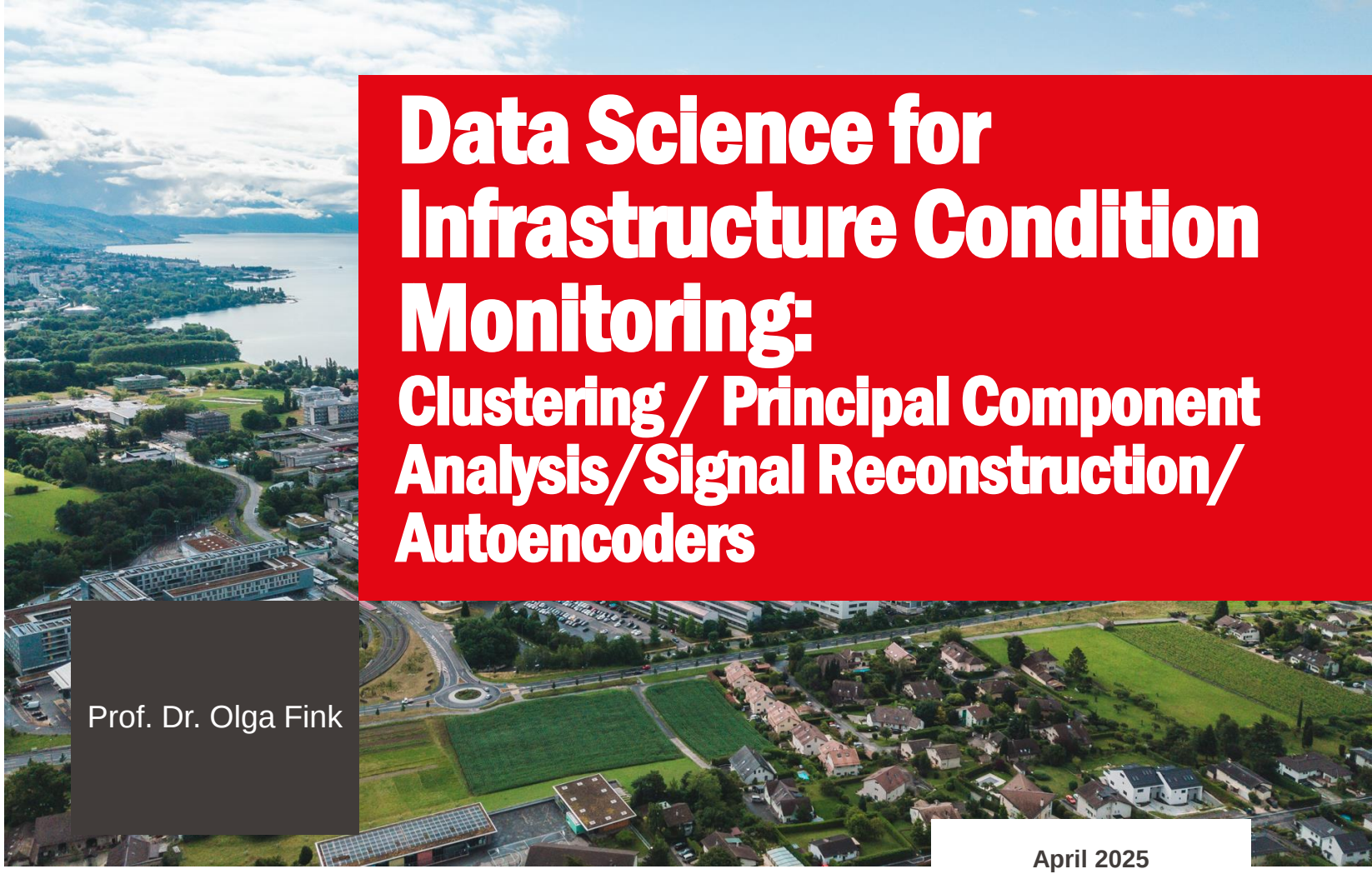


Data Science for Infrastructure Condition Monitoring:

Clustering / Principal Component Analysis/Signal Reconstruction/ Autoencoders

Prof. Dr. Olga Fink

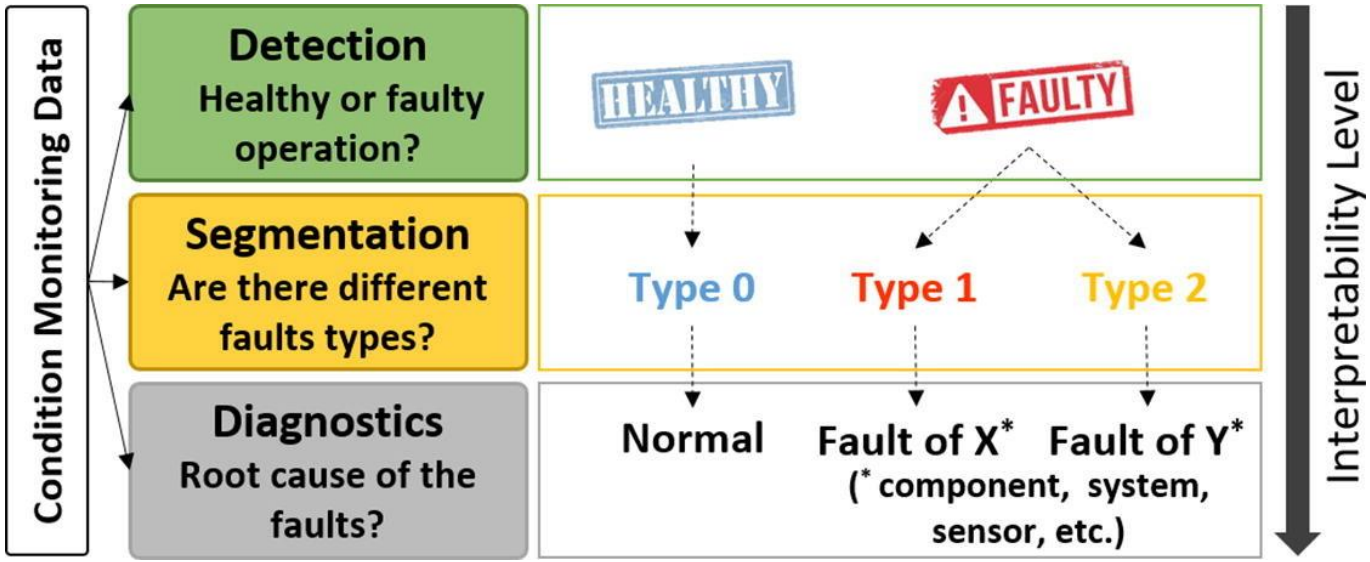
École polytechnique fédérale de Lausanne

April 2025

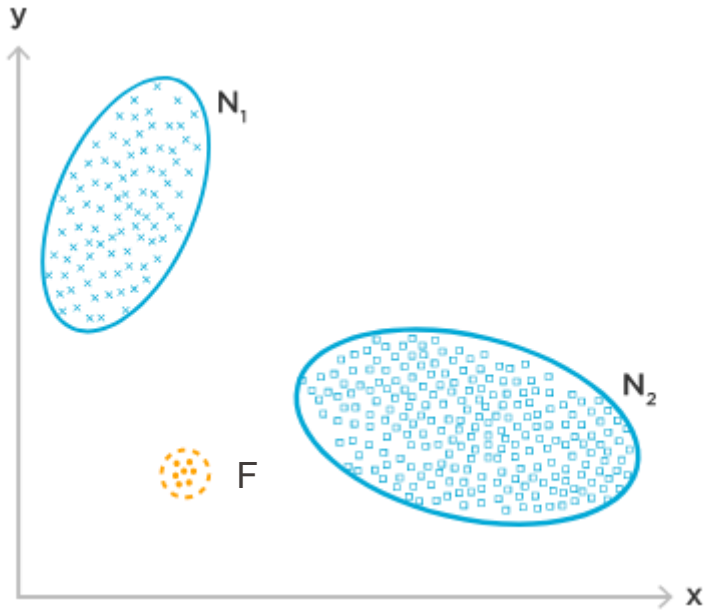
Clustering algorithms

Clustering for fault detection

- assumption that similar data points tend to cluster together in groups, as determined by their proximity to local centroids.
- data instances that fall outside of these groups are considered as data anomalies



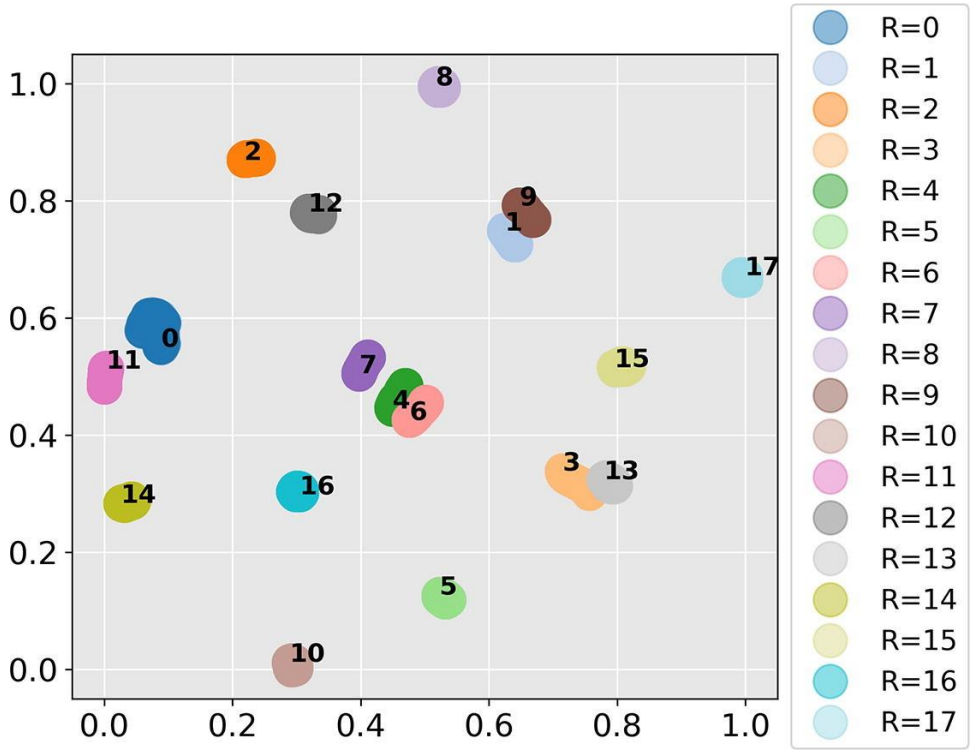
Clustering for fault detection



Clustering for fault dignostics

- Segmentation of different fault types in an unsupervised way
- Only mapping between which cluster belongs to which fault type is missing

Clustering for fault diagnostics



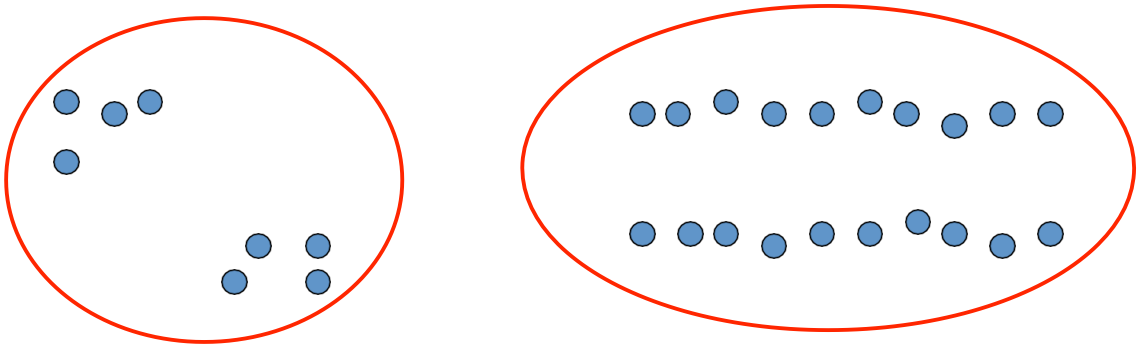
Arias Chao, M., Adey, B. T., & Fink, O. (2021). Implicit supervision for fault detection and segmentation of emerging fault types with Deep Variational Autoencoders. *Neurocomputing*, *454*, 324-338.

Cluster analysis

- Cluster: A collection of data objects
  - similar (or related) to one another within the same group
  - dissimilar (or unrelated) to the objects in other groups
- Cluster analysis (or *clustering*, *data segmentation, …*)
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- The subgroups are chosen such that the intra-cluster differences are minimized and the inter-cluster differences are maximized.
- Unsupervised learning: no predefined classes
- Applications of cluster analysis:
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

Clustering

- Basic idea: group together similar instances
- In the context of feature selection: group together similar features (and replace the groups by a «representative» feature)



- How do we define similarity?
  - Classical: Euclidean distance, Manhatten distance
  - Correlation-based distances

Clustering for pre-processing

- Summarization:
  - Preprocessing for regression, PCA, classification, and association analysis
- Compression:
  - Image processing: vector quantization
- Finding K-nearest Neighbors
  - Localizing search to one or a small number of clusters
- Outlier detection
  - Outliers are often viewed as those “far away” from any cluster

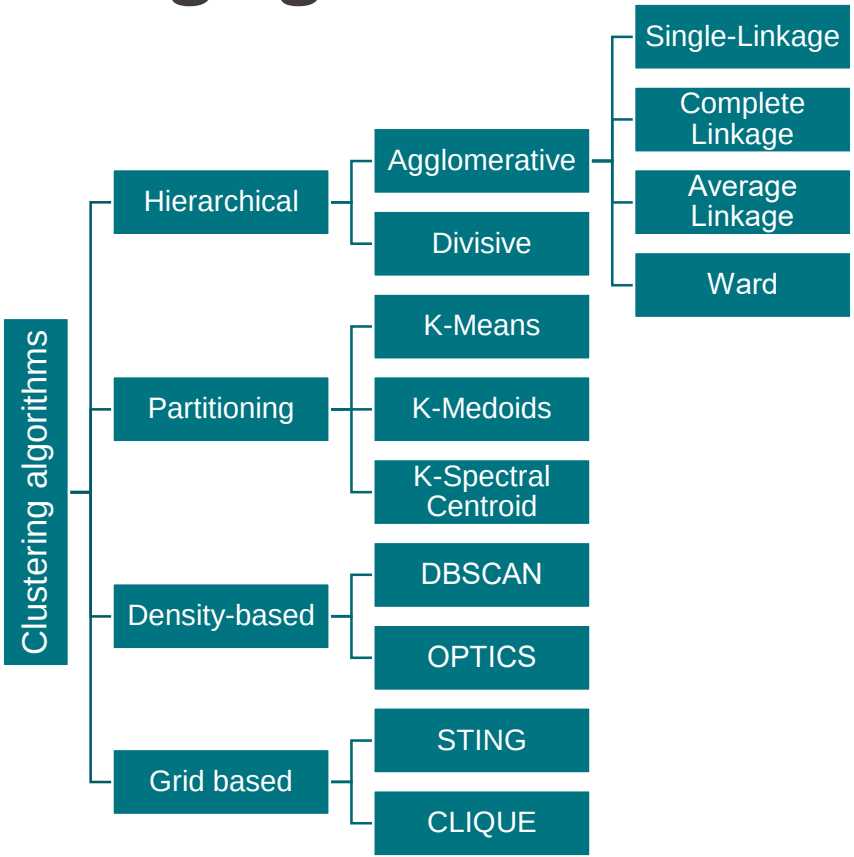
Quality of the cluster analysis

- A good clustering method will produce high quality clusters
  - high intra-class similarity: **cohesive** within clusters
  - low inter-class similarity: **distinctive** between clusters
- The quality of a clustering method depends on
  - the similarity measure used by the method
  - its implementation, and
  - Its ability to discover some or all of the hidden patterns

Requirements and Challenges

- Scalability
  - Clustering all the data instead of only on samples
- Ability to deal with different types of attributes
  - Numerical, binary, categorical, ordinal, linked, and mixture of these
- Constraint-based clustering
  - User may give inputs on constraints
  - Use domain knowledge to determine input parameters
- Interpretability and usability
- Others
  - Discovery of clusters with arbitrary shape
  - Ability to deal with noisy data
  - Incremental clustering and insensitivity to input order
  - High dimensionality

Major Clustering algorithms



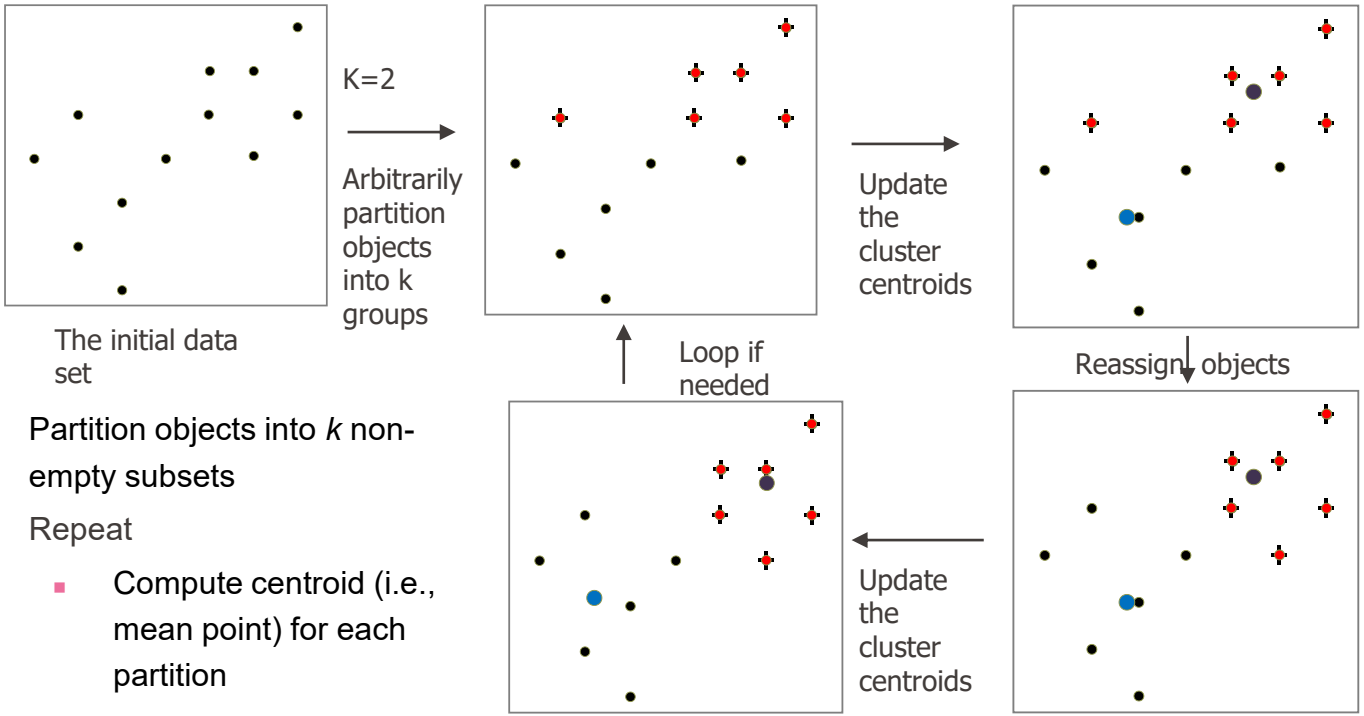
Partitioning based clustering

- Partitioning based clustering algorithms divide the dataset into initial ‘K’ clusters and iteratively improve the clustering quality based on a objective function.
- K-means is an example of a partitioning based clustering algorithm.
- Partitioning based algorithm are sensitive to initialization.

K-means procedure

- An iterative clustering algorithm
- Initialize: Pick *k* random points as cluster centers (k pre-defined)
- Alternate:
  1. Assign data points to closest cluster center
  2. Change the cluster center to the average of its assigned points
- Stop when no points' assignments change

An Example of *K-Means* Clustering

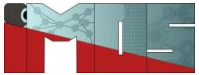


- Partition objects into *k* non-empty subsets
- Repeat
  - Compute centroid (i.e., mean point) for each partition
  - Assign each object to the cluster of its nearest centroid
- Until no change

Comments on the *K-Means* Method

- Strength: *Efficient*: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, $k, t << n$.
  - Comparing: PAM: $O(k(n-k)^2)$, CLARA: $O(ks^2 + k(n-k))$
- Comment: Often terminates at a *local optimal*.
- Weakness
  - Applicable only to objects in a continuous n-dimensional space
    - Using the k-modes method for categorical data
    - In comparison, k-medoids can be applied to a wide range of data
  - Need to specify *k,* the *number* of clusters, in advance (there are ways to automatically determine the best k (see Hastie et al., 2009)
  - Sensitive to noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

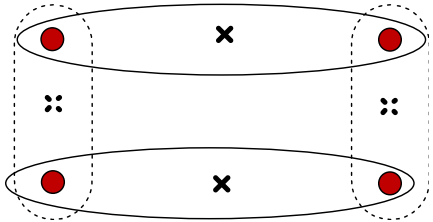
What Is the Problem of the K-Means Method?



- The k-means algorithm is sensitive to outliers !
  - Since an object with an extremely large value may substantially distort the distribution of the data
- K-Medoids: Instead of taking the **mean** value of the object in a cluster as a reference point, **medoids** can be used, which is the **most centrally located** object in a cluster

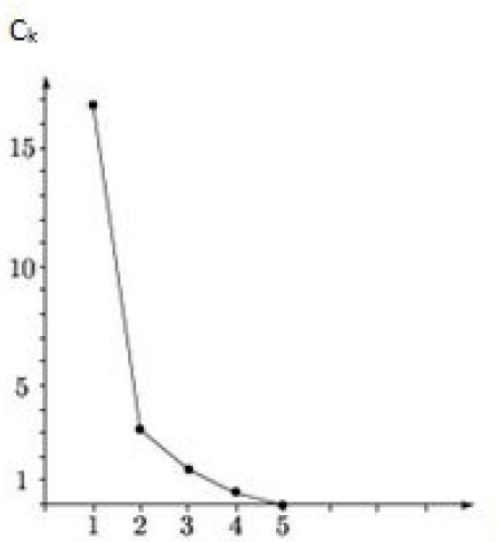
Variations of the K-Means Method

- Most of the variants of the *k-means* differ in
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means



- Handling categorical data: *k-modes*
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters
  - A mixture of categorical and numerical data: *k-prototype* method

Number of clusters



Number of clusters

C_k: Total within-clusters sum of squares

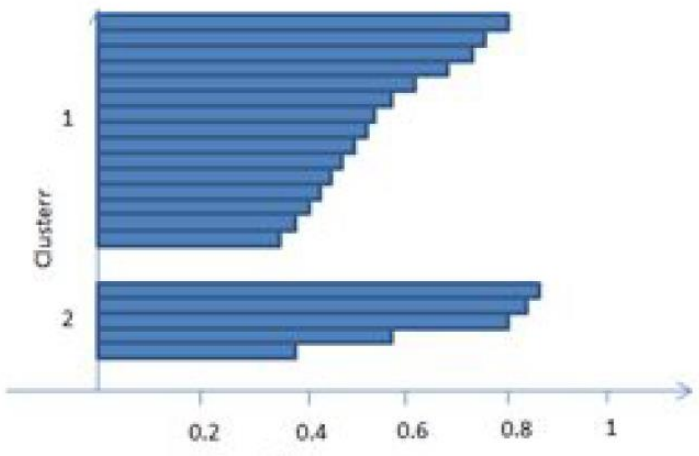
Silhoutte Coefficient

- Can be used to study the separation distance between the resulting clusters
- A measure how close each point in one cluster is to points in the neighboring clusters → can be assessed visually in silhoutte plots
- a(i) average distance between i and all observations within the same cluster
- b(i) be the smallest average distance of i to all points in any other cluster (excluding the cluster that it is member of)

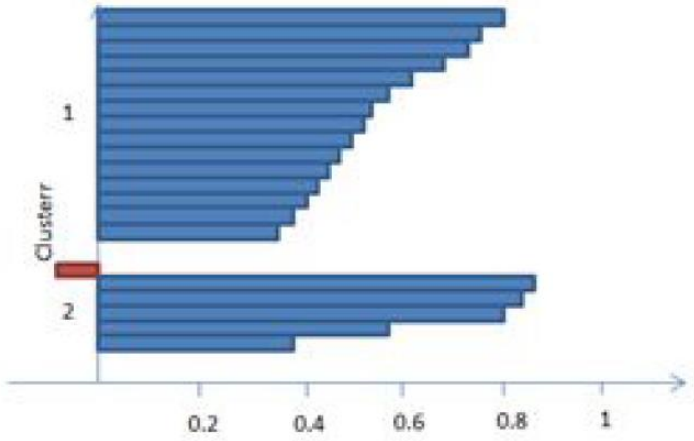
$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$

Silhoutte Plot: example



Silhoutte Coefficient



Silhoutte Coefficient

Spectral clustering

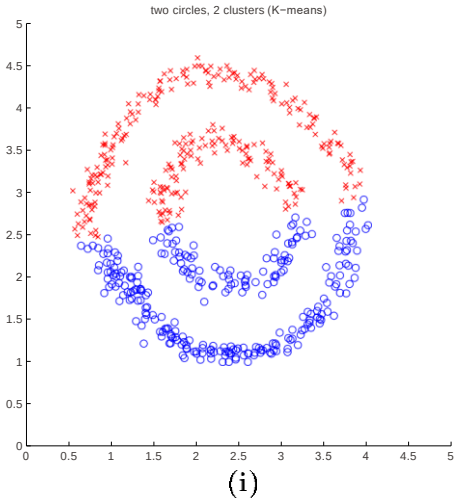
- Spectral clustering methods are attractive:
  - Easy to implement,
  - Reasonably fast especially for sparse data sets up to several thousands.

- Spectral clustering treats the data clustering as a graph partitioning problem without making any assumption on the form of the data clusters.

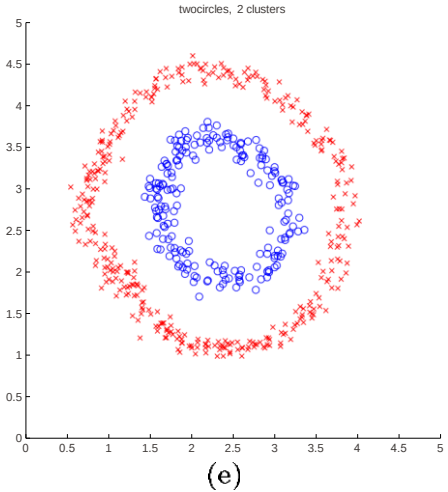
Source: Hamad & Biela

Spectral clustering

K-means → compactness



Spectral clustering → connectivity



[Shi & Malik ‘00; Ng, Jordan, Weiss NIPS ‘01]

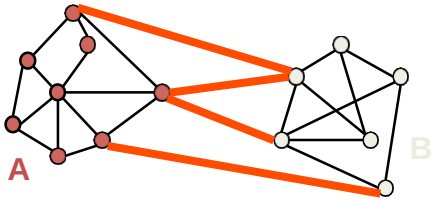
Key idea of spectral clustering

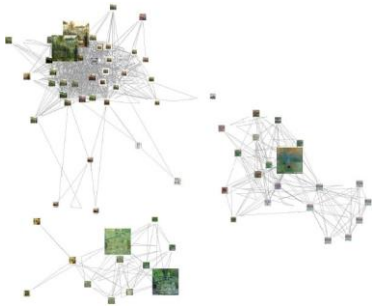
- Builds a similarity graph from the data.
- Computes the Laplacian matrix of the graph.
- Uses the eigenvectors of the Laplacian to embed the data into a lower-dimensional space.

→ great at capturing **manifold structures**, where clusters are connected but not necessarily blob-shaped.

Spectral clustering

Group points based on links in a graph





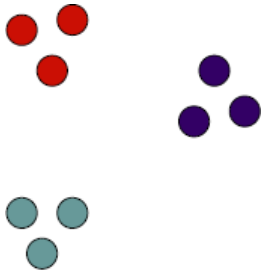
Source: James Hays

Spectral Clustering

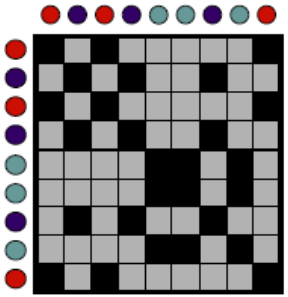
Goal: Given data points $X_1, \ldots, X_n$ and similarities $w(X_i, X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

Similarity Graph: G(V,E,W)

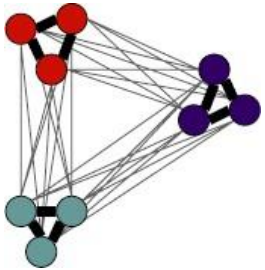
V – Vertices (Data points) E – Edge if similarity > 0
W - Edge weights (similarities)



Data



Similarities



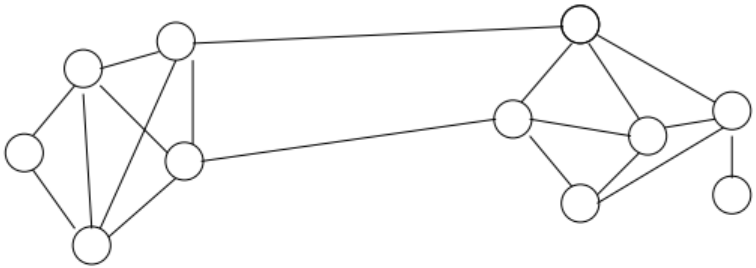
Similarity graph

Source: Hamad & Biela

Partition the graph so that edges within a group have large weights and edges across groups have small weights.

Graph notation

- $W = (w_{ij})$ adjacency matrix of the graph
- $d_i = \sum_j w_{ij}$ degree of a vertex
- $D = diag(d_1, \ldots, d_n)$ degree matrix
- $|A|$ = number of vertices in A
- $\text{vol}(A) = \sum_{i \in A} d_i$



In the following: vector $f = (f_1, ..., f_n)$ interpreted as function on the graph with $f(X_i) = f_i$.

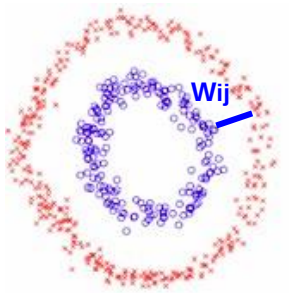
Source: Hein & Luxburg

Similarity graph construction

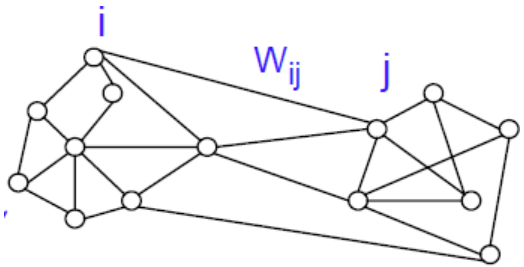
Similarity Graphs: Model local neighborhood relations between data points

E.g. Gaussian kernel similarity function

$$W_{ij} = e^{\frac{\|x_i - x_j\|^2}{2\sigma^2}}$$ → Controls size of neighborhood



Data clustering



$G = \{V,E\}$

Graph construction

- Different ways to construct a graph representing the relationships between data points :
  - Fully connected graph: All vertices having non-null similarities are connected to each other.
  - r-neighborhood graph: Each vertex is connected to vertices falling inside a ball of radius r where r is a real value that has to be tuned in order to catch the local structure of data.
  - k-nearest neighbor graph: Each vertex is connected to its k-nearest neighbors where k is an integer number which controls the local relationships of data.
  - r-neighborhood and k-nearest neighbor combined

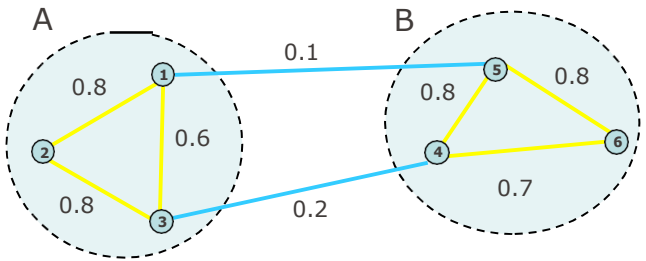
Source: Hamad & Biela

2-way Normalized Cuts

1. Compute the affinity matrix W, compute the degree matrix (D), D is diagonal and
   $D(i,i) = \sum_{jEV} W(i,j)$
2. Solve $(D - W)y = \lambda Dy$, where $D - W$ is called the Laplacian matrix
3. Use the eigenvector with the second smallest eigen-value to bipartition the graph into two parts.

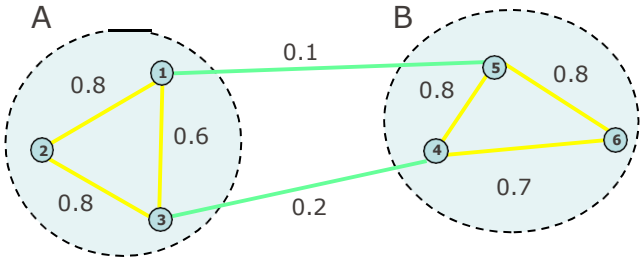
Source: D. Sonntag

Illustrative example



Source: Hamad & Biela

Graph and similarity matrix



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	0.8	0.6	0	0.1	0
x_2	0.8	0	0.8	0	0	0
x_3	0.6	0.8	0	0.2	0	0
x_4	0.8	0	0.2	0	0.8	0.7
x_5	0.1	0	0	0.8	0	0.8
x_6	0	0	0	0.7	0.8	0

Source: Hamad & Biela

Spectral bipartitioning: Illustrative example

Pre-processing

Build Laplacian matrix L of the graph

x_1	1.5	-0.8	-0.6	0	-0.1	0
x_2	-0.8	1.6	-0.8	0	0	0
x_3	-0.6	-0.8	1.6	-0.2	0	0
x_4	-0.8	0	-0.2	2.5	-0.8	-0.7
x_5	-0.1	0	0	0.8	1.7	-0.8
x_6	0	0	0	-0.7	-0.8	1.5

Decomposition : Find

- eigenvalues Λ and
- eigenvectors X of matrix L

$\Lambda =$

0.0
0.3
2.2
2.3
2.5
3.0

$X =$

0.4	0.2	0.1	0.4	-0.2	-0.9
0.4	0.2	0.1	-0.	0.4	0.3
0.4	0.2	-0.2	0.0	-0.2	0.6
0.4	-0.4	0.9	0.2	-0.4	-0.6
0.4	-0.7	-0.4	-0.8	-0.6	-0.2
0.4	-0.7	-0.2	0.5	0.8	0.9

- Map vertices to the corresponding components of 2nd eigenvector (also referred to as Fiedler vector)

x_1	0.2
x_2	0.2
x_3	0.2
x_4	-0.4
x_5	-0.7
x_6	-0.7

How do we find the clusters?

Source: Hamad & Biela

Spectral Clustering Algorithms

x_1	0.2
x_2	0.2
x_3	0.2
x_4	-0.4
x_5	-0.7
x_6	-0.7

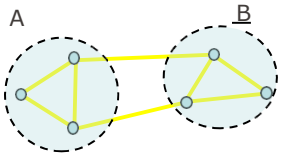
Split at value 0

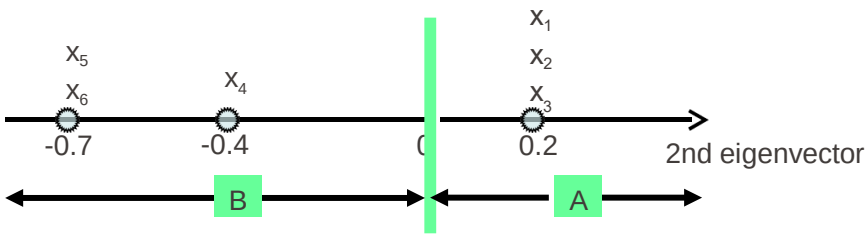
Cluster A: Positive points

Cluster B: Negative points

x_1	0.2
x_2	0.2
x_3	0.2

x_4	-0.4
x_5	-0.7
x_6	-0.7





Source: Hamad & Biela

Spectral Clustering for k > 2 Clusters

1. Construct Similarity Graph
   - Nodes = data points
   - Edges = similarity (e.g., Gaussian kernel)

2. Compute Graph Laplacian

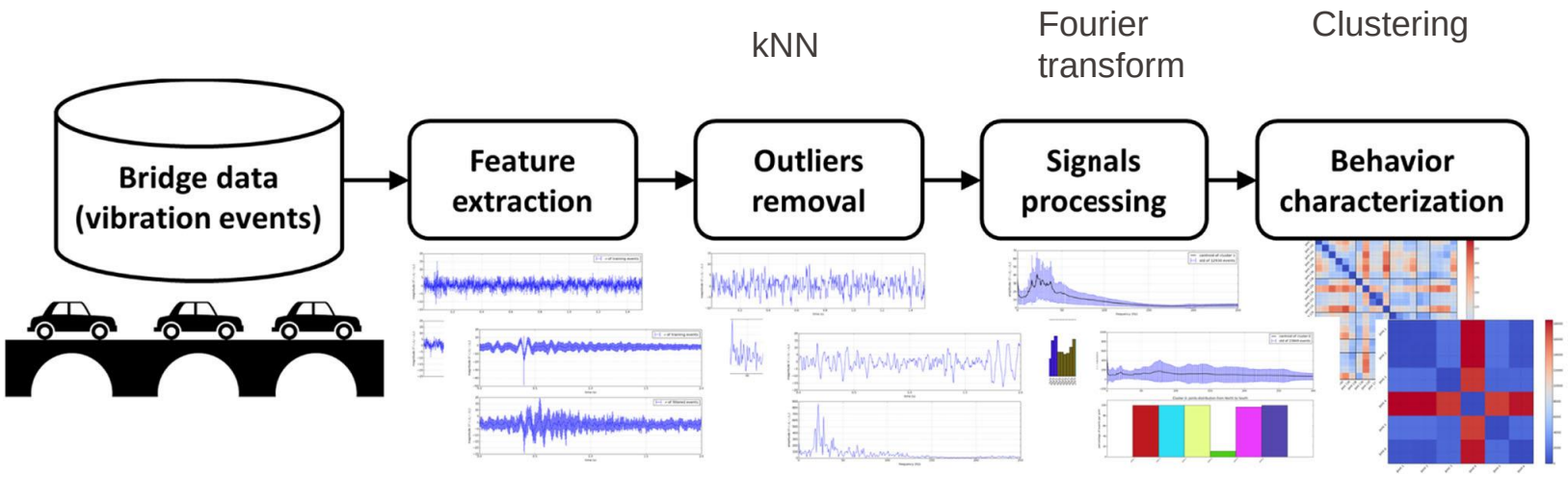
3. Compute Eigenvectors
   - Take first k eigenvectors of L
   - Each point becomes a k-dimensional vector

4. Cluster in Spectral Space
   - Apply k-means on rows of eigenvector matrix

Intuition:
- Eigenvectors capture graph structure
- Clustering in this space separates complex shapes
- Spectral clustering projects onto directions that capture graph structure

Example Clustering for Structural Health Monitoring

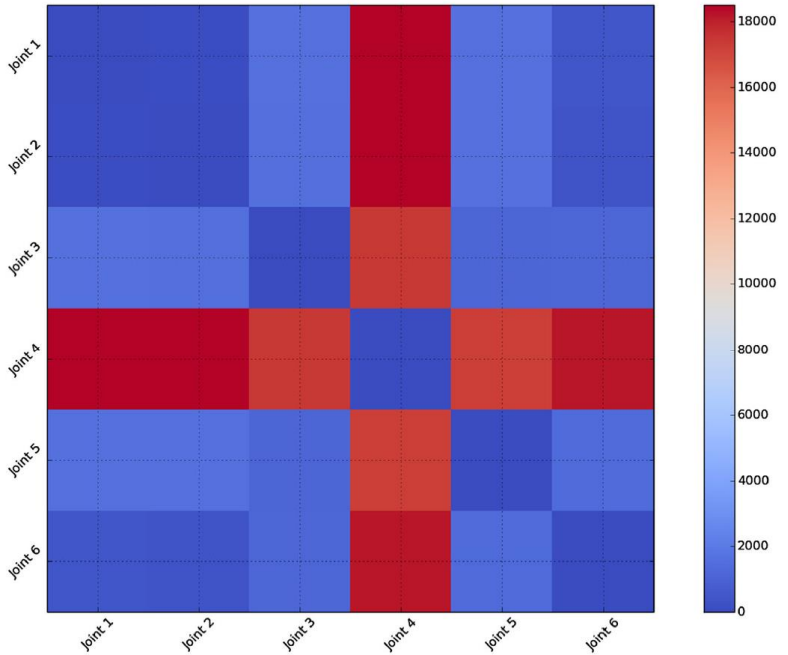
Flowchart of the proposed clustering based approach for SHM



Diez, Alberto, Nguyen Lu Dang Khoa, Mehrisadat Makki Alamdari, Yang Wang, Fang Chen, and Peter Runcie. "A clustering approach for structural health monitoring on bridges." *Journal of Civil Structural Health Monitoring* 6 (2016): 429-445.

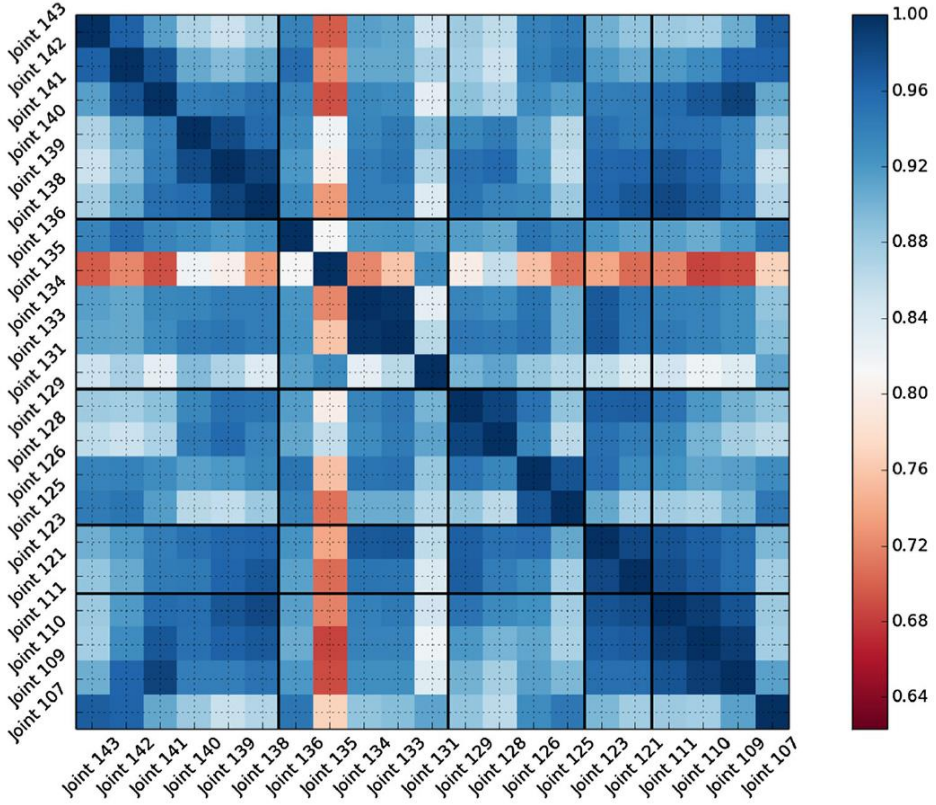
6-Joints case

- Comprises Joints 1, 2 and 3 in North main span and Joints 4, 5 and 6 in North pylon
- a map of pairwise distances among representatives of all joints was generated.
- A joint representative is calculated as the mean values of all events of each joint, after outlier removal phase.



Diez, Alberto, Nguyen Lu Dang Khoa, Mehrisadat Makki Alamdari, Yang Wang, Fang Chen, and Peter Runcie. "A clustering approach for structural health monitoring on bridges." *Journal of Civil Structural Health Monitoring* 6 (2016): 429-445.

71-Joint case, span 7: map of pairwise distances using cross correlation



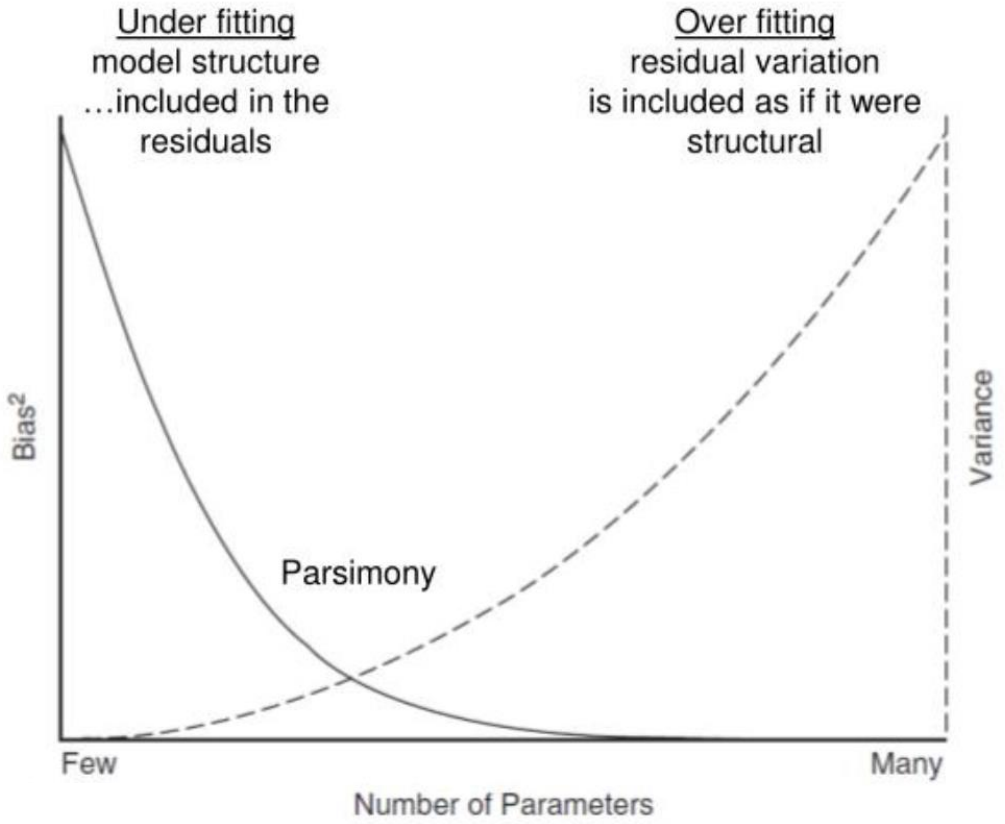
Diez, Alberto, Nguyen Lu Dang Khoa, Mehrisadat Makki Alamdari, Yang Wang, Fang Chen, and Peter Runcie. "A clustering approach for structural health monitoring on bridges." *Journal of Civil Structural Health Monitoring* 6 (2016): 429-445.

Principal component analysis (PCA)

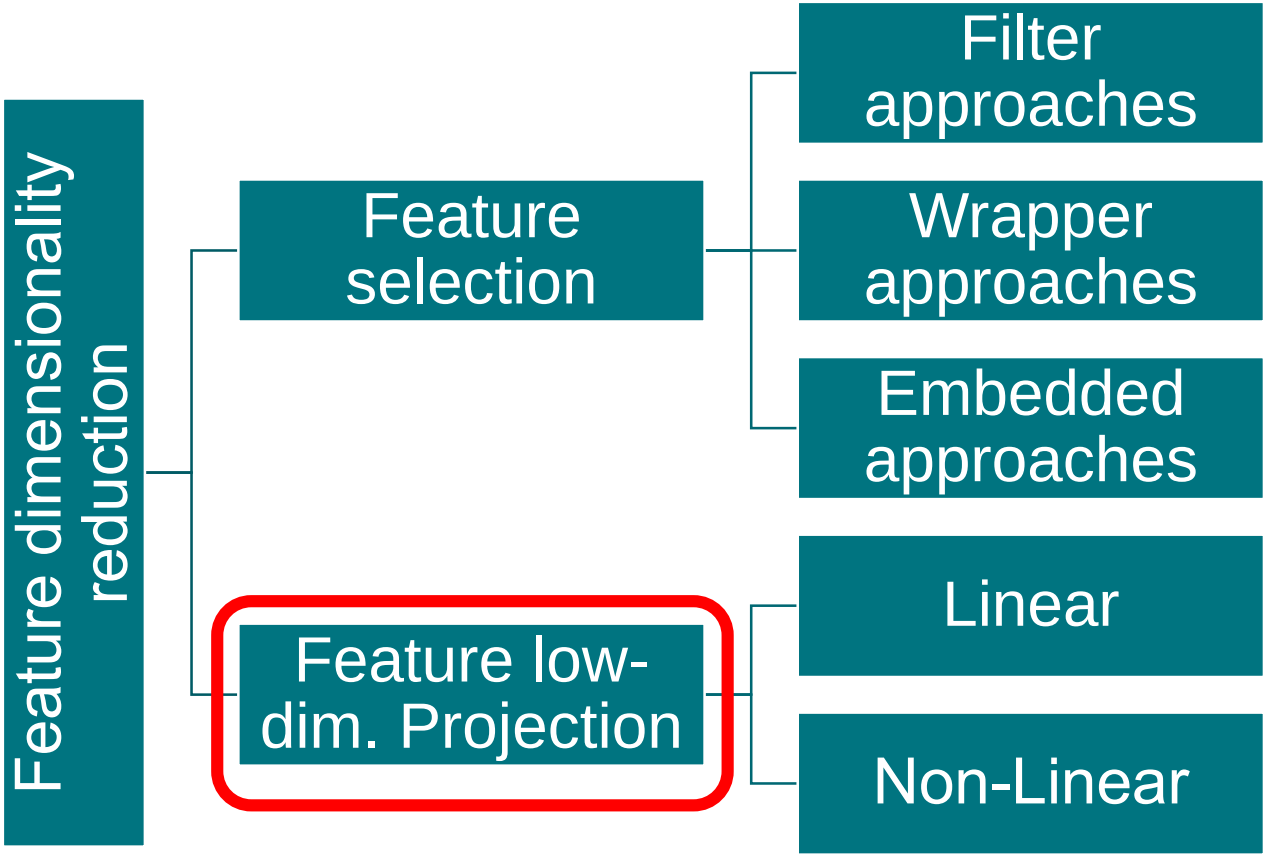
Why dimensionality reduction?

- **Handling High-Dimensional Data**: In many engineering fields, data collected from sensors, simulations, or experiments can be extremely high-dimensional. High-dimensional data can be difficult to process and analyze effectively due to the sheer volume of variables involved.
- **Avoiding the Curse of Dimensionality**: As the number of dimensions increases, the volume of the space increases exponentially, which requires exponentially more data to maintain the same level of statistical reliability. This phenomenon is known as the curse of dimensionality. Dimensionality reduction helps mitigate this issue by reducing the number of random variables under consideration, simplifying models without significant loss of information.
- **Improved Model Performance and Generalization**: Reducing the dimensionality of data can lead to improvements in the performance of predictive models. By eliminating irrelevant or redundant features, models can be made more efficient and less prone to overfitting. This means that models are better at generalizing from training data to unseen data, which is crucial for robust engineering solutions.
- **Cost and Time Efficiency**: Processing fewer dimensions requires less computational resources, thus reducing both the time and cost associated with data analysis and model training. This is particularly important in engineering applications where real-time or near-real-time processing is crucial, such as in control systems or online monitoring systems.
- **Enhanced Data Visualization**: High-dimensional data is challenging to visualize, but reducing the number of dimensions can allow for effective visualization techniques. Visualization is essential for exploring data, identifying patterns, and communicating findings clearly in many engineering disciplines.
- **Feature Extraction and Engineering Insight**: Dimensionality reduction techniques can also be used for feature extraction, where new features are derived from the original set that retain most of the important information. This can lead to insights into the underlying processes and mechanisms, which can be invaluable in fields like mechanical engineering, electrical engineering, and systems biology.
- **Data Compression**: In many applications, reducing the dimensionality of data can significantly decrease the storage space required, which is beneficial for systems with limited memory or for transmitting data over bandwidth-limited channels.

Underfitting vs overfitting

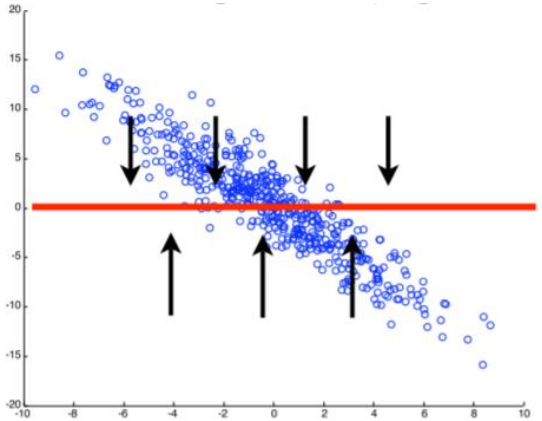


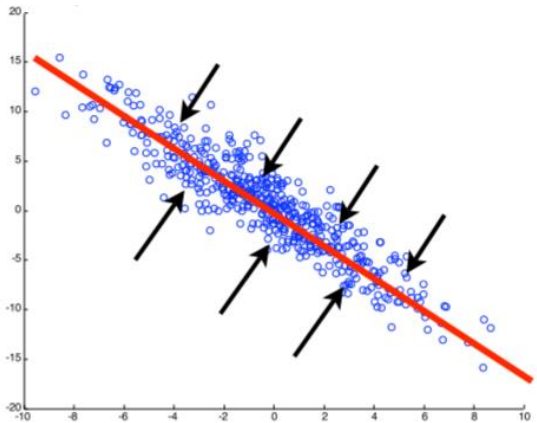
Different approaches to dimensionality reduction



Feature selection vs. Feature low-dim. Projection

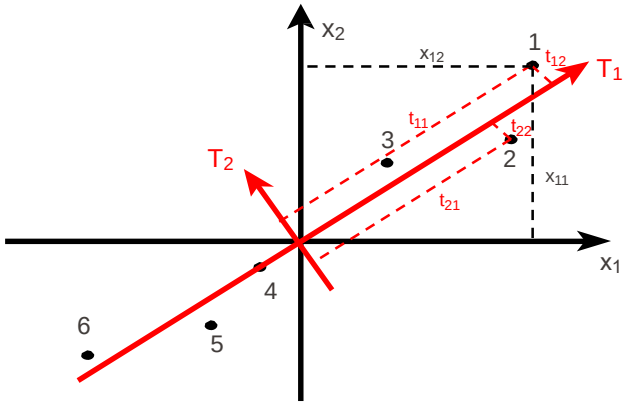
- Removing features → Equivalent to projecting data onto lower-dimensional linear subspace perpendicular to the feature removed





Principal component analysis

- PCA seeks to preserve as much of the randomness (variance) in the high-dimensional space as possible
- Projection of the dataset onto a lower dimensional space
- In the direction of maximum variance



Key concepts of PCA

- Dimensionality Reduction: PCA reduces the number of variables (features) in a dataset while preserving as much variability (information) as possible.
- Principal Components: New uncorrelated variables that are **linear combinations of the original variables**. **Each principal component captures a portion of the total variance in the data**.
- Variance Maximization: The first principal component captures the maximum variance, the second captures the next highest variance orthogonal to the first, and so on.
- Orthogonality: Principal components are orthogonal (uncorrelated) to each other, which eliminates redundancy.

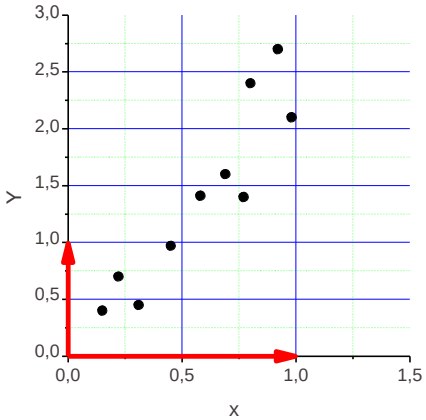
PCA main ideas

- Splitting a measured signal (i.e. usually a mixture of many different signal components) into a set of underlying variables (into a new base)
- Identification of some (few) main source signals from a large number of mixed signals
- Dimensionality reduction
- The goal of a PCA is to express a noisy data set in a new base, expecting this new representation to filter out the noise and reveal hidden structures in the data.
- What are the important dynamics underlying a data set and which dynamics are of lesser importance? How can these be filtered out using a PCA?

Uses of PCA

- Pre-processing method before classification to reduce computational costs of the classifier.
- Compression method for ease of data storage and retrieval → also used for signal reconstruction
- Feature extraction method
- Noise Reduction
- Visualization
- Detection of Changes in Variance (→ fault or anomaly detection)

Steps to perform a PCA



Problem:
How to find the unit vector with maximum variance ?

Solution → eigenvalue problem :
Calculate the largest eigenvalue λ and the corresponding eigenvector x to the covariance matrix C

$$Cx = \lambda x$$

$Cx = \lambda x$ is equivalent to $(C - \lambda E_n)x = 0$

E_n = unit matrix

$$\det(C - \lambda E_n)x = 0$$

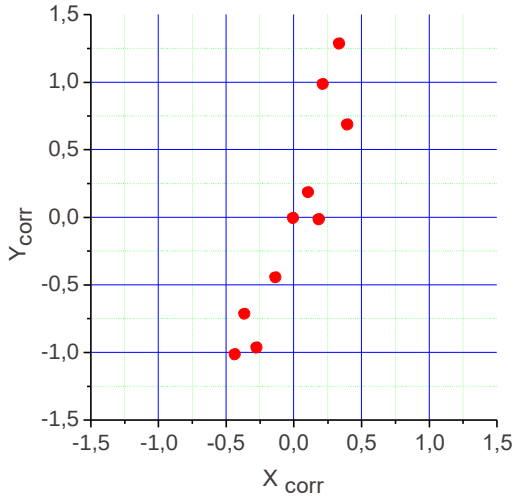
Step 1: Subtraction of mean values

Subtraction of mean values X_μ , Y_μ
(for each of the dimensions)

→ Creates a data set whose mean values in all dimensions are zero

$$\frac{1}{n}\sum_{i=1}^{n}(X_i - X_\mu) = \frac{1}{n}\sum_{i=1}^{n}X_{i,corr} = 0$$

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_\mu) = \frac{1}{n}\sum_{i=1}^{n}Y_{i,corr} = 0$$



In the example:

X_μ = 0.587
Y_μ = 1.413

Step 2: Calculation of the covariance matrix

Calculation of the covariance matrix (of the mean-corrected data)

n-dimensional dataset → Covariance matrix has n rows and n columns;

→ quadratic matrix

→ main diagonal = variance

→ symmetrical matrix:

cov(x,y) = cov(y,x)

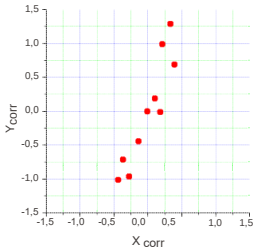
$$\text{var}(X,X) = \frac{\sum_{i=1}^{n}(X_i - X_\mu)(X_i - X_\mu)}{n-1} \qquad \text{cov}(X,Y) = \frac{\sum_{i=1}^{n}(X_i - X_\mu)(Y_i - Y_\mu)}{n-1}$$

For *n* = 2:

$$C = \begin{pmatrix} var(x,x) & cov(x,y) \\ cov(y,x) & var(y,y) \end{pmatrix}$$

In the example

$$C = \begin{pmatrix} 0.0862 & 0.2175 \\ 0.2175 & 0.6440 \end{pmatrix}$$



Step 3: Eigenvalues and Eigenvectors of the Covariance Matrix

Example: Eigenvalues of the Covariance Matrix

$$\det(C - \lambda E) = 0 \quad \Longrightarrow \quad \begin{vmatrix} 0.0862 - \lambda & 0.2175 \\ 0.2175 & 0.6440 - \lambda \end{vmatrix} = 0$$

example: $\lambda_1 = 0.7188; \quad \lambda_2 = 0.0114$

λ_1 and λ_2 are the two eigenvalues of the covariance matrix.

Step 3: Eigenvalues and Eigenvectors of the Covariance Matrix

The eigenvectors belonging to the two eigenvalues result from the solution of $(C - \lambda E_n)x = 0$

$$\left[\begin{pmatrix} \mathrm{var}(x,x) & \mathrm{cov}(x,y) \\ \mathrm{cov}(y,x) & \mathrm{var}(y,y) \end{pmatrix} - \lambda_{1,2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right] \bullet x = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

In the example:

$$\left[\begin{pmatrix} 0.0862 & 0.2175 \\ 0.2175 & 0.6440 \end{pmatrix} - \lambda_{1,2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right] \bullet x = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Eigenvector for $\lambda_1 = 0.7188$: $x_1 = (0.3251, 0.9456); \quad |x_1| = 1$

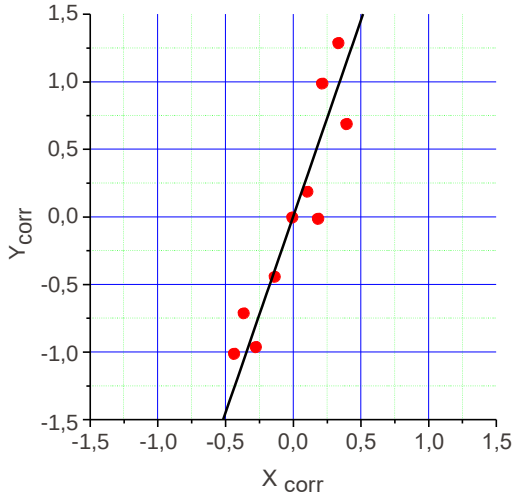
Eigenvector for $\lambda_2 = 0.0114$: $x_2 = (-0.9456, 0.3251); |x_2| = 1$

Step 3: Eigenvalues and Eigenvectors of the Covariance Matrix

dataset with the eigenvector for

$\lambda_1 = 0.7188$:

$x_1 = (0.3251, 0.9456)$



Step 3: Eigenvalues and Eigenvectors of the Covariance Matrix

dataset with the eigenvector for

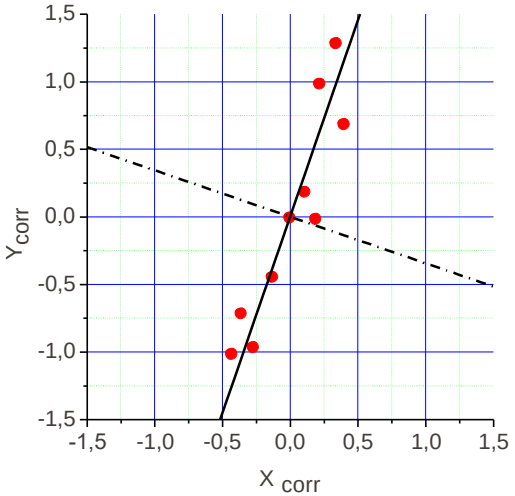
$\lambda_1 = 0.7188$:

$x_1 = (0.3251, 0.9456)$

and for:

$\lambda_2 = 0.0114$:

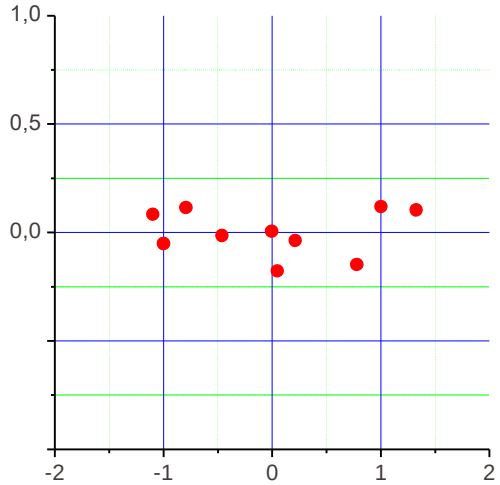
$x_2 = (-0.9456, 0.3251)$



Step 4: transformation of the new data set

Data transformation:

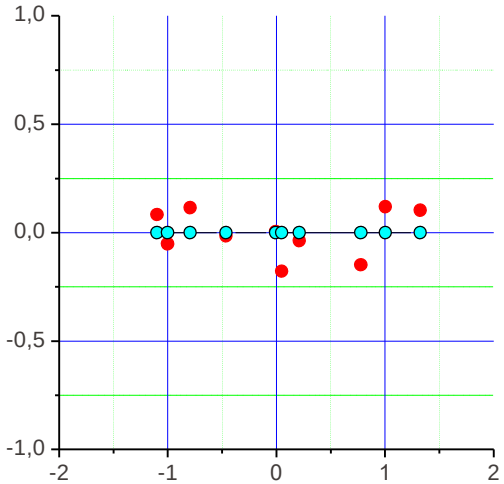
$$Y = XP_{\lambda}$$



Step 4: transformation of the new data set

New Dataset Y with

- One PC
- Two PCs



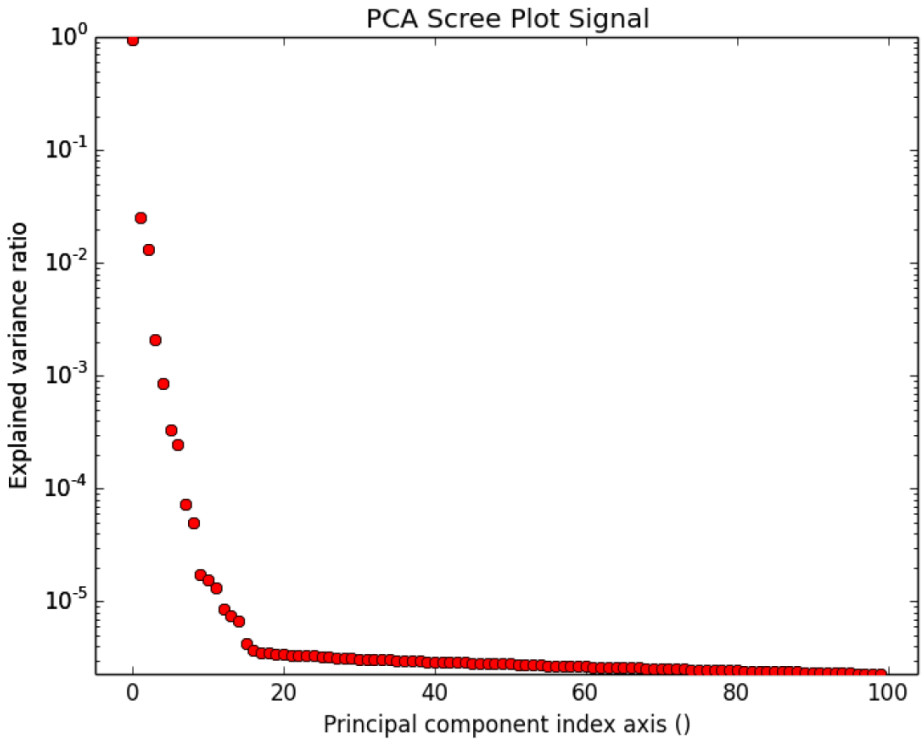
How PCA works

- Standardization:
  - Purpose: Ensures that each feature contributes equally to the analysis.
  - Method: Subtract the mean and divide by the standard deviation for each feature.
- Covariance Matrix Computation:
  - Purpose: Measures how variables change together.
  - Method: Calculate the covariance matrix of the standardized data.
- Eigenvalue and Eigenvector Calculation:
  - Eigenvalues: Indicate the amount of variance captured by each principal component.
  - Eigenvectors: Define the direction of the principal components.
- Selecting Principal Components:
  - Criteria: Choose components with the highest eigenvalues.
  - Methods:
    - Scree Plot: Visualize the eigenvalues to determine the "elbow point.“
    - Explained Variance Ratio: Select components that cumulatively explain a desired amount of total variance (e.g., 95%).
- Transforming the Data:
  - Projection: Multiply the original data by the selected eigenvectors to obtain the principal components.
  - Result: A reduced dataset with uncorrelated features.

Summary of the most important assumptions and limitations of a PCA

- Linearity of the problem: data set is a linear combination of a certain base → Problem solution by means of linear algebra (see also 4.)
- PCA uses the eigenvectors of the covariance matrix and finds only independent base vectors assuming a Gaussian probability distribution.
- Assumption that large variances reflect important dynamics. PCA essentially only performs a rotation of the coordinate system in the direction of maximum variance.
- Large variance principal components represent interesting dynamics; small variance components represent noise.
- Role of SNR (Signal to Noise Ratio)
- Main components are orthogonal → Simplification that makes PCA solvable by means of linear algebra.

Scree test for determining the number of PCs



Source: Hyperspy 2011

Importance of Q and T^2 Statistics in PCA

- In **Principal Component Analysis (PCA)**, the **Q-statistic** (also known as the **Squared Prediction Error (SPE)**) and **Hotelling's T^2 statistic** are essential metrics used for assessing the fit of observations within the PCA model. They play a crucial role in:
  - **Outlier Detection**
  - **Process Monitoring**
  - **Fault Detection**
  - **Quality Control**
- These statistics help identify observations that deviate significantly from the modeled behavior, enabling analysts to take corrective actions or investigate anomalies.

Calculation of the Q and T²-Statistics

$$T_i^2 = \sum_{j=1}^{r} \frac{t_{sij}^2}{\lambda_j} = \boldsymbol{t}_{\mathrm{si}} \Lambda^{-1} \boldsymbol{t}_{\mathrm{si}}^{\mathrm{T}} = \boldsymbol{x}_i \mathbf{P} \Lambda^{-1} \mathbf{P}^{\mathrm{T}} \boldsymbol{x}_i^{\mathrm{T}}$$

$\boldsymbol{t}_{si} = \boldsymbol{x}_i \mathbf{P}$ Projection of the sample x_i on the principal component

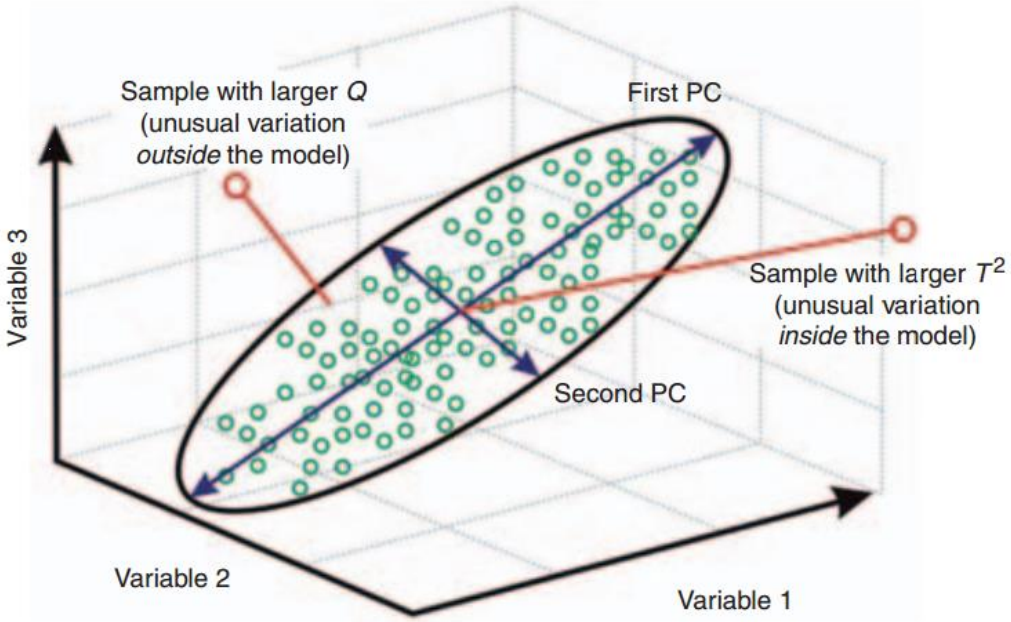
$\boldsymbol{x}_i$ Sample vector representing all the measurements at the point i

$$Q_i = \tilde{\boldsymbol{x}}_i \tilde{\boldsymbol{x}}_i^{\mathrm{T}} = \boldsymbol{x}_i \left(\mathbf{I} - \mathbf{P}\mathbf{P}^{\mathrm{T}} \right) x_i^{\mathrm{T}}$$

$\tilde{\boldsymbol{x}}_i$ Projection of the sample x_i on the residuals

- Hotelling's T² statistic measures the variation of an observation within the principal component (score) space defined by the PCA model.
- It quantifies how far an observation's scores are from the center (mean) of the model, considering the variability captured by the selected principal components.
- The Q-statistic measures the residual variation of an observation not explained by the PCA model.
- It quantifies the distance between the original observation and its reconstruction from the retained principal components.

Q and T^2-Statistic

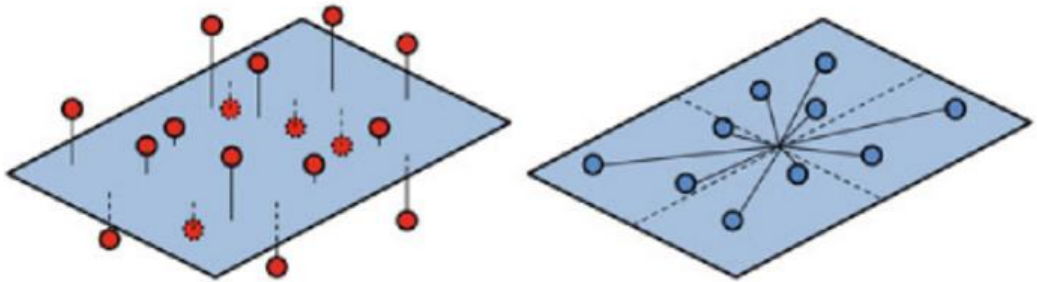


Mujica, L. E., J. Rodellar, A. Fernandez, and A. Güemes. "Q-statistic and T2-statistic PCA-based measures for damage assessment in structures." *Structural Health Monitoring* 10, no. 5 (2011): 539-553.

Q and T^2-Statistic to analyse the samples

Q-Statistic

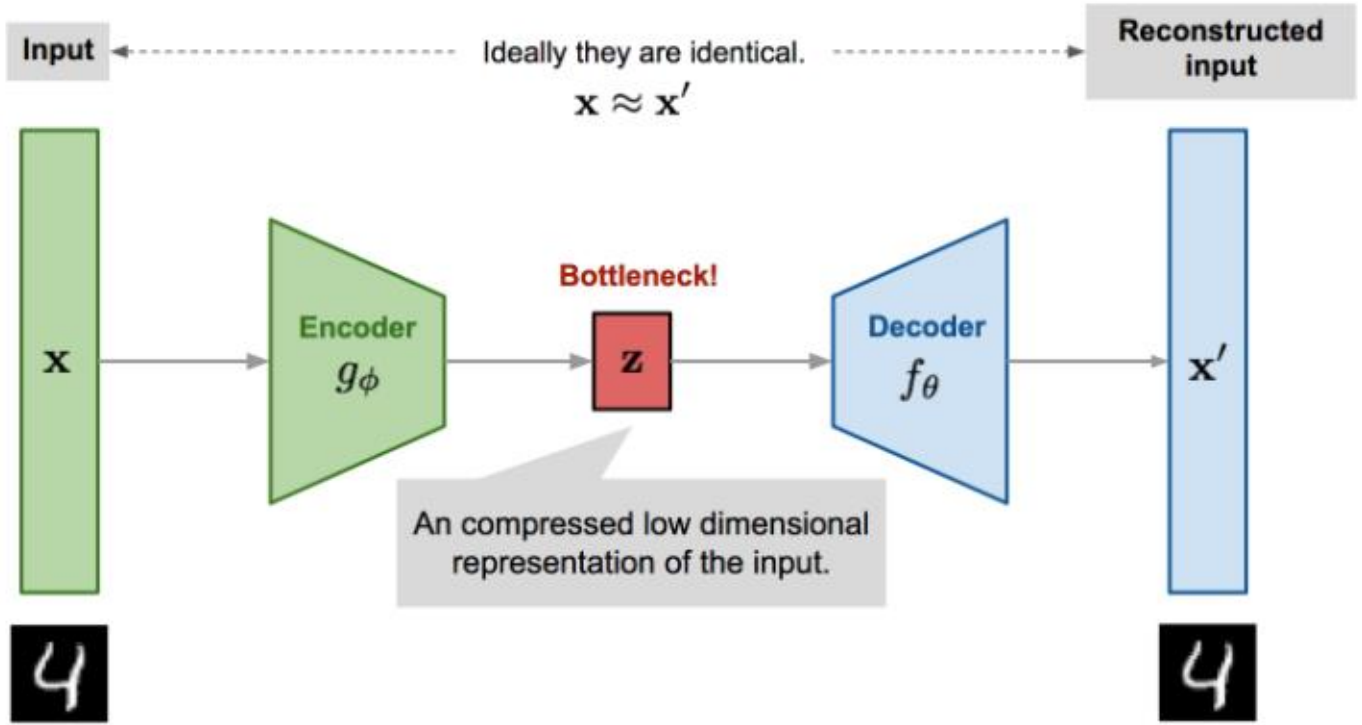
T^2-Statistic



Other dimensionality reduction approaches

- Nonlinear PCA (NLPCA)
- Kernel PCA
- Exploratory Projection Pursuit (EPP)
  - EPP seeks an M-dimensional (M=2,3 typically) linear projection of the data that maximizes a measure of «interestingness»
  - Interestingness is measured as departure from multivariate normality
    - This measure is not the variance and is commonly scale-free. In most implementations, it is also affine invariant, so it does not depend on the correlations between features
- Kernel LDA
- T-distributed Stochastic Neighbor Embedding (t-SNE) (for visualization purposes)

Autoencoder (will be considered later)



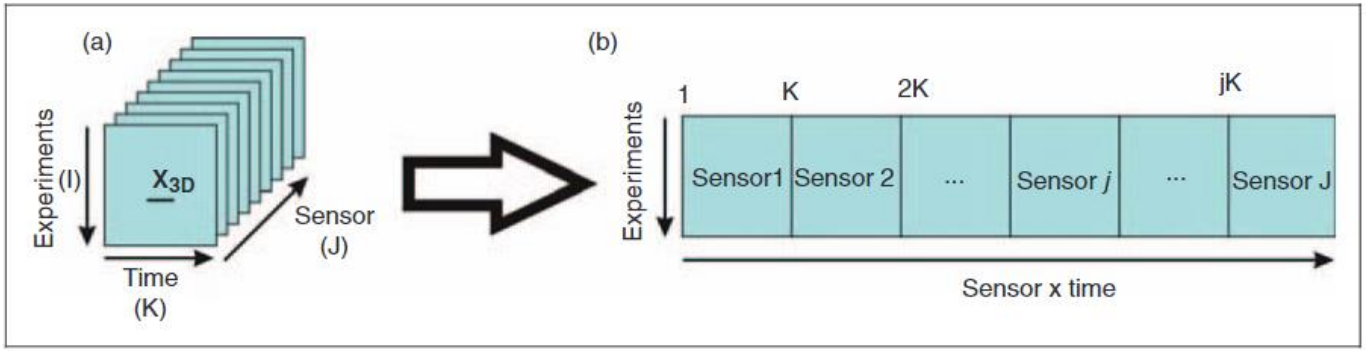
Example Principal component analysis (PCA)

Damage assessment in structures

- two structures are used for experimental assessment:
  - a steel sheet and
  - a turbine blade of an aircraft.
- The analysis has been performed in two ways:
  - (i) by exciting the structure with low-frequency vibrations using a shaker and using several piezoelectric (PZT) sensors attached on the surface
  - (ii) by exciting at high-frequency vibrations using a single PZT as actuator and several PZTs as sensors.
- A known vibration signal is applied and the dynamical responses are analyzed

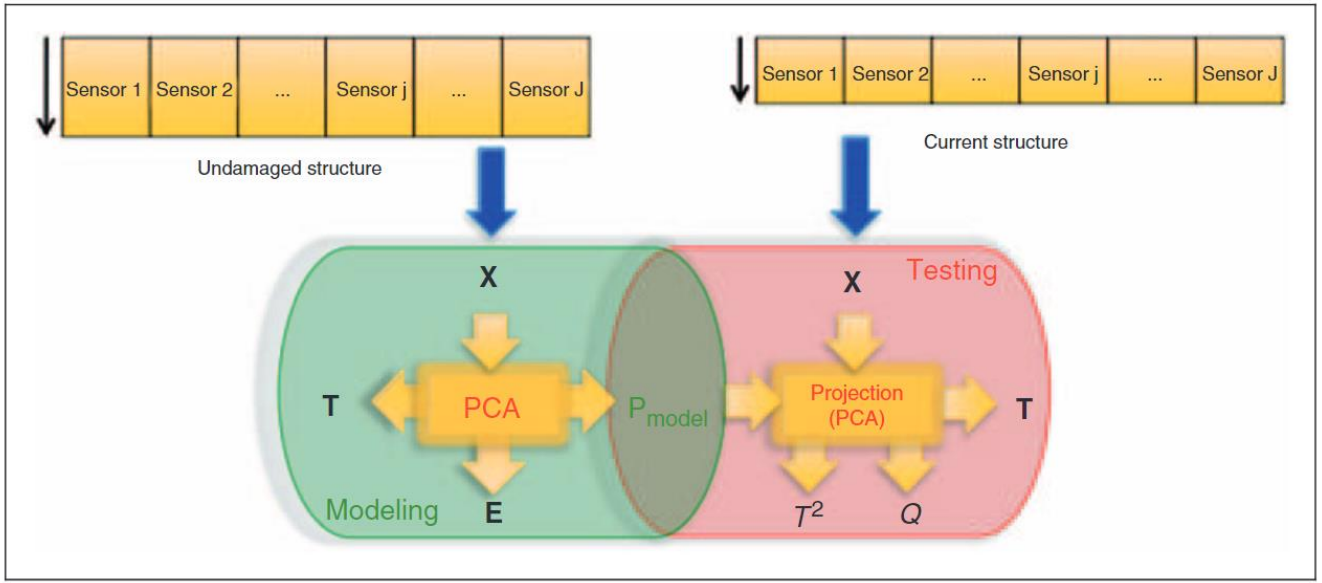
Mujica, L. E., J. Rodellar, A. Fernandez, and A. Güemes. "Q-statistic and T2-statistic PCA-based measures for damage assessment in structures." *Structural Health Monitoring* 10, no. 5 (2011): 539-553.

Collected data



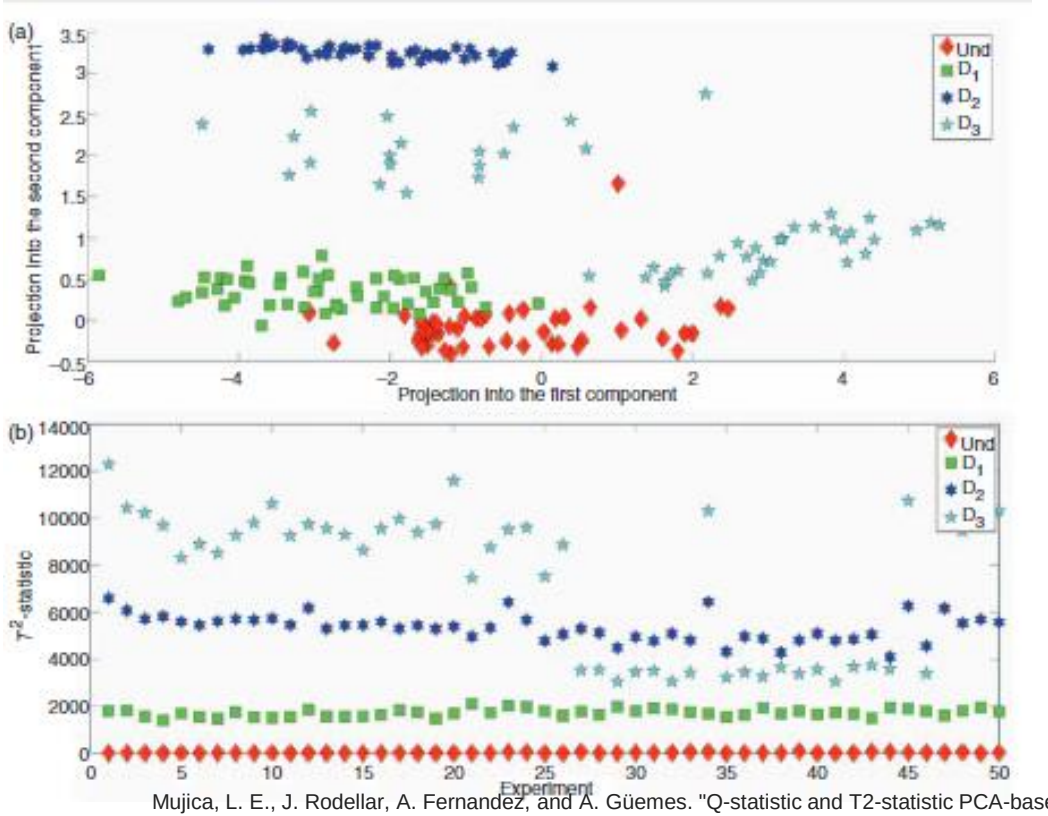
Mujica, L. E., J. Rodellar, A. Fernandez, and A. Güemes. "Q-statistic and T2-statistic PCA-based measures for damage assessment in structures." *Structural Health Monitoring* 10, no. 5 (2011): 539-553.

Proposed procedure



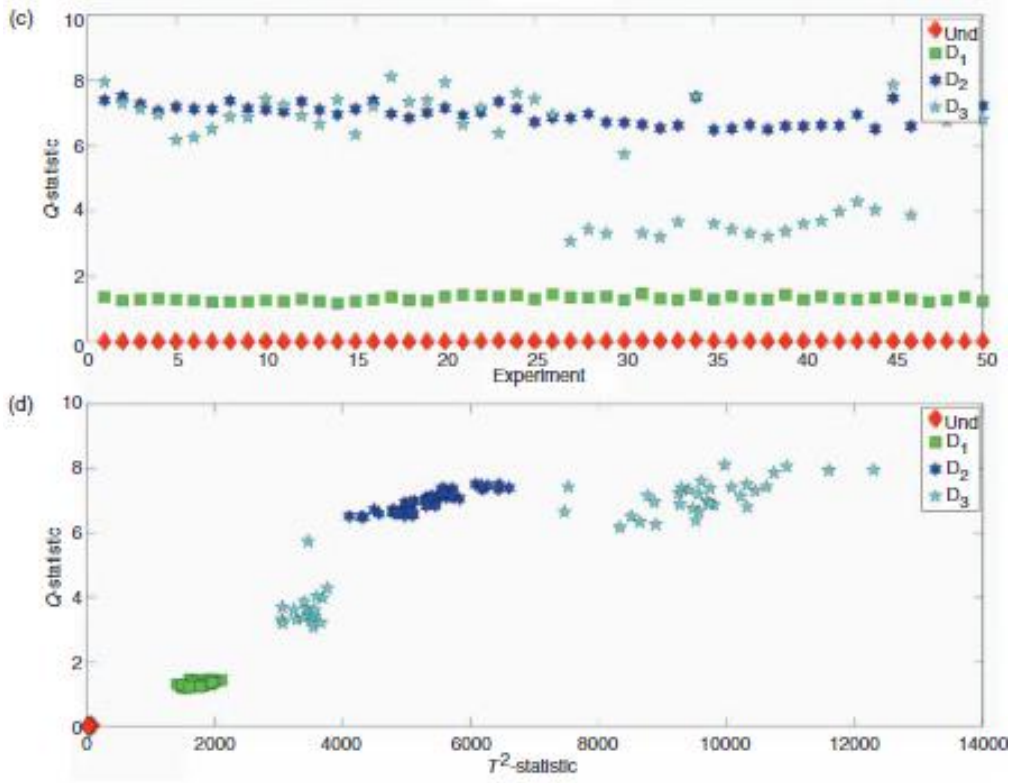
Mujica, L. E., J. Rodellar, A. Fernandez, and A. Güemes. "Q-statistic and T2-statistic PCA-based measures for damage assessment in structures." *Structural Health Monitoring* 10, no. 5 (2011): 539-553.

Results: Steel sheet low frequency



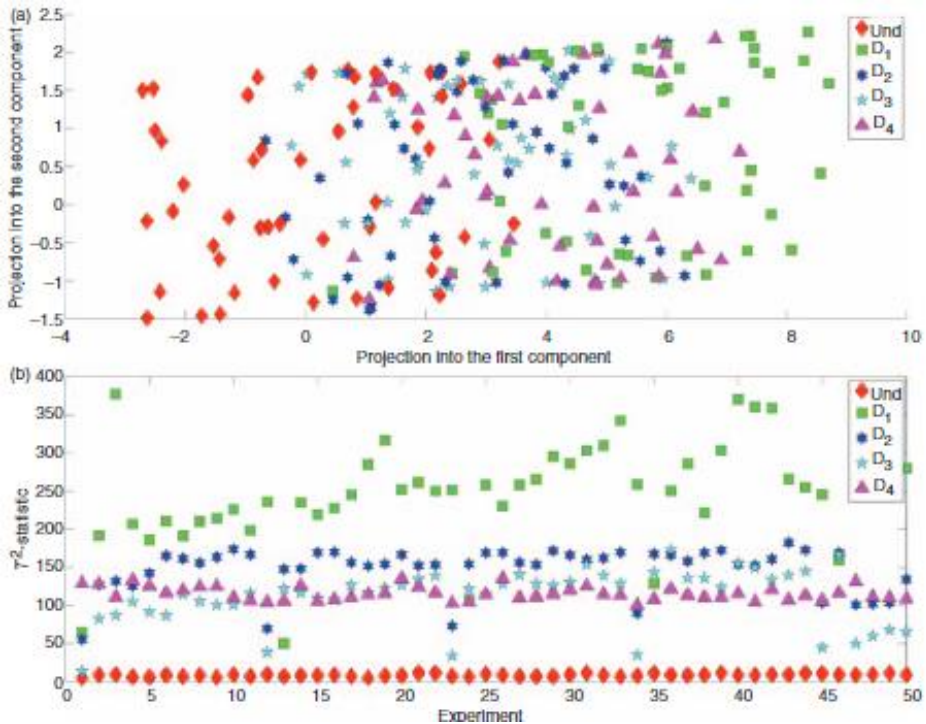
Mujica, L. E., J. Rodellar, A. Fernandez, and A. Güemes. "Q-statistic and T2-statistic PCA-based measures for damage assessment in structures." *Structural Health Monitoring* 10, no. 5 (2011): 539-553.

Results: Steel sheet low frequency



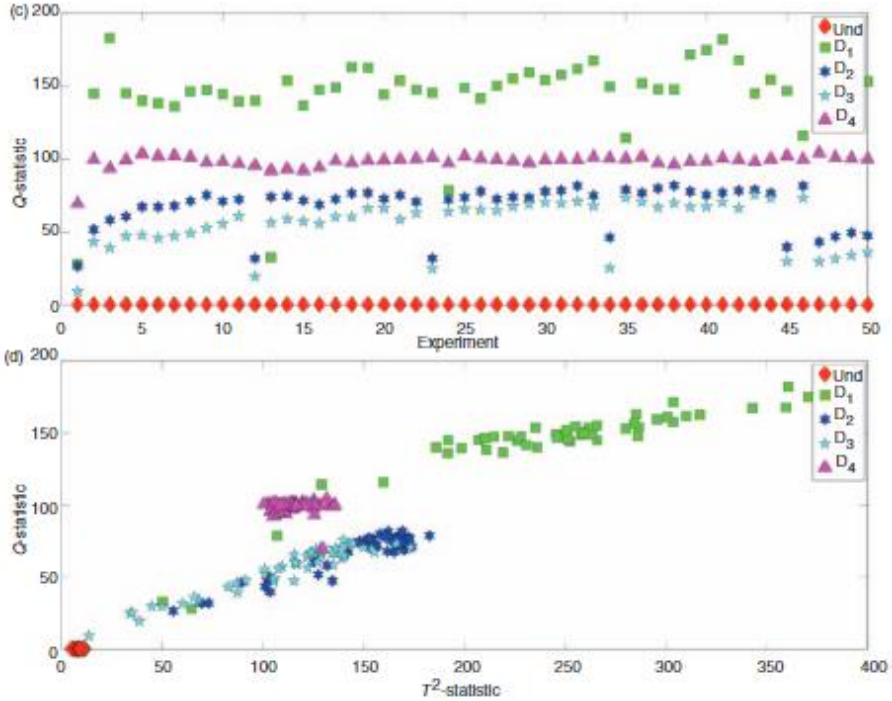
Mujica, L. E., J. Rodellar, A. Fernandez, and A. Güemes. "Q-statistic and T2-statistic PCA-based measures for damage assessment in structures." *Structural Health Monitoring* 10, no. 5 (2011): 539-553.

Results: Blade low frequency



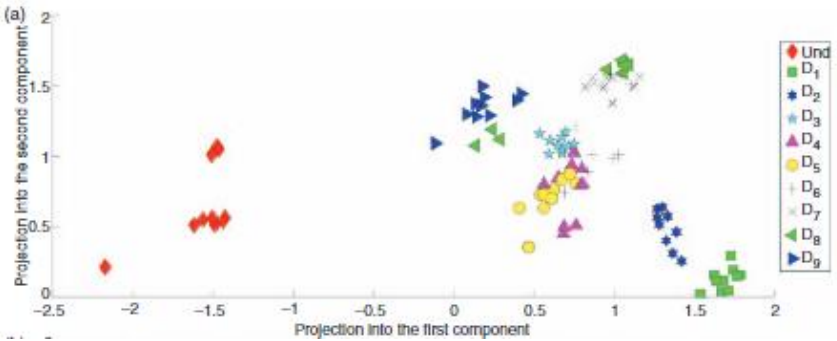
Mujica, L. E., J. Rodellar, A. Fernandez, and A. Güemes. "Q-statistic and T2-statistic PCA-based measures for damage assessment in structures." *Structural Health Monitoring* 10, no. 5 (2011): 539-553.

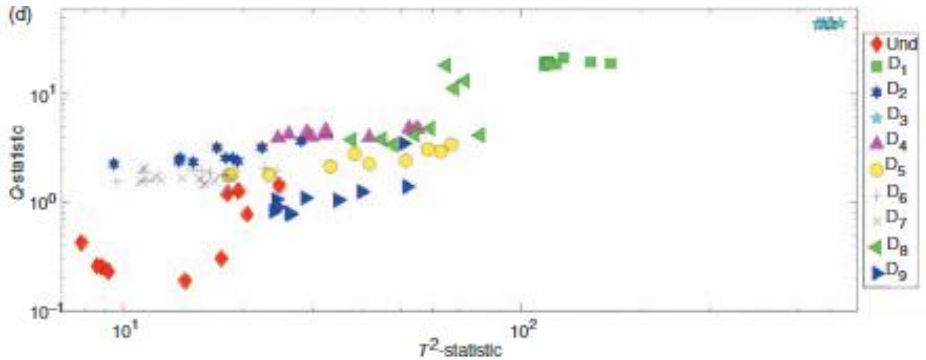
Results: Blade low frequency



Mujica, L. E., J. Rodellar, A. Fernandez, and A. Güemes. "Q-statistic and T2-statistic PCA-based measures for damage assessment in structures." *Structural Health Monitoring* 10, no. 5 (2011): 539-553.

Results: Blade high frequency

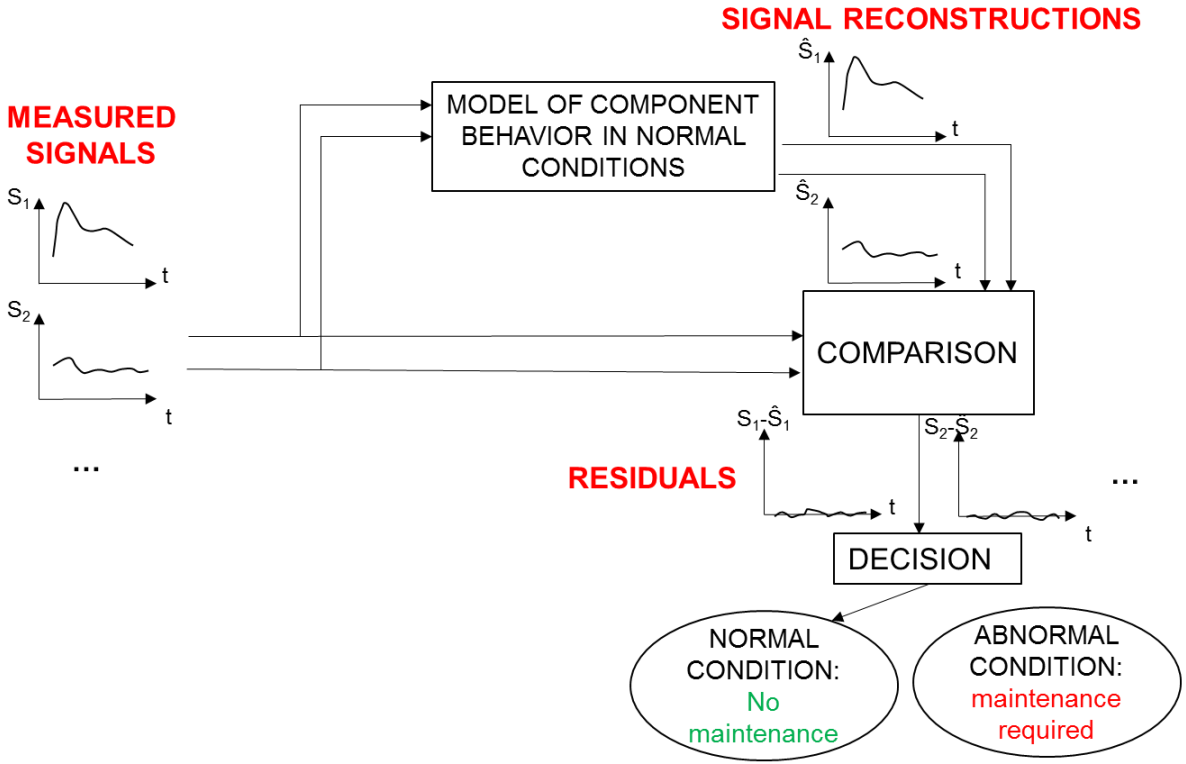




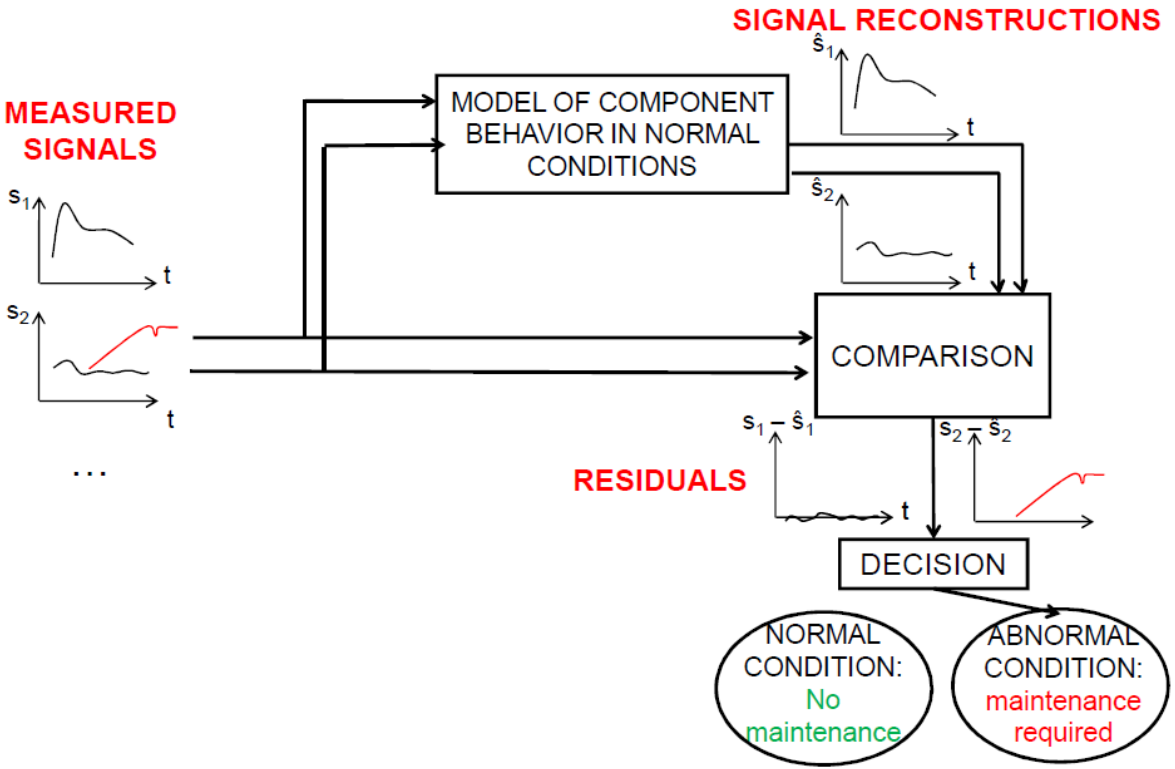
Mujica, L. E., J. Rodellar, A. Fernandez, and A. Güemes. "Q-statistic and T2-statistic PCA-based measures for damage assessment in structures." *Structural Health Monitoring* 10, no. 5 (2011): 539-553.

Fault detection with signal reconstruction

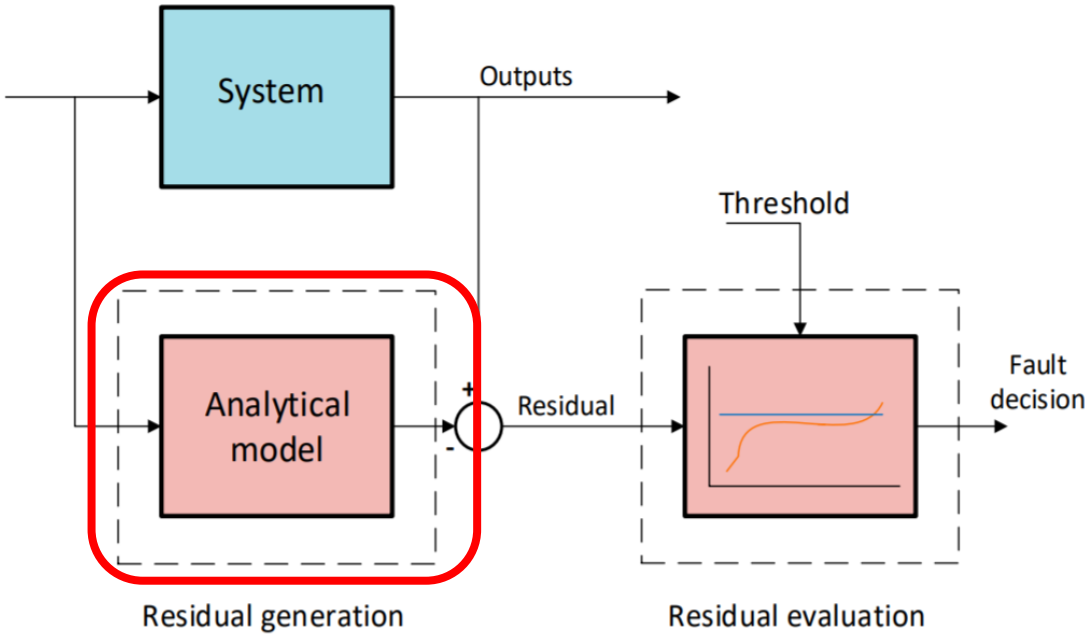
Fault detection with residual-based or reconstruction-based approaches



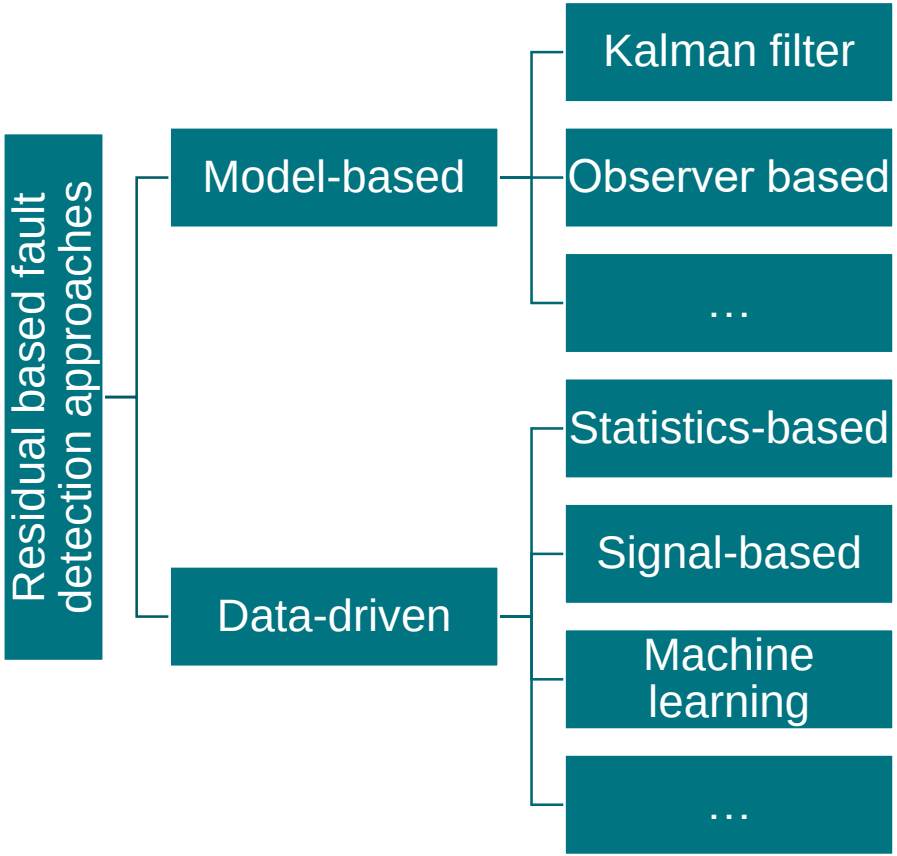
Fault detection with residual-based or reconstruction-based approaches



Basic idea of residual based methods

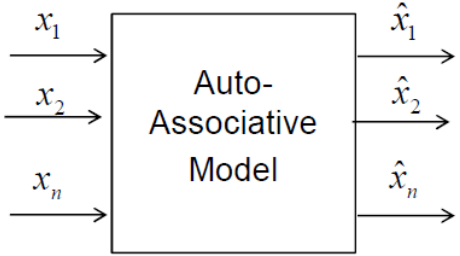


Different approaches to residual based methods



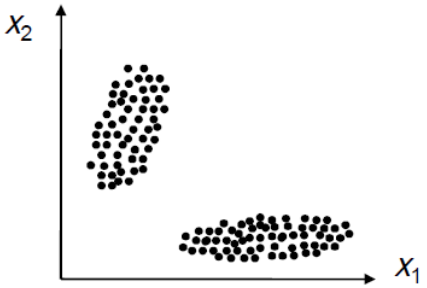
Auto-associative models

Auto-associative model



$$\hat{x}_i = f(x_1, x_2, \ldots, x_n)$$
$$\forall i = 1, \ldots, n$$

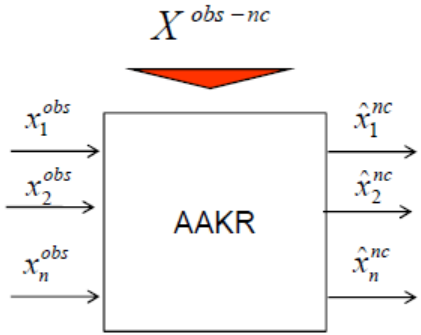
Empirical model built using training patterns = historical signal measurements in **normal system condition**



$$X^{obs-nc} = \begin{bmatrix} x_{11}^{obs-nc} & \cdots & x_{1j} & \cdots & x_{1n}^{obs-nc} \\ \cdots & & \cdots & \cdots & \cdots \\ x_{k1} & \cdots & x_{kj} & \cdots & x_{kn} \\ \cdots & & \cdots & \cdots & \cdots \\ x_{N1}^{obs-nc} & \cdots & x_{Nj} & \cdots & x_{Nn}^{obs-nc} \end{bmatrix}$$

Basic principles of auto-associative models

- Training pattern $X^{obs-nc} = \begin{bmatrix} x_{11}^{obs-nc} & \dots & x_{1j} & \dots & x_{1n}^{obs-nc} \\ \dots & \dots & \dots & \dots & \dots \\ x_{k1} & \dots & x_{kj} & \dots & x_{kn} \\ \dots & \dots & \dots & \dots & \dots \\ x_{N1}^{obs-nc} & \dots & x_{Nj} & \dots & x_{Nn}^{obs-nc} \end{bmatrix}$ = historical signal measurements in normal system condition
- Test pattern: input $\vec{x}^{obs} = (x_1^{obs}, \dots, x_n^{obs})$ = measured signals at current time
- ★ Test pattern: output $\vec{\hat{x}}^{nc} = (\hat{x}_1^{nc}, \dots, \hat{x}_n^{nc})$ = signal reconstructions (expected values of the signals in normal condition)

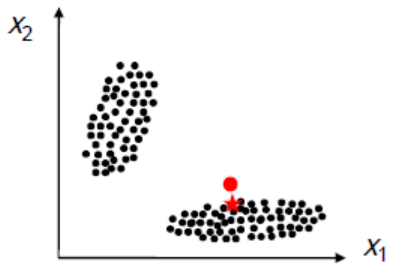


Example: normal condition

● Training pattern $X^{obs-nc} = \begin{bmatrix} x_{11}^{obs-nc} & \dots & x_{1j} & \dots & x_{1n}^{obs-nc} \\ \dots & \dots & \dots & \dots & \dots \\ x_{k1} & \dots & x_{kj} & \dots & x_{kn} \\ \dots & \dots & \dots & \dots & \dots \\ x_{N1}^{obs-nc} & \dots & x_{Nj} & \dots & x_{Nn}^{obs-nc} \end{bmatrix}$ = historical signal measurements in normal system condition

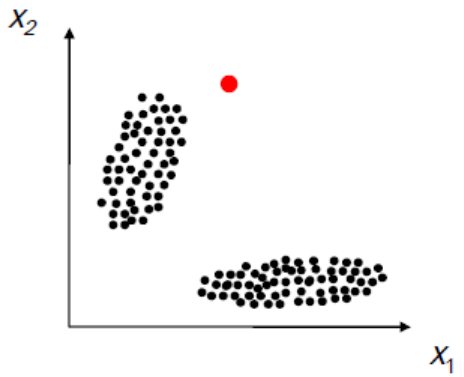
● Test pattern: input $\vec{x}^{obs} = (x_1^{obs}, \dots, x_n^{obs})$ = measured signals at current time

★ Test pattern: output $\vec{\hat{x}}^{nc} = (\hat{x}_1^{nc}, \dots, \hat{x}_n^{nc})$ = weighted sum of the training patterns:



Example: abnormal condition (1/2)

- Signal values at current time: $\vec{x}^{obs} = (x_1^{obs}, \ldots, x_n^{obs})$
- Signal reconstructions?
- Normal or abnormal condition?

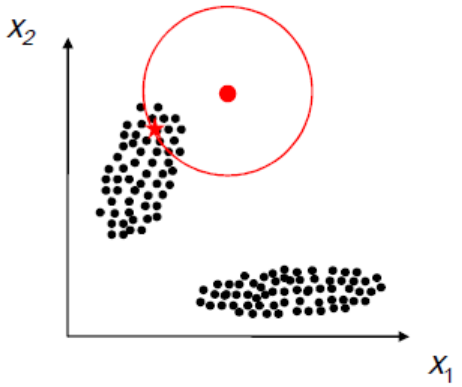


- available historical signal measurements in normal system condition

Example: abnormal condition (2/2)

•Signal values at current time: $\vec{x}^{obs} = (x_1^{obs}, \ldots, x_n^{obs})$ ●

•Signal reconstructions: $\hat{\vec{x}}^{nc} = (\hat{x}_1^{nc}, \ldots, \hat{x}_n^{nc})$ ★ based on the available historical signal measurements in normal system condition ●



$\vec{x}^{obs} \neq \hat{\vec{x}}^{nc}$

abnormal condition

•available historical signal measurements in normal plant condition

Model is the data itself: Auto-associative kernel regression for fault detection

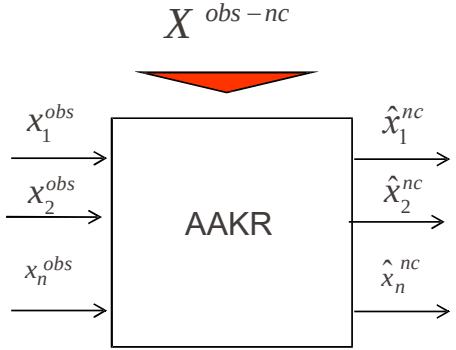
- Training pattern

$$X^{obs-nc} = \begin{bmatrix} x_{11}^{obs-nc} & \dots & x_{1j} & \dots & x_{1n}^{obs-nc} \\ \dots & \dots & \dots & \dots & \dots \\ x_{k1} & \dots & x_{kj} & \dots & x_{kn} \\ \dots & \dots & \dots & \dots & \dots \\ x_{N1}^{obs-nc} & \dots & x_{Nj} & \dots & x_{Nn}^{obs-nc} \end{bmatrix} =$$

historical signal measurements in normal system condition

- Test pattern: input $x^{obs} = (x_1^{obs}, \dots, x_n^{obs})$ = measured signals at current time

- ★ Test pattern: output $\hat{x}^{nc} = (x_1^{nc}, \dots, x_n^{nc})$ = signal reconstructions (expected values of the signals in normal condition)



Auto-Associative Kernel Regression

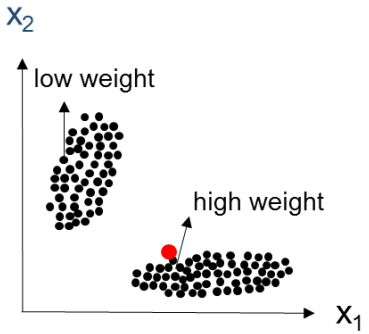
Output $\vec{\bar{X}}^{nc} = (\bar{X}_1^{nc},...,\bar{X}_n^{nc})$ = weighted sum of the training patterns:

On all the training pattern

$$\bar{X}_j^{nc} = \frac{\sum_{k=1}^{N} w(k) \bullet x_{kj}^{obs-nc}}{\sum_{k=1}^{N} w(k)}$$

weights $w(k)$ = similarity measures between $\vec{x}^{obs}$ and $\vec{x}_k^{obs-nc}$ (the test and the k-th training pattern):

$$w(k) = \frac{1}{\sqrt{2\pi}h} e^{-\frac{d^2(k)}{2h^2}}$$



with $d^2(k) = \sum_{j=1}^{n} (x_1^{obs} - x_{kj}^{obs-nc})^2$ Euclidean distance between $\vec{x}^{obs}$ and $\vec{x}_k^{obs-nc}$

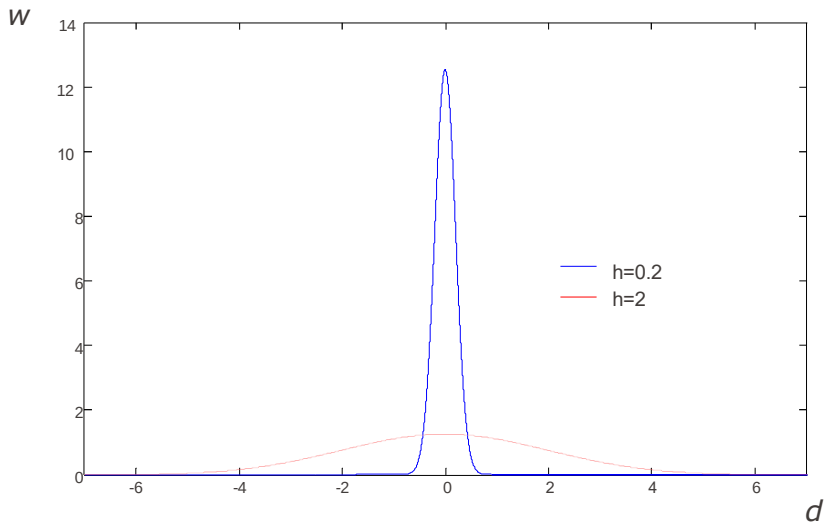
h = bandwidth parameter

Bandwidth parameter

d= 0 → w=0.4/h
d=h → w=0.24/h

$$\frac{w(d=h)}{w(d=3h)} = \frac{0.2}{0.004} = 60$$

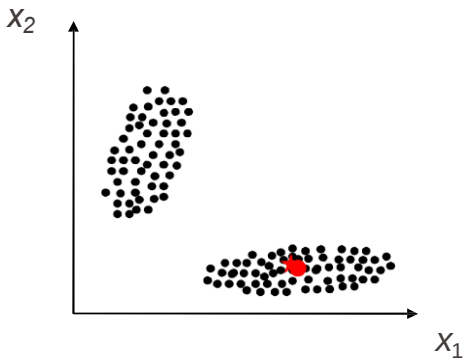
d=2h → w=0.05/h d=3h
→ w=0.004/h



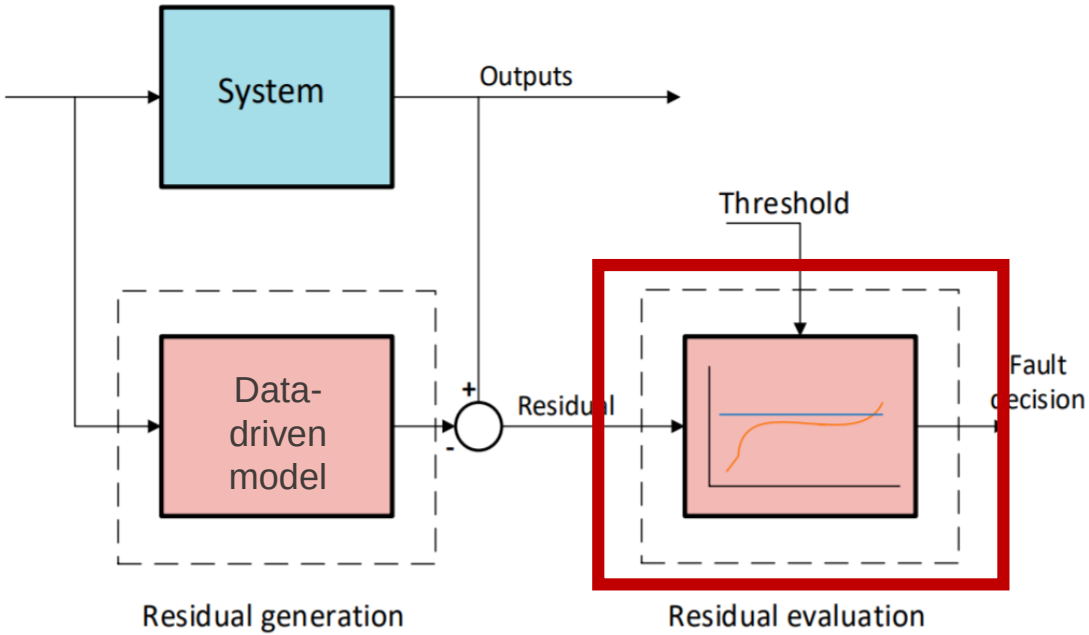
AAKR Performance: Accuracy

Accuracy:

- depends on the training set:
  - $\uparrow N \rightarrow \uparrow$ Accuracy



Basic idea of residual based methods



Thresholds

- Determine the thresholds based on the validation dataset (optionally: add additional margin)
- Two types of thresholds: for all the signals/residuals + for single signals/residuals → fault isolation
- Perform statistical tests on the distributions of the residuals

Autoencoders

Autoencoders

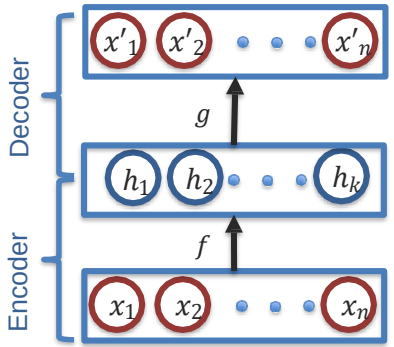
- Why autoencoders?
  - Map high-dimensional data to two dimensions for visualization

    Compression (i.e. reducing the file size)
    - Note: autoencoders don’t do this for free — it requires other ideas as well.

  - Learn abstract features in an unsupervised way so you can apply them to a supervised task
    - Unlabled data can be much more plentiful than labeled data

Autoencoders

- Network is trained to output the input (learn identify function).
- Two parts encoder/decoder
  - $x' = g(f(x))$
  - g - decoder
  - f - encoder



Trivial solution unless:
- Constrain number of units in Layer 2 (learn compressed representation), or
- Constrain Layer 2 to be **sparse**

Basic principles of an autoencoder

If the input is $x \in \mathbb{R}^n$ an autoencoder will produce a $h \in \mathbb{R}^d$ where $d < n$, which is designed to contain most of the important features of $\mathbf{x}$ to reconstruct it.

Autoencoder performs the following steps:

- **Encoder**: Perform a dimensionality reduction step on the data, $\mathbf{x} \in \mathbb{R}^n$ to obtain features $\mathbf{h} \in \mathbb{R}^d$.
- **Decoder**: Map the features $\mathbf{h} \in \mathbb{R}^d$ to closely reproduce the input, $\hat{\mathbf{x}} \in \mathbb{R}^n$.

Thus, the autoencoder implements the following problem:

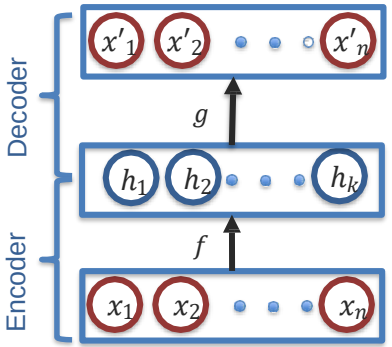
Let $\mathbf{x} \in \mathbb{R}^n$, $f(\cdot) : \mathbb{R}^n \to \mathbb{R}^d$ and $g(\cdot) : \mathbb{R}^d \to \mathbb{R}^n$. Let

$$\hat{\mathbf{x}} = g(f(\mathbf{x}))$$

Define a loss function, $\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}})$, and minimize $\mathcal{L}$ with respect to the parameters of $f(\cdot)$ and $g(\cdot)$.

There are different loss functions that you could consider, but a common one is the squared loss:

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$$

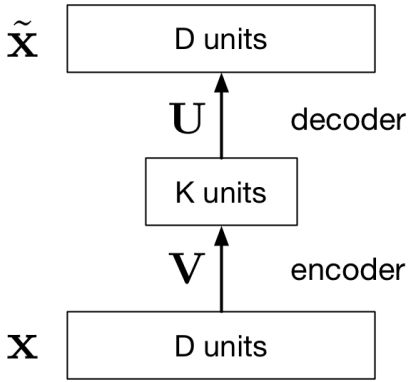


Source: J.C. Kao, UCLA

Simplest AE: Principal Component Analysis

- The simplest kind of autoencoder has one hidden layer, linear activations, and squared error loss.

$$\mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|^2$$



- This network computes $\tilde{\mathbf{x}} = \mathbf{UVx}$, which is a linear function.
- If $K \geq D$, we can choose **U** and **V** such that **UV** is the identity. This isn't very interesting.
- But suppose $K < D$:
  - **V** maps **x** to a K-dimensional space, so it's doing dimensionality reduction.
  - The output must lie in a K-dimensional subspace, namely the column space of **U**.

Source: R. Grosse

AE: PCA

- The autoencoder should learn to choose the subspace which minimizes the squared distance from the data to the projections.

- This is equivalent to the subspace which maximizes the variance of the projections.

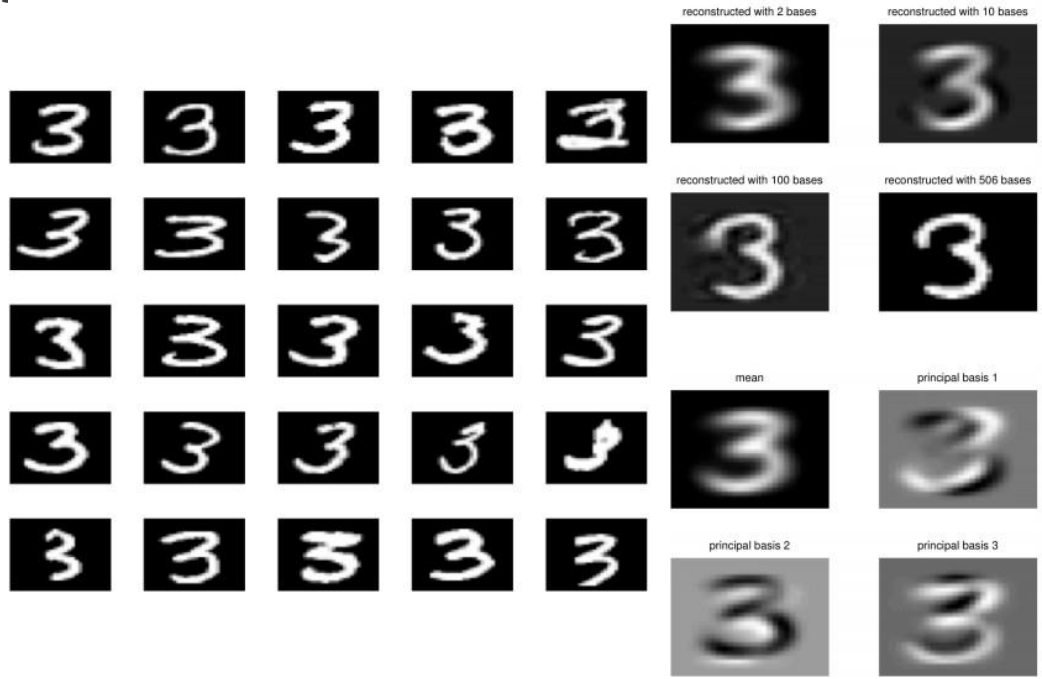
AE: PCA

PCA for faces (“Eigenfaces”)

Source: R. Grosse

AE: PCA

PCA for digits



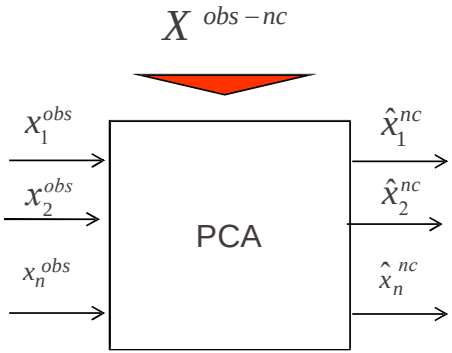
Source: R. Grosse

PCA as AE for fault detection

Training pattern $X^{obs-nc} = \begin{bmatrix} x_{11}^{obs-nc} & \dots & x_{1j} & \dots & x_{1n}^{obs-nc} \\ \dots & \dots & \dots & \dots & \dots \\ x_{k1} & \dots & x_{kj} & \dots & x_{kn} \\ \dots & \dots & \dots & \dots & \dots \\ x_{N1}^{obs-nc} & \dots & x_{Nj} & \dots & x_{Nn}^{obs-nc} \end{bmatrix} =$ historical signal measurements in normal condition

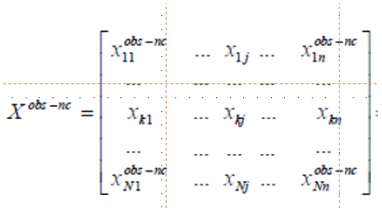
Test pattern: input $x^{obs} = (x_1^{obs}, \dots, x_n^{obs})$ = measured signals at current time

Test pattern: output $\hat{x}^{nc} = (x_1^{nc}, \dots, x_n^{nc})$ = signal reconstructions (expected values of the signals in normal condition)



PCA as AE for fault detection

- Historical data



- Measured signals at present time: $\vec{x}^{obs} = (x_1^{obs},...,x_n^{obs})$

- Find P from X^{obs-nc}
- Trasform: $\vec{x}^{obs} P$

- Antitrasform $\hat{\vec{x}}^{nc} = \vec{x}^{obs} P P^T$ Signal reconstructions

$\vec{x}^{obs} \cong \hat{x}^{nc}$ → normal condition

$\vec{x}^{obs} \neq \hat{x}^{nc}$ → abnormal condition

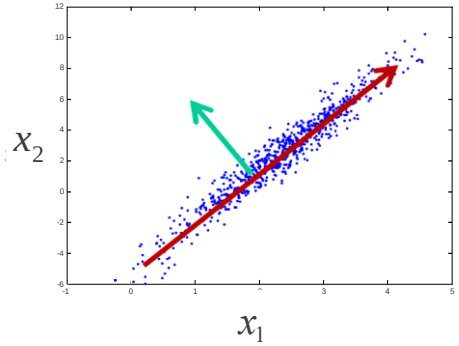
PCA as AE for anomaly detection

1) Find Principal Components:

- PC_1 is the direction of maximum variance
- PC_2,…, PC_n are orthogonal to PC_1 and describe maximum residual variance

$\rightarrow PC_1$

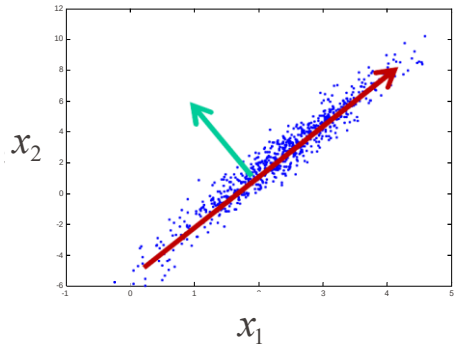
$\rightarrow PC_2$



PCA as AE for anomaly detection

Step 1) Find Principal Components:

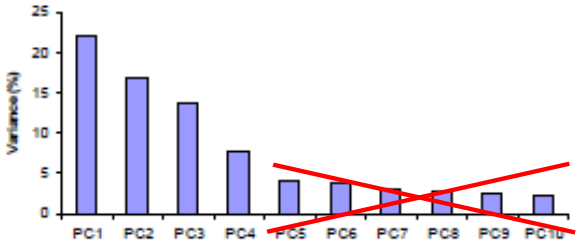
- PC_1 is the direction of maximum variance
- PC_2,…, PC_n are orthogonal to PC1 and describe maximum residual variance



PC_1

PC_2

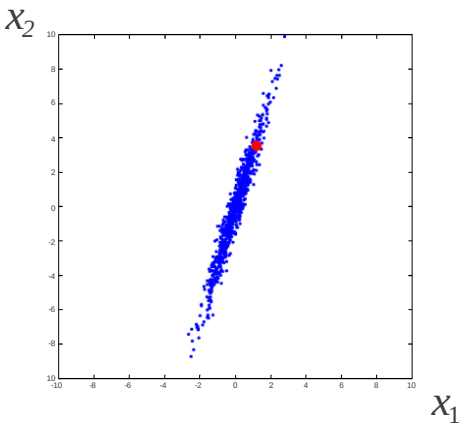
Step 2) PCA approximation: ignore the PCs of lower significance.



- Lost small information
- Example: number of dimensions from *n*=10 to $\lambda = 4$

PCA as AE for anomaly detection Example 1

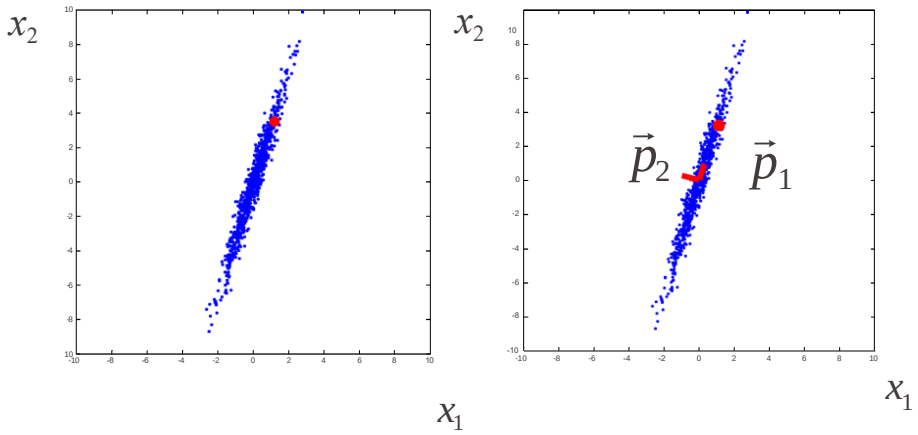
- Measured signals at present time: $\vec{x}^{obs} = (x_1^{obs}, x_2^{obs})$
- Signal reconstructions?
- Normal or abnormal condition?



- available historical signal measurements in normal system condition

PCA as AE for anomaly detection Example 1

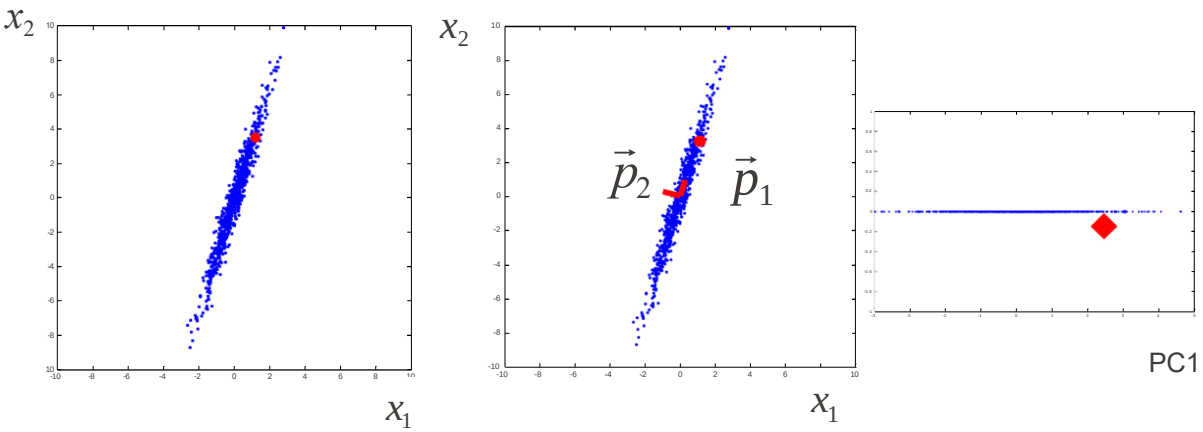
- Measured signals at present time: $\vec{x}^{obs} = (x_1^{obs}, x_2^{obs})$
- Step 1: find principal components: $\vec{p}_1, \vec{p}_2$



- available historical signal measurements in normal system condition

PCA as AE for anomaly detection Example 1

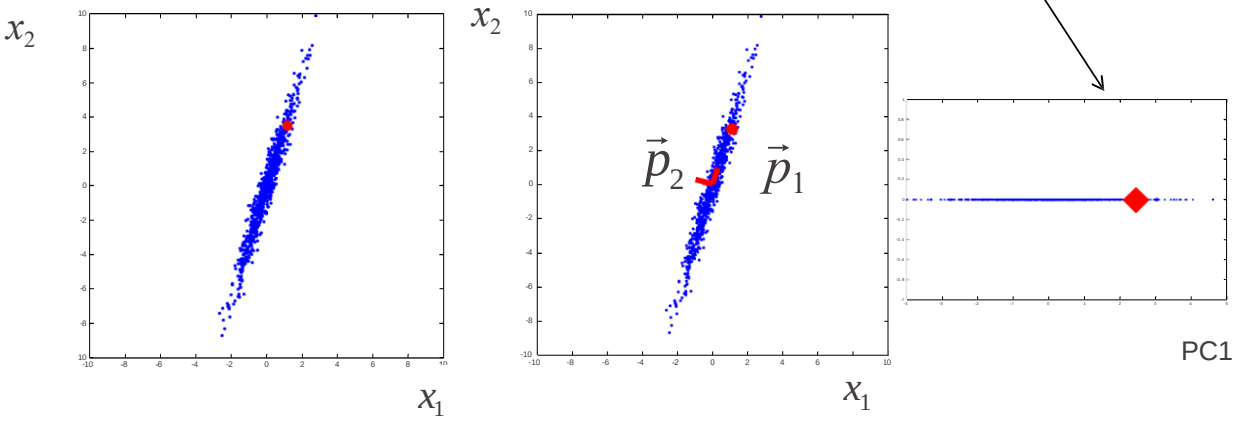
- Measured signals at present time: $\vec{x}^{obs} = (x_1^{obs}, x_2^{obs})$
- Step 1: find principal components $\vec{p}_1, \vec{p}_2$



PCA as AE for anomaly detection Example 1

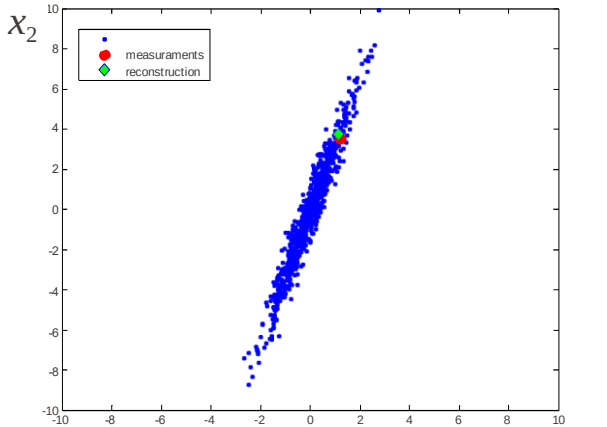
- Measured signals at present time: $\vec{x}^{obs} = (x_1^{obs}, x_2^{obs})$
- Step 1: find principal components: $\vec{p}_1, \vec{p}_2$
- Step 2 (PCA approximation): keep only i.e.1 PC

$\vec{x}^{obs} \cdot \vec{p}_1$



PCA as AE for anomaly detection Example 1

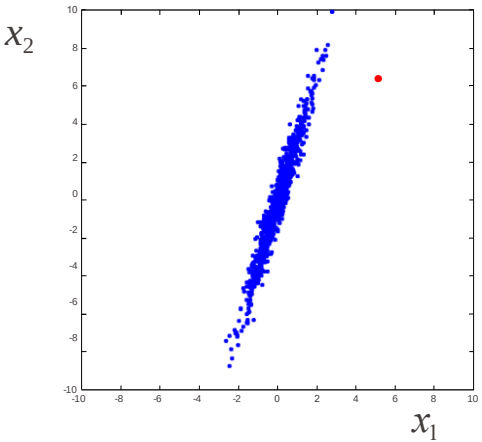
- Measured signals at present time: $\vec{x}^{obs} = (x_1^{obs}, x_2^{obs})$
- Step 1: find principal components: $\vec{p}_1, \vec{p}_2$
- Step 2 (PCA approximation): keep only 1 PC: $\vec{x}^{obs} \cdot \vec{p}_1$
- Step 3 (Reconstruct to the original coordinates): $\vec{x}^{nc} = \vec{x}^{obs} P\, P^T$



$\vec{x}^{obs} \cong \hat{x}^{nc}$ normal condition

PCA as AE for anomaly detection Example 2

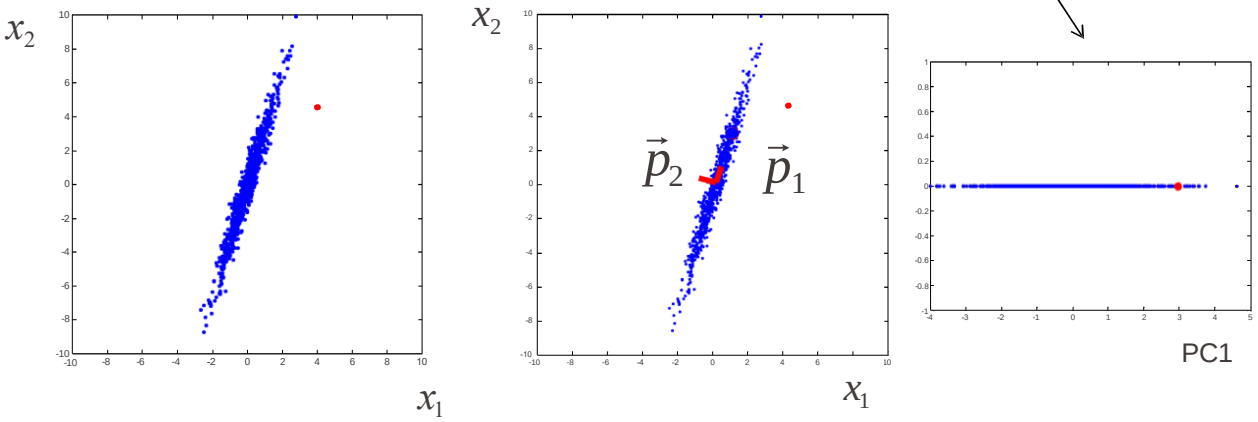
- Measured signals at present time: $\vec{x}^{obs} = (x_1^{obs}, x_2^{obs})$
- Signal reconstructions?
- Normal or abnormal condition?



- available historical signal measurements in normal system condition

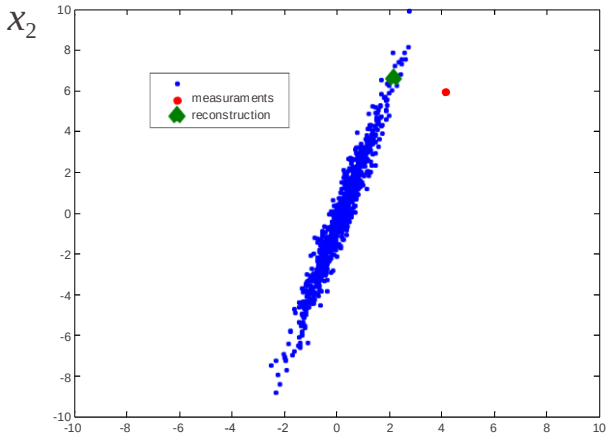
PCA as AE for anomaly detection Example 2

- Measured signals at present time: $\vec{x}^{obs} = (x_1^{obs}, x_2^{obs})$
- Step 1: find principal components: $\vec{p}_1, \vec{p}_2$
- Step 2 (PCA approximation): keep only i.e.1 PC $\vec{x}^{obs} \cdot \vec{p}_1$



PCA as AE for anomaly detection Example 2

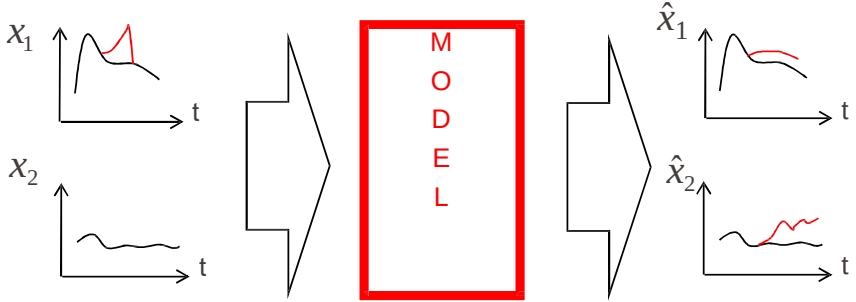
- Measured signals at present time: $\vec{x}^{obs} = (x_1^{obs}, x_2^{obs})$
- Step 1: find principal components: $\vec{p}_1, \vec{p}_2$
- Step 2 (PCA approximation): keep only 1 PC: $\vec{x}^{obs} \cdot \vec{p}_1$
- Step 3 (Reconstruct to the original coordinates): $\vec{x}^{nc} = \vec{x}^{obs} P P^T$



$\vec{x}^{obs} \neq \hat{x}^{nc}$ → Abnormal condition

PCA as AE evaluation

- Performance:
  - Accuracy = satisfactory
  - Low robustness and spillover effects on highly correlated signals



  - Unsatisfactory for dataset characterized by highly non linear relationships.