Clustering - part 1

Rebecka Jörnsten, Mathematical Sciences

MSA220/MVE441 Statistical Learning for Big Data

8th May 2025

Classification without classes

In **classification** the main idea was to determine

$$p(i|\mathbf{x}) \quad \text{or} \quad p(\mathbf{x}, i) = p(\mathbf{x}|i)p(i)$$

through model approximations (LDA, logistic regression), rules/partitioning (CART, random forests) or directly from data (kNN).

Classification without classes

In **classification** the main idea was to determine

$$p(i|\mathbf{x}) \quad \text{or} \quad p(\mathbf{x}, i) = p(\mathbf{x}|i)p(i)$$

through model approximations (LDA, logistic regression), rules/partitioning (CART, random forests) or directly from data (kNN).

What if we do not have any classes?

Classification without classes

In **classification** the main idea was to determine

$$p(i|\mathbf{x}) \quad \text{or} \quad p(\mathbf{x}, i) = p(\mathbf{x}|i)p(i)$$

through model approximations (LDA, logistic regression), rules/partitioning (CART, random forests) or directly from data (kNN).

What if we do not have any classes? Clustering

Classification without classes

In **classification** the main idea was to determine

$$p(i|\mathbf{x}) \quad \text{or} \quad p(\mathbf{x}, i) = p(\mathbf{x}|i)p(i)$$

through model approximations (LDA, logistic regression), rules/partitioning (CART, random forests) or directly from data (kNN).

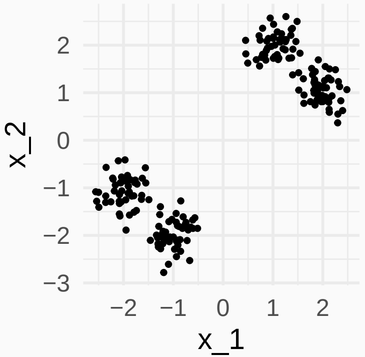
What if we do not have any classes? Clustering

Goals

- Find groups in data
- Summarize high-dimensional data
- Data exploration

Clustering

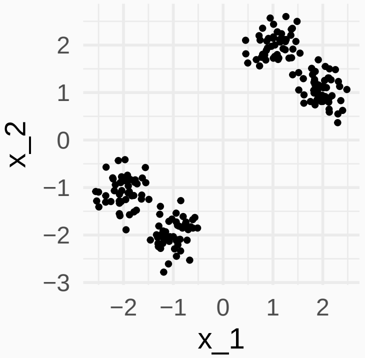
Clustering is a harder problem than classification



Clustering

Clustering is a harder problem than classification

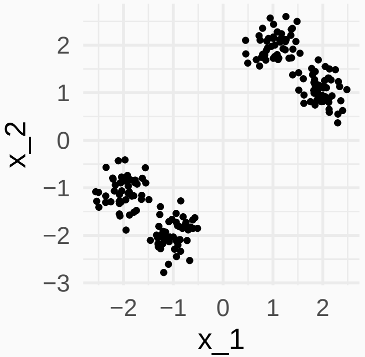
- What is a cluster?



Clustering

Clustering is a harder problem than classification

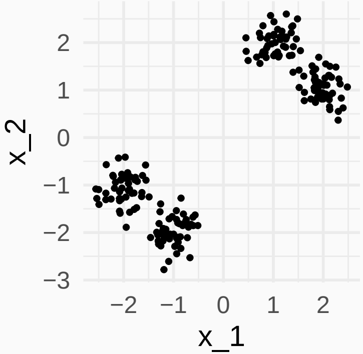
- What is a cluster?
- How many clusters are there?



Clustering

Clustering is a harder problem than classification

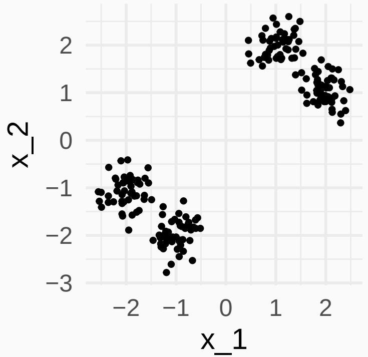
- What is a cluster?
- How many clusters are there?
- How do we find them? Can they have any shape?



Clustering

Clustering is a harder problem than classification

- What is a cluster?
- How many clusters are there?
- How do we find them? Can they have any shape?



We need to able to measure **dissimilarity** between features to determine which samples/objects are close together or far apart.

Note: In clustering *classes* are often called **labels** and *features* are **attributes**

Dissimilarity measures

A **dissimilarity measure** for features x_1, x_2 is a function such that

$$d(x_1, x_2) \geq 0 \quad \text{and} \quad d(x_1, x_2) = d(x_2, x_1)$$

Dissimilarity measures

A **dissimilarity measure** for features x_1, x_2 is a function such that

$$d(x_1, x_2) \geq 0 \quad \text{and} \quad d(x_1, x_2) = d(x_2, x_1)$$

Dissimilarity across all features can be defined as

$$D(\mathbf{x}_1, \mathbf{x}_2) = \sum_{j=1}^{p} d_j(x_1^{(j)}, x_2^{(j)})$$

Dissimilarity measures

A **dissimilarity measure** for features x_1, x_2 is a function such that

$$d(x_1, x_2) \geq 0 \quad \text{and} \quad d(x_1, x_2) = d(x_2, x_1)$$

Dissimilarity across all features can be defined as

$$D(\mathbf{x}_1, \mathbf{x}_2) = \sum_{j=1}^{p} d_j(x_1^{(j)}, x_2^{(j)})$$

Typical examples

- For quantitative features: ℓ_1 or ℓ_2 norm, correlation between whole feature vectors, ...
- For categorical variables with K levels: Loss matrix $\mathbf{L} \in \mathbb{R}^{K \times K}$ such that $\mathbf{L}_{rs} = \mathbf{L}_{sr}$, $\mathbf{L}_{rr} = 0$ and $\mathbf{L}_{rs} \geq 0$. Then $d(r, s) = \mathbf{L}_{rs}$

Challenges in Clustering

Two main challenges

Challenges in Clustering

Two main challenges

1. How many clusters are there?

Challenges in Clustering

Two main challenges

1. How many clusters are there?
2. Given a number of clusters, how do we find them?

Challenges in Clustering

Two main challenges

1. How many clusters are there?
2. Given a number of clusters, how do we find them?

Focus on Challenge 2 first.

Challenges in Clustering

Two main challenges

1. How many clusters are there?
2. Given a number of clusters, how do we find them?

Focus on Challenge 2 first.

Idea: Partition the observations into K groups/clusters so that **pairwise dissimilarities within groups** are **smaller than between groups.**

Note: A partition of the observations is called a **clustering** $C(\mathbf{x}) = i$

Combinatorial Clustering (I)

Total amount of dissimilarity for an arbitrary clustering C

$$T = \underbrace{\sum_{l=1}^{n} \sum_{m<l} D(\mathbf{x}_l, \mathbf{x}_m)}_{\text{Total point scatter}}$$

Combinatorial Clustering (I)

Total amount of dissimilarity for an arbitrary clustering C

$$T = \underbrace{\sum_{l=1}^{n} \sum_{m<l} D(\mathbf{x}_l, \mathbf{x}_m)}_{\text{Total point scatter}}$$

$$= \sum_{i=1}^{K} \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} \left(\sum_{\substack{m<l \\ C(\mathbf{x}_m)=i}} D(\mathbf{x}_l, \mathbf{x}_m) + \sum_{\substack{m<l \\ C(\mathbf{x}_m)\neq i}} D(\mathbf{x}_l, \mathbf{x}_m) \right)$$

Combinatorial Clustering (I)

Total amount of dissimilarity for an arbitrary clustering C

$$T = \underbrace{\sum_{l=1}^{n} \sum_{m<l} D(\mathbf{x}_l, \mathbf{x}_m)}_{\text{Total point scatter}}$$

$$= \sum_{i=1}^{K} \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} \left(\sum_{\substack{m<l \\ C(\mathbf{x}_m)=i}} D(\mathbf{x}_l, \mathbf{x}_m) + \sum_{\substack{m<l \\ C(\mathbf{x}_m)\neq i}} D(\mathbf{x}_l, \mathbf{x}_m) \right)$$

$$= \underbrace{\sum_{i=1}^{K} \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} \sum_{\substack{m<l \\ C(\mathbf{x}_m)=i}} D(\mathbf{x}_l, \mathbf{x}_m)}_{\substack{=: W(C) \\ \text{Within cluster point scatter}}} + \underbrace{\sum_{i=1}^{K} \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} \sum_{\substack{m<l \\ C(\mathbf{x}_m)\neq i}} D(\mathbf{x}_l, \mathbf{x}_m)}_{\substack{=: B(C) \\ \text{Between cluster point scatter}}}$$

Combinatorial Clustering (II)

Note that T does not depend on the clustering. Therefore

$$W(C) = T - B(C)$$

and **minimizing within cluster point scatter** is equivalent to **maximizing between cluster point scatter**.

Combinatorial Clustering (II)

Note that T does not depend on the clustering. Therefore

$$W(C) = T - B(C)$$

and **minimizing within cluster point scatter** is equivalent to **maximizing between cluster point scatter**.

As in the case of decision trees/CART looking at all possible partitions and finding the global minimum of $W(C)$ is too computational expensive.

Combinatorial Clustering (II)

Note that T does not depend on the clustering. Therefore

$$W(C) = T - B(C)$$

and **minimizing within cluster point scatter** is equivalent to **maximizing between cluster point scatter**.

As in the case of decision trees/CART looking at all possible partitions and finding the global minimum of $W(C)$ is too computational expensive.

Use **greedy algorithms** to find local minima.

An approximation to Combinatorical Clustering (I)

Consider the **special case** $D(\mathbf{x}_l, \mathbf{x}_m) = \|\mathbf{x}_l - \mathbf{x}_m\|^2$

An approximation to Combinatorical Clustering (I)

Consider the **special case** $D(\mathbf{x}_l, \mathbf{x}_m) = \|\mathbf{x}_l - \mathbf{x}_m\|^2$ then

$$W(C) = \sum_{i=1}^{K} \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} \sum_{\substack{m<l \\ C(\mathbf{x}_m)=i}} \|\mathbf{x}_l - \mathbf{x}_m\|^2$$

An approximation to Combinatorical Clustering (I)

Consider the **special case** $D(\mathbf{x}_l, \mathbf{x}_m) = \|\mathbf{x}_l - \mathbf{x}_m\|^2$ then

$$W(C) = \sum_{i=1}^{K} \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} \sum_{\substack{m<l \\ C(\mathbf{x}_m)=i}} \|\mathbf{x}_l - \mathbf{x}_m\|^2$$

$$= \sum_{i=1}^{K} n_i \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} \|\mathbf{x}_l - \mathbf{m}_i\|^2$$

where

$$n_i = \sum_{l=1}^{n} \mathbb{1}(C(\mathbf{x}_l) = i) \quad \text{and} \quad \mathbf{m}_i = \frac{1}{n_i} \sum_{C(\mathbf{x}_l)=i} \mathbf{x}_l$$

An approximation to Combinatorical Clustering (II)

The goal now is to solve

$$\arg\min_{C} \sum_{i=1}^{K} n_i \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} \|\mathbf{x}_l - \mathbf{m}_i(C)\|^2$$

which still requires to visit all possible partitions.

An approximation to Combinatorical Clustering (II)

The goal now is to solve

$$\arg\min_{C} \sum_{i=1}^{K} n_i \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} \|\mathbf{x}_l - \mathbf{m}_i(C)\|^2$$

which still requires to visit all possible partitions.

Observation: For a fixed clustering rule C it holds that

$$\mathbf{m}_i(C) = \arg\min_{\mathbf{m}} \sum_{C(\mathbf{x}_l)=i} \|\mathbf{x}_l - \mathbf{m}\|^2$$

An approximation to Combinatorical Clustering (II)

The goal now is to solve

$$\arg\min_{C} \sum_{i=1}^{K} n_i \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} \|\mathbf{x}_l - \mathbf{m}_i(C)\|^2$$

which still requires to visit all possible partitions.

Observation: For a fixed clustering rule C it holds that

$$\mathbf{m}_i(C) = \arg\min_{\mathbf{m}} \sum_{C(\mathbf{x}_l)=i} \|\mathbf{x}_l - \mathbf{m}\|^2$$

Approximative solution: Consider the larger problem

$$\arg\min_{\substack{C \\ m_i \text{ for } 1\leq i\leq K}} \sum_{i=1}^{K} n_i \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} \|\mathbf{x}_l - \mathbf{m}_i\|^2$$

k-means

This approximation can be solved iteratively for the clustering C and the cluster centres. This is called the **k-means** algorithm.

Computational procedure:

1. **Initialize:** Randomly choose K observations as cluster centres $\mathbf{m}_i$ and set $J_{\max}$ to a positive integer.
2. For steps $j = 1, \ldots, J_{\max}$
   2.1 **Cluster allocation:** $C(\mathbf{x}_l) = \underset{1 \leq i \leq K}{\arg\min} \|\mathbf{x}_l - \mathbf{m}_i\|^2$
   2.2 **Cluster centre update:** $\mathbf{m}_i = \frac{1}{n_i} \sum_{C(\mathbf{x}_l)=i} \mathbf{x}_l$
   2.3 Stop if clustering C did not change

Notes on k-means

- **Dependence on initial selection:** Run repeatedly to see if k-means provides stable results

Notes on k-means

- **Dependence on initial selection:** Run repeatedly to see if k-means provides stable results
- Since k-means uses the ℓ_2 norm it has all the typical problems (**sensitive to outliers and noise**)

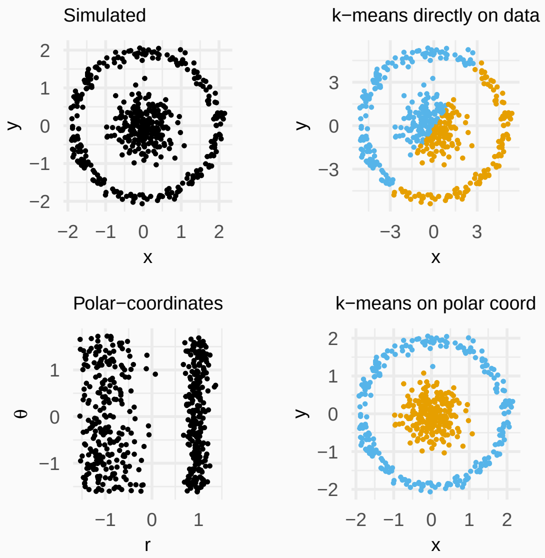
Notes on k-means

- **Dependence on initial selection:** Run repeatedly to see if k-means provides stable results
- Since k-means uses the ℓ_2 norm it has all the typical problems (**sensitive to outliers and noise**)
- **Clusters tend to be circular:** k-means looks in a circular fashion around each cluster centre and assigns an observation to the closest centre

Notes on k-means

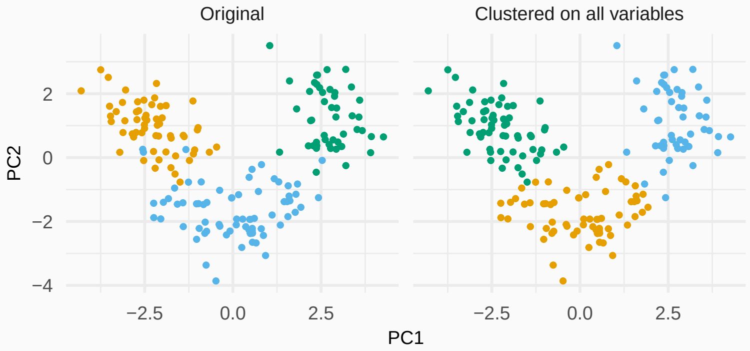
- **Dependence on initial selection:** Run repeatedly to see if k-means provides stable results
- Since k-means uses the ℓ_2 norm it has all the typical problems (**sensitive to outliers and noise**)
- **Clusters tend to be circular:** k-means looks in a circular fashion around each cluster centre and assigns an observation to the closest centre
- **Problems with unequal cluster size:** If some clusters have less samples than others, then k-means tends to add those to the bigger clusters
- **Always finds K clusters** (not unique to k-means)

k-means and circular clusters



Using k-means on the wine dataset

UCI Wine dataset: $K = 3$ classes. Let's see if k-means recovers the classes given only the features/attributes.



Partition around medoids (PAM) or k-medoids

Restrictions of k-means: Features have to be continuous and the ℓ_2 norm has to be used as a distance measure.

Partition around medoids (PAM) or k-medoids

Restrictions of k-means: Features have to be continuous and the ℓ_2 norm has to be used as a distance measure.

Idea: Similar approximation but use general distance measure. Also, use one of the observations as cluster centre (a **medoid**), not the centroid.

Partition around medoids (PAM) or k-medoids

Restrictions of k-means: Features have to be continuous and the ℓ_2 norm has to be used as a distance measure.

Idea: Similar approximation but use general distance measure. Also, use one of the observations as cluster centre (a **medoid**), not the centroid.

Solve

$$\underset{\substack{C \\ l_i \text{ for } 1 \leq i \leq K}}{\arg\min} \sum_{i=1}^{K} n_i \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} D(\mathbf{x}_l, \mathbf{x}_{l_i})$$

Partition around medoids (PAM) or k-medoids

Restrictions of k-means: Features have to be continuous and the ℓ_2 norm has to be used as a distance measure.

Idea: Similar approximation but use general distance measure. Also, use one of the observations as cluster centre (a **medoid**), not the centroid.

Solve

$$\underset{\substack{C \\ l_i \text{ for } 1 \le i \le K}}{\arg\min} \sum_{i=1}^{K} n_i \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} D(\mathbf{x}_l, \mathbf{x}_{l_i})$$

Notation: For observed feature vectors $\mathbf{x}_l$ and $\mathbf{x}_m$ set $\mathbf{D}_{l,m} = D(\mathbf{x}_l, \mathbf{x}_m)$. This results in $\mathbf{D} \in \mathbb{R}^{n \times n}$.

PAM/k-medoids algorithm

Computational procedure:

PAM/k-medoids algorithm

Computational procedure:

1. **Initialize:** Randomly choose K observation indices as cluster centres l_i and set $J_{\max}$ to a positive integer

PAM/k-medoids algorithm

Computational procedure:

1. **Initialize:** Randomly choose K observation indices as cluster centres l_i and set $J_{\max}$ to a positive integer
2. For steps $j = 1, \dots, J_{\max}$

PAM/k-medoids algorithm

Computational procedure:

1. **Initialize:** Randomly choose K observation indices as cluster centres l_i and set $J_{\max}$ to a positive integer
2. For steps $j = 1, \ldots, J_{\max}$
    2.1 **Cluster allocation:** $C(\mathbf{x}_l) = \underset{1 \le i \le K}{\arg\min}\, \mathbf{D}_{l,l_i}$

PAM/k-medoids algorithm

Computational procedure:

1. **Initialize:** Randomly choose K observation indices as cluster centres l_i and set $J_{\max}$ to a positive integer
2. For steps $j = 1, \dots, J_{\max}$
   - 2.1 **Cluster allocation:** $C(\mathbf{x}_l) = \underset{1 \leq i \leq K}{\arg\min}\, \mathbf{D}_{l,l_i}$
   - 2.2 **Cluster centre update:** $l_i = \underset{\substack{1 \leq l \leq n \\ C(\mathbf{x}_l) = i}}{\arg\min} \sum_{C(\mathbf{x}_m)=i} \mathbf{D}_{l,m}$

PAM/k-medoids algorithm

Computational procedure:

1. **Initialize:** Randomly choose K observation indices as cluster centres l_i and set $J_{\max}$ to a positive integer
2. For steps $j = 1, \dots, J_{\max}$
   - 2.1 **Cluster allocation:** $C(\mathbf{x}_l) = \underset{1 \le i \le K}{\arg\min}\, \mathbf{D}_{l,l_i}$
   - 2.2 **Cluster centre update:** $l_i = \underset{\substack{1 \le l \le n \\ C(\mathbf{x}_l)=i}}{\arg\min} \sum_{C(\mathbf{x}_m)=i} \mathbf{D}_{l,m}$
   - 2.3 Stop if clustering C did not change

PAM/k-medoids algorithm

Computational procedure:

1. **Initialize:** Randomly choose K observation indices as cluster centres l_i and set $J_{\max}$ to a positive integer
2. For steps $j = 1, \dots, J_{\max}$
   2.1 **Cluster allocation:** $C(\mathbf{x}_l) = \underset{1 \le i \le K}{\arg\min}\, \mathbf{D}_{l,l_i}$
   2.2 **Cluster centre update:** $l_i = \underset{\substack{1 \le l \le n \\ C(\mathbf{x}_l)=i}}{\arg\min} \sum_{C(\mathbf{x}_m)=i} \mathbf{D}_{l,m}$
   2.3 Stop if clustering C did not change

Computational Complexity: Step 2.2 is now quadratic in n_i instead of linear as in k-means

PAM/k-medoids algorithm

Computational procedure:

1. **Initialize:** Randomly choose K observation indices as cluster centres l_i and set $J_{\max}$ to a positive integer
2. For steps $j = 1, \ldots, J_{\max}$
   2.1 **Cluster allocation:** $C(\mathbf{x}_l) = \underset{1 \le i \le K}{\arg\min}\, \mathbf{D}_{l,l_i}$
   2.2 **Cluster centre update:** $l_i = \underset{\substack{1 \le l \le n \\ C(\mathbf{x}_l) = i}}{\arg\min} \sum_{C(\mathbf{x}_m) = i} \mathbf{D}_{l,m}$
   2.3 Stop if clustering C did not change

Computational Complexity: Step 2.2 is now quadratic in n_i instead of linear as in k-means

Note: All PAM requires is a matrix of distances $\mathbf{D}$ and no additional distance computations are necessary. Very diverse types of features can be used.

Cluster validation and selection of cluster count

Cluster validation

Internal indices

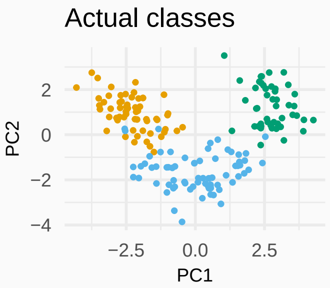
- Focus on between and within cluster scatter
- Aim is to achieve high between cluster scatter and low within cluster scatter

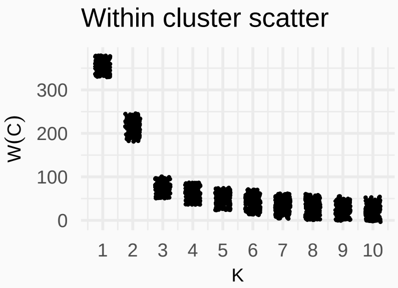
External indices

- Focus on comparison of final clustering with reference classes
- Used to e.g. determine which types of clusters can be found in data, or to evaluate different clustering algorithms on a reference dataset

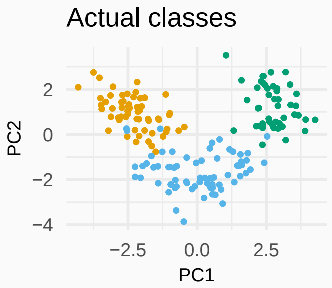
Examples of internal indices

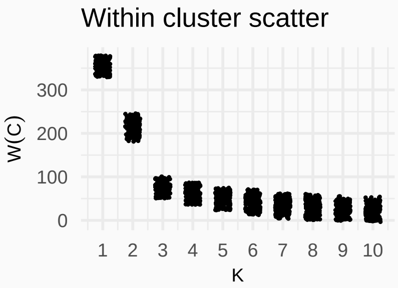
Elbow heuristic for k-means





Elbow heuristic for k-means

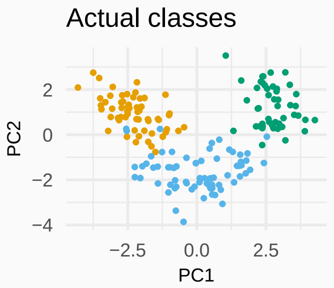


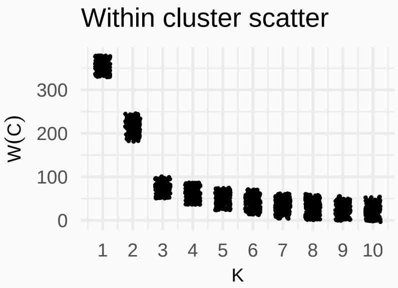


Observations:

- $W(C)$ decreases with cluster count K

Elbow heuristic for k-means

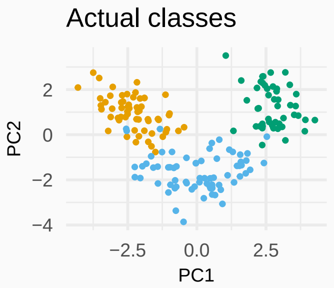


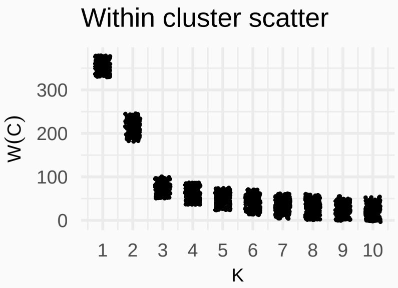


Observations:

- $W(C)$ decreases with cluster count K
- Decreases are less substantial if data does not support more clusters

Elbow heuristic for k-means





Observations:

- $W(C)$ decreases with cluster count K
- Decreases are less substantial if data does not support more clusters
- K is chosen such that **following decreases are substantially smaller**.

Silhouette Width

For every observation $\mathbf{x}_l$ define (with $\mathbf{D}_{l,m} = D(\mathbf{x}_l, \mathbf{x}_m)$)

Silhouette Width

For every observation $\mathbf{x}_l$ define (with $\mathbf{D}_{l,m} = D(\mathbf{x}_l, \mathbf{x}_m)$)

1. **Average distance within cluster:**

$$a_l = \frac{1}{n_{C(\mathbf{x}_l)}} \sum_{C(\mathbf{x}_m)=C(\mathbf{x}_l)} \mathbf{D}_{l,m}$$

Silhouette Width

For every observation $\mathbf{x}_l$ define (with $\mathbf{D}_{l,m} = D(\mathbf{x}_l, \mathbf{x}_m)$)

1. **Average distance within cluster:**

$$a_l = \frac{1}{n_{C(\mathbf{x}_l)}} \sum_{C(\mathbf{x}_m)=C(\mathbf{x}_l)} \mathbf{D}_{l,m}$$

2. **Average distance to nearest cluster:**

$$b_l = \arg\min_{\substack{1\leq i\leq K \\ i\neq C(\mathbf{x}_l)}} \frac{1}{n_i} \sum_{C(\mathbf{x}_m)=i} \mathbf{D}_{l,m}$$

Silhouette Width

For every observation $\mathbf{x}_l$ define (with $\mathbf{D}_{l,m} = D(\mathbf{x}_l, \mathbf{x}_m)$)

1. **Average distance within cluster:**

$$a_l = \frac{1}{n_{C(\mathbf{x}_l)}} \sum_{C(\mathbf{x}_m)=C(\mathbf{x}_l)} \mathbf{D}_{l,m}$$

2. **Average distance to nearest cluster:**

$$b_l = \arg\min_{\substack{1\leq i\leq K \\ i\neq C(\mathbf{x}_l)}} \frac{1}{n_i} \sum_{C(\mathbf{x}_m)=i} \mathbf{D}_{l,m}$$

3. **Silhouette width:** $s_l = \dfrac{b_l - a_l}{\max(a_l, b_l)} \in [-1, 1]$

Silhouette Width

For every observation $\mathbf{x}_l$ define (with $\mathbf{D}_{l,m} = D(\mathbf{x}_l, \mathbf{x}_m)$)

1. **Average distance within cluster:**
$$a_l = \frac{1}{n_{C(\mathbf{x}_l)}} \sum_{C(\mathbf{x}_m)=C(\mathbf{x}_l)} \mathbf{D}_{l,m}$$
2. **Average distance to nearest cluster:**
$$b_l = \underset{\substack{1\leq i\leq K\\ i\neq C(\mathbf{x}_l)}}{\arg\min} \frac{1}{n_i} \sum_{C(\mathbf{x}_m)=i} \mathbf{D}_{l,m}$$
3. **Silhouette width:** $s_l = \dfrac{b_l - a_l}{\max(a_l, b_l)} \in [-1, 1]$

and overall **average silhouette width:** $S = \dfrac{1}{n}\sum_{l=1}^{n} s_l.$

Notes on silhouette width

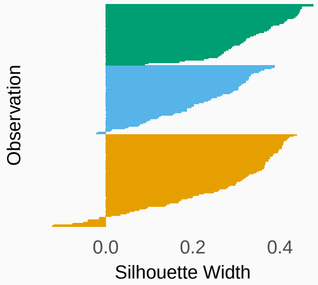
- **Interpretation**
  - Close to 1 when observation is well located inside the cluster and separated from the nearest cluster
  - Close to 0 when observation is between two clusters
  - Negative if observation on average closer to another cluster.
    Warning sign: Hints at which observations should be investigated.
  - Average silhouette width should be maximal for a good clustering

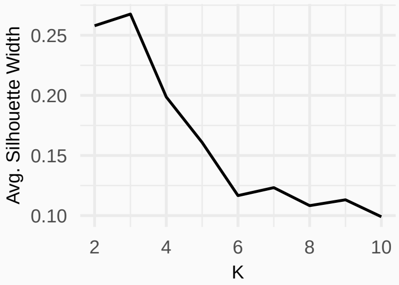
Notes on silhouette width

- **Interpretation**
  - Close to 1 when observation is well located inside the cluster and separated from the nearest cluster
  - Close to 0 when observation is between two clusters
  - Negative if observation on average closer to another cluster. **Warning sign:** Hints at which observations should be investigated.
  - Average silhouette width should be maximal for a good clustering
- **Limitations**
  - Needs at least two clusters
  - Based on the same ideas as PAM/k-medoids and therefore considers clusters to be spherical

Silhouette Width: Example

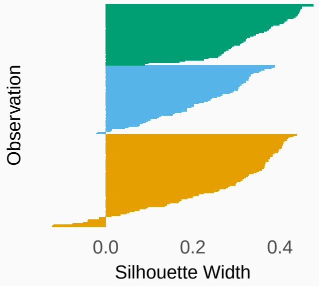
Clustering of the **UCI wine data** using k-medoids with the ℓ_2 metric. Sorted per cluster and arranged in decreasing order of silhouette width.

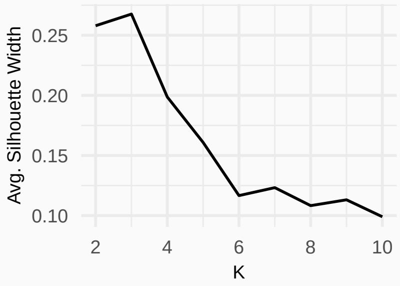




Silhouette Width: Example

Clustering of the **UCI wine data** using k-medoids with the ℓ_2 metric. Sorted per cluster and arranged in decreasing order of silhouette width.

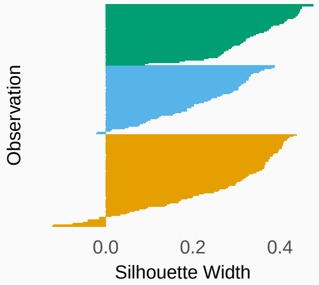


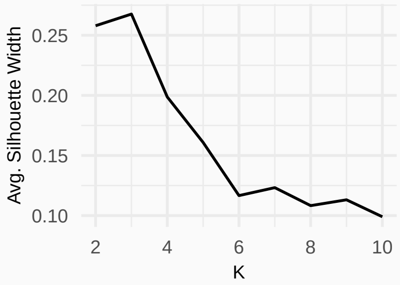


- Silhouette width gives a clear signal that more than three clusters lead to decreasing performance

Silhouette Width: Example

Clustering of the **UCI wine data** using k-medoids with the ℓ_2 metric. Sorted per cluster and arranged in decreasing order of silhouette width.

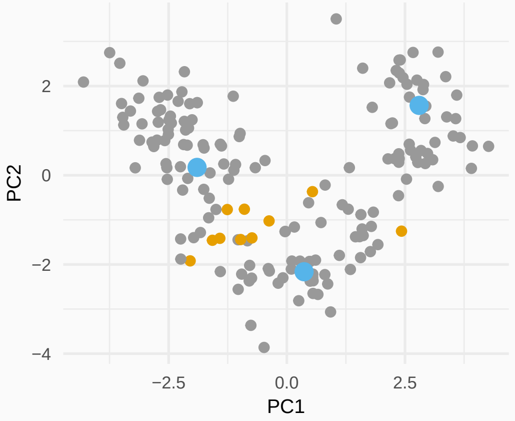


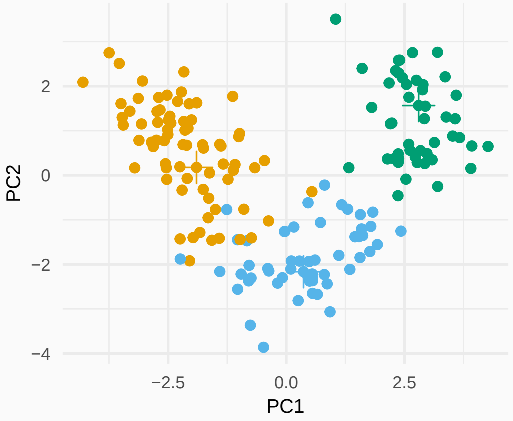


- Silhouette width gives a clear signal that more than three clusters lead to decreasing performance
- However, two and three clusters are indicated of similar quality.

Observations with negative Silhouette width

Observations in orange have negative silhouette width. Cluster medoids are shown in blue.





An example of an external index

Mutual information and entropy

Let C be a clustering for K clusters and c a classification rule for M classes.
Denote $S_i = \{\mathbf{x}_l : C(\mathbf{x}_l) = i\}$, $S^j = \{\mathbf{x}_l : c(\mathbf{x}_l) = j\}$, and $S_i^j = S_i \cap S^j$.

Mutual information and entropy

Let C be a clustering for K clusters and c a classification rule for M classes. Denote $S_i = \{\mathbf{x}_l : C(\mathbf{x}_l) = i\}$, $S^j = \{\mathbf{x}_l : c(\mathbf{x}_l) = j\}$, and $S_i^j = S_i \cap S^j$.

We are interested in how well the two rules agree on a dataset.

Mutual information and entropy

Let C be a clustering for K clusters and c a classification rule for M classes. Denote $S_i = \{\mathbf{x}_l : C(\mathbf{x}_l) = i\}$, $S^j = \{\mathbf{x}_l : c(\mathbf{x}_l) = j\}$, and $S_i^j = S_i \cap S^j$.

We are interested in how well the two rules agree on a dataset.

Mutual Information: Amount of information that can be obtained about one rule by knowing the other rule

$$I(C,c) = \sum_{i=1}^{K} \sum_{j=1}^{M} \mathbb{P}(S_i^j) \log \frac{\mathbb{P}(S_i^j)}{\mathbb{P}(S_i)\,\mathbb{P}(S^j)} \approx \sum_{i=1}^{K} \sum_{j=1}^{M} \frac{|S_i^j|}{n} \log \frac{n|S_i^j|}{|S_i||S^j|}$$

Mutual information and entropy

Let C be a clustering for K clusters and c a classification rule for M classes. Denote $S_i = \{\mathbf{x}_l : C(\mathbf{x}_l) = i\}$, $S^j = \{\mathbf{x}_l : c(\mathbf{x}_l) = j\}$, and $S_i^j = S_i \cap S^j$.

We are interested in how well the two rules agree on a dataset.

Mutual Information: Amount of information that can be obtained about one rule by knowing the other rule

$$I(C,c) = \sum_{i=1}^{K}\sum_{j=1}^{M} \mathbb{P}(S_i^j) \log \frac{\mathbb{P}(S_i^j)}{\mathbb{P}(S_i)\,\mathbb{P}(S^j)} \approx \sum_{i=1}^{K}\sum_{j=1}^{M} \frac{|S_i^j|}{n} \log \frac{n|S_i^j|}{|S_i||S^j|}$$

Entropy: Information present in each rule

$$H(C) = -\sum_{i=1}^{K} \mathbb{P}(S_i) \log \mathbb{P}(S_i) \approx -\sum_{i=1}^{K} \frac{|S_i|}{n} \log \frac{|S_i|}{n}$$

and analogously for c.

Normalised mutual information

Mutual information can be seen as a measure for how much more information about the true classes we obtain by being given the cluster labels.

Normalised mutual information

Mutual information can be seen as a measure for how much more information about the true classes we obtain by being given the cluster labels.

If the clustering is **completely random**, we gain no knowledge, i.e. $I(C, c) = 0$. If the clustering is **perfect**, then mutual information is maximal.

Normalised mutual information

Mutual information can be seen as a measure for how much more information about the true classes we obtain by being given the cluster labels.

If the clustering is **completely random**, we gain no knowledge, i.e. $I(C, c) = 0$. If the clustering is **perfect**, then mutual information is maximal.

However, mutual information is also maximal if $K = n$, i.e. each observation is in its own cluster. Since $H(C)$ is maximal if $K = n$, normalisation can solve this problem.

Normalised mutual information

Mutual information can be seen as a measure for how much more information about the true classes we obtain by being given the cluster labels.

If the clustering is **completely random**, we gain no knowledge, i.e. $I(C, c) = 0$. If the clustering is **perfect**, then mutual information is maximal.

However, mutual information is also maximal if $K = n$, i.e. each observation is in its own cluster. Since $H(C)$ is maximal if $K = n$, normalisation can solve this problem.

Note that $I(C, c) \leq (H(C) + H(c))/2$ which leads to the definition of **normalised mutual information**

$$\mathrm{NMI}(C, c) = \frac{I(C, c)}{(H(C) + H(c))/2} \in [0, 1].$$

Consensus clustering

Consensus clustering

- Any method that comprises many steps is subject to instability since each step is a source of error
- How many features, how many eigenvalues?
- In addition, many clustering methods are quite sensitive to small data perturbations

Consensus clustering

- If you can do things once, you can do it 100 times!
- Add some randomness to the procedure and run it many times
- Retain clusters that are *stable* across multiple runs!

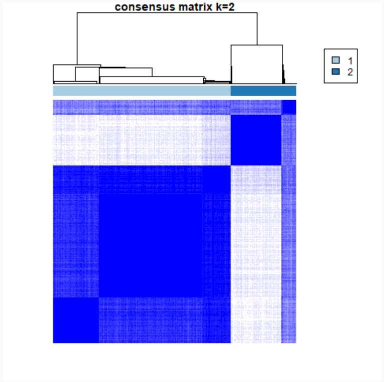
Consensus clustering

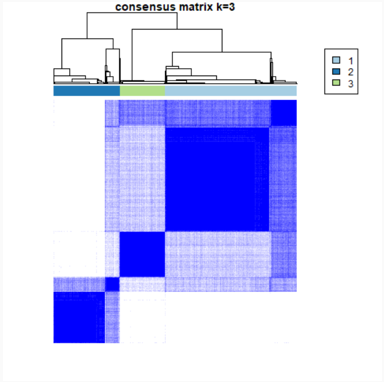
- How add randomness?
- Resampling of observations... but also
- Subset of features
- Subset of features + PCA
- Random projections
- ....

Consensus clustering

- Each run produces a clustering result
- How do we combine these?
- Some methods compare the clusters in terms of overlap
- Other methods use a similar idea to RF clustering: for each pair of objects, count how many times they appear in a cluster together. Use this is a new similarity metric and use e.g. hierarchical clustering (more on thursday) to produce a final result.
- I like the latter approach because it gives you a lot of flexibility in which clustering procedures to compare across runs.

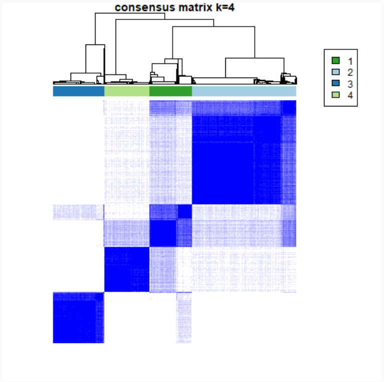
Example of results

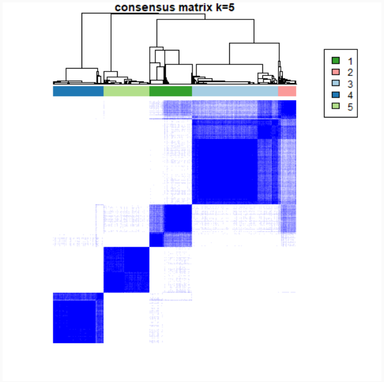




Consensus matrices (i.e. proportion of times clusters are in agreement for all pairs of observations.

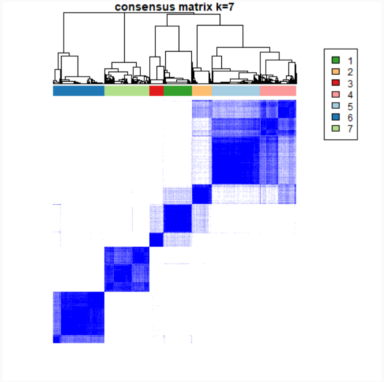
Example of results

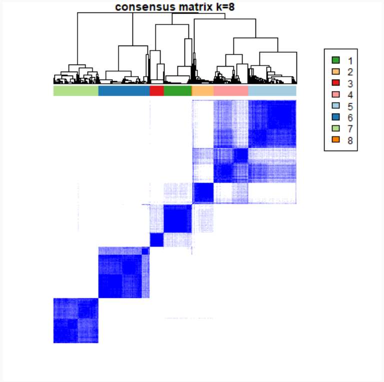




Consensus matrices (i.e. proportion of times clusters are in agreement for all pairs of observations.

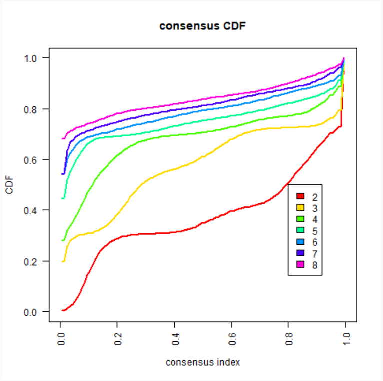
Example of results

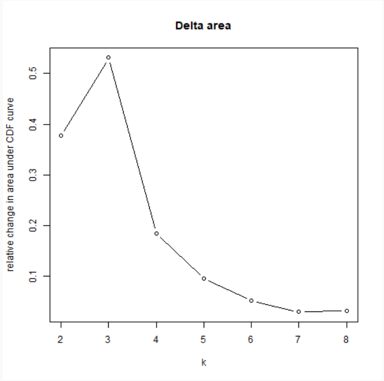




Consensus matrices (i.e. proportion of times clusters are in agreement for all pairs of observations.

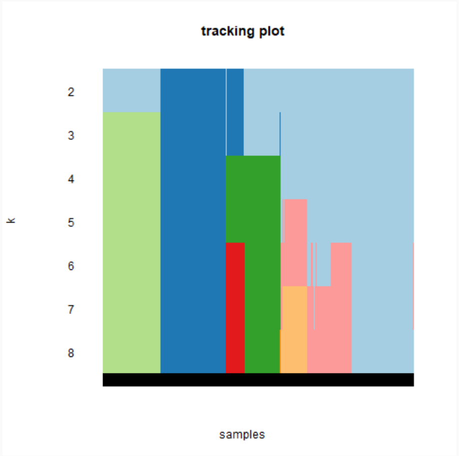
Example of results





CDF of consensus matrices can be used to choose the number of clusters. Check where adding clusters ”stops paying off” - can be assessed by comparing the CDF differences (right panel).

Example of results



It can be interesting to look at how clusters are formed as you increase the number of clusters.

Graphical Lasso

- Remember our discussion about the covariance matrix and the inverse covariance matrix in class
- The *sparse* inverse covariance matrix has non-zero entries where there is a direct correlation between items
- That is, if there is a partial correlation remaining once we account for all other dependencies.
- Can also utilize this for *network modeling*
- Nice visualizations of complex data!
- Related to clustering in the sense that....
- ... observations are represented in a network - neighbors are more similar.
- But also a more complex question - neighbors are close once dependency on other observations taken into account

Graphical Lasso

- Lots of methods for network modeling (Bayesian networks, information theoretic, directed/mechanistic,...)
- Here we will focus on *sparse modeling*
- Assume data comes from a multivariate normal model $N(\mu, \Sigma)$
- The inverse of the covariance matrix Σ, Θ, is called the *precision matrix*

Graphical Lasso

- The inverse of the covariance matrix Σ, Θ, is called the *precision matrix*
- Fact: The precision matrix is non-zero for entry i, j only if the *partial correlation* between i, j is non-zero
- Partial correlation = correlation between i, j once dependency on all other observations accounted for
- $\theta_{i,j} = Cov(X_i, X_j \mid X_k, k \neq i, j)$
- Can compute the partial correlation from residual correlation from regression of i on all other variables and j on all other variables

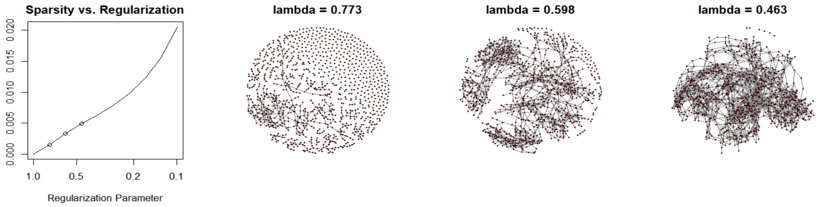
Graphical Lasso

- In practice, can't compute the inverse $\hat{\Theta}$ of the $p \times p$ $\hat{\Sigma}$ if $p > n$
- Sparse modeling to the rescue - which we will learn more about after the easter break.
- However, what we do is essentially regularize the inverse estimates, shrinking some elements toward 0.
- Specifically: We maximize the gaussian log-likeihood with penalty $\lambda \sum_{j<i} |\theta_{i,j}|$
- Methods: gradient based glasso, lasso-regression based neighborhood selection.
- Packages glasso and huge

Graphical Lasso

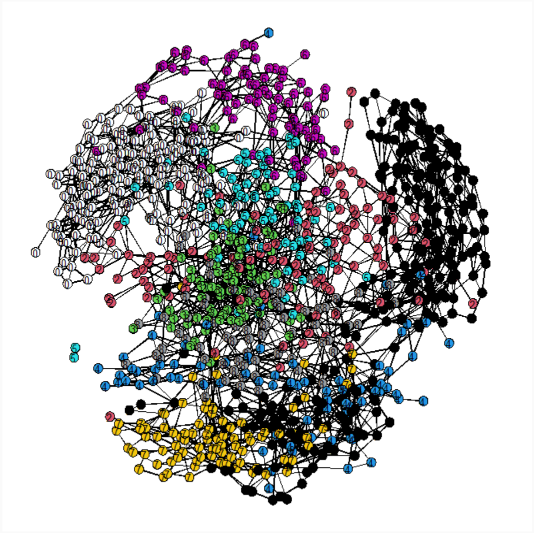
- Does it work?
- Like sparse regression, there are some caveats. Too many highly correlated Xs, we cannot identify the network model.
- Is the data sparse?
- Fixes: randomized lasso. Run glasso many times with random penalties: check how often a graph-link is selected.
- High-dimensional data? First filter. If a set of variables has no correlation with any member of another set exceeding λ, you can run glasso separately on the sets (implemented in huge package).

Examples: Graphical Lasso



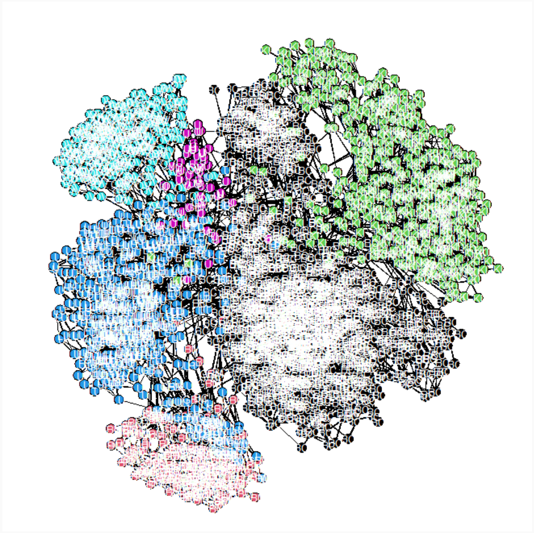
Networks at different levels of sparsity regulation.

Examples: Graphical Lasso



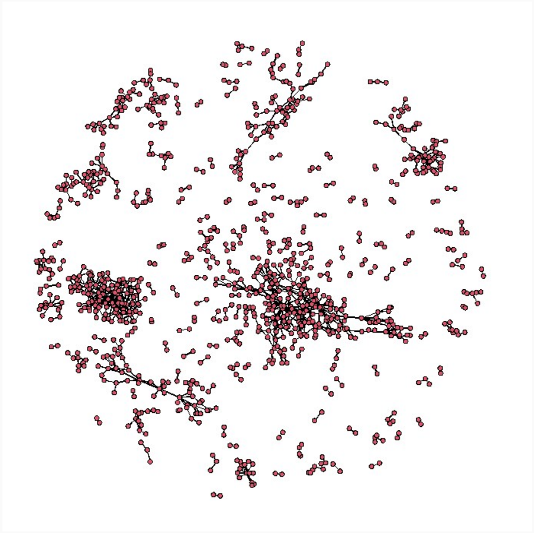
Networks on the digits data

Examples: Graphical Lasso



Networks from the cancer data.

Examples: Graphical Lasso



Networks from the cancer data - but now on the genes instead of the patients.

Take-home message

- Clustering - a more difficult task than classification because there is no ”ground truth”
- Remember - clustering algorithms may make implicit assumptions about the data and what constitutes a good cluster - i.e. kmeans looking for spherical clusters
- Think carefully about scaling/standardizing the data and what impact this may have
- Selecting the number of clusters is another challenge
- Resampling can be used for consensus clustering - and also to select the number of clusters based on stability
- Lots of algorithms out there!!! Good idea to compare a few.

Take-home message

- Clustering is a more challenging problem than classification and needs to answer two questions:
  - What is a cluster?
  - How many clusters are there?
- The clustering algorithm defines what shapes are considered as clusters.
- Clustering results can be validated by external indices and cluster count can be selected through internal indices.