called the variable space.
- Each object is a row in the data table (matrix) and is visualised as a point in the variable space.

	Variable 1	Variable 2	Variable 3
Object 1			
Object 2			
Object 3			
Object 4			



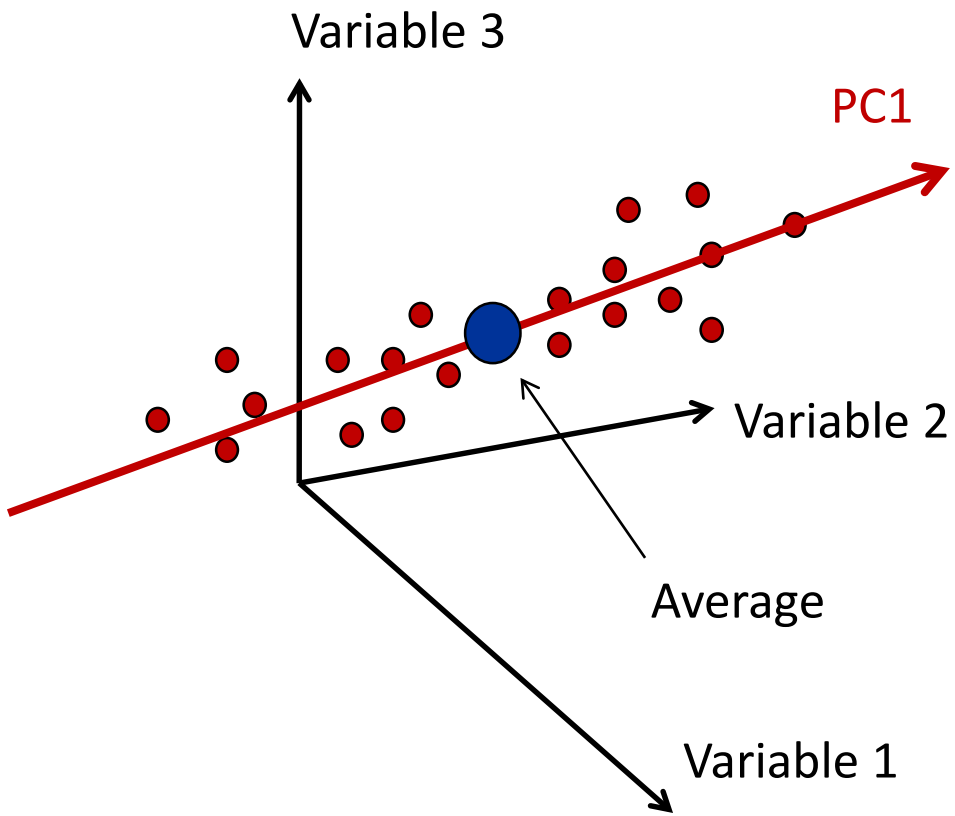
The principles of projection (2/4)

- Each sample / object is represented as a point in the variable space
- The whole data table constitutes a swarm of points in the variable space
- Similar samples are close to each other, dissimilar samples are distant



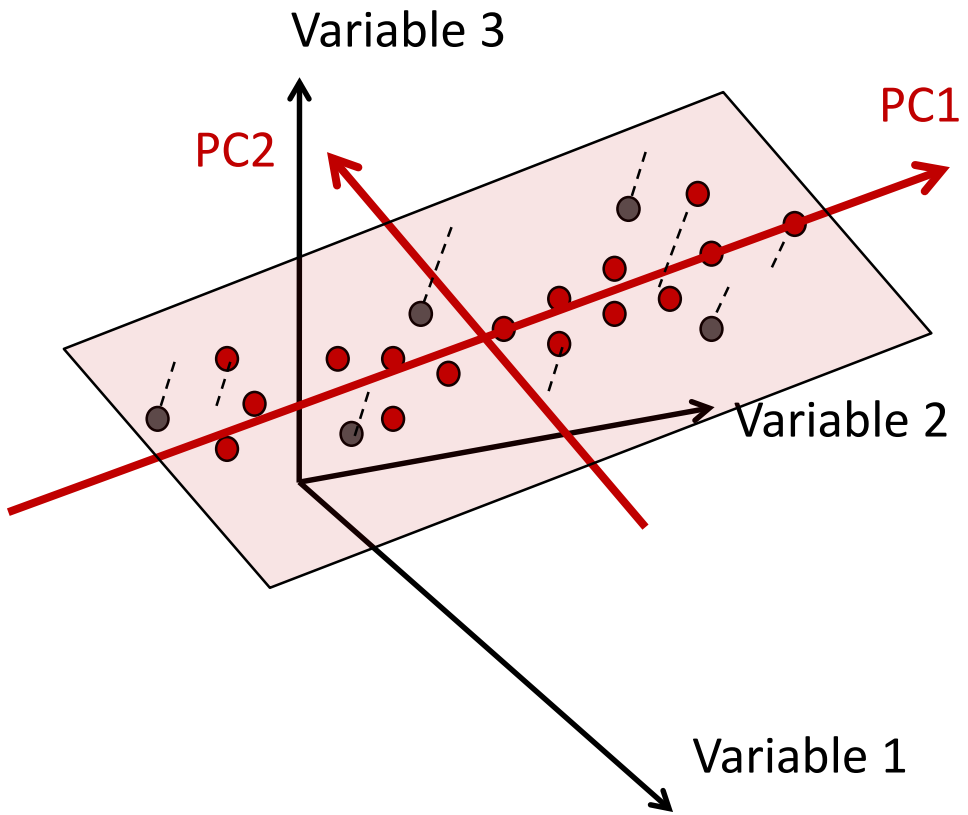
The principles of projection (3/4)

- Variation among samples can be summarised by a straight line through the center of the swarm
- The direction of largest elongation is often the most informative, as it captures most of the variability between samples
- This line is the direction of the first principal component (PC1)

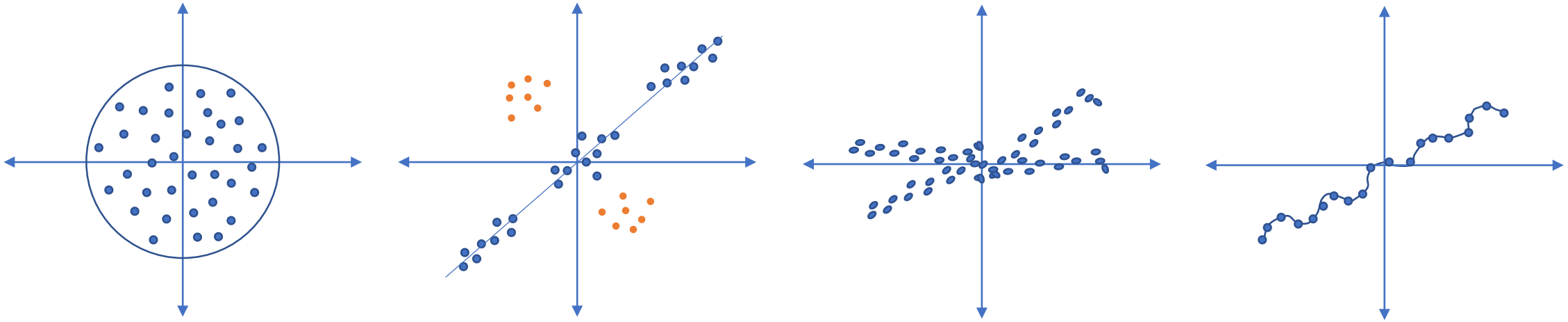


The principles of projection (4/4)

- With 2 principal components we can build a plane onto which the projections of the swarm lie closer yet to the original variable space
- We have found the 2-dimensional 'window' which best describes the data
- The process can be continued to find more PCs



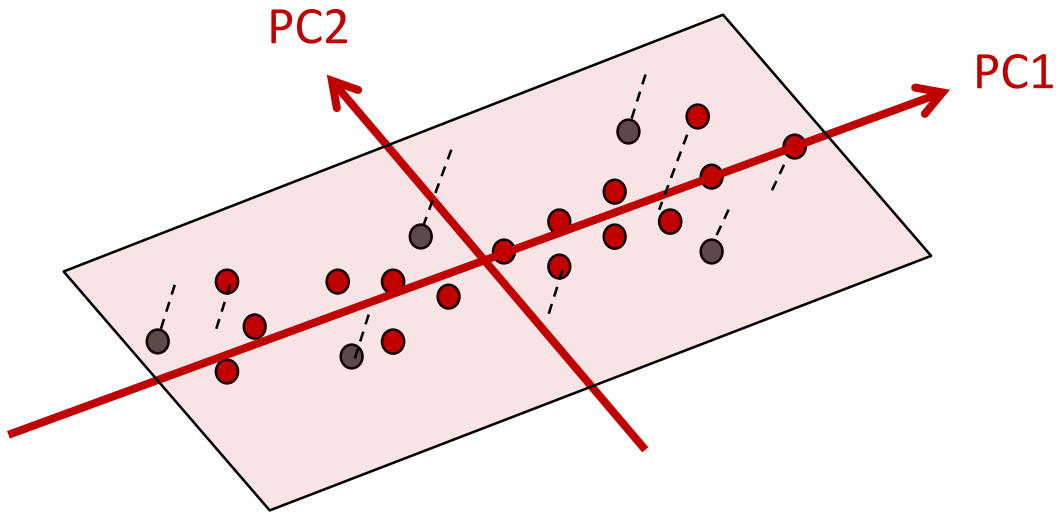
Limitations

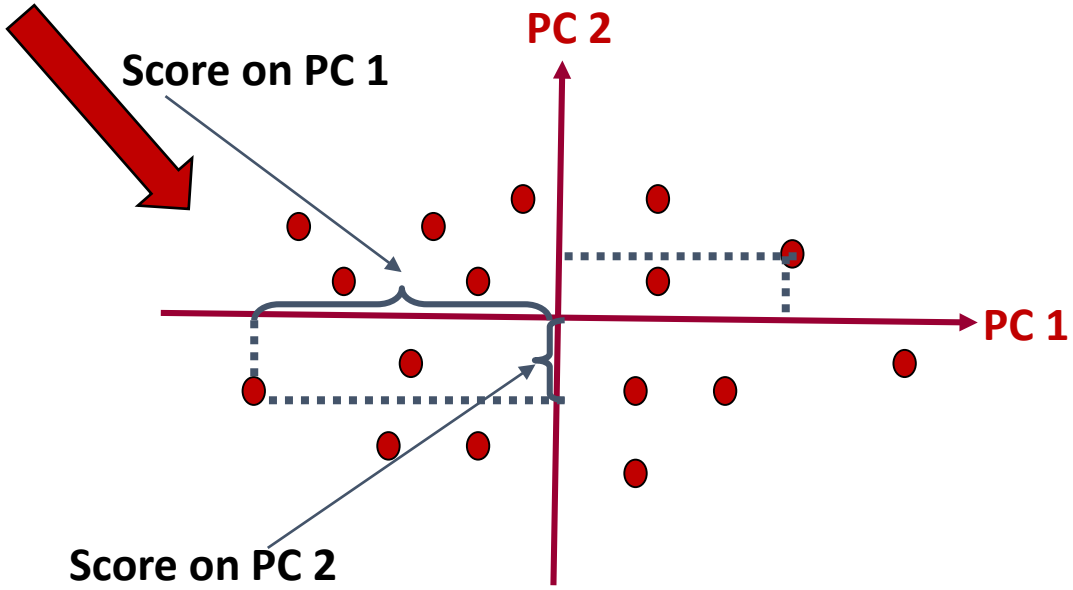


After projection internal structure of the original data gets lost

Score plot – map of samples

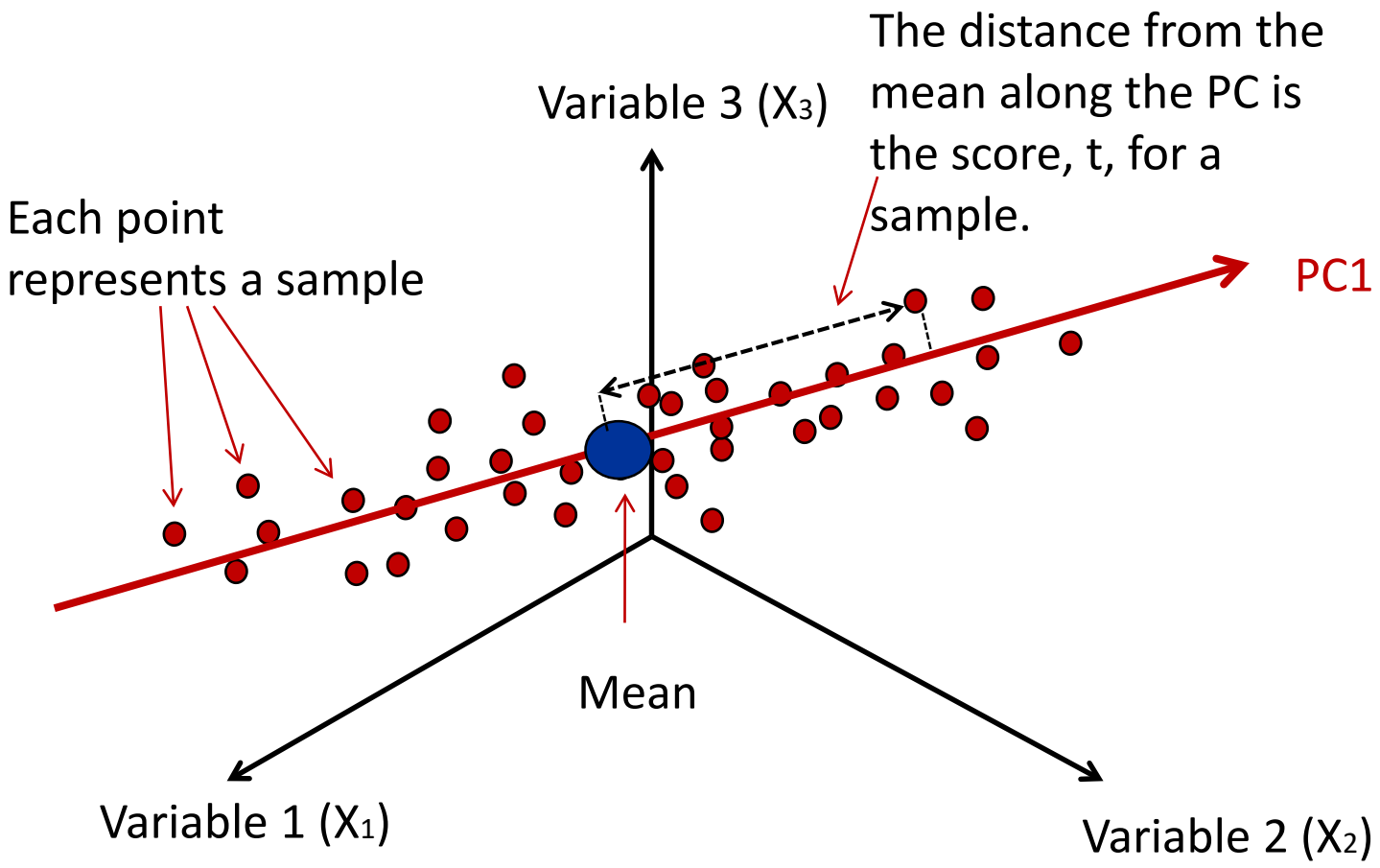
We can now draw a 2-dimensional plot of the projected objects by using PC1 and PC2 as a new coordinate system



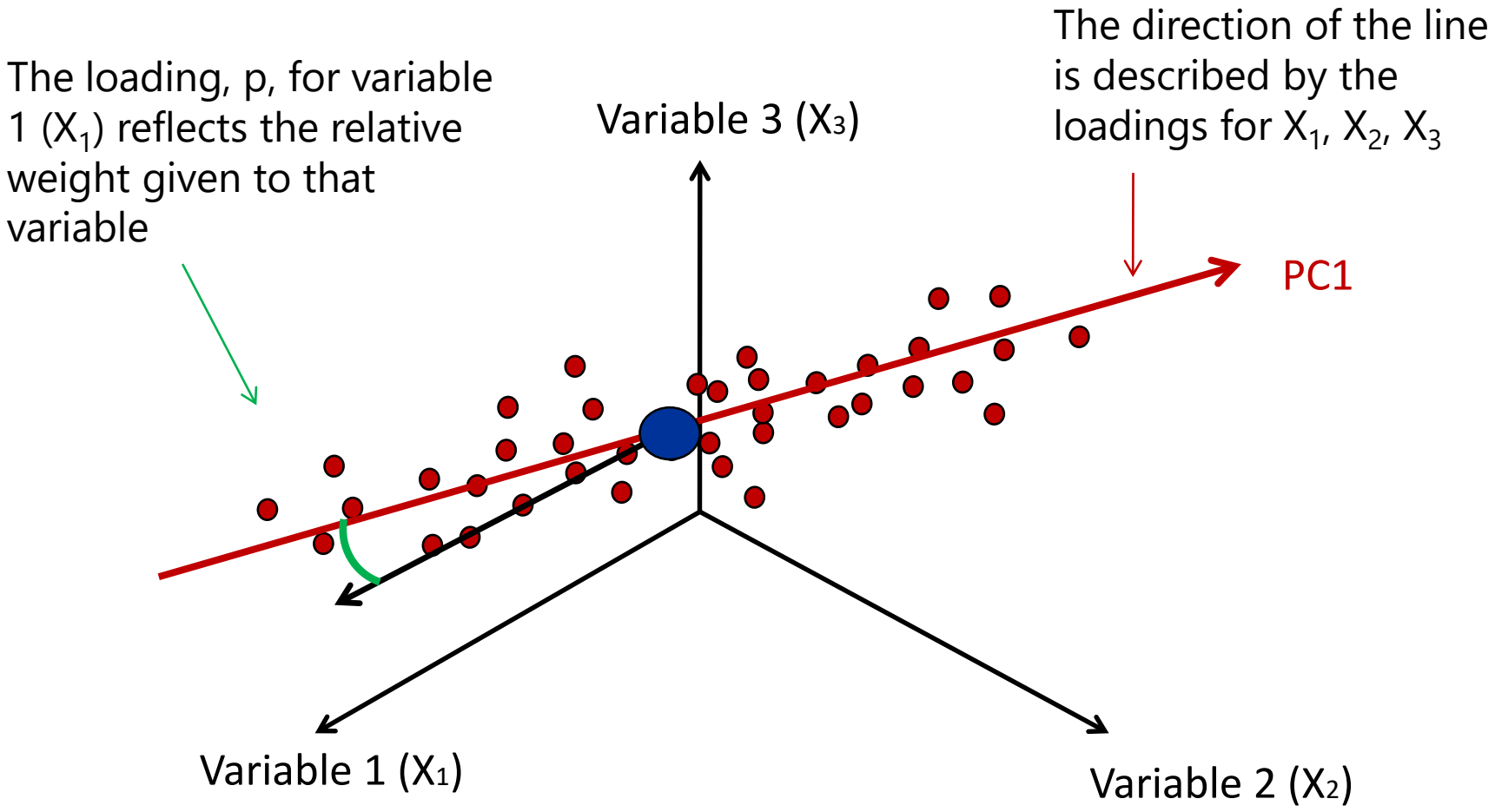


This map of objects in the Principal components plot is called a score plot

Score


The distance from the mean along the PC is the score, t, for a sample.
Variable 3 (X_3)
Each point represents a sample
PC1
Mean
Variable 1 (X_1)
Variable 2 (X_2)

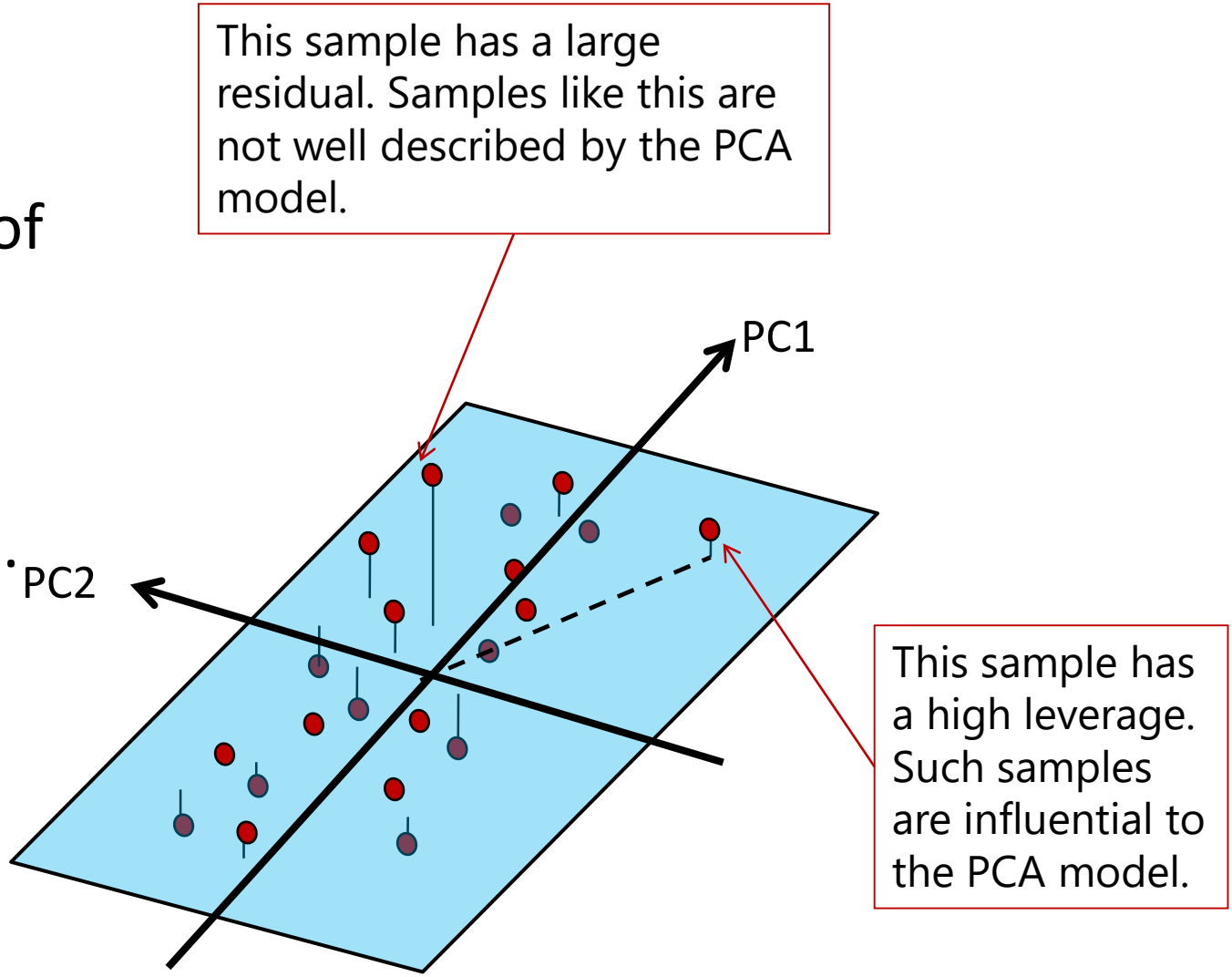
Loading


The loading, p, for variable 1 (X_1) reflects the relative weight given to that variable
Variable 3 (X_3)
The direction of the line is described by the loadings for X_1, X_2, X_3
PC1
Variable 1 (X_1)
Variable 2 (X_2)

Outlier detection

This example shows a projection of a set of samples onto PC1 and PC2.

- Residual: Distance to (multidimensional) model plane.
- Leverage: Distance from center along (multidimensional) model plane



Interpreting a PCA model

Number of components to take into account

The number of components in the model characterises the structure of the data: the fewer PCs needed, the simpler the model (and easier interpretation?)

Scores

The score plot shows how the data are distributed. Sample patterns, groupings, similarities and differences can be studied

Loadings (or Correlation Loadings)

The loading plot is useful to understand the correlations between the variables

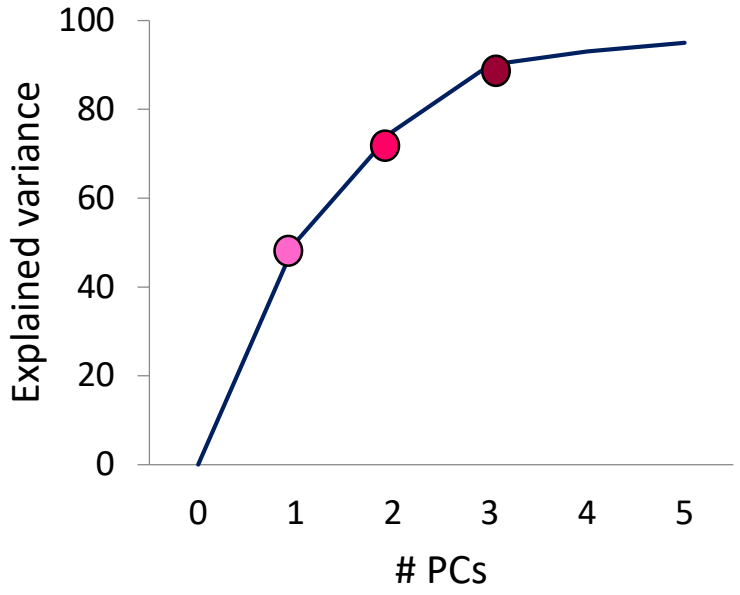
Residuals and plots for detecting outliers

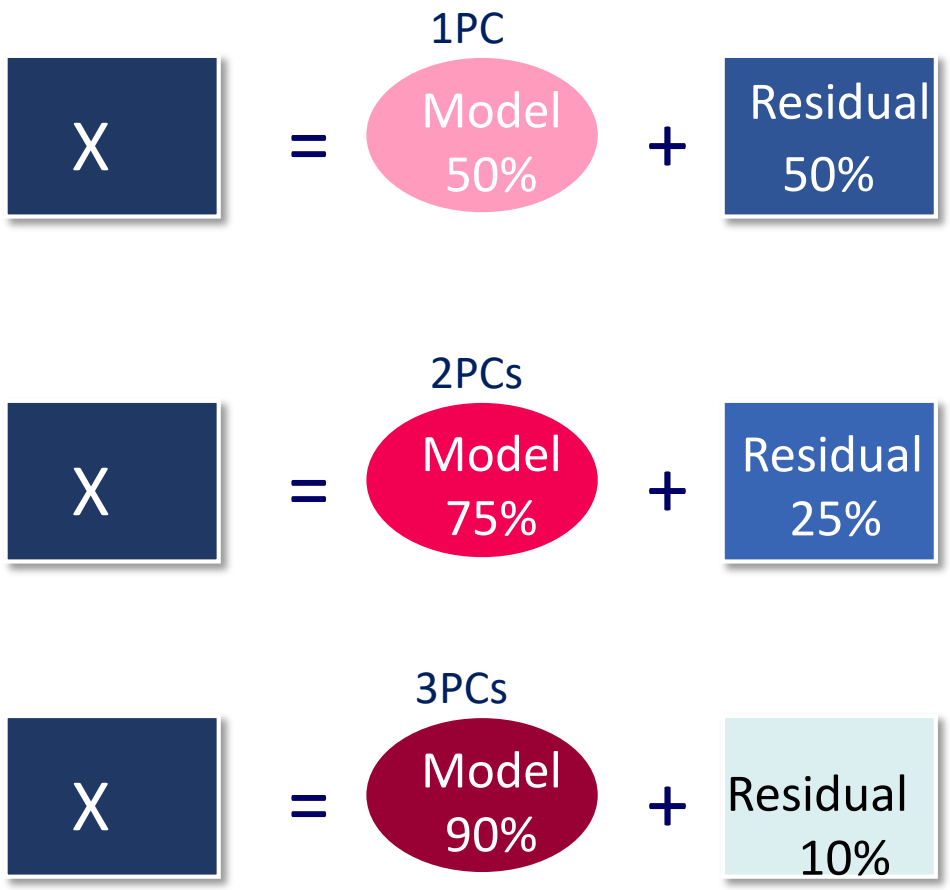
Various plots of residuals and other diagnostic tools give information about possible outliers and anomalies

Choice of the number of components

Keep adding components as long as they carry structured information:

- Interpretable differences between samples
- Explains a significant amount of variation

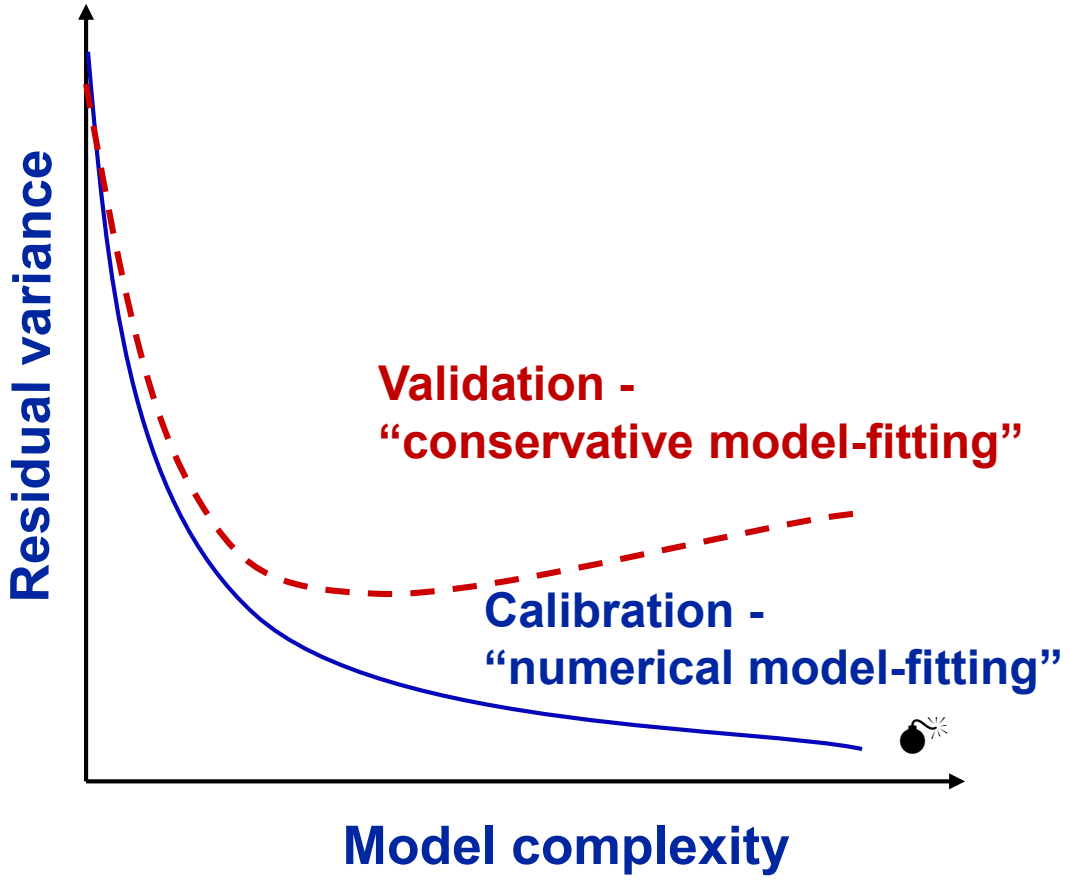




Finding the correct model rank: Validation is essential!

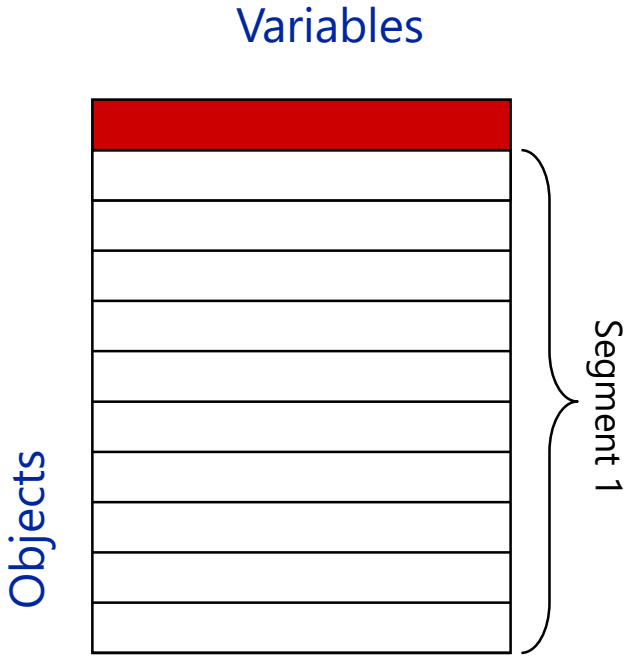
The validation level is important and must reflect the various sources of variations:

- Raw material variations
- Sensors
- Time
- Location
- … specific to your application!



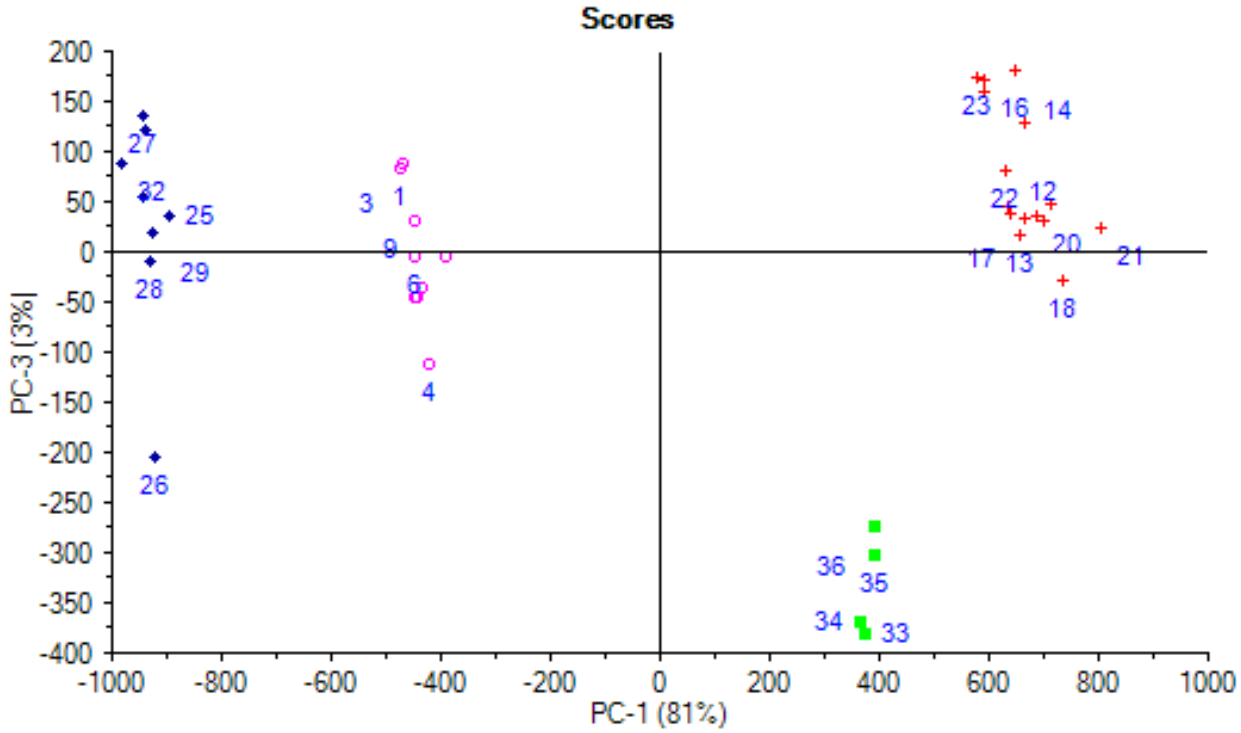
Cross-validation for PCA

- Leave out some objects, make submodels until all objects have been taken out once
- Be clear about at what level you want to validate
- If you have no obvious groups of samples or background knowledge, a 10-segment random CV is suggested



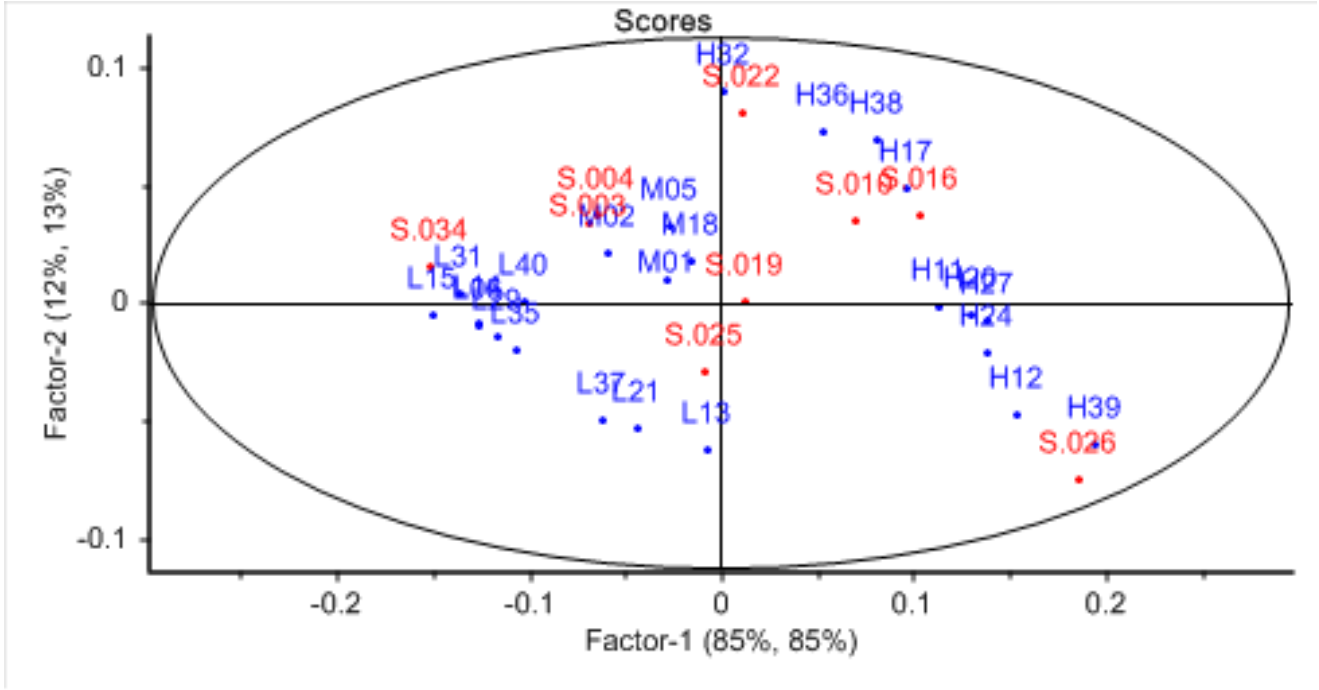
Scores

- Score plot on the plane spanned by the first two dominant PCs gives idea about:
  - Any specific pattern in the data
  - Tells about the similarity or dissimilarity of the data points



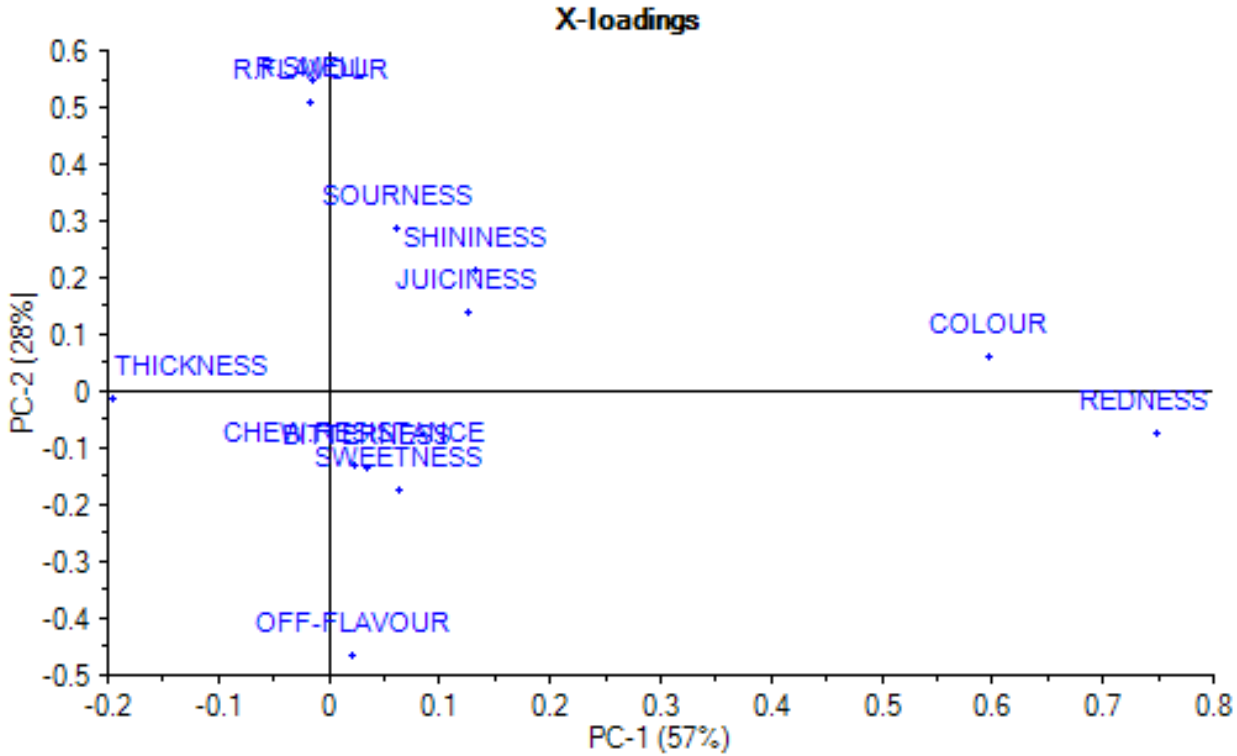
Calibration and validation scores

- Calibration and Validation (Test) scores in the same plot, Use this plot to determine whether the test set covers the entire span of the calibration set or determine if any cross validation segments/samples are different from the rest of the set.



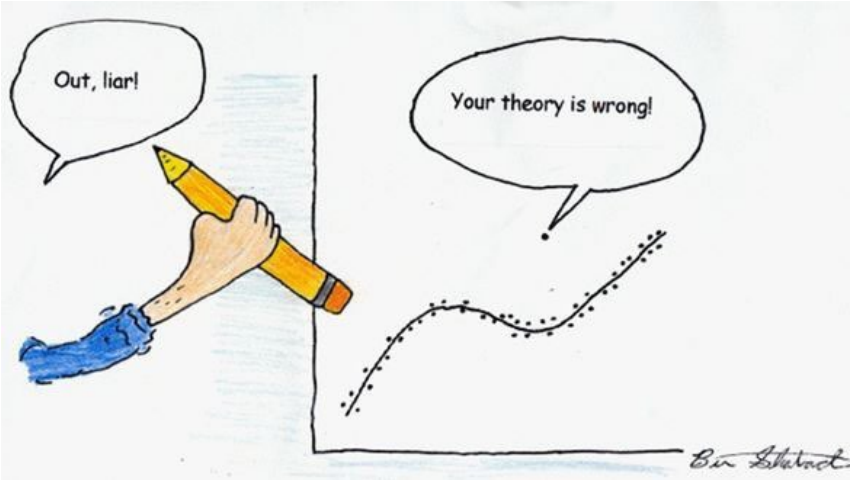
X-variable correlation structure

- Variables **close** to each other in the loadings plot will have a **high positive correlation** if *the two components explain a large portion of the variance* of X. The same is true for variables in the same quadrant lying close to a straight line through the origin. Variables in diagonally opposed quadrants will have a tendency to be negatively correlated.



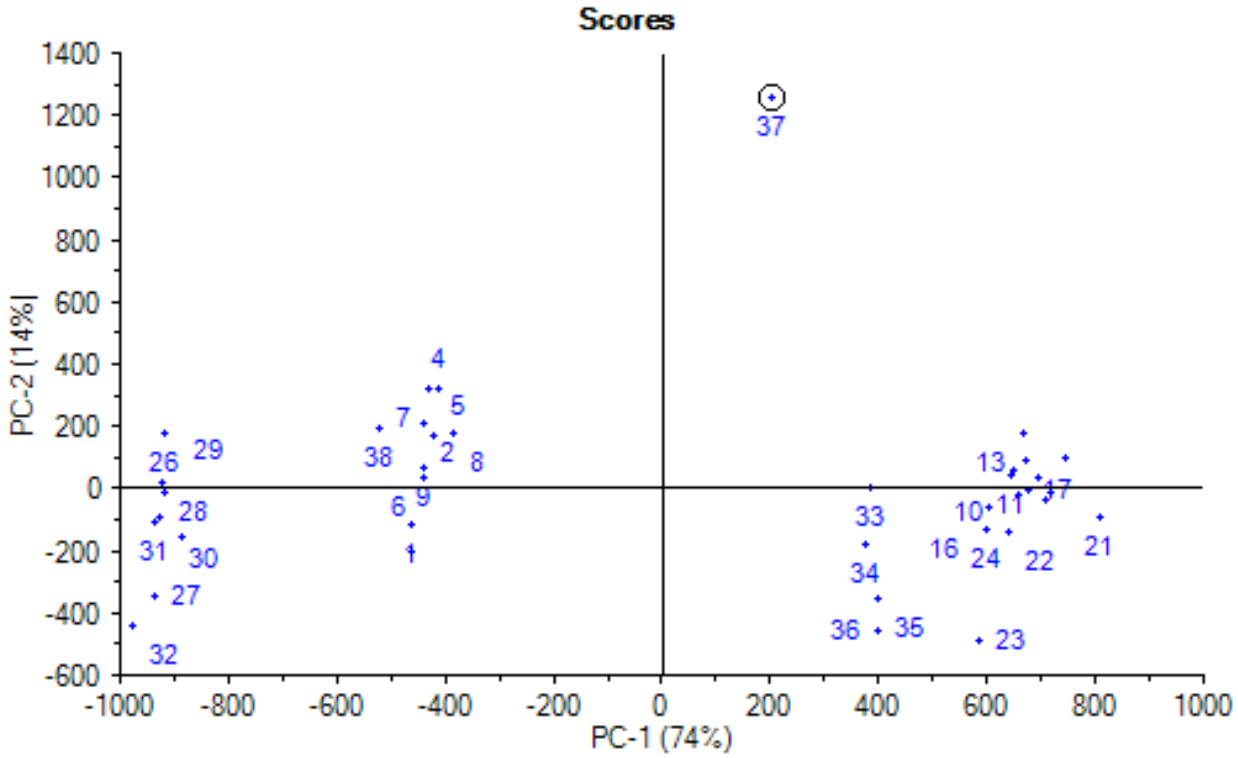
Outlier detection

- An outlier is an object which deviates from the other objects in a model and may not belong to the same population as the majority
- Outliers can disturb the model
- Causes of outliers
  - Measurement error
  - Wrong labelling
  - Deviating products / processes
  - Noise
  - Extreme / interesting samples



Detecting outliers in the score plot

- Are some samples very different from the rest? This can indicate that they are outliers, as shown in the figure below. Outliers should be investigated: there may have been errors in data collection or transcription, or those samples may have to be removed if they do not belong to the population of interest.

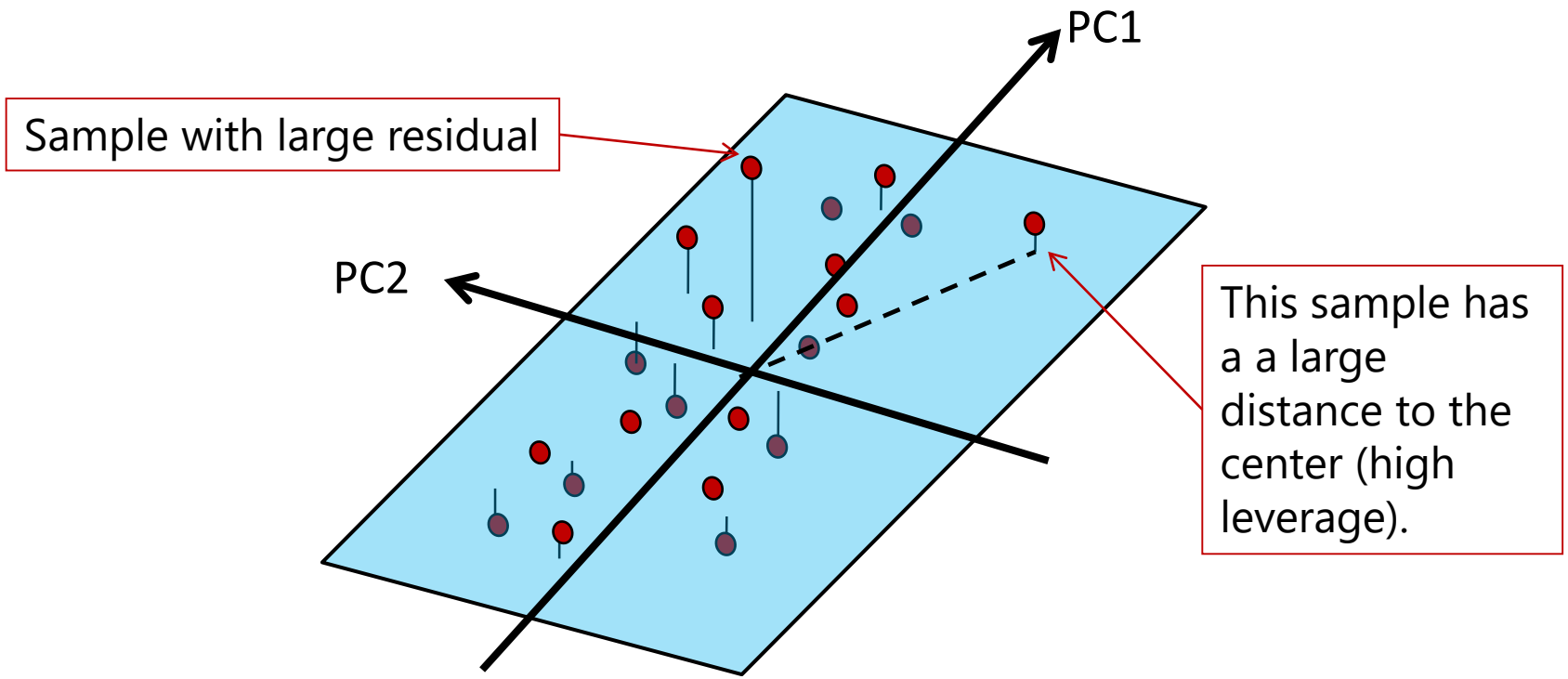


Outlier-sensitive prediction uncertainty

Based on similarity to calibration objects:

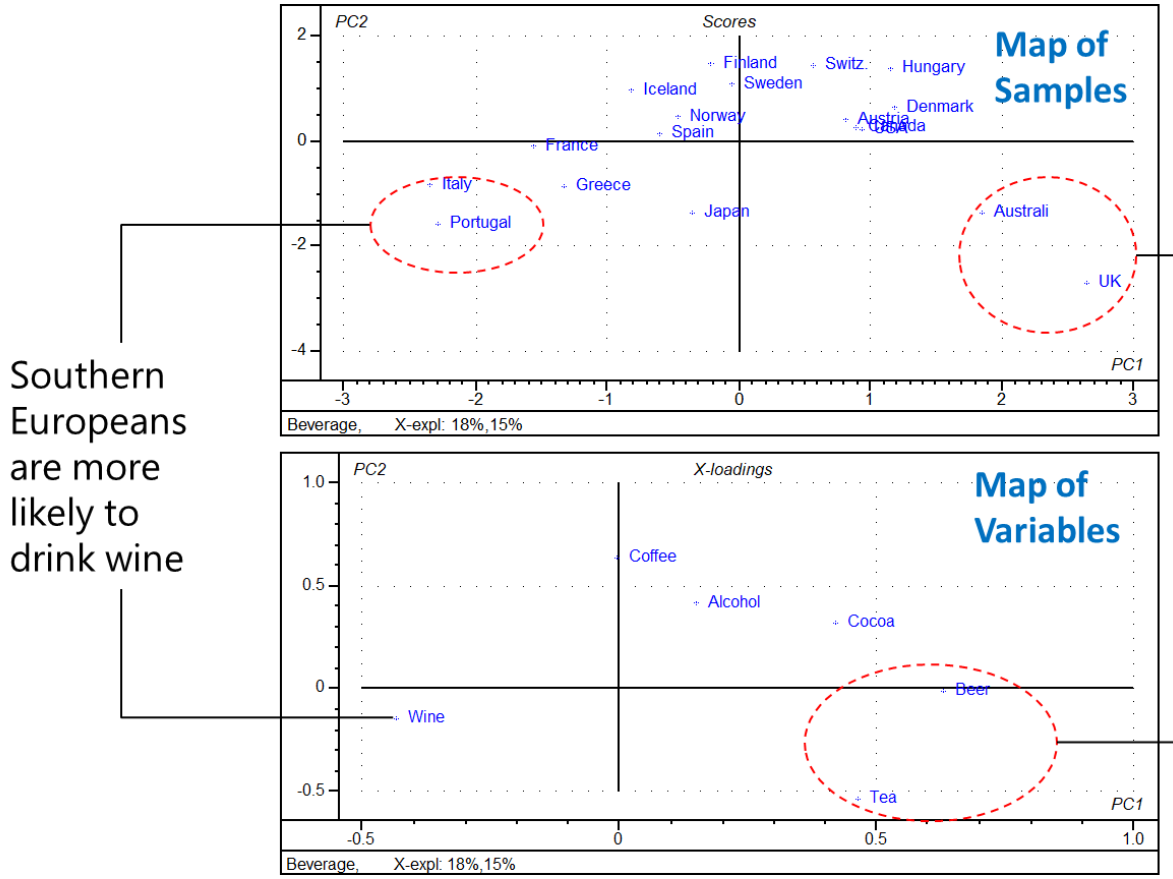
- Similar: Reliable
- Dissimilar: Unreliable

} conventional outlier detection tools



Scores and loading plots together

- Score plot on the plane spanned by the first two dominant PCs gives idea about:
  - Any specific pattern in the data
  - Tells about the similarity or dissimilarity of the data points



Southern Europeans are more likely to drink wine

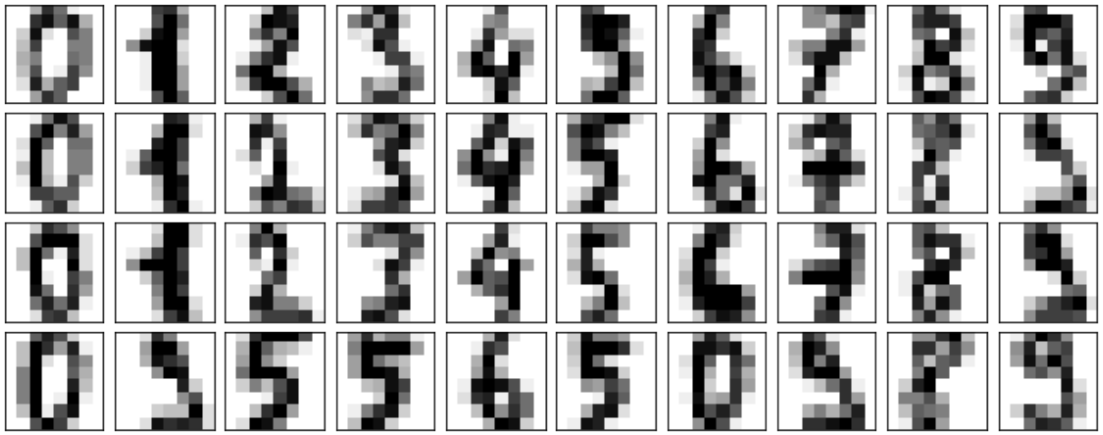
Australians and the English are more likely to drink Beer and Tea

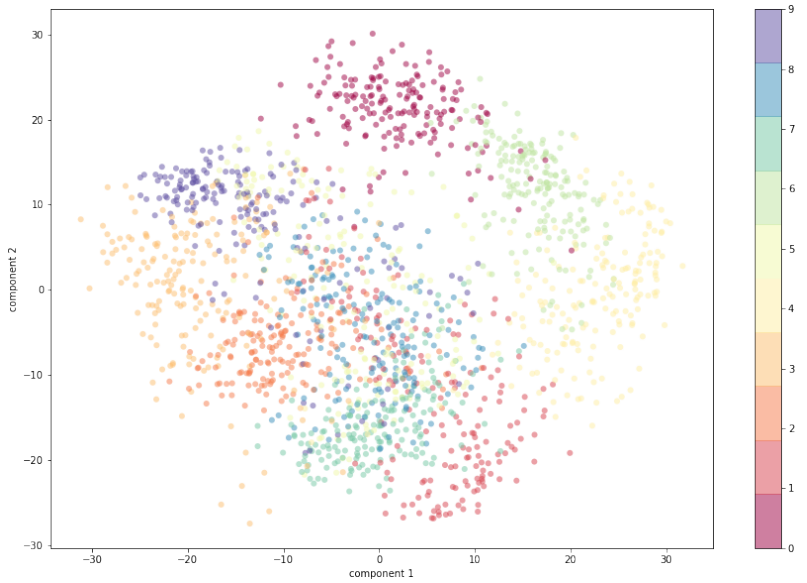
PCA: Applications

- Clustering
- Noise Reduction
- Developing Insight
- Outlier Detection
- Dimensionality Reduction
- Model order reduction / Reduced Order Model

Clustering and noise reduction

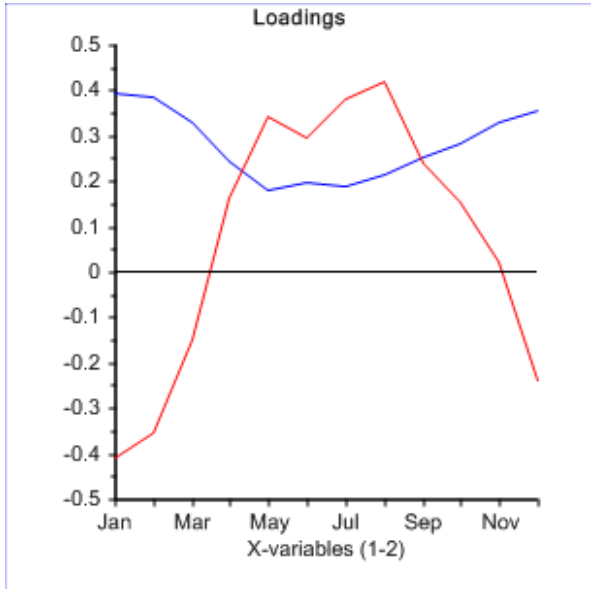
https://github.com/adil-rasheed/TK8117/blob/master/Lecture4/PCA-Clustering.ipynb

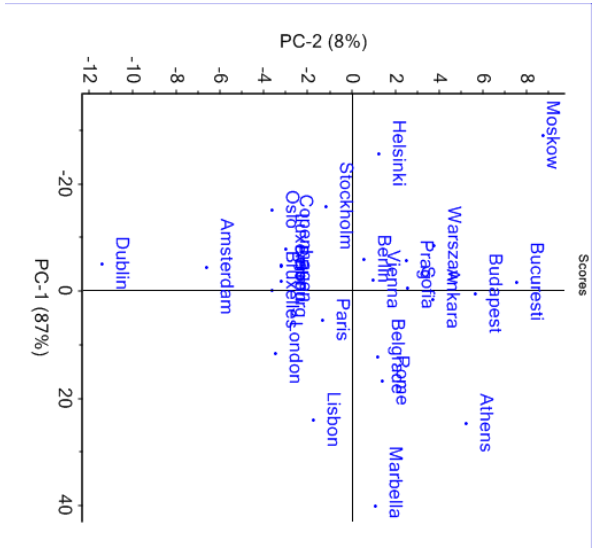




Developing insight

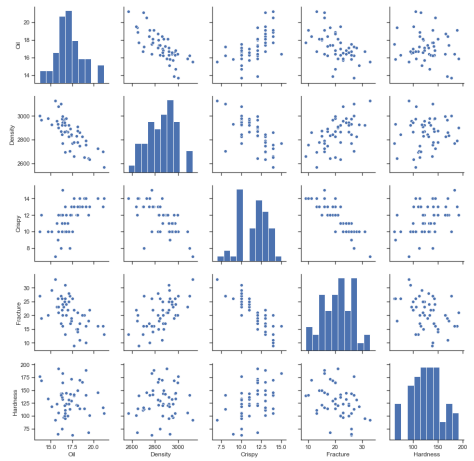
Score plot
Loadings plot

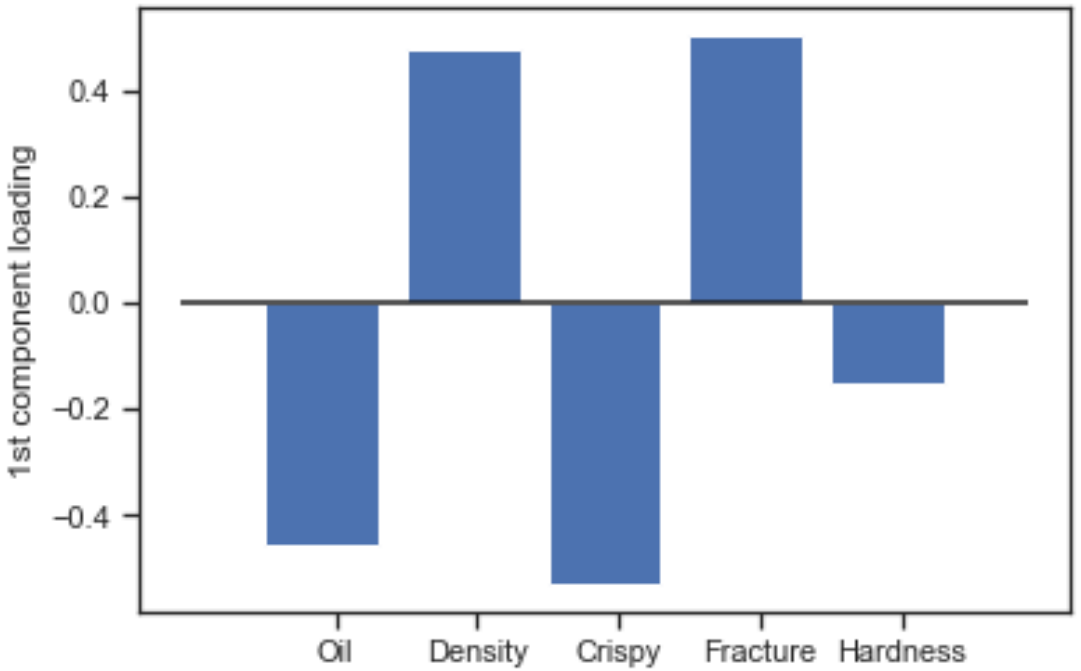




Developing insight: Pastry characterization

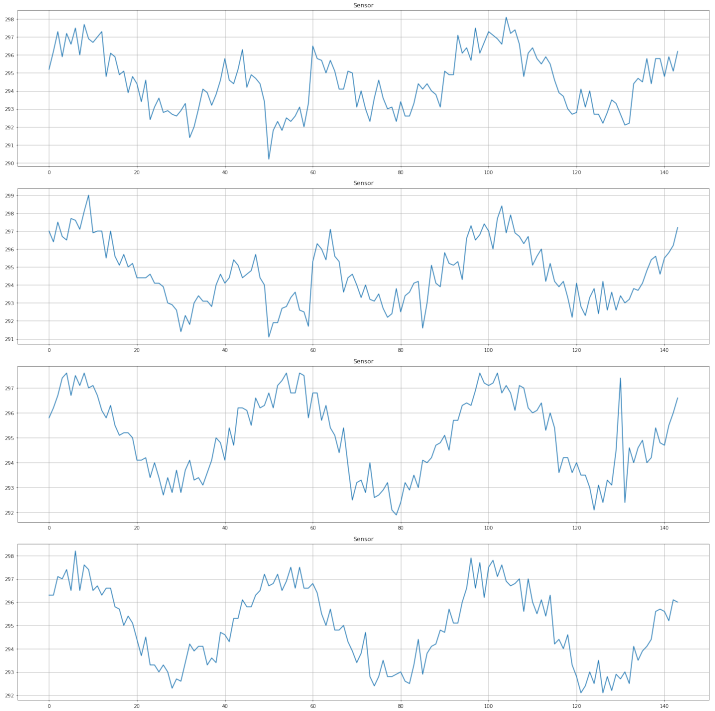
- https://github.com/adil-rasheed/TK8117/blob/d52fdeb305d043592faecfa1afd5fadf1d0f6c57/Lecture4/PCA-GettingInsight.ipynb

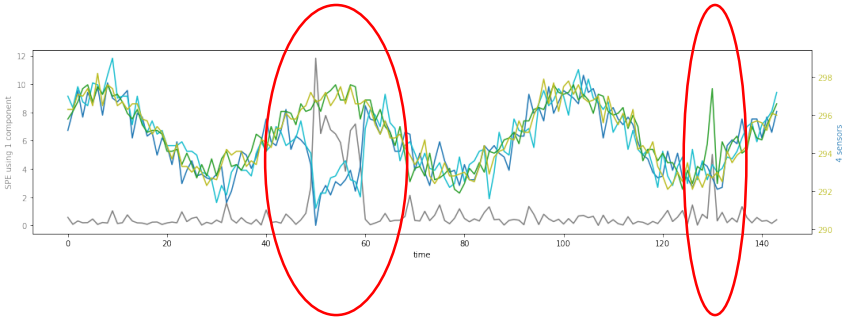




Outlier detection

- https://github.com/adil-rasheed/TK8117/blob/d52fdeb305d043592faecfa1afd5fadf1d0f6c57/Lecture4/PCA_Residuals.ipynb





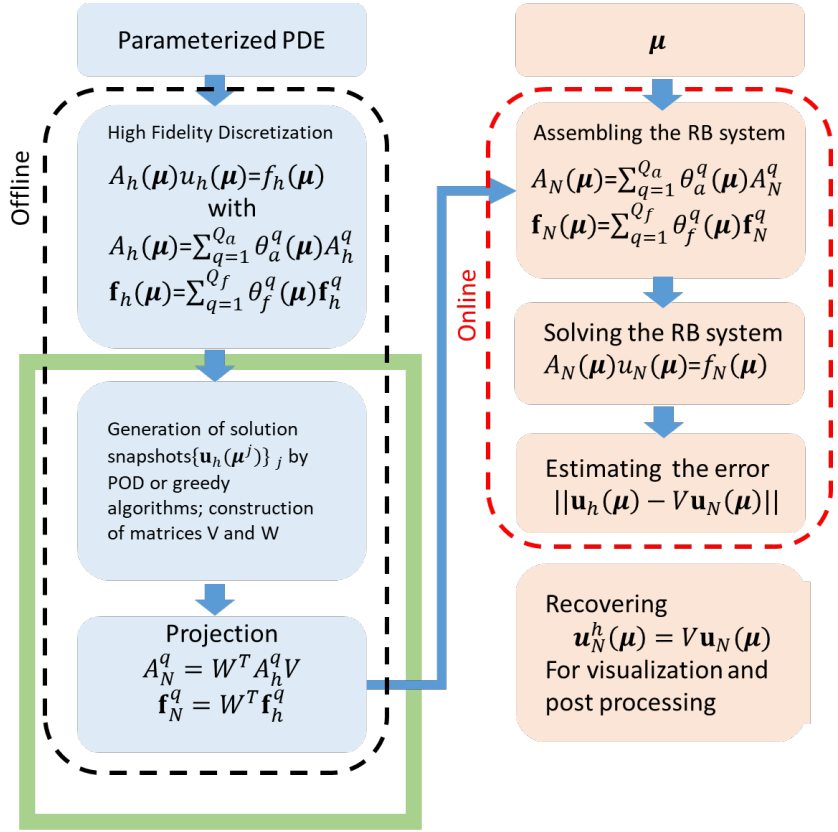
Reduced Order Modeling

Explicitly combining partial differential equations with unsupervised dimensionality reduction algorithms

μ can be any parameter

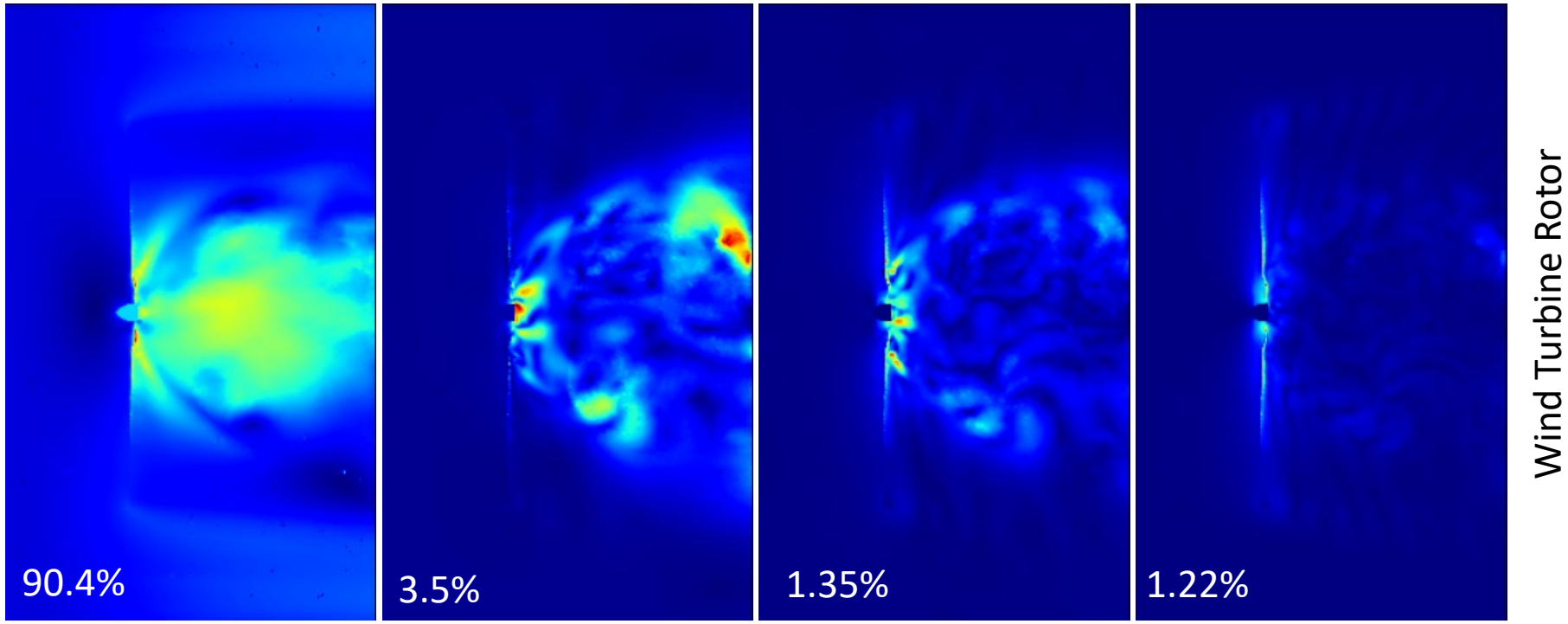
Chosen set of values for μ

Unsupervised Learning Algorithms for optimally Spanning the parameter space
And for projecting high dimensional solutions to a lower dimensional space to construct the reduced bases



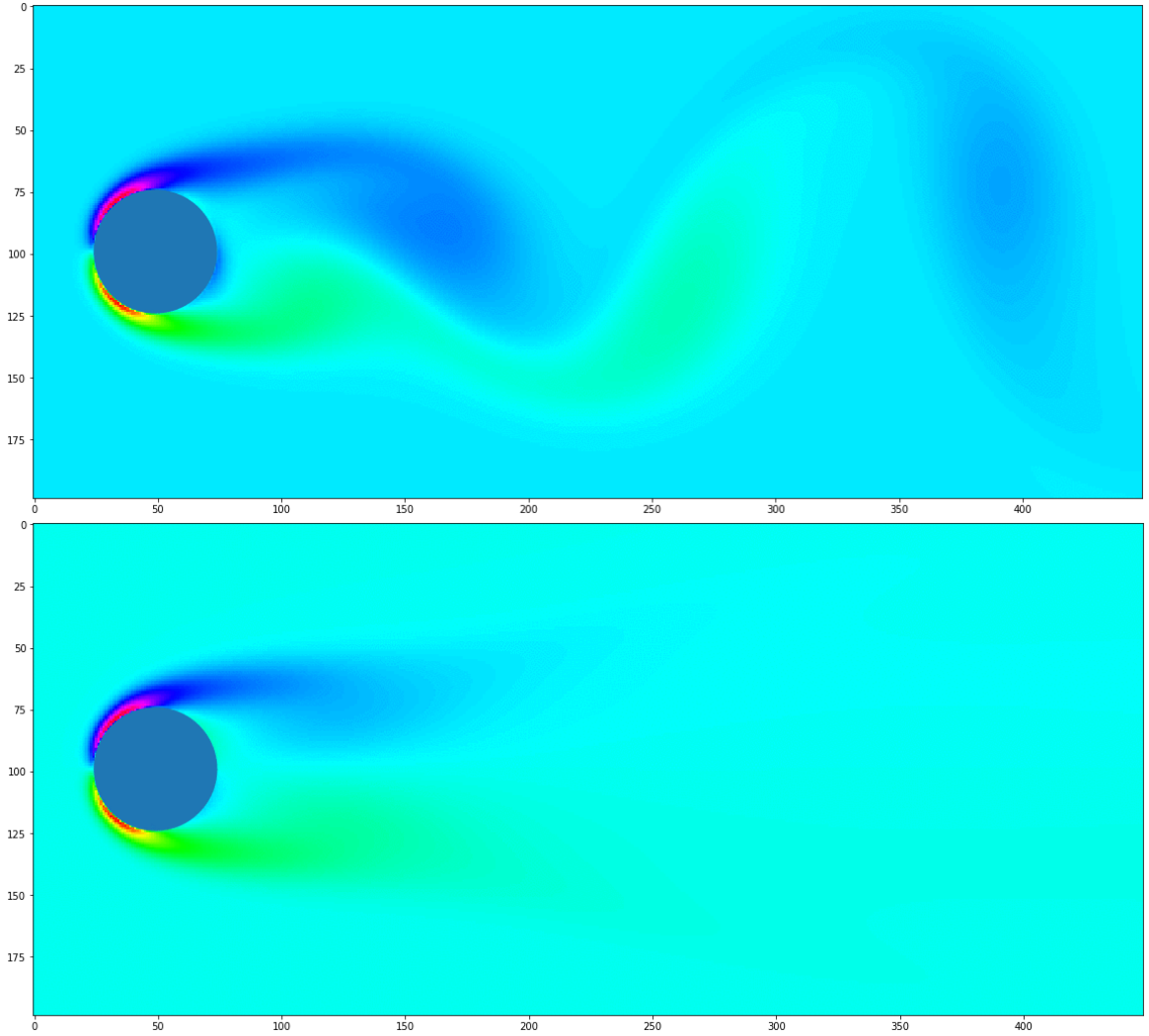
The Reduced bases are used to compute reduced dimensional solutions

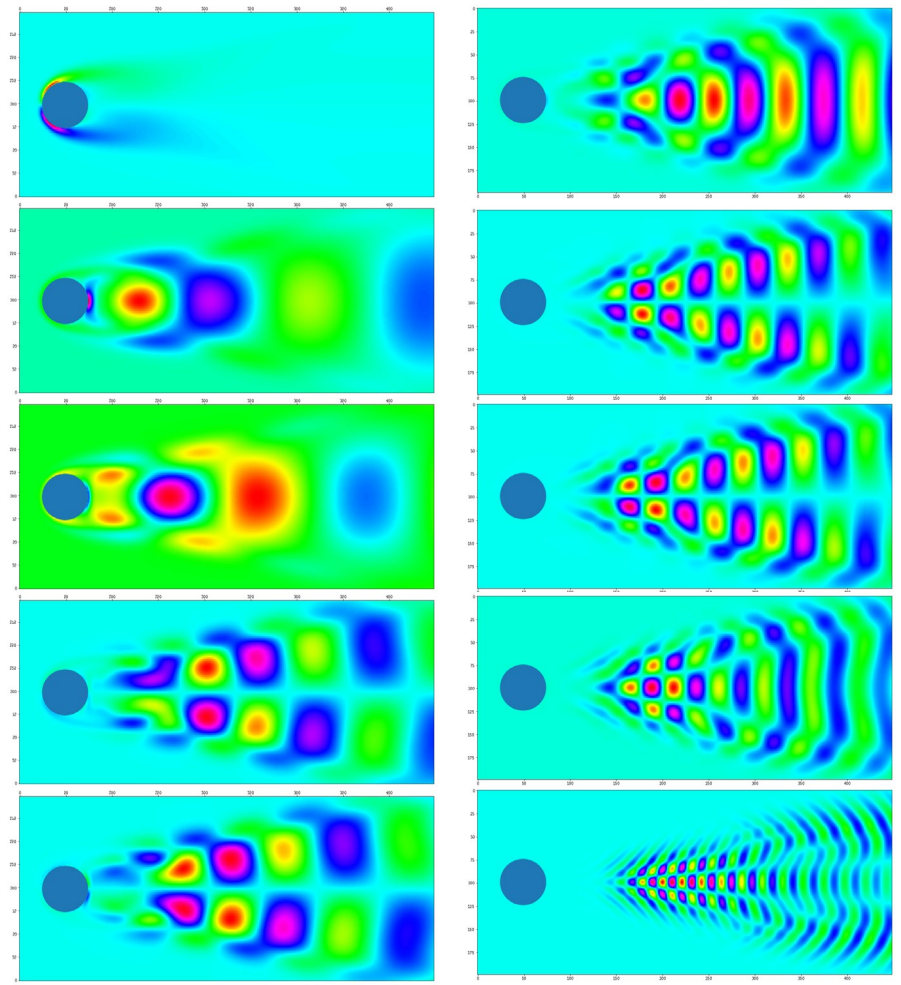
N << h

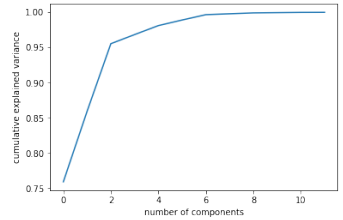


PCA shows that only a few modes are sufficient to explain the full physics. The technique can be used to develop Reduced Bases to be used in a Reduce Order Modeling

Analyzing simulation results

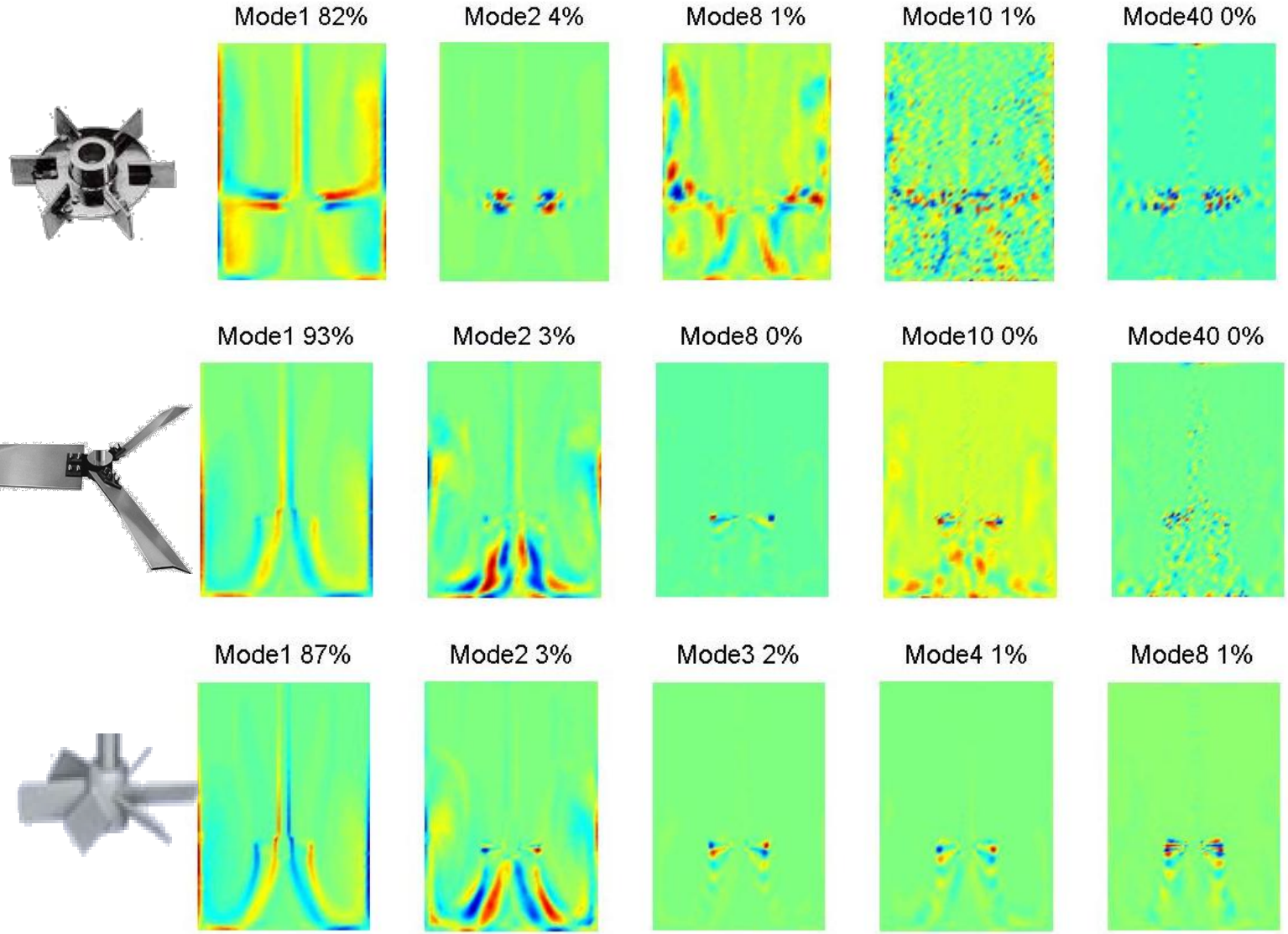




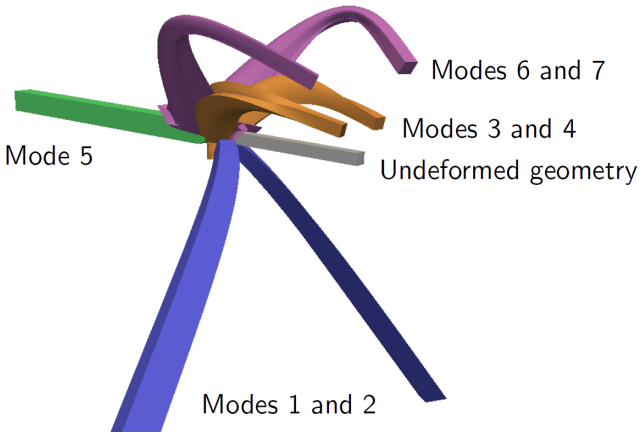


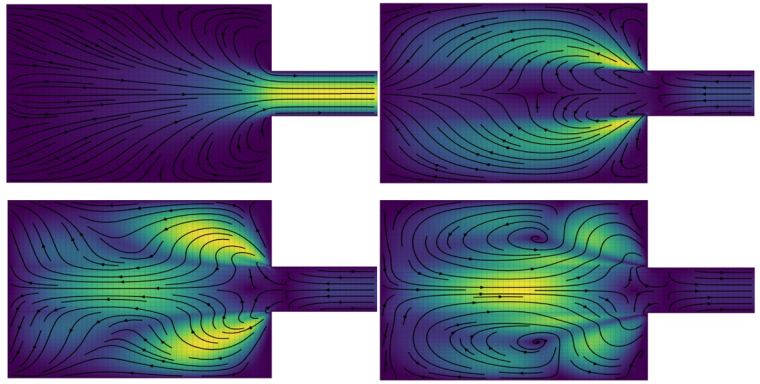
https://github.com/adil-rasheed/TK8117/blob/master/Lecture4/PCA-Cylinder.ipynb

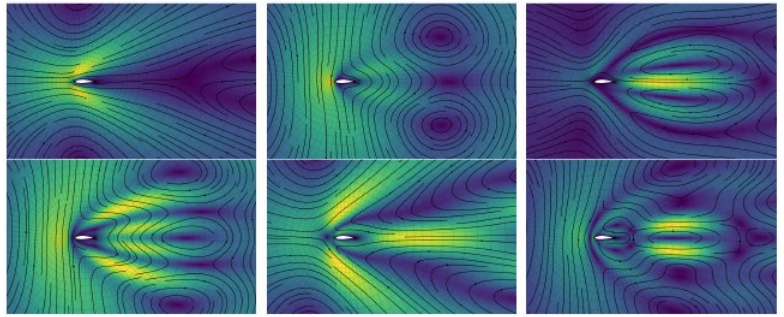
Designing rotating machinary

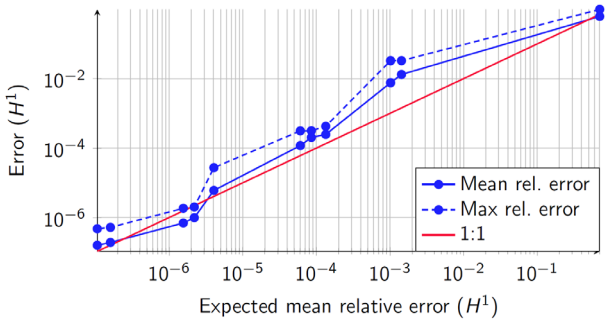


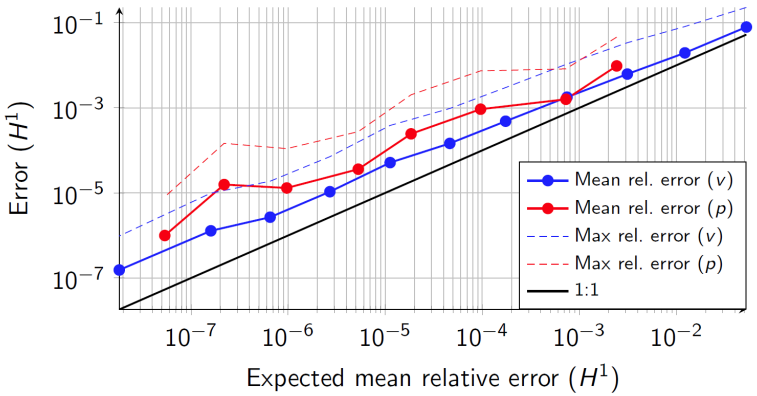
Reduced Order Modeling

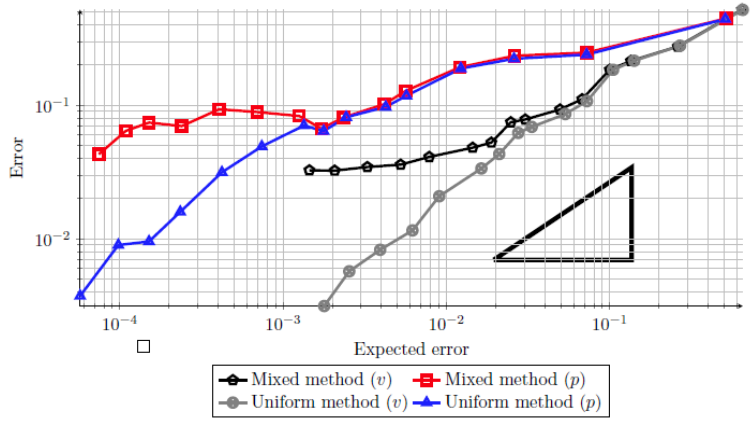












The high-fidelity method ($150 \times 10 \times 10$ elements) took about 5 seconds on average to solve each problem instance.

Each reduced method completed the solution in 200 microseconds on average. (The systems are too small to see a noticeable dependence on M.)

Speedup factor: 25000

The high-fidelity method (18847 DoFs) took about **25 seconds** on average to solve each problem instance.

Each reduced method completed the solution in **1 millisecond** or less on average.

Speedup factor: 25000

	♯ DoFs	Speedup	Relative error (velocity)	Relative error (pressure)
High fidelity	110	1	0	0
Uniform method	5	15370	1.07×10^{-1}	3.07×10^{0}
	10	4122	3.02×10^{-2}	9.06×10^{-1}
	15	1972	1.01×10^{-3}	3.3×10^{-1}
	20	1011	3.03×10^{-3}	3.01×10^{-1}
Mixed method	5	25981	1.10×10^{-1}	3.12×10^{0}
	10	6902	4.75×10^{-2}	1.98×10^{0}
	15	2936	3.51×10^{-2}	5.0×10^{-1}
	20	1764	3.01×10^{-2}	5.3×10^{-1}

Requires large number of computationally expensive simulations so Design of Experiments is necessary to do it optimally.

Timeseries data analyais

Unscambler Demonstration

Download the data from blackboard: Ozone Timeseries

Data presentation

- Kurdistan Chawshin: Lithography classification
- Abu Md Ariful Islam: Water Injection
- Erlend Torje Berg Lundby: Data from the Aluminium extraction process