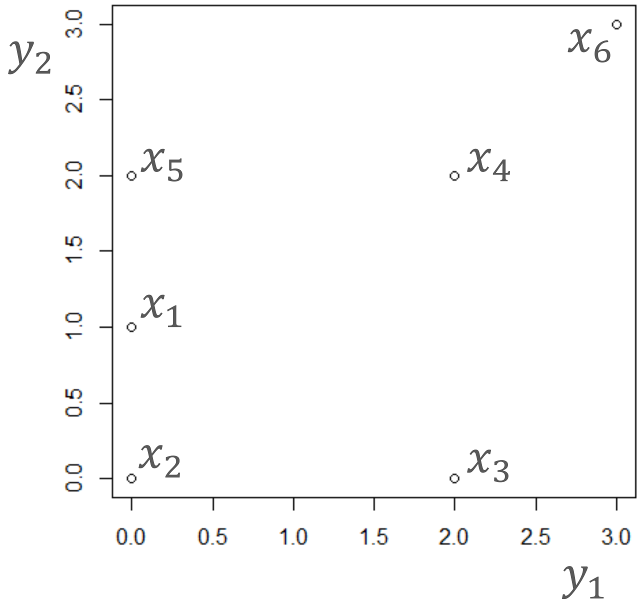
Spectral Clustering

Clustering in General

Spectral Clustering Overview

Dataset

i	y_1	y_2
x_1	0	1
x_2	0	0
x_3	2	0
x_4	2	2
x_5	0	2
x_6	3	3

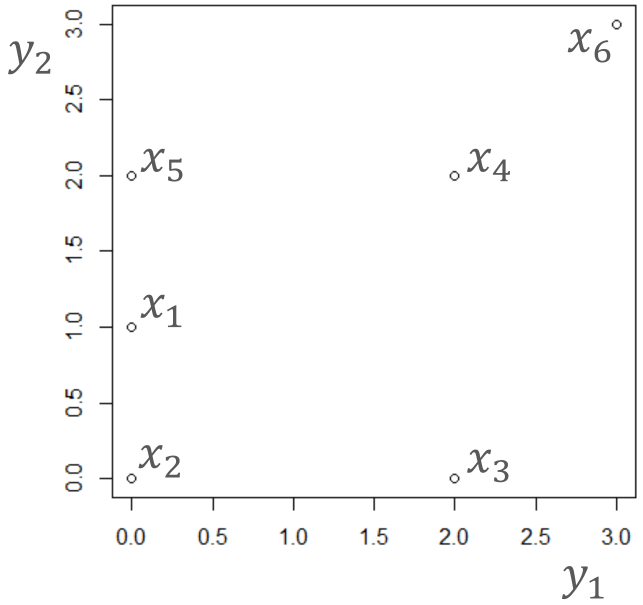


Clustering in General

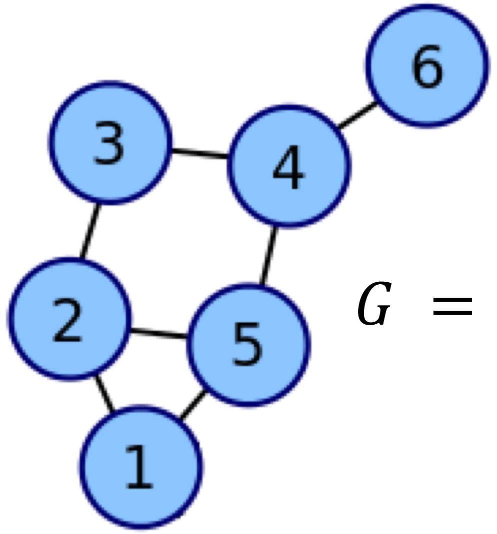
Spectral Clustering Overview

Dataset

i	y_1	y_2
x_1	0	1
x_2	0	0
x_3	2	0
x_4	2	2
x_5	0	2
x_6	3	3



Similarity Graph

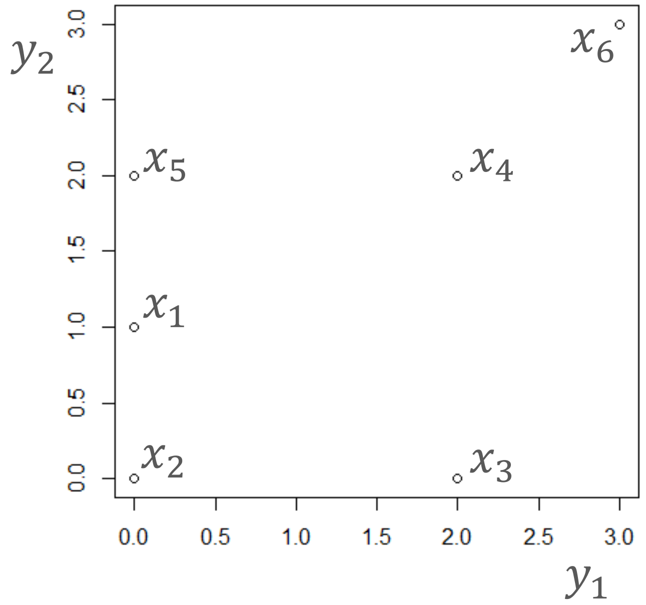


$$G = (V, E, W)$$

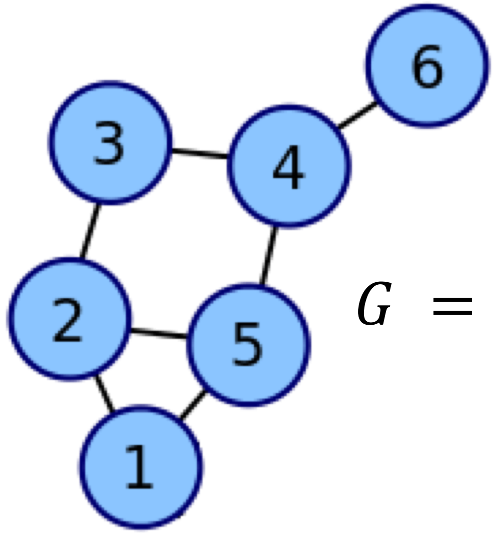
Clustering in General
Spectral Clustering Overview

Dataset

i	y_1	y_2
x_1	0	1
x_2	0	0
x_3	2	0
x_4	2	2
x_5	0	2
x_6	3	3



Similarity Graph



$G = (V, E, W)$

Eigenvalue Problem

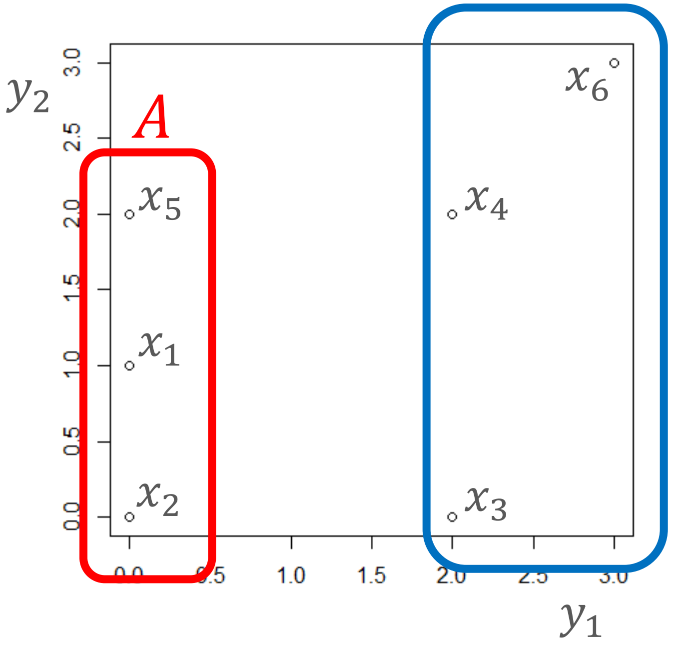
$L\vec{u} = \lambda\vec{u}$

and **???**

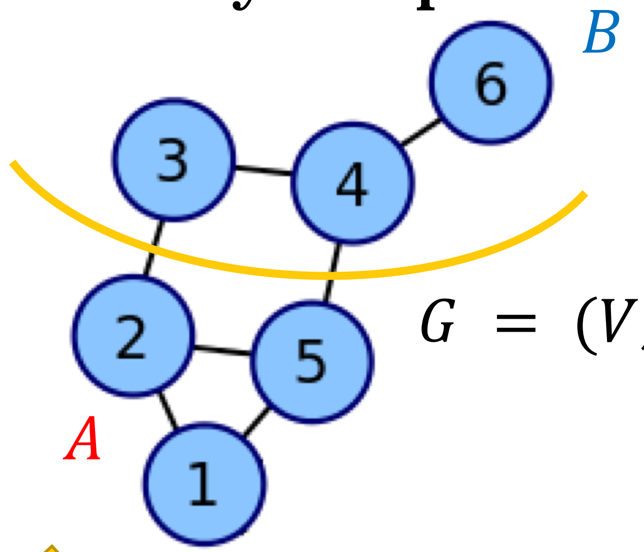
Clustering in General
Spectral Clustering Overview

i	y_1	y_2
x_1	0	1
x_2	0	0
x_3	2	0
x_4	2	2
x_5	0	2
x_6	3	3

Dataset



Similarity Graph



$G = (V, E, W)$

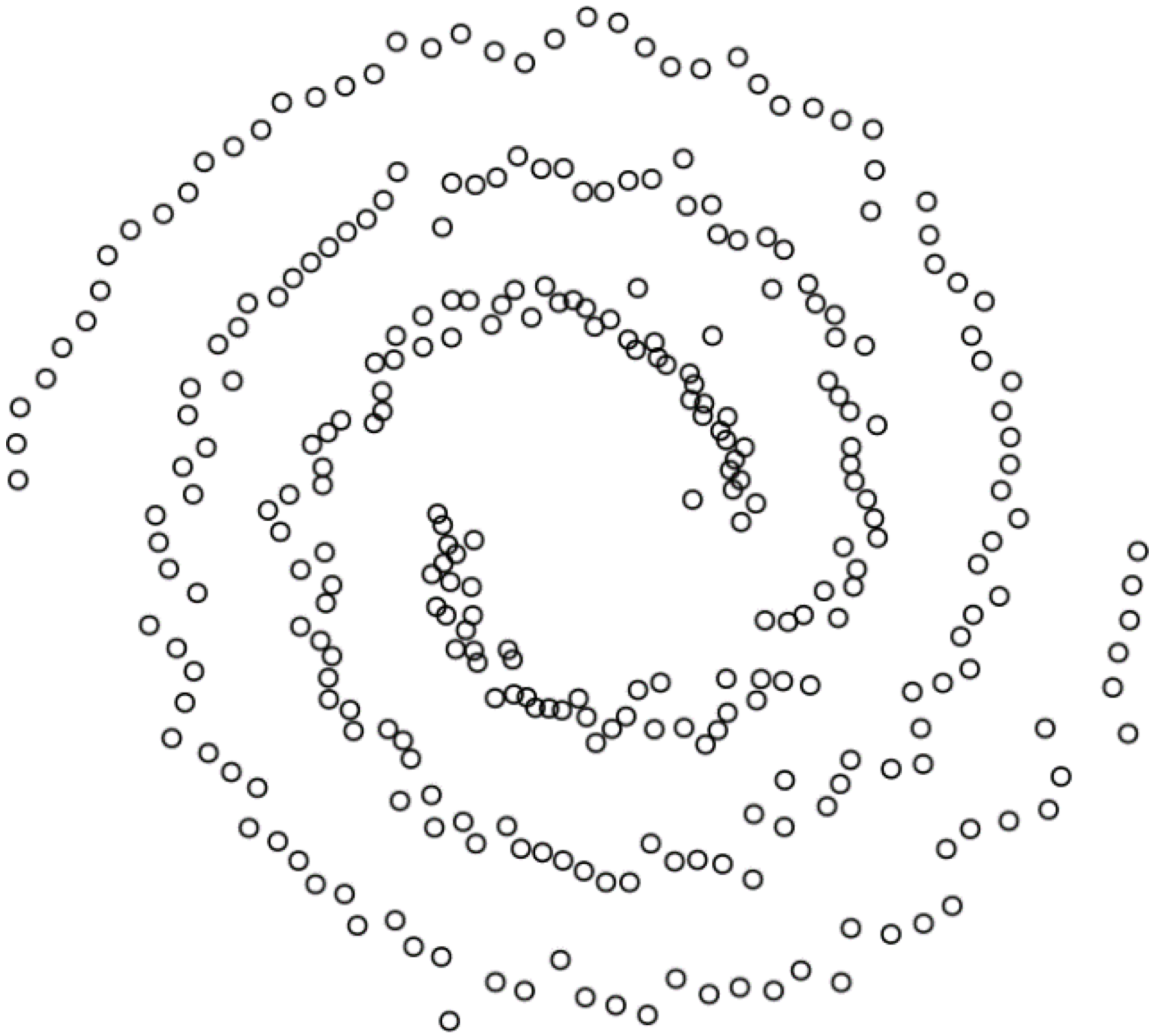
Eigenvalue Problem

$L\vec{u} = \lambda\vec{u}$

and **???**

??????

Clustering in General
Motivation for Spectral Clustering



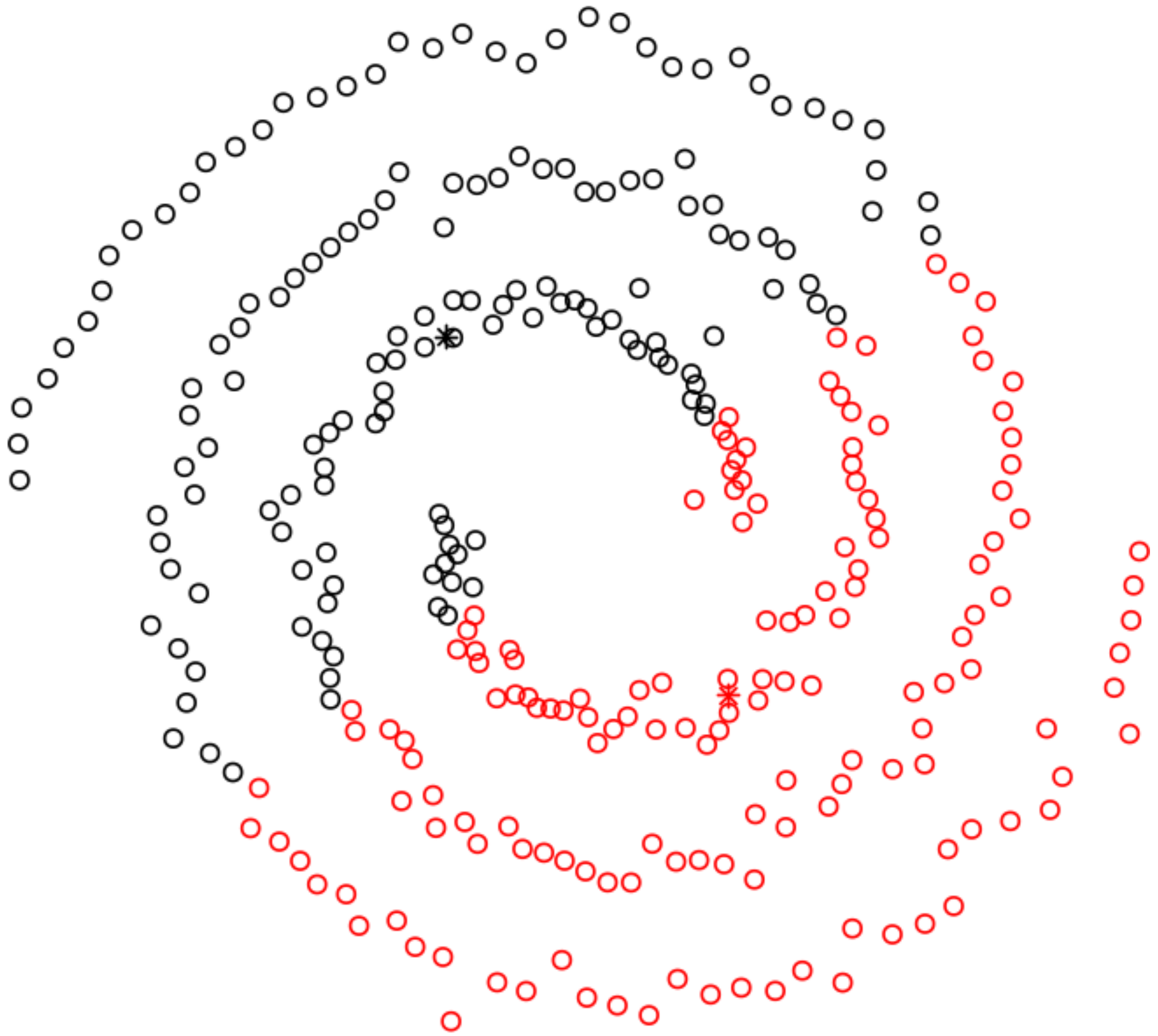
Spectral clustering makes no assumption on the shape of a cluster

- compactness vs. connectedness

Spectral clustering can be implemented efficiently for large datasets

- computing eigenvalues is numerically "efficient"

Clustering in General
Motivation for Spectral Clustering



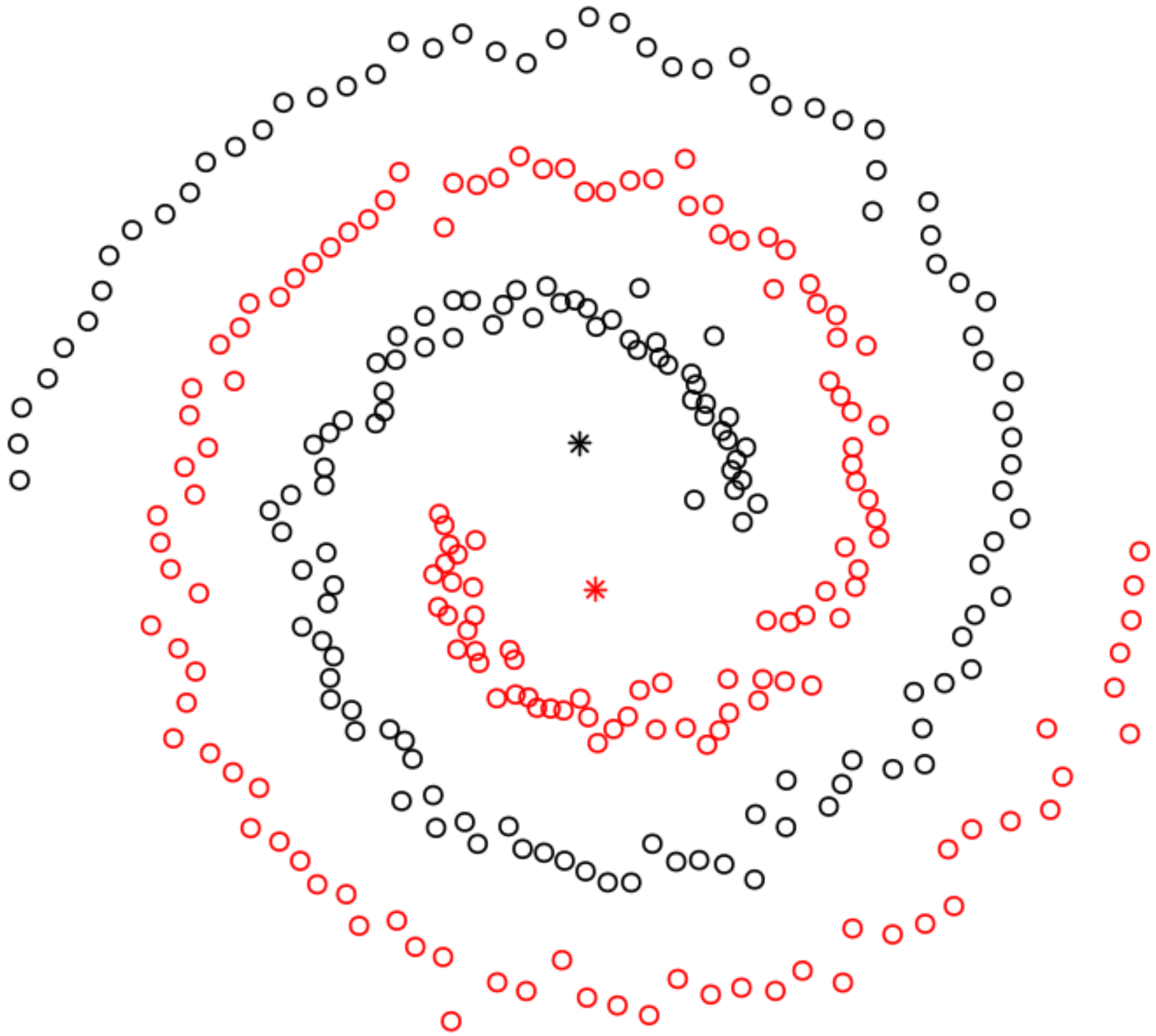
Spectral clustering makes no assumption on the shape of a cluster

- compactness vs. connectedness

Spectral clustering can be implemented efficiently for large datasets

- computing eigenvalues is numerically "efficient"

Clustering in General
Motivation for Spectral Clustering



Spectral clustering makes no assumption on the shape of a cluster

- compactness vs. connectedness

Spectral clustering can be implemented efficiently for large datasets

- computing eigenvalues is numerically "efficient"

Data Pre-Processing
Similarity Graph

In graph theory, the notation of a **similarity graph** is $G = (V, E, W)$.

1. Data points x are **vertices** $v \in V$.
2. A pair of vertices v_i, v_j are connected by an **edge** $e_{ij} = 1$ if the **similarity weight** $w_{ij} > \tau$ for a given threshold $\tau \in [0,1)$.
3. The edges e_{ij} form the **adjacency matrix** E.
4. The weights w_{ij} form the **similarity matrix** W.
5. The (diagonal) **degree matrix** D provides information about the number of edges attached to a vertex: $d_{ii} = \sum_{j=1}^{n} e_{ij}$.

External requirements: threshold τ, similarity measure w.

Data Pre-Processing
Similarity Graph – Example

With the **Gower** similarity measure on data with m features

$$w_G(v_i, v_j) = 1 - \frac{1}{m}\sum_{k=1}^{m} \frac{|x_{i,k} - x_{j,k}|}{\text{range of } k^{\text{th}} \text{ feature}}$$

the similarity matrix of the previous data is

$$W = \begin{pmatrix} 0 & 5/6 & 1/2 & 1/2 & 5/6 & 1/6 \\ 5/6 & 0 & 2/3 & 1/3 & 2/3 & 0 \\ 1/2 & 2/3 & 0 & \boxed{2/3} & 1/3 & 1/3 \\ 1/2 & 1/3 & \boxed{2/3} & 0 & 2/3 & 2/3 \\ 5/6 & 2/3 & 1/3 & 2/3 & 0 & 1/3 \\ 1/6 & 0 & 1/3 & 2/3 & 1/3 & 0 \end{pmatrix}$$

For instance, $w_G(v_3, v_4) = w_{34} = w_{43} = 1 - \frac{1}{2}\left\{\frac{|x_{3,1} - x_{4,1}|}{r_1} + \frac{|x_{3,2} - x_{4,2}|}{r_2}\right\}$.

But $r_1 = r_2 = 3$, so $w_{34} = w_{43} = 1 - \frac{1}{6}\{|2-2| + |0-2|\} = \frac{2}{3}$.

Data Pre-Processing

Similarity Graph – Example

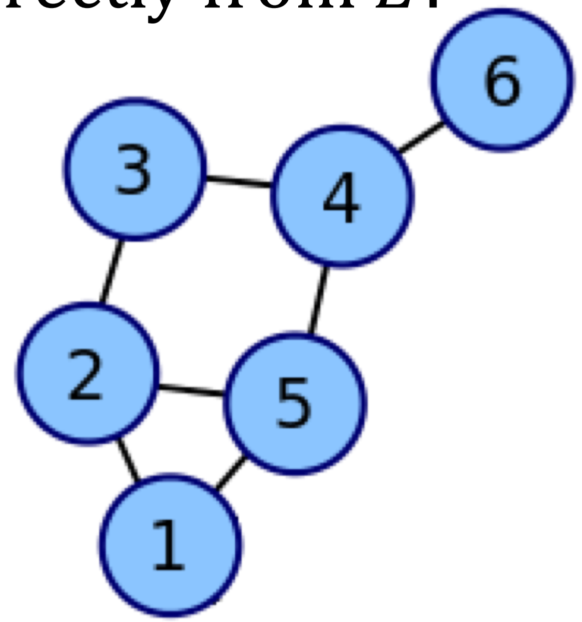
- Let's use a threshold value $\tau = 0.6$. The adjacency matrix is thus

$$E = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

- Incidentally, the degree matrix is

$$D = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

- The similarity graph G is read directly from E:



- Now all that is left is to **partition** the graph!

Graph Partitions
Graph Cuts

- A **graph cut** partitions a graph into two sub-graphs (**clusters**) A, B.
- The goal is to partition the graph so that edges within a group have large weights (so the vertices they join are **similar**) and edges across groups have small weights (so the vertices they join are **dissimilar**).
- We focus on one way to do this: the **Normalized Cut**.

Other partition schemes: Min Cut, Ratio Cut, Min Max Cut

- An objective function $J(A, B)$ must be minimized against the set of all possible partitions (A, B).
- The partition which minimizes J gives rise to the first clustering level.
- The procedure can be repeated as necessary on the cluster sub-graphs.

Graph Partitions

Normalized Cut – Example

Objective function:

$$J_{\text{NCut}} = \text{Cut}(A, B)\left(\frac{1}{\text{Vol}(A)} + \frac{1}{\text{Vol}(B)}\right)$$

Sum of all the weights on edges emanating from B

Sum of all the weights on edges emanating from A

Sum of all the weights on edges starting in one group and ending in the other

Advantages:

- Takes into consideration the size of partitioned groups
- Tends to avoid isolating vertices
- Takes into consideration intra-group variance

Limitations

- Not an easy optimization problem to solve (**NP-hard!!**)

Graph Partitions
Normalized Cut – Example

Objective function:

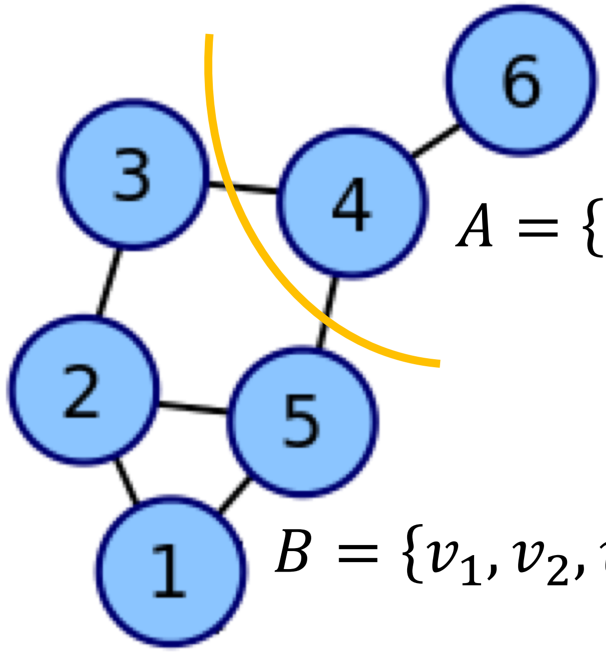
$$J_{\text{NCut}} = \text{Cut}(A,B)\left(\frac{1}{\text{Vol}(A)} + \frac{1}{\text{Vol}(B)}\right)$$

Advantages:

- Takes into consideration the size of partitioned groups
- Tends to avoid isolating vertices
- Takes into consideration intra-group variance

Limitations

- Not an easy optimization problem to solve (**NP-hard!!**)



$A = \{v_4, v_6\}$

$B = \{v_1, v_2, v_3, v_5\}$

$\text{Cut}(A,B) = \sum_{i\in A, j\in B} w_{ij} = 3$

$\text{Vol}(A) = \sum_{i\in A, j\in V} w_{ij} = 13/3$

$\text{Vol}(B) = \sum_{i\in V, j\in B} w_{ij} = 32/3$

$J_{\text{NCut}}(A,B) = 0.97$

The Eigenvalue Problem

How Spectral Clustering Got Its Name

Spectral clustering is a **compromise**: it solves an *easier* problem than Normalized Cut optimization, but with *similar* solutions.

The **Laplacian matrix** is a spectral representation of a graph.

- Simple Laplacian: $L = D - E$ Careful! There are competing definitions.
- Symmetric Laplacian: $L_S = D^{-1/2} L D^{-1/2}$
- Asymmetric Laplacian (random walk): $L_A = D^{-1} L$

In the case of two clusters, J_{NCut} is minimized when finding the eigenvector f for the **second smallest** eigenvalue of L_S, leading to the name of the method (special case of general algorithm, see later). L is positive semi-definite and its smallest eigenvalue is 0

The clustering is recovered by sending $v_i \in A$ when $f_i > 0$, and $v_i \in B$ otherwise (or *vice-versa*). Deterministic?

Interlude
Eigenvalues and Laplacian Matrices

An **eigenvalue** λ of a matrix T is a complex number (potentially with no imaginary part) such that $\dim \ker(T - \lambda I) > 0$.

In other words, λ is an eigenvalue of T if there exists (at least) an **eigenvector** $\vec{v} \neq 0$ such $T\vec{v} = \lambda\vec{v}$.

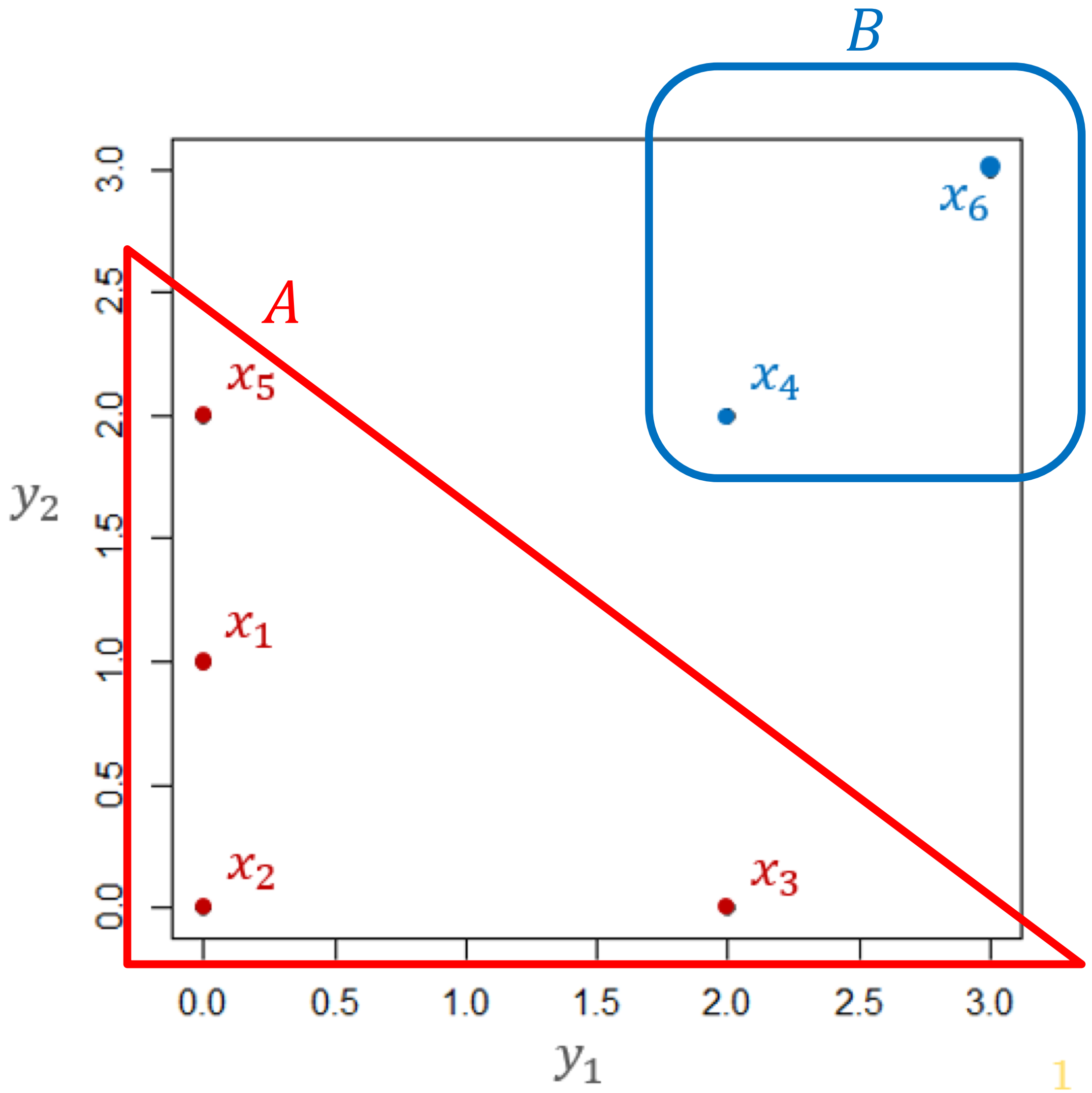
The Laplacian matrix L of a graph is a matrix representation of that graph.

The Laplacian matrix has a bevy of nice properties that ensure that its eigenvalues behave "as they should"; for instance, the dimension of the eigenspace associated with the eigenvalue $\lambda = 0$ measures the number of connected components in the graph.

A first guess for # of clusters?

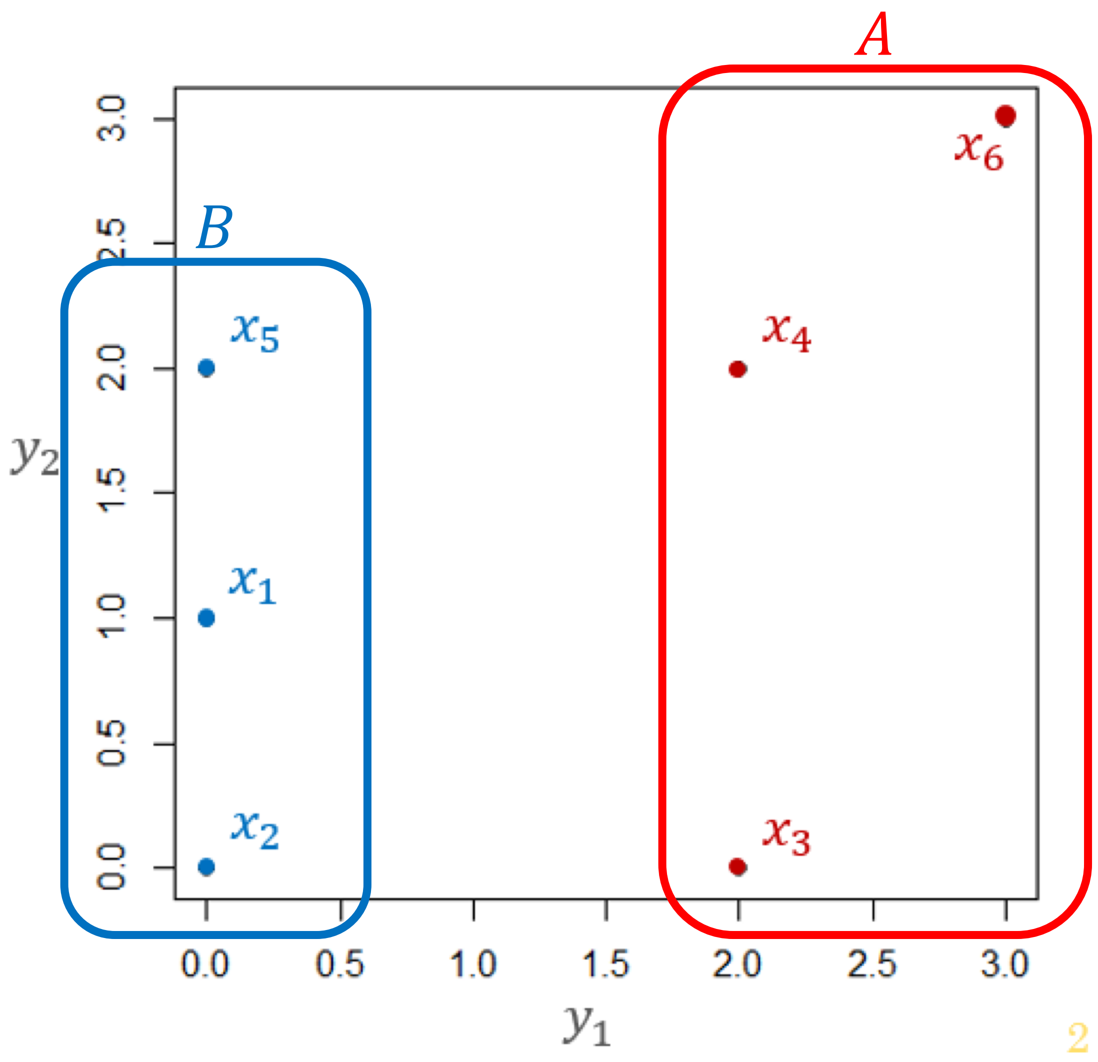
The Eigenvalue Problem
Simple Laplacian – Example

i	f_i	A/B
x_1	-0.39	A
x_2	-0.30	A
x_3	-0.18	A
x_4	0.03	B
x_5	-0.16	A
x_6	0.84	B



The Eigenvalue Problem
Symmetric Laplacian – Example

i	f_i	A/B
x_1	0.36	B
x_2	0.54	B
x_3	-0.03	A
x_4	-0.43	A
x_5	0.14	B
x_6	-0.61	A



The Eigenvalue Problem

Spectral Clustering –Algorithm
(version from von Luxburg's tutorial, with different D and L)

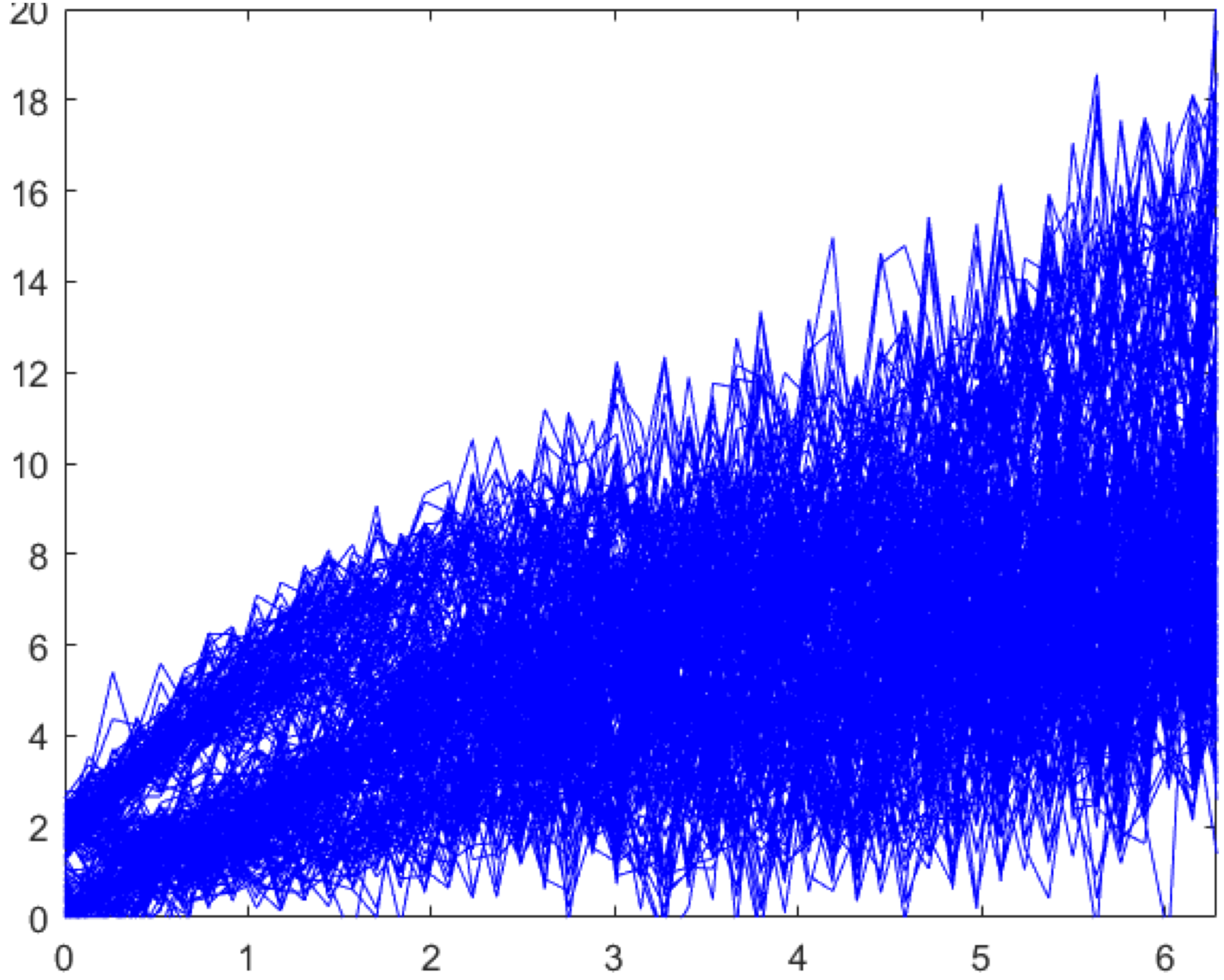
Algorithm to cluster $\{x_1, \dots, x_n\}$ into k clusters: Choice of # of clusters

1. Form similarity matrix W. Choice of similarity measure
2. Define the degree matrix D. Choice of adjacency threshold
3. Construct the Laplacian matrix L. Choice of Laplacian
4. Compute the first k orthogonal eigenvectors $\{\mu_1, \dots \mu_k\}$ of the Laplacian L corresponding to its k **smallest** eigenvalues.
5. Construct U, using $\mu_1, \dots, \mu_k$ as **columns**.
6. Normalize the **rows** of U so that they each have unit length; call the new matrix Y.
7. Cluster the rows of Y into k clusters. Choice of clustering method
8. Assign the original point x_i to cluster j if the i^{th} row of Y was assigned to cluster j.

Other algorithms: un-normalized spectral clustering, Shi and Malik's algorithm (see von Luxburg's tutorial).

Examples and Case Studies

Latent Classes – Time Series

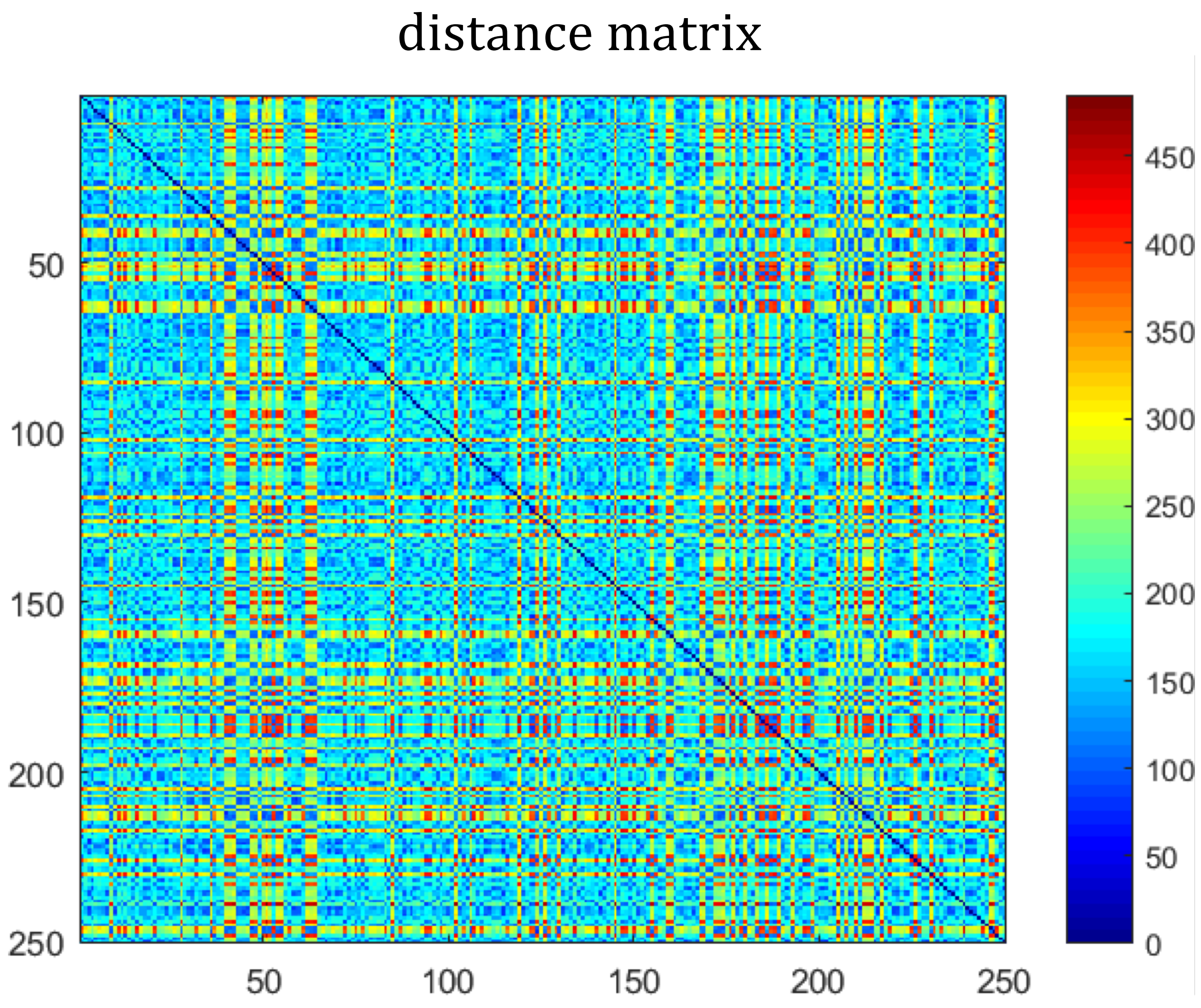


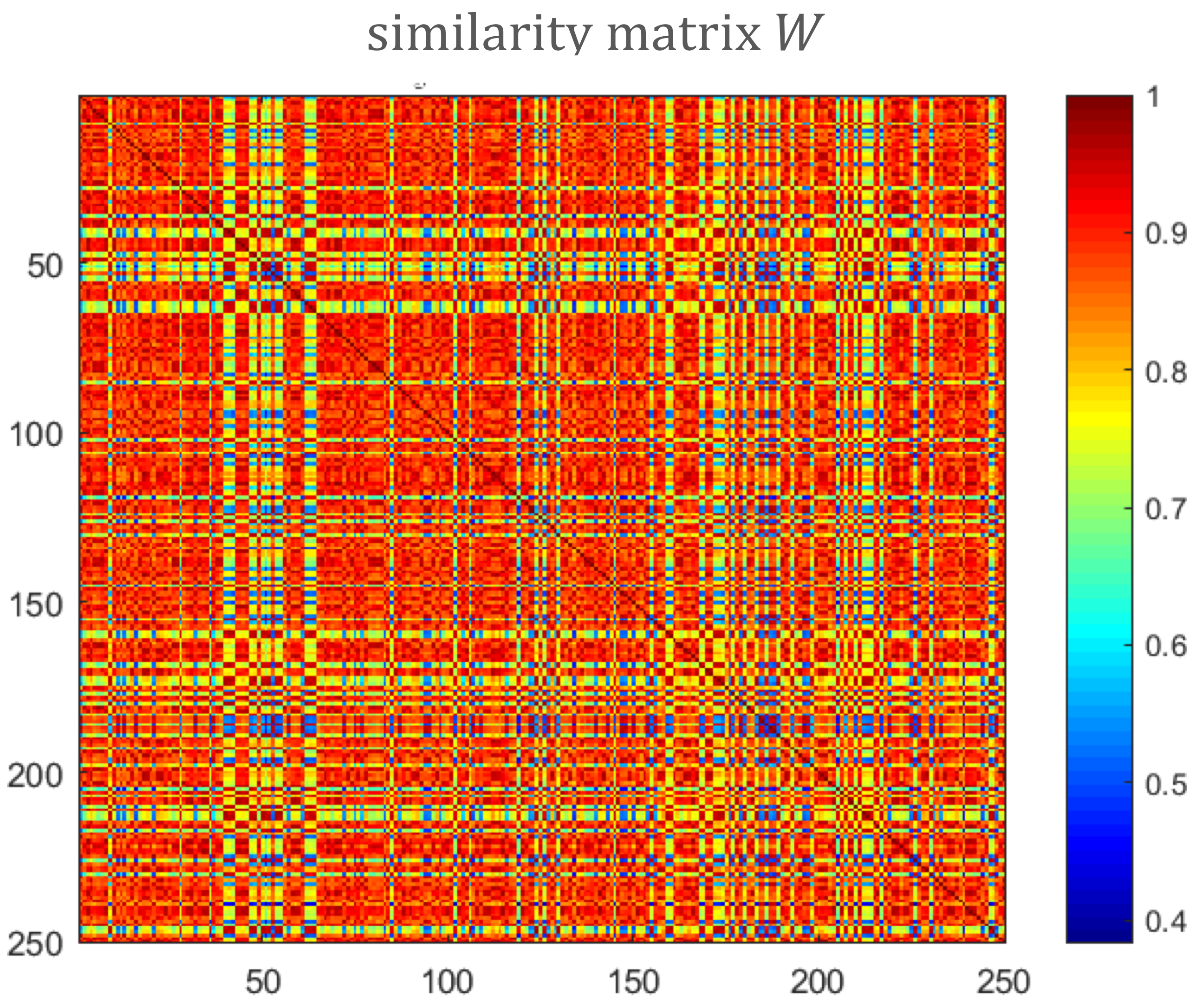
- 250 times series
- average absolute gap between series used as distance d
- Gaussian similarity measure

$$w = \exp\left(-\frac{d^2}{2\sigma^2}\right)$$

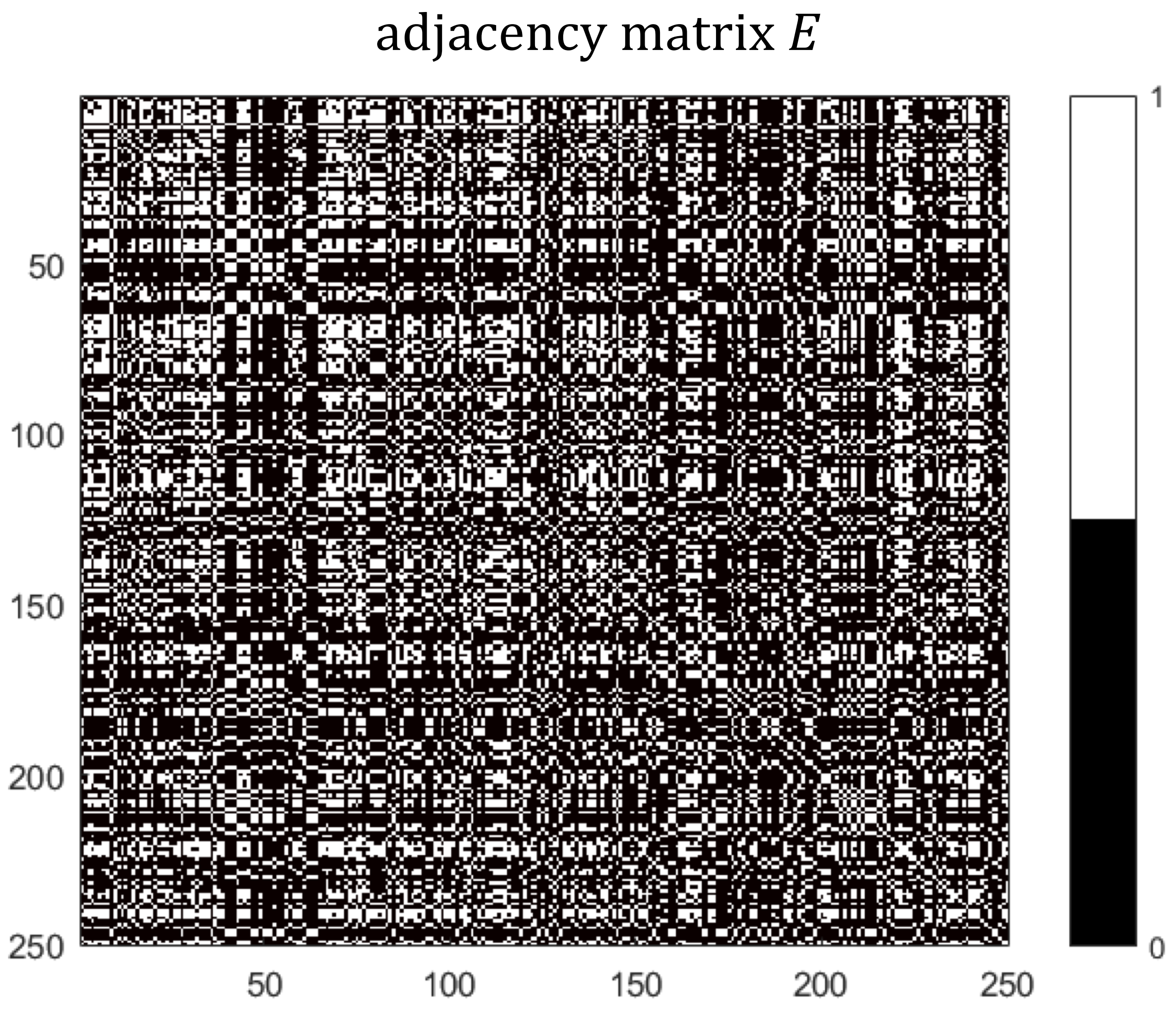
- $\sigma = 300$
- adjacency threshold $\tau = 0.9$
- $k = 5$ clusters

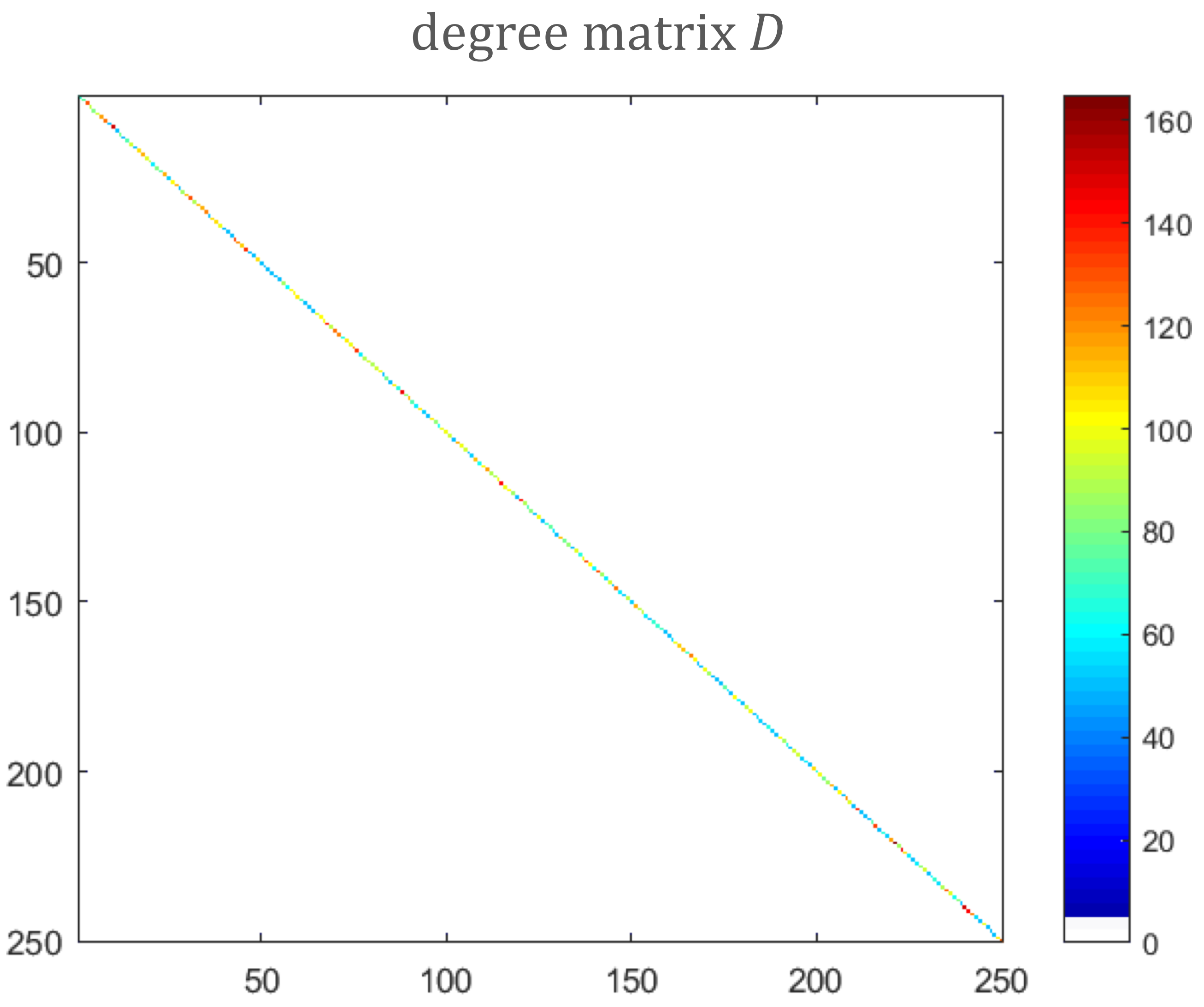
Examples and Case Studies
Latent Classes – Time Series





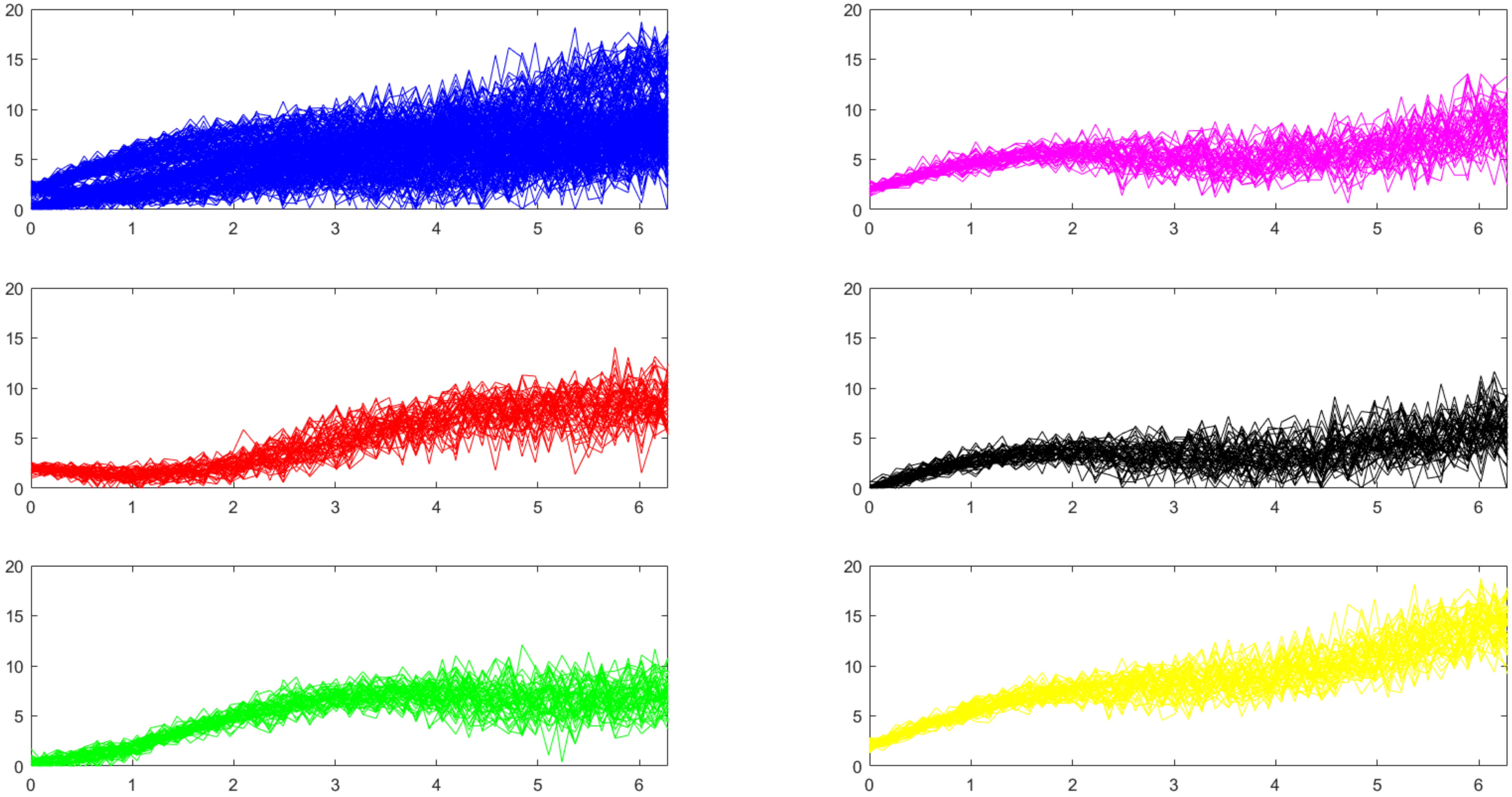
Examples and Case Studies
Latent Classes – Time Series



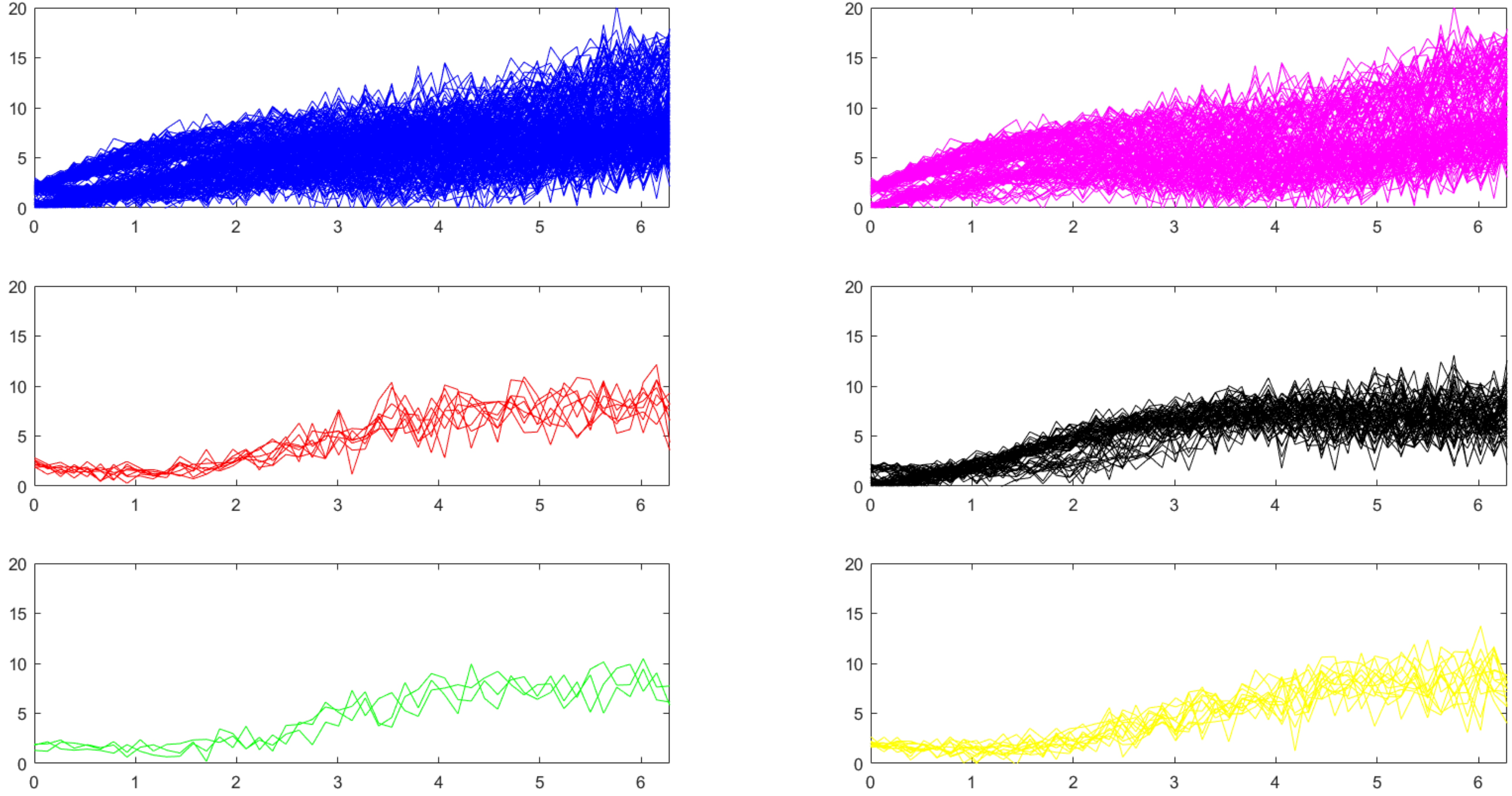


Examples and Case Studies

Latent Classes – Time Series

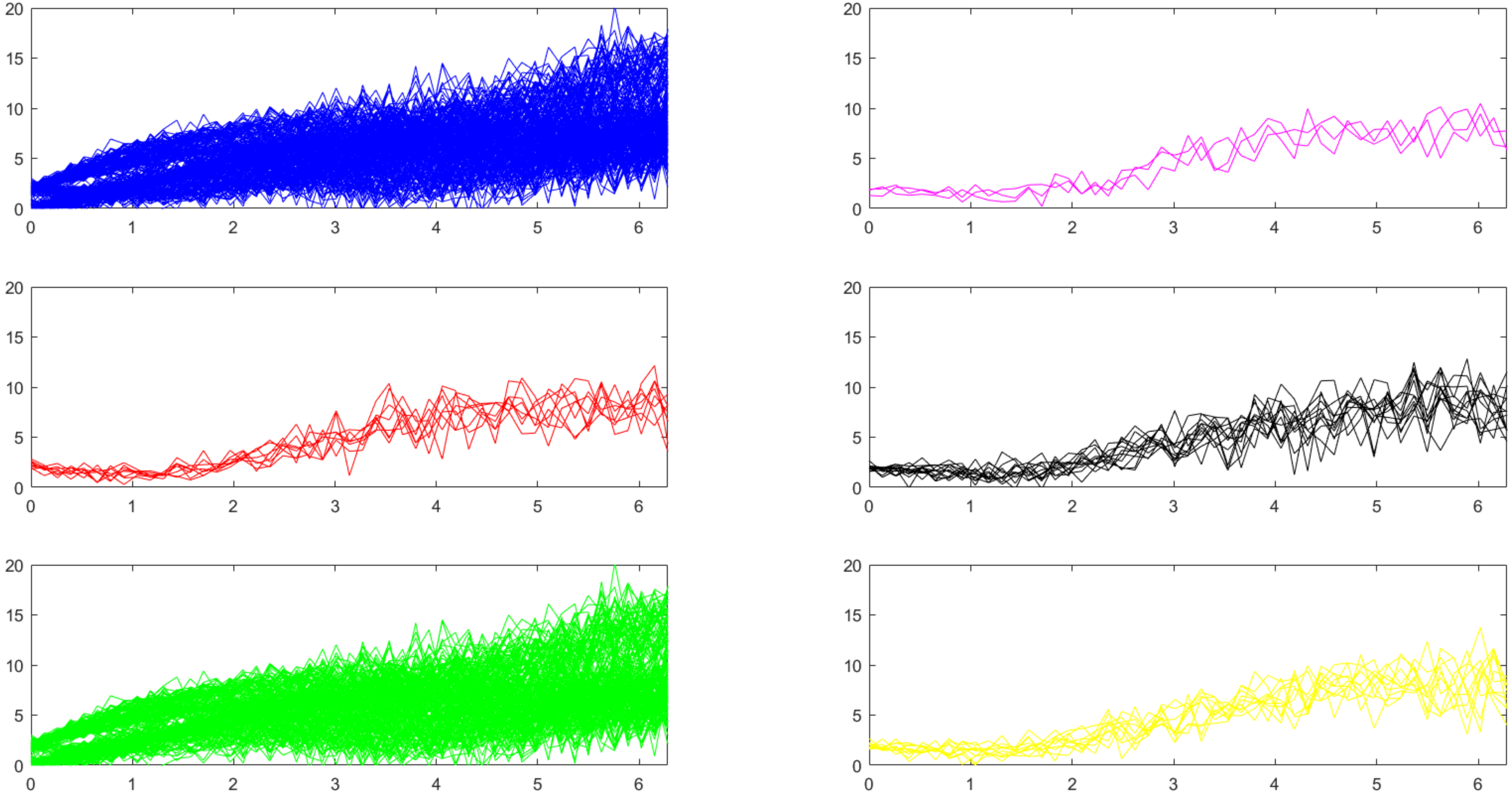


Examples and Case Studies
Latent Classes – Time Series



Examples and Case Studies

Latent Classes – Time Series



Examples and Case Studies
Signal Processing – Spectral Clustering for Speech Separation

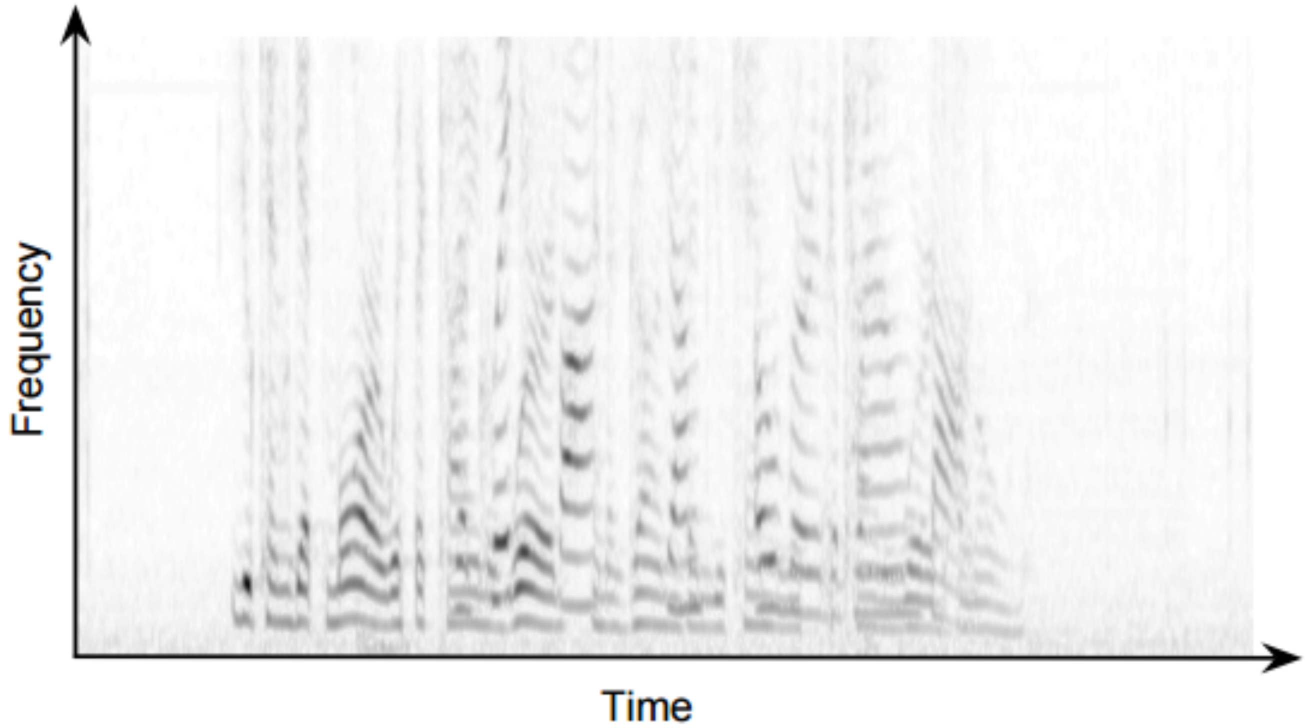
Project: Francis R. Bach and Michael I. Jordan combined prior relevant knowledge with learning similarity algorithm, to explain spectral clustering.

Goal: apply the algorithm to separate two speakers from a one-microphone blind source.

Data: Two speakers give speech and their voice signal is collected by a one-microphone blind source.

Spectrogram of speech (two simultaneous English speakers).

The gray intensity is proportional to the amplitude of the spectrogram.



Examples and Case Studies
Signal Processing – Spectral Clustering for Speech Separation

Method:

- Assume partitions are known in the given sample data.
- Perform spectral clustering on the similarity matrices
- Obtain the same partitions as assumed previously

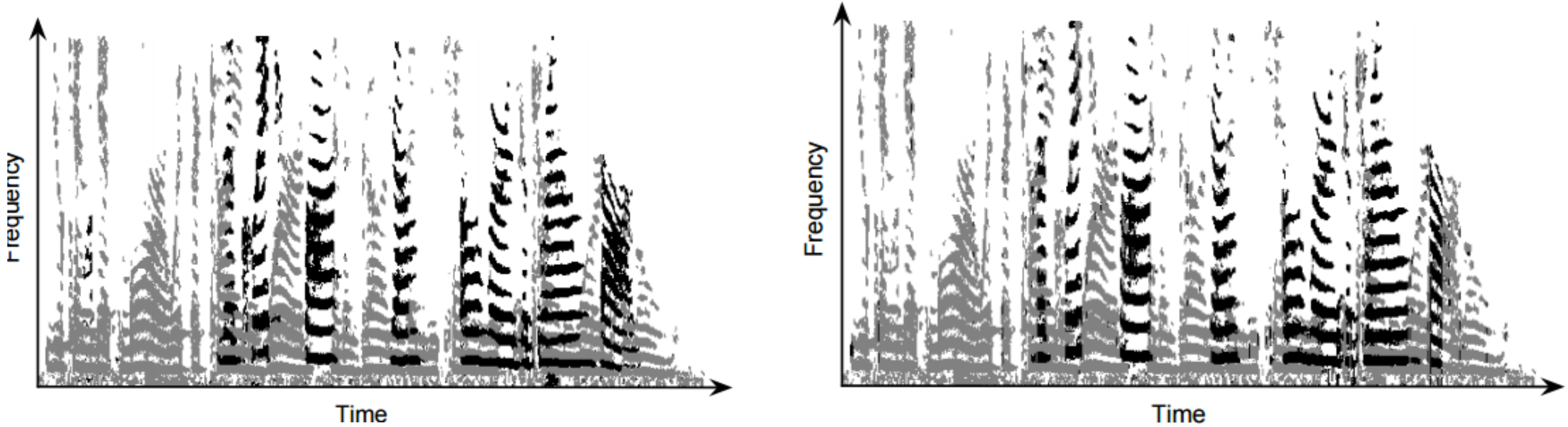
Algorithm: Similar to NJW.

Challenges:

- Limited to the setting of ideal acoustics and equal-strength mixing of two speakers
- Training examples can be created by mixing previously captured signals
- Spectral clustering needs to be robust to irrelevant features
- Computation challenge of spectral clustering applied to speech separation

Examples and Case Studies
Signal Processing – Spectral Clustering for Speech Separation

The result is an optimized segmenter for spectrograms of speech mixtures.



Selected result: (Left) Optimal segmentation for the spectrogram of English speakers, where the two speakers are “black” and “grey”; this segmentation is obtained from the known separated signals. (Right) The blind segmentation obtained with our algorithm.

F.R. Bach, M.I. Jordan, *Learning Spectral Clustering, With Application to Speech Separation*, Journal of Mach. Learn. Res. 7

Examples and Case Studies

Sensor Detection – A Spectral Clustering Approach to Validating Sensors via Their Peers in Distributed Sensor Networks

Project: H. T. Kung and Dario Vlah describe a spectral clustering approach to identify bad sensors, by using a simple model problem.

Motivation: current status and environment affect sensors performance and impractical to bring calibrate device to test each sensor

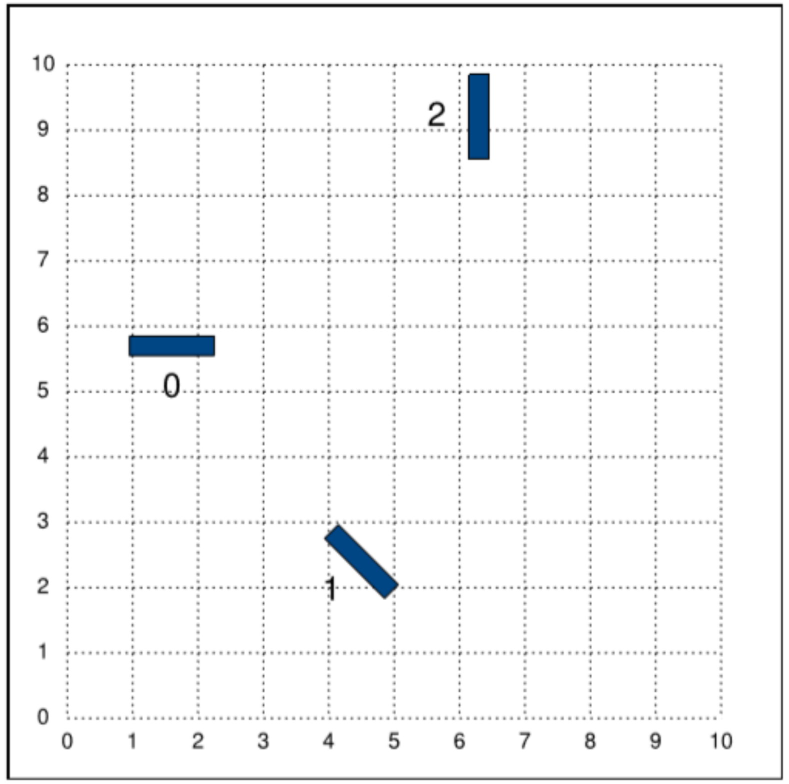
Goal: using peer sensors to detect badly performing sensors

Method: simulation and spectral clustering

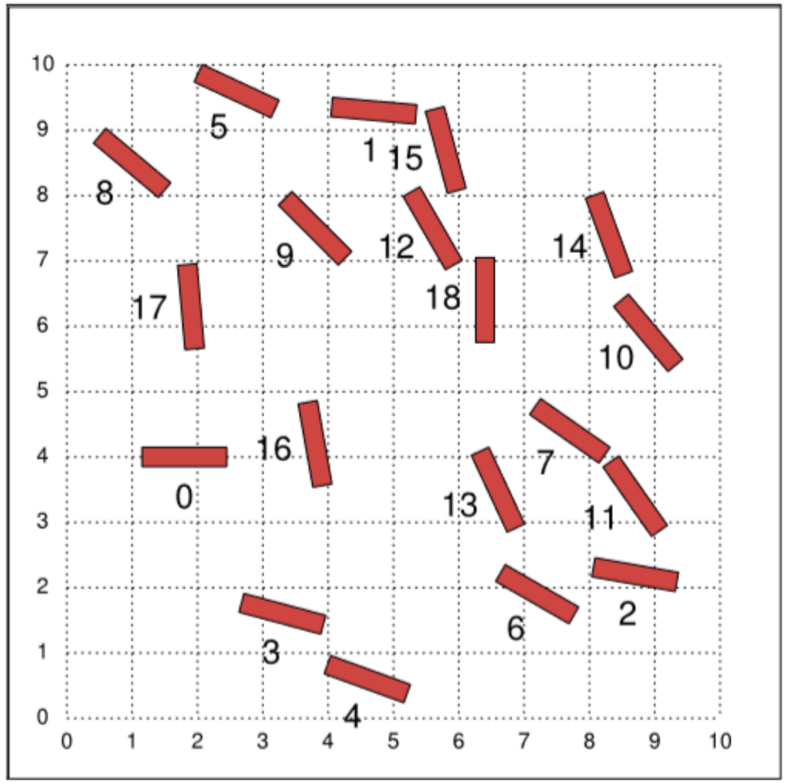
Examples and Case Studies
Sensor Detection – A Spectral Clustering Approach to Validating Sensors via Their Peers in Distributed Sensor Networks

Model design: sensors are indexed by their antenna orientations

- assume that the matching of a sensor and a target is based on the degree to which their antenna orientations match
- use of non-principal eigenvector with the principal one, to detect clustering structures



(a) 3 targets with 3 antenna orientations



(b) 19 sensors with 19 antenna orientations

Sensors and targets in the same region

Examples and Case Studies

Sensor Detection – A Spectral Clustering Approach to Validating Sensors via Their Peers in Distributed Sensor Networks

Simulation of large systems on the same model design

- Data: 100 sensors and 10 targets
- Assumption 1: sensors and targets are evenly partitioned into three groups, with antenna orientations of 0, 45 and 90 degrees
- Assumption 2: some randomly selected sensors are bad sensors in the sense that their measurements can be off by any amount from -100% to +100%

Results:

- When the number k of leading eigenvectors used increases, the accuracy performance improves
- The number of false positives decreases with the number of bad sensors input to the simulator.
- Spectral clustering achieves almost perfect performance in specific circumstances.

Clustering Validation
Is a Clustering Scheme Any Good?

There is **NO** optimal validation approach.

Possibilities include:

- comparing with the optimal clustering (external)
- comparing with other clustering methods (external)
- visualizing the clusters (external)
- Davies-Bouldin, Within-SS (internal) ⇐ # of clusters
- repeated clusterings (internal) ⇐ co-clustered items

Scenario 1: given data D, true clustering C, algorithm A produces C':

- **is C' "close" to C?**

Scenario 2: given data D, true clustering C, algorithm A produces C'; algorithm A^* produces C^*, and so forth.

- **are C', C^*, … , "close" to C? Which one is "closer"?**

Clustering Validation
Is a Clustering Scheme Any Good?

A distance measurement $d(C, C')$ between clusterings is needed…

Let $C = \{C_1, \dots, C_k\}$ be a **clustering** of a set of n data points $\{x_1, \dots, x_n\}$.

The **quadratic cost** is the function defined by

$$\Lambda(C) = -\mathrm{Trace}\big(Z^T(C) \cdot W \cdot Z(C)\big),$$

where Z is the matrix representation of C:

$$z_{ik} = \begin{cases} 1 & \text{if } x_i \in C_k \\ 0 & \text{if } x_i \notin C_k \end{cases}$$

In some sense, the clustering scheme for which $\Lambda(C)$ is minimized is **optimal** against quadratic cost.

For a given choice of similarity measure