Unsupervised Learning

- No goal class (either Y nor G).
- We are interested in relations in the data:

  Clustering Are the data organized in natural clusters? (Clustering, Segmentation)
  EM algorithm for clustering
  (Dirichlet Process Mixture Models)
  (Spectral Clustering)

  Association Rules Are there some frequent combinations, implication relations? (Market Basket Analysis) *later*

  Other The Elements of Statistical Learning Chapter 14

  SOM Self Organizing Maps

  PCA Principal Component Analysis Linear Algebra; k linear combinations of features minimizing reconstruction error (= first k principal components).
  - Principal Curves and Surfaces, Kernel and Spare Principal Components

  ICA Independent Component Analysis.

Unsupervised Learning

- No goal class (either Y nor G).
- We are interested in relations in the data:

  Clustering Are the data organized in natural clusters? (Clustering, Segmentation)
  EM algorithm for clustering
  (Dirichlet Process Mixture Models)
  (Spectral Clustering)

  Association Rules Are there some frequent combinations, implication relations? (Market Basket Analysis) *later*

  Other The Elements of Statistical Learning Chapter 14

  SOM Self Organizing Maps

  PCA Principal Component Analysis Linear Algebra; k linear combinations of features minimizing reconstruction error (= first k principal components).
  - Principal Curves and Surfaces, Kernel and Spare Principal Components

  ICA Independent Component Analysis.

Unsupervised Learning

- No goal class (either Y nor G).
- We are interested in relations in the data:

  Clustering Are the data organized in natural clusters? (Clustering, Segmentation)
  EM algorithm for clustering
  (Dirichlet Process Mixture Models)
  (Spectral Clustering)

  Association Rules Are there some frequent combinations, implication relations? (Market Basket Analysis) *later*

  Other The Elements of Statistical Learning Chapter 14

  SOM Self Organizing Maps

  PCA Principal Component Analysis Linear Algebra; k linear combinations of features minimizing reconstruction error (= first k principal components).
  - Principal Curves and Surfaces, Kernel and Spare Principal Components

  ICA Independent Component Analysis.

Unsupervised Learning

- No goal class (either Y nor G).
- We are interested in relations in the data:

  Clustering Are the data organized in natural clusters? (Clustering, Segmentation)
  EM algorithm for clustering
  (Dirichlet Process Mixture Models)
  (Spectral Clustering)

  Association Rules Are there some frequent combinations, implication relations? (Market Basket Analysis) *later*

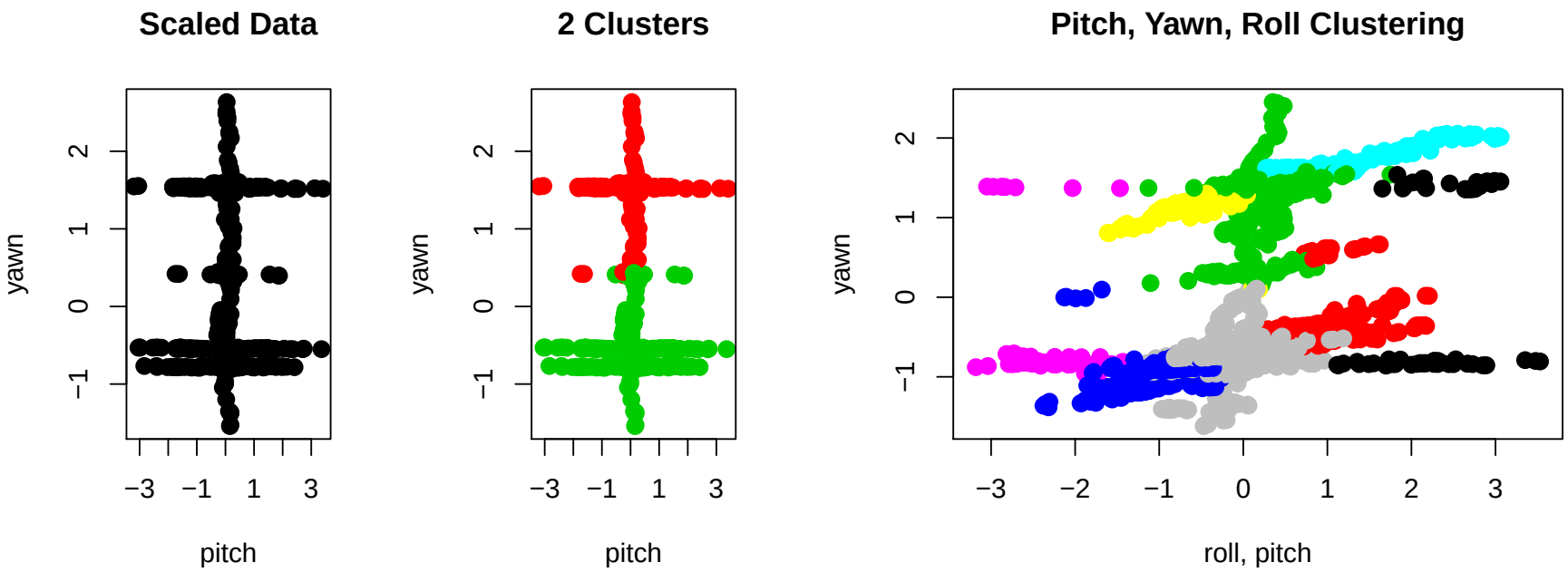
  Other The Elements of Statistical Learning Chapter 14

  SOM Self Organizing Maps

  PCA Principal Component Analysis Linear Algebra; k linear combinations of features minimizing reconstruction error (= first k principal components).
  - Principal Curves and Surfaces, Kernel and Spare Principal Components

  ICA Independent Component Analysis.

Clustering Example



- We set the color of items, no colour in train data.
- We want to assign same color to nearby points.

K – means !

K–means

1: **procedure** $K-\text{MEANS}$:(X data, K the number of clusters)
2: select randomly K centers of clusters μ_k
3: # either random data points or random points in the feature space
4: **repeat**
5: **for** each data record **do**
6: $C(x_i) \leftarrow argmin_{k \in \{1,\ldots,K\}} d(x_i, \mu_k)$
7: **end for**
8: **for** each cluster k **do** # find new centers μ_k
9: $\mu_k = \sum_{x_i : C(x_i)=k} \frac{x_i}{|C(k)|}$.
10: **end for**
11: **until** no chance in assignment
12: **end procedure**

K – means

K–means

The t iterations of K–means algorithm take $O(tkpN)$ time.

- To find global optimum is NP-hard.
- The result depends on initial values.
- May get stuck in local minimum.
- May not be robust to data sampling.
  - We may generate datasets by bootstrap method.
  - The cluster centers found in different dataset may be quite different.

  (for example, different bootstrap samples may give very different clustering results).
- Each record must belong to some cluster. Sensitive to outliers.

K – means

K–means

The t iterations of K–means algorithm take $O(tkpN)$ time.

- To find global optimum is NP-hard.
- The result depends on initial values.
- May get stuck in local minimum.
- May not be robust to data sampling.
  - We may generate datasets by bootstrap method.
  - The cluster centers found in different dataset may be quite different.

  (for example, different bootstrap samples may give very different clustering results).
- Each record must belong to some cluster. Sensitive to outliers.

K – means

K–means

The t iterations of K–means algorithm take $O(tkpN)$ time.

- To find global optimum is NP-hard.
- The result depends on initial values.
- May get stuck in local minimum.
- May not be robust to data sampling.
  - We may generate datasets by bootstrap method.
  - The cluster centers found in different dataset may be quite different.

  (for example, different bootstrap samples may give very different clustering results).
- Each record must belong to some cluster. Sensitive to outliers.

K – means

K–means

The t iterations of K–means algorithm take $O(tkpN)$ time.

- To find global optimum is NP-hard.
- The result depends on initial values.
- May get stuck in local minimum.
- May not be robust to data sampling.
  - We may generate datasets by bootstrap method.
  - The cluster centers found in different dataset may be quite different.

  (for example, different bootstrap samples may give very different clustering results).
- Each record must belong to some cluster. Sensitive to outliers.

K – means

K–means

The t iterations of K–means algorithm take $O(tkpN)$ time.

- To find global optimum is NP-hard.
- The result depends on initial values.
- May get stuck in local minimum.
- May not be robust to data sampling.
  - We may generate datasets by bootstrap method.
  - The cluster centers found in different dataset may be quite different.

  (for example, different bootstrap samples may give very different clustering results).
- Each record must belong to some cluster. Sensitive to outliers.

K – means

K–means

The t iterations of K–means algorithm take $O(tkpN)$ time.

- To find global optimum is NP-hard.
- The result depends on initial values.
- May get stuck in local minimum.
- May not be robust to data sampling.
  - We may generate datasets by bootstrap method.
  - The cluster centers found in different dataset may be quite different.

  (for example, different bootstrap samples may give very different clustering results).
- Each record must belong to some cluster. Sensitive to outliers.

K – means

K–means

The t iterations of K–means algorithm take $O(tkpN)$ time.

- To find global optimum is NP-hard.
- The result depends on initial values.
- May get stuck in local minimum.
- May not be robust to data sampling.
  - We may generate datasets by bootstrap method.
  - The cluster centers found in different dataset may be quite different.

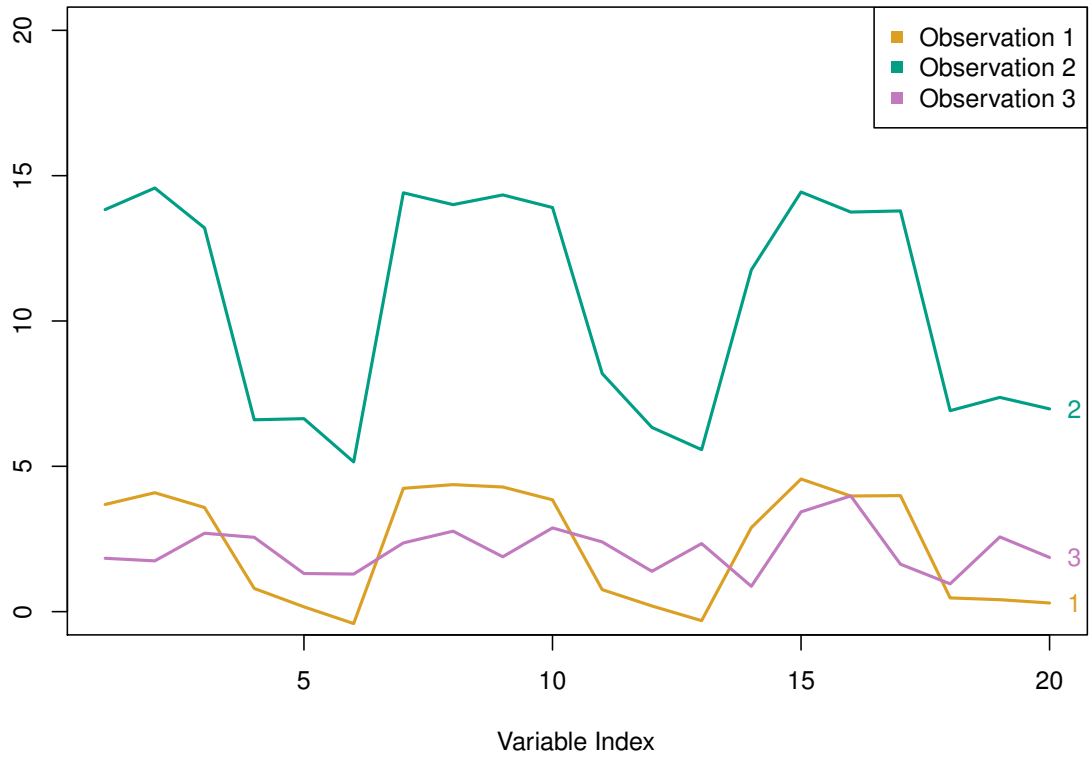
  (for example, different bootstrap samples may give very different clustering results).
- Each record must belong to some cluster. Sensitive to outliers.

Distance measures

the most common distance measures:

Euclidian	$d(x_i, x_j) = \sqrt{\sum_{r=1}^{p}(x_{ir} - x_{jr})^2}$
Hamming (Manhattan)	$d(x_i, x_j) = \sum_{r=1}^{p} \lvert x_{ir} - x_{jr} \rvert$
overlap (překrytí) categorical variables	$d(x_i, x_j) = \sum_{r=1}^{p} I(x_{ir} \neq x_{jr})$
cosine similarity	$s(x_i, x_j) = \frac{\sum_{r=1}^{p}(x_{ir} \cdot x_{jr})}{\sqrt{\sum_{r=1}^{p}(x_{jr} \cdot x_{jr}) \cdot \sum_{r=1}^{p}(x_{ir} \cdot x_{ir})}}$
cosine distance	$d(x_i, x_j) = 1 - \frac{\sum_{r=1}^{p}(x_{ir} \cdot x_{jr})}{\sqrt{\sum_{r=1}^{p}(x_{jr} \cdot x_{jr}) \cdot \sum_{r=1}^{p}(x_{ir} \cdot x_{ir})}}$

Other Distance Measures



Correlation Proximity

- Euclidian distance: Observations 1 and 3 are close.
- Correlation distance: 1 and 2 look very similar.

$$\rho_{X,Y} = corr(X, Y) = \frac{cov(X, Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

Distance – key issue, application dependent

- The result depends on the choice of distance measure $d(x_i, \mu_k)$.
- The choice is application dependent.
- Scaling of the data is recommended.
- Weights for equally important attributes j are: $w_j = \frac{1}{\hat{d}_j}$ where

$$\hat{d}_j = \frac{1}{N^2} \sum_{i_1=1}^{N} \sum_{i_2=1}^{N} d_j(x_{i_1}, x_{i_2}) = \frac{1}{N^2} \sum_{i_1=1}^{N} \sum_{i_2=1}^{N} (x_{i_1}[j] - x_{i_2}[j])^2$$

- Total distance as a weighted sum of attribute distances.
- Distance may be specified directly by a symmetric matrix, 0 at the diagonal, should fulfill triangle inequality

$$d(x_i, x_\ell) \leq d(x_i, x_r) + d(x_r, x_\ell).$$

Distance – key issue, application dependent

- The result depends on the choice of distance measure $d(x_i, \mu_k)$.
- The choice is application dependent.
- Scaling of the data is recommended.
- Weights for equally important attributes j are: $w_j = \frac{1}{\hat{d}_j}$ where

$$\hat{d}_j = \frac{1}{N^2} \sum_{i_1=1}^{N} \sum_{i_2=1}^{N} d_j(x_{i_1}, x_{i_2}) = \frac{1}{N^2} \sum_{i_1=1}^{N} \sum_{i_2=1}^{N} (x_{i_1}[j] - x_{i_2}[j])^2$$

- Total distance as a weighted sum of attribute distances.
- Distance may be specified directly by a symmetric matrix, 0 at the diagonal, should fulfill triangle inequality

$$d(x_i, x_\ell) \leq d(x_i, x_r) + d(x_r, x_\ell).$$

Distance – key issue, application dependent

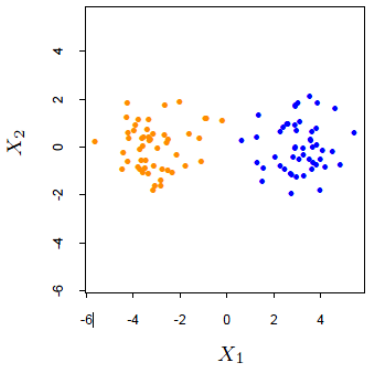
- The result depends on the choice of distance measure $d(x_i, \mu_k)$.
- The choice is application dependent.
- Scaling of the data is recommended.
- Weights for equally important attributes j are: $w_j = \frac{1}{\hat{d}_j}$ where

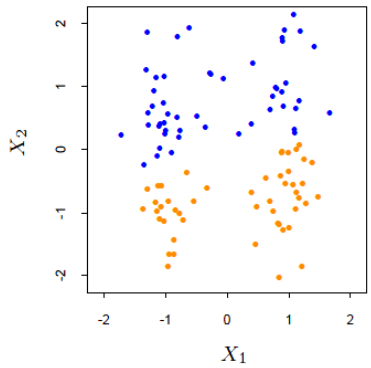
$$\hat{d}_j = \frac{1}{N^2} \sum_{i_1=1}^{N} \sum_{i_2=1}^{N} d_j(x_{i_1}, x_{i_2}) = \frac{1}{N^2} \sum_{i_1=1}^{N} \sum_{i_2=1}^{N} (x_{i_1}[j] - x_{i_2}[j])^2$$

- Total distance as a weighted sum of attribute distances.
- Distance may be specified directly by a symmetric matrix, 0 at the diagonal, should fulfill triangle inequality

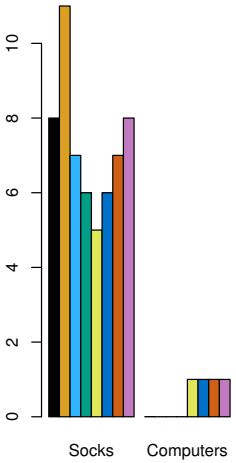
$$d(x_i, x_\ell) \leq d(x_i, x_r) + d(x_r, x_\ell).$$

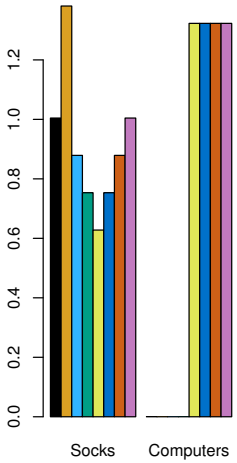
Alternative Ideas

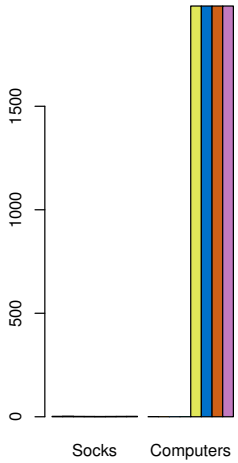




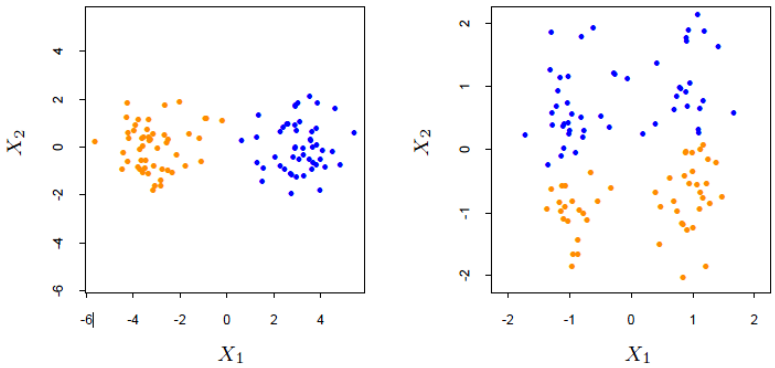
- Scaling may remove natural clusters
- Weighting Attributes
  - Consider internet shop offering socks and computers.
  - Compare: number of sales, standardized data, $



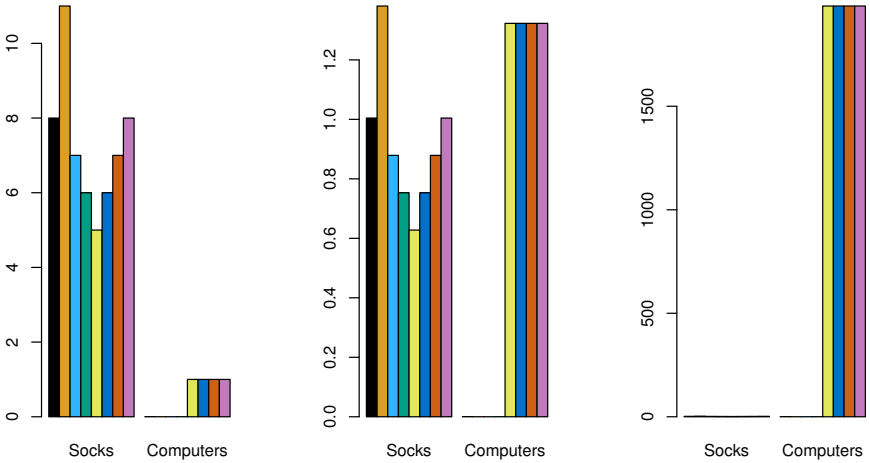




Alternative Ideas



- Scaling may remove natural clusters
- Weighting Attributes
  - Consider internet shop offering socks and computers.
  - Compare: number of sales, standardized data, $



Number of Clusters

- We may focus on the **Within cluster variation** measure:

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} d(x_i, x_{i'})$$

- Notice that $W(C)$ is decreasing also for uniformly distributed data.
- We look for small drop of $W(C)$ as a function of K or maximal difference between $W(C)$ on our data and on the uniform data.
- **Total** cluster variation is the sum of **between** cluster variation and **within** cluster variation

$$T(C) = \frac{1}{2} \sum_{i,i'=1}^{N} d(x_i, x_{i'}) = W(C) + B(C)$$

$$= \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \Big(\sum_{C(i')=k} d(x_i, x_{i'}) \Big) + \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \Big(\sum_{C(i')\neq k} d(x_i, x_{i'}) \Big)$$

Number of Clusters

- We may focus on the **Within cluster variation** measure:

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} d(x_i, x_{i'})$$

- Notice that $W(C)$ is decreasing also for uniformly distributed data.
- We look for small drop of $W(C)$ as a function of K or maximal difference between $W(C)$ on our data and on the uniform data.
- **Total** cluster variation is the sum of **between** cluster variation and **within** cluster variation

$$\begin{aligned} T(C) &= \frac{1}{2} \sum_{i,i'=1}^{N} d(x_i, x_{i'}) = W(C) + B(C) \\ &= \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \Big(\sum_{C(i')=k} d(x_i, x_{i'}) \Big) + \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \Big(\sum_{C(i')\neq k} d(x_i, x_{i'}) \Big) \end{aligned}$$

Number of Clusters

- We may focus on the **Within cluster variation** measure:

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} d(x_i, x_{i'})$$

- Notice that $W(C)$ is decreasing also for uniformly distributed data.
- We look for small drop of $W(C)$ as a function of K or maximal difference between $W(C)$ on our data and on the uniform data.
- **Total** cluster variation is the sum of **between** cluster variation and **within** cluster variation

$$T(C) = \frac{1}{2} \sum_{i,i'=1}^{N} d(x_i, x_{i'}) = W(C) + B(C)$$

$$= \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \left(\sum_{C(i')=k} d(x_i, x_{i'}) \right) + \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \left(\sum_{C(i')\neq k} d(x_i, x_{i'}) \right)$$

Number of Clusters

- We may focus on the **Within cluster variation** measure:

$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')=k} d(x_i, x_{i'})$$

- Notice that $W(C)$ is decreasing also for uniformly distributed data.
- We look for small drop of $W(C)$ as a function of K or maximal difference between $W(C)$ on our data and on the uniform data.
- **Total** cluster variation is the sum of **between** cluster variation and **within** cluster variation

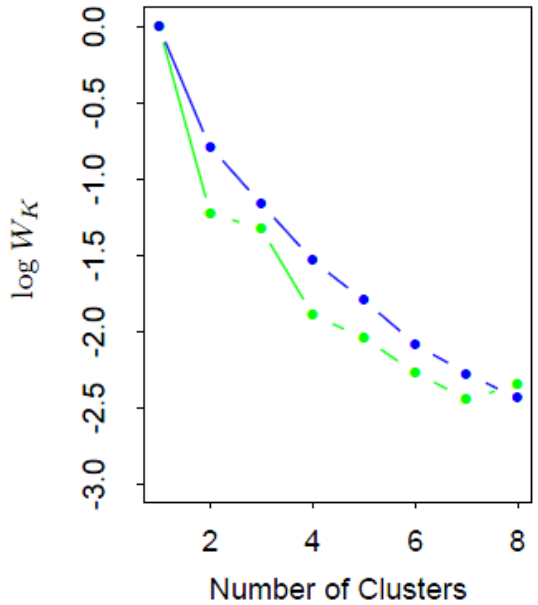
$$\begin{aligned} T(C) &= \frac{1}{2}\sum_{i,i'=1}^{N} d(x_i, x_{i'}) = W(C) + B(C) \\ &= \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\Big(\sum_{C(i')=k} d(x_i, x_{i'})\Big) + \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\Big(\sum_{C(i')\neq k} d(x_i, x_{i'})\Big) \end{aligned}$$

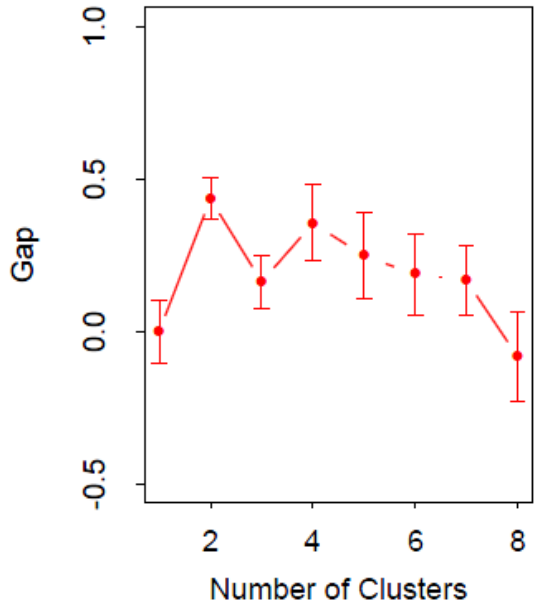
GAP function for Number of Clusters

- denote W_k the expected W for uniformly distributed data and k clusters, the average over 20 runs
- GAP is expected $log(W_k)$ minus observed $log(W(k))$

$$K^* = argmin\{k|G(k) \geq G(k+1) - s'_{k+1}\}$$

$$s'_k = s_k\sqrt{1+\frac{1}{20}} \text{ where } s_k \text{ is the standard deviation of } log(W_k)$$



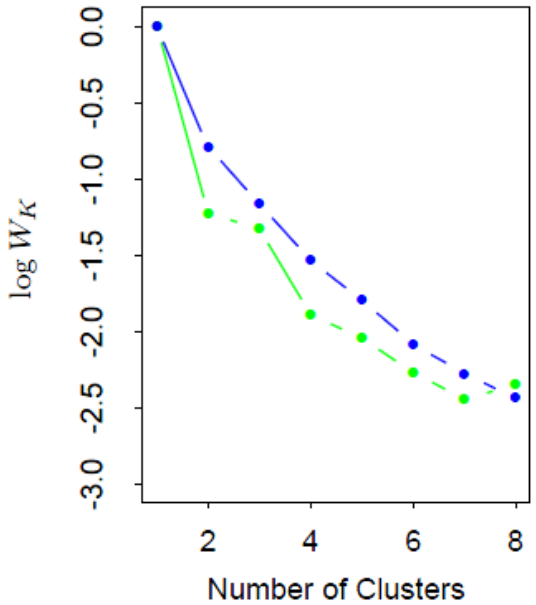


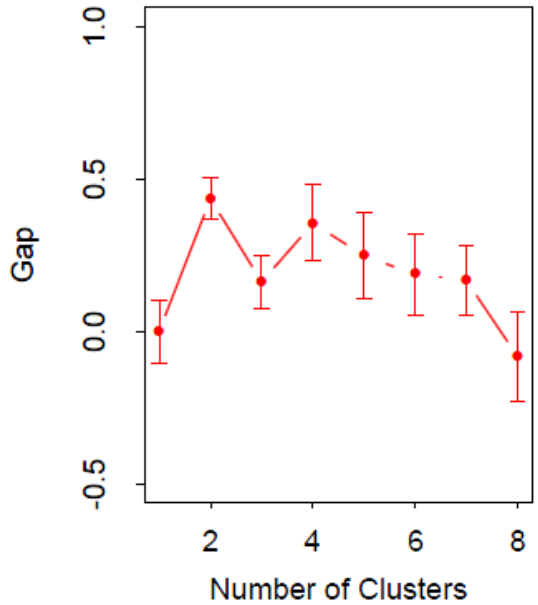
GAP function for Number of Clusters

- denote W_k the expected W for uniformly distributed data and k clusters, the average over 20 runs
- GAP is expected $log(W_k)$ minus observed $log(W(k))$

$$K^* = argmin\{k|G(k) \geq G(k+1) - s'_{k+1}\}$$

$$s'_k = s_k\sqrt{1+\frac{1}{20}} \text{ where } s_k \text{ is the standard deviation of } log(W_k)$$





Silhouette

For each data sample x_i we define

- $a(i) = \frac{1}{|C_i|-1} \sum_{j \in C_i, i \neq j} d(i,j)$ if $|C_i| > 1$
- $b(i) = min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$

Definition (Silhouette)

Silhouette s is defined

- $s(i) = \frac{b(i)-a(i)}{max\{a(i),b(i)\}}$ if $|C_i| > 1$
- $s(i) = 0$ for $|C_i| = 1$.

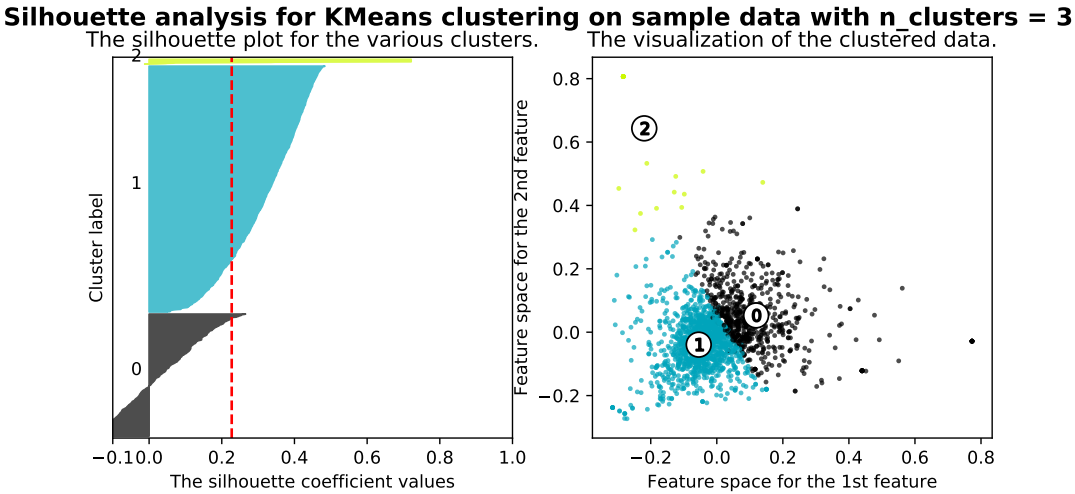
Optimal number of clusters k may be selected by the SC.

Definition (Silhouette Score)

The Silhouette score is $\frac{1}{N} \sum_i^N s(i)$.

Silhouette is always between

- $-1 \leq s(i) \leq 1$.



Note: One cluster $(-1,1),(1,1)$, other cluster $(0,-1.2),(0,-1.1)$, the point $(0,0)$ is assigned to the first cluster but has a negative silhouette. https://stackoverflow.com/a/66751204

Silhouette

For each data sample x_i we define

- $a(i) = \frac{1}{|C_i|-1} \sum_{j \in C_i, i \neq j} d(i,j)$ if $|C_i| > 1$
- $b(i) = min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$

Definition (Silhouette)

Silhouette s is defined

- $s(i) = \frac{b(i)-a(i)}{max\{a(i),b(i)\}}$ if $|C_i| > 1$
- $s(i) = 0$ for $|C_i| = 1$.

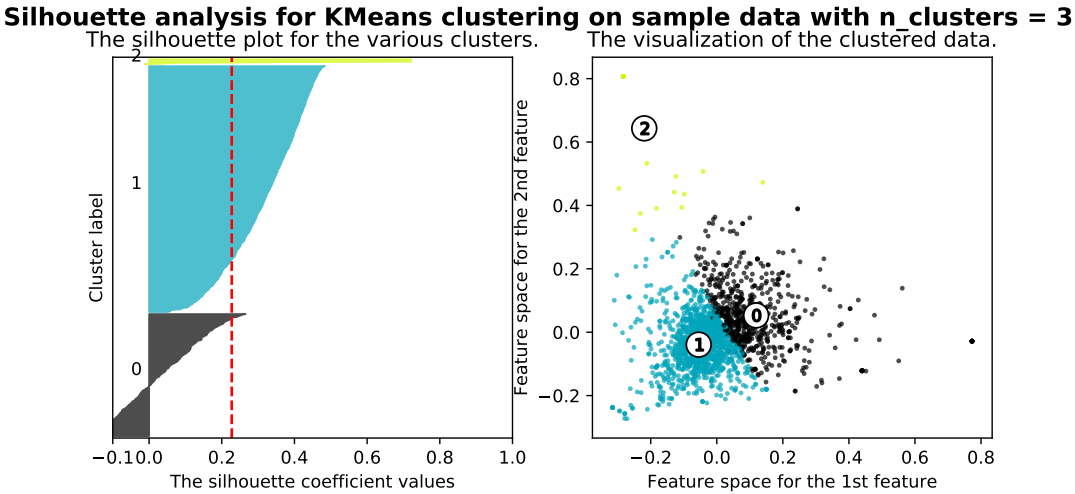
Optimal number of clusters k may be selected by the SC.

Definition (Silhouette Score)

The Silhouette score is $\frac{1}{N} \sum_i^N s(i)$.

Silhouette is always between

- $-1 \leq s(i) \leq 1$.



Note: One cluster $(-1,1),(1,1)$, other cluster $(0,-1.2),(0,-1.1)$, the point $(0,0)$ is assigned to the first cluster but has a negative silhouette. https://stackoverflow.com/a/66751204

Silhouette

For each data sample x_i we define

- $a(i) = \frac{1}{|C_i|-1} \sum_{j \in C_i, i \neq j} d(i,j)$ if $|C_i| > 1$
- $b(i) = min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$

Definition (Silhouette)

Silhouette s is defined

- $s(i) = \frac{b(i)-a(i)}{max\{a(i),b(i)\}}$ if $|C_i| > 1$
- $s(i) = 0$ for $|C_i| = 1$.

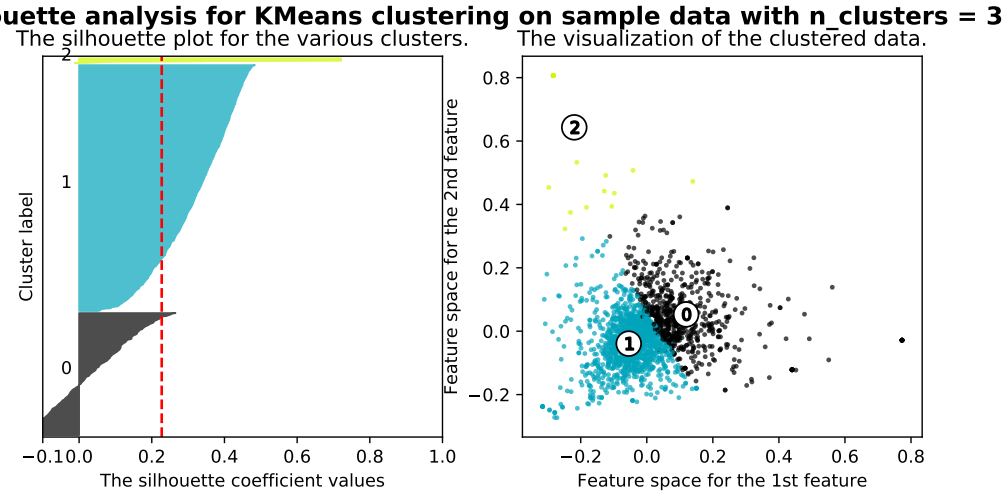
Optimal number of clusters k may be selected by the SC.

Definition (Silhouette Score)

The Silhouette score is $\frac{1}{N} \sum_i^N s(i)$.

Silhouette is always between

- $-1 \leq s(i) \leq 1$.



Note: One cluster $(-1,1),(1,1)$, other cluster $(0,-1.2),(0,-1.1)$, the point $(0,0)$ is assigned to the first cluster but has a negative silhouette. https://stackoverflow.com/a/66751204

Silhouette

For each data sample x_i we define

- $a(i) = \frac{1}{|C_i|-1} \sum_{j \in C_i, i \neq j} d(i,j)$ if $|C_i| > 1$
- $b(i) = min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$

Optimal number of clusters k may be selected by the SC.

Definition (Silhouette)

Silhouette s is defined

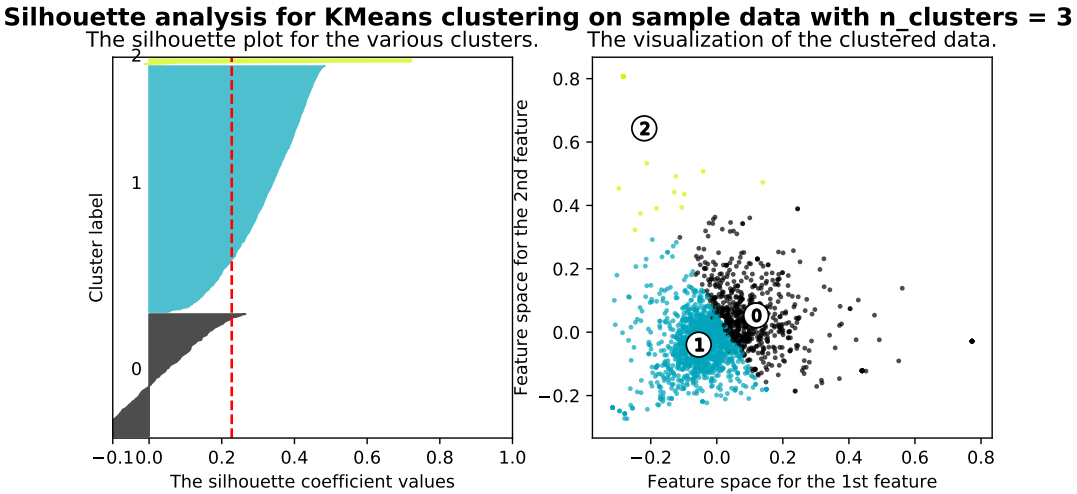
- $s(i) = \frac{b(i)-a(i)}{max\{a(i),b(i)\}}$ if $|C_i| > 1$
- $s(i) = 0$ for $|C_i| = 1$.

Definition (Silhouette Score)

The Silhouette score is $\frac{1}{N} \sum_i^N s(i)$.

Silhouette is always between

- $-1 \leq s(i) \leq 1$.



Note: One cluster $(-1,1),(1,1)$, other cluster $(0,-1.2),(0,-1.1)$, the point $(0,0)$ is assigned to the first cluster but has a negative silhouette. https://stackoverflow.com/a/66751204

Country Similarity Example

- Data from a political science survey: values are average pairwise dissimilarities of countries from a questionnaire given to political science students.

	BEL	BRA	CHI	CUB	EGY	FRA	IND	ISR	USA	USS	YUG
BRA	5.58										
CHI	7.00	6.50									
CUB	7.08	7.00	3.83								
EGY	4.83	5.08	8.17	5.83							
FRA	2.17	5.75	6.67	6.92	4.92						
IND	6.42	5.00	5.58	6.00	4.67	6.42					
ISR	3.42	5.50	6.42	6.42	5.00	3.92	6.17				
USA	2.50	4.92	6.25	7.33	4.50	2.25	6.33	2.75			
USS	6.08	6.67	4.25	2.67	6.00	6.17	6.17	6.92	6.17		
YUG	5.25	6.83	4.50	3.75	5.75	5.42	6.08	5.83	6.67	3.67	
ZAI	4.75	3.00	6.08	6.67	5.00	5.58	4.83	6.17	5.67	6.50	6.92

K–medoids

1: **procedure** $K-$MEDOIDS:(X data, K the number of clusters)
2: select randomly K data samples to be centroids of clusters
3: **repeat**
4: **for** each data record **do**
5: assign to the closest cluster
6: **end for**
7: **for** each cluster k **do** # find new centroids $i_k^* \in C_k$
8: $i_k^* \leftarrow argmin_{\{i:C(i)=k\}} \sum_{C(i')=k} d(x_i, x_{i'})$
9: **end for**
10: **until** no chance in assignment
11: **end procedure**

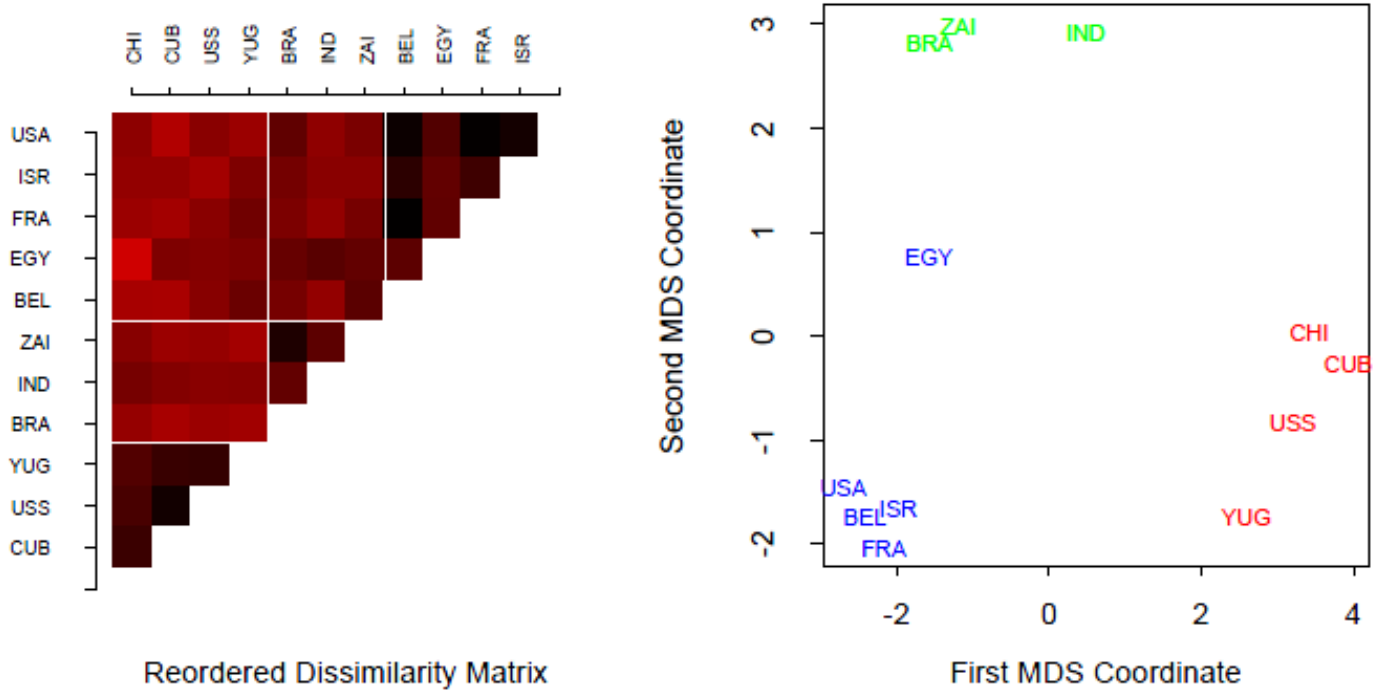
- To find a centroid requires quadratic time compared to linear k–means.
- We may use any distance, for example number of differences in binary attributes.

Complexity

The t iterations of K–medoids take $O(tkpN^2)$.

Clusters of Countries

- Survey of country dissimilarities.
- Left: dissimilarities
  - Reordered and blocked according to 3-medoid clustering.
  - Heat map is coded from most similar (dark red) to least similar (bright red).
- Right: Two-dimensional multidimensional scaling plot
  - with 3-medoid clusters indicated by different colors.



Multidimensional Scaling

- The right figure on previous slide was done by Multidimesional scaling.
- We know only distances of countries, not a metric space.
- We try to keep proximity of countries (*least squares scaling*).
- We choose the number of dimensions p.

Definition (Multidimensional Scaling)

For a given data $x_1, \ldots, x_N$ with their distance matrix d, we search $(z_1, \ldots, z_N) \in \mathbb{R}^p$ projections of data minimizing stress function

$$S_D(z_1, \ldots, z_N) = \left[\sum_{i \neq \ell} (d[x_i, x_\ell] - \|z_i - z_\ell\|)^2 \right]^{\frac{1}{2}}.$$

- It is evaluated gradiently.
- Note: Spectral clustering.

Hierarchical clustering – Bottom Up

Start with each data sample in its own cluster. Iteratively join two nearest clusters.

Measures for join

- closest points (**single linkage**)
- maximally distant points (**complete linkage**)
- average linkage, $d_{GA}(C_A, C_B) = \frac{1}{|C_A||C_B|} \sum_{x \in C_A, y \in C_B} d(x, y)$
- Ward distance minimizes the sum of squared differences within all clusters.

$$Ward(C_A, C_B) = \sum_{x \in C_A} d(x, \mu_{A \cup B})^2 + \sum_{y \in C_B} d(y, \mu_{A \cup B})^2 - \sum_{x \in C_A} d(x, \mu_A)^2 - \sum_{y \in C_B} d(y, \mu_B)^2$$

$$= \frac{|C_A||C_B|}{|C_A| + |C_B|} d(\mu_A, \mu_B)^2$$

Hierarchical clustering – Bottom Up

Start with each data sample in its own cluster. Iteratively join two nearest clusters.
Measures for join

- closest points (**single linkage**)
- maximally distant points (**complete linkage**)
- **average linkage**, $d_{GA}(C_A, C_B) = \frac{1}{|C_A| \cdot |C_B|} \sum_{x_i \in C_A, x_j \in C_B} d(x_i, x_j)$
- **Ward distance** minimizes the sum of squared differences within all clusters.

$$
\begin{aligned}
Ward(C_A, C_B) &= \sum_{i \in C_A \cup C_B} d(x_i, \mu_{A \cup B})^2 - \sum_{i \in C_A} d(x_i, \mu_A)^2 - \sum_{i \in C_B} d(x_i, \mu_B)^2 \\
&= \frac{|C_A| \cdot |C_B|}{|C_A| + |C_B|} \cdot d(\mu_A, \mu_B)^2
\end{aligned}
$$

where μ_i are the centers of clusters (A, B and joined cluster).
Ward is a variance-minimizing approach and in this respect is similar to the k-means objective function but tackled with an agglomerative hierarchical approach.

Hierarchical clustering – Bottom Up

Start with each data sample in its own cluster. Iteratively join two nearest clusters.
Measures for join

- closest points (**single linkage**)
- maximally distant points (**complete linkage**)
- **average linkage**, $d_{GA}(C_A, C_B) = \frac{1}{|C_A| \cdot |C_B|} \sum_{x_i \in C_A, x_j \in C_B} d(x_i, x_j)$
- **Ward distance** minimizes the sum of squared differences within all clusters.

$$
\begin{aligned}
Ward(C_A, C_B) &= \sum_{i \in C_A \cup C_B} d(x_i, \mu_{A \cup B})^2 - \sum_{i \in C_A} d(x_i, \mu_A)^2 - \sum_{i \in C_B} d(x_i, \mu_B)^2 \\
&= \frac{|C_A| \cdot |C_B|}{|C_A| + |C_B|} \cdot d(\mu_A, \mu_B)^2
\end{aligned}
$$

- where μ are the centers of clusters A, B and joined cluster $A \cup B$
- It is a variance-minimizing approach and in this sense is similar to the k-means objective function but tackled with an agglomerative hierarchical approach

Hierarchical clustering – Bottom Up

Start with each data sample in its own cluster. Iteratively join two nearest clusters. Measures for join

- closest points (**single linkage**)
- maximally distant points (**complete linkage**)
- **average linkage**, $d_{GA}(C_A, C_B) = \frac{1}{|C_A| \cdot |C_B|} \sum_{x_i \in C_A, x_j \in C_B} d(x_i, x_j)$
- **Ward distance** minimizes the sum of squared differences within all clusters.

$$Ward(C_A, C_B) = \sum_{i \in C_A \cup C_B} d(x_i, \mu_{A \cup B})^2 - \sum_{i \in C_A} d(x_i, \mu_A)^2 - \sum_{i \in C_B} d(x_i, \mu_B)^2$$

$$= \frac{|C_A| \cdot |C_B|}{|C_A| + |C_B|} \cdot d(\mu_A, \mu_B)^2$$

  - where μ are the centers of clusters (A, B and joined cluster).
  - It is a variance-minimizing approach and in this sense is similar to the k-means objective function but tackled with an agglomerative hierarchical approach.

Hierarchical clustering – Bottom Up

Start with each data sample in its own cluster. Iteratively join two nearest clusters.
Measures for join

- closest points (**single linkage**)
- maximally distant points (**complete linkage**)
- **average linkage**, $d_{GA}(C_A, C_B) = \frac{1}{|C_A| \cdot |C_B|} \sum_{x_i \in C_A, x_j \in C_B} d(x_i, x_j)$
- **Ward distance** minimizes the sum of squared differences within all clusters.

$$\begin{aligned} Ward(C_A, C_B) &= \sum_{i \in C_A \cup C_B} d(x_i, \mu_{A \cup B})^2 - \sum_{i \in C_A} d(x_i, \mu_A)^2 - \sum_{i \in C_B} d(x_i, \mu_B)^2 \\ &= \frac{|C_A| \cdot |C_B|}{|C_A| + |C_B|} \cdot d(\mu_A, \mu_B)^2 \end{aligned}$$

- where μ are the centers of clusters (A, B and joined cluster).
- It is a variance-minimizing approach and in this sense is similar to the k-means objective function but tackled with an agglomerative hierarchical approach.

Hierarchical clustering – Bottom Up

Start with each data sample in its own cluster. Iteratively join two nearest clusters.
Measures for join

- closest points (**single linkage**)
- maximally distant points (**complete linkage**)
- **average linkage**, $d_{GA}(C_A, C_B) = \frac{1}{|C_A| \cdot |C_B|} \sum_{x_i \in C_A, x_j \in C_B} d(x_i, x_j)$
- **Ward distance** minimizes the sum of squared differences within all clusters.

$$\begin{aligned} Ward(C_A, C_B) &= \sum_{i \in C_A \cup C_B} d(x_i, \mu_{A \cup B})^2 - \sum_{i \in C_A} d(x_i, \mu_A)^2 - \sum_{i \in C_B} d(x_i, \mu_B)^2 \\ &= \frac{|C_A| \cdot |C_B|}{|C_A| + |C_B|} \cdot d(\mu_A, \mu_B)^2 \end{aligned}$$

- where μ are the centers of clusters (A, B and joined cluster).
- It is a variance-minimizing approach and in this sense is similar to the k-means objective function but tackled with an agglomerative hierarchical approach.

Hierarchical clustering – Bottom Up

Start with each data sample in its own cluster. Iteratively join two nearest clusters.
Measures for join

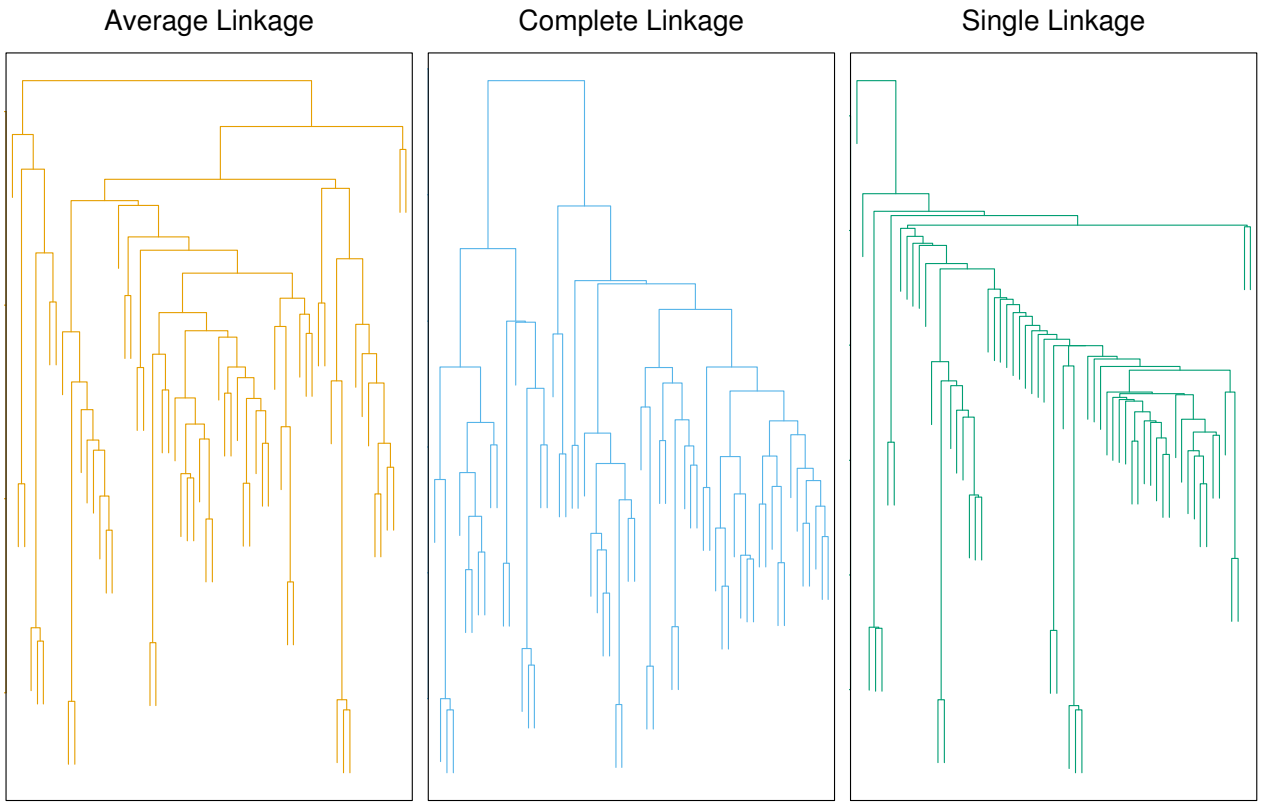
- closest points (**single linkage**)
- maximally distant points (**complete linkage**)
- **average linkage**, $d_{GA}(C_A, C_B) = \frac{1}{|C_A| \cdot |C_B|} \sum_{x_i \in C_A, x_j \in C_B} d(x_i, x_j)$
- **Ward distance** minimizes the sum of squared differences within all clusters.

$$\begin{aligned} Ward(C_A, C_B) &= \sum_{i \in C_A \cup C_B} d(x_i, \mu_{A \cup B})^2 - \sum_{i \in C_A} d(x_i, \mu_A)^2 - \sum_{i \in C_B} d(x_i, \mu_B)^2 \\ &= \frac{|C_A| \cdot |C_B|}{|C_A| + |C_B|} \cdot d(\mu_A, \mu_B)^2 \end{aligned}$$

- where μ are the centers of clusters (A, B and joined cluster).
- It is a variance-minimizing approach and in this sense is similar to the k-means objective function but tackled with an agglomerative hierarchical approach.

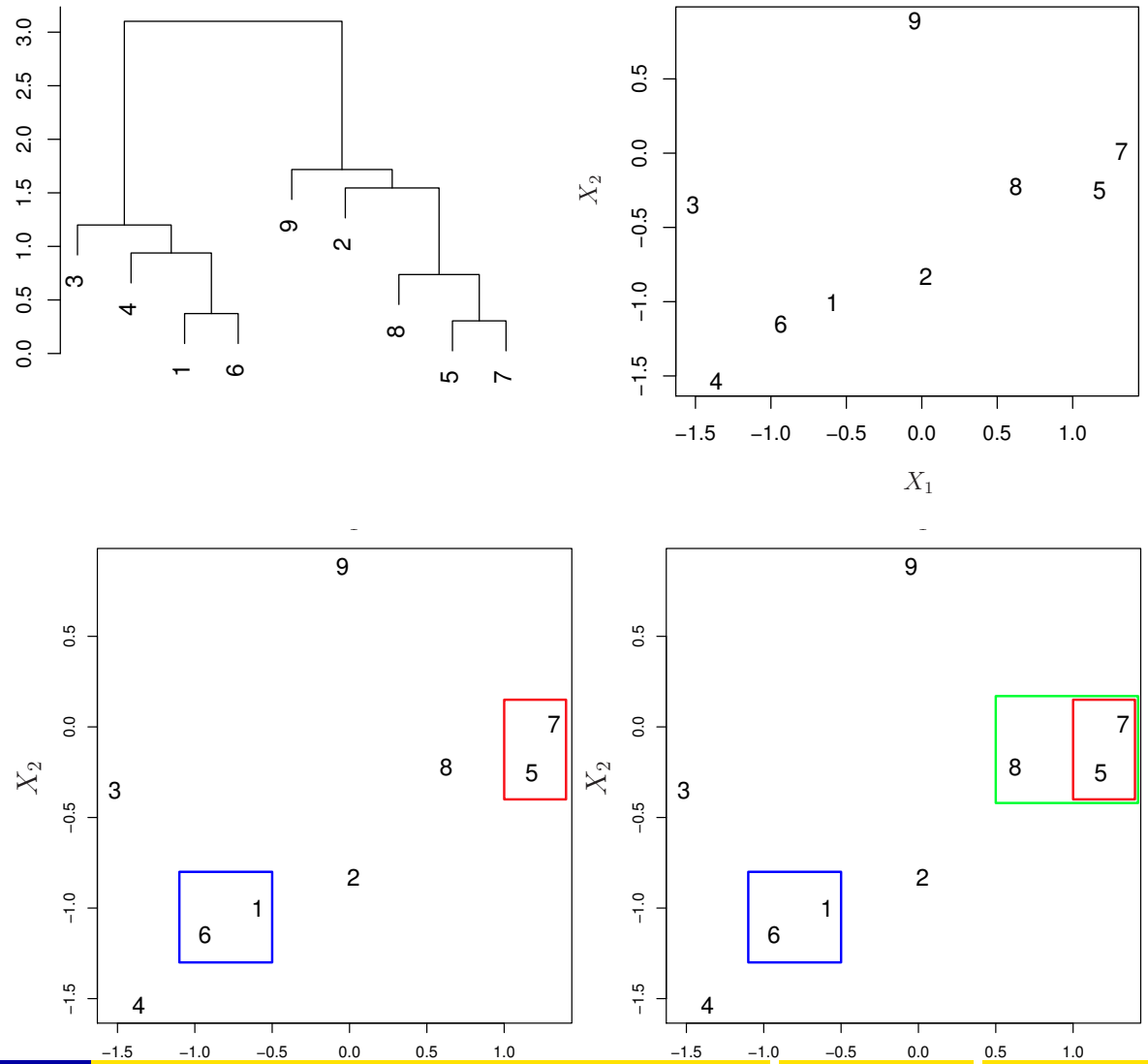
Dendrograms

- Dendrogram is the result plot of a hierarchical clustering.
- Cutting the tree of a fixed high splits samples at leaves into clusters.
  - The length of the two legs of the U-link represents the distance between the child clusters.



Interpretation of Dendrograms – 2 and 9 are NOT close

Samples fused at very bottom are close each other.



Mean Shift Clustering

Mean Shift Clustering

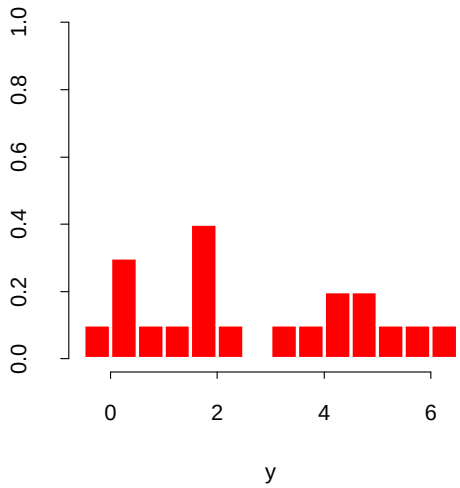
1: **procedure** MEAN SHIFT CLUSTERING:(X data, $K(\cdot)$ the kernel, λ the bandwidth)
2: $\mathcal{C} \leftarrow \emptyset$
3: **for** each data record **do**
4: **repeat** # shift each mean x to the weighted average
5: $m(x) \leftarrow \frac{\sum_{i=1}^{N} K(x_i - x) x_i}{\sum_{i=1}^{N} K(x_i - x)}$
6: **until** no chance in assignment
7: add the new $m(x)$ to $\mathcal{C}$
8: **end for**
9: return prunned $\mathcal{C}$
10: **end procedure**

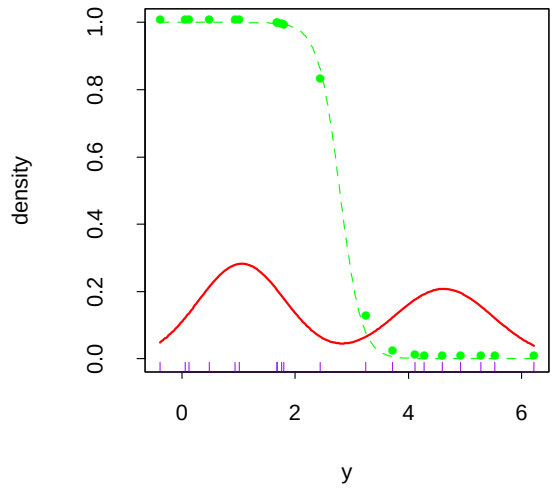
Kernels:

- flat kernel λ ball
- Gaussian kernel $K(x_i - x) = e^{\frac{\|x_i - x\|^2}{\lambda^2}}$

Gaussian Mixture Model

- Assume the data come from a set of k gaussian distributions
- each with
  - prior probability π_k
  - mean μ_k
  - covariance matrix Σ_k
  - $\phi_{\mu_k,\Sigma_k}(x) = \frac{1}{\sqrt{(2\pi)^p|\Sigma_k|}} e^{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}$.
- We want to find the maximum likelihood estimate of the model parameters.
- We use (more general) EM algorithm.





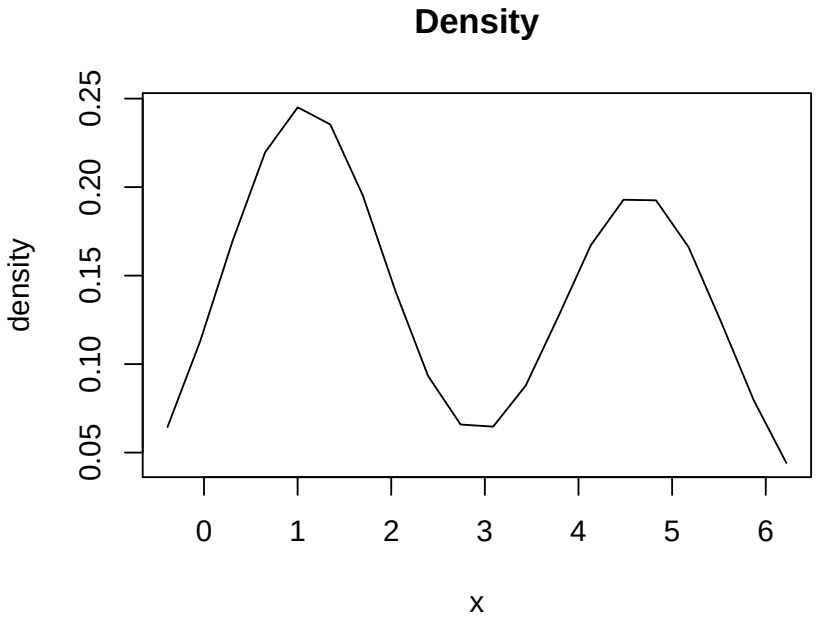
EM learning of Mixture of K Gaussians !

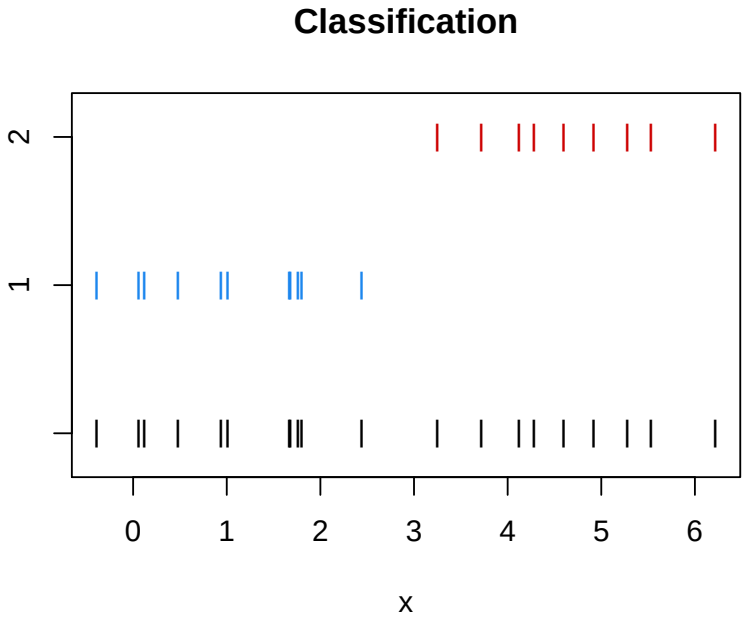
- Model parameters $\pi_1, \ldots, \pi_k, \mu_1, \ldots, \mu_k, \Sigma_1, \ldots, \Sigma_k$ such that $\sum_{k=1}^{K} \pi_k = 1$.
- **E**xpectation: weights of unobserved 'fill–ins' k of variable C:

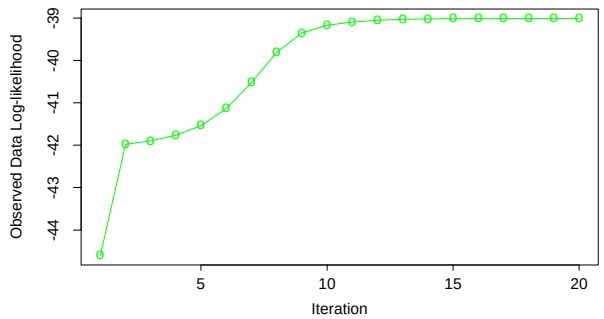
$$\begin{aligned} p_{ik} &= P(C = k|x_i) = \alpha \cdot P(x_i|C_i = k) \cdot P(C_i = k) \\ &= \frac{\pi_k \phi_{\theta_k}(x_i)}{\sum_{l=1}^{K} \pi_l \phi_{\theta_l}(x_i)} \\ p_k &= \sum_{i=1}^{N} p_{ik} \end{aligned}$$

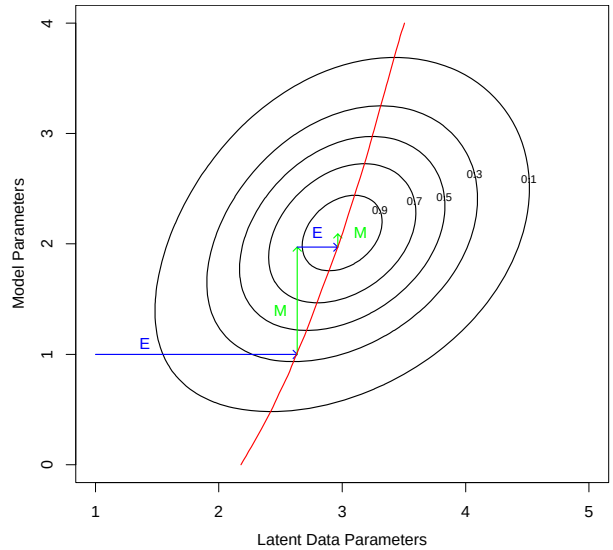
- **M**aximize: mean, variance and cluster 'prior' for each cluster k:

$$\begin{aligned} \mu_k &\leftarrow \sum_i \frac{p_{ik}}{p_k} x_i \\ \Sigma_k &\leftarrow \sum_i \frac{p_{ik}}{p_k} (x_i - \mu_k)(x_i - \mu_k)^T \\ \pi_k &\leftarrow \frac{p_k}{\sum_{l=1}^{K} p_l}. \end{aligned}$$


Density
density
0.05
0.10
0.15
0.20
0.25
0
1
2
3
4
5
6
x


Classification
1
2
0
1
2
3
4
5
6
x


Observed Data Log-likelihood
-39
-40
-41
-42
-43
-44
5
10
15
20
Iteration


Model Parameters
0
1
2
3
4
1
2
3
4
5
Latent Data Parameters
E
M
E
M
0.9
0.7
0.5
0.3
0.1

Kernel Density Estimation

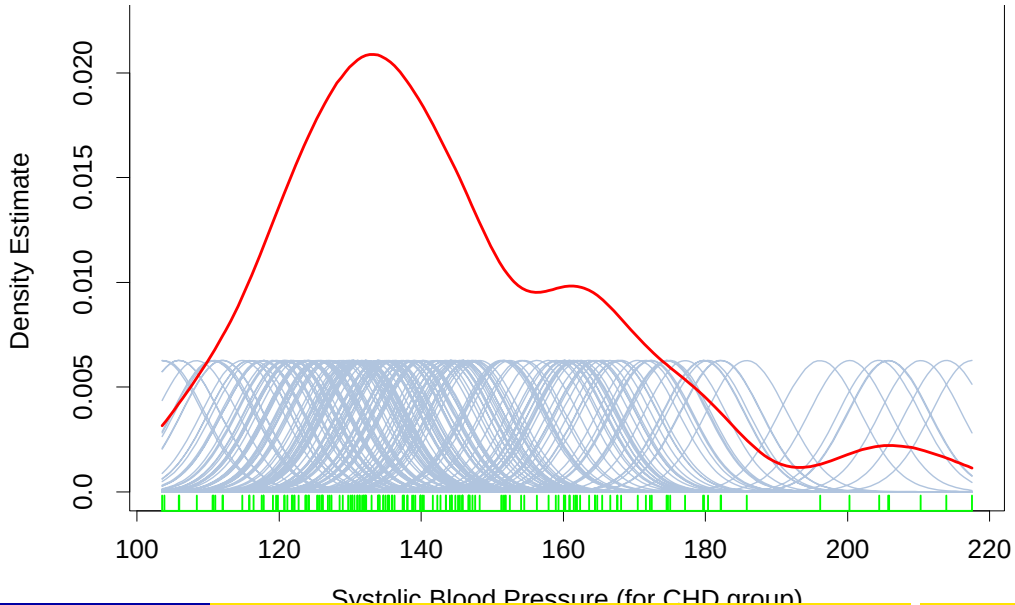
- Kernel Density Estimation is an unsupervised procedure
- We smooth the density estimate in the neighbourhood $\mathcal{N}(x_0)$ with lenghtscale λ

$$\hat{f}_X(x_0) = \frac{\#x_i \in \mathcal{N}(x_0)}{N\lambda}$$

- by the Parzen kernel estimate

$$\hat{f}_X(x_0) = \frac{1}{N\lambda} \sum_{i=1}^{N} K_\lambda(x_0, x_i),$$

- Popular choice for K_λ is the Gaussian kernel density ϕ_λ.



Kernel Density Classification

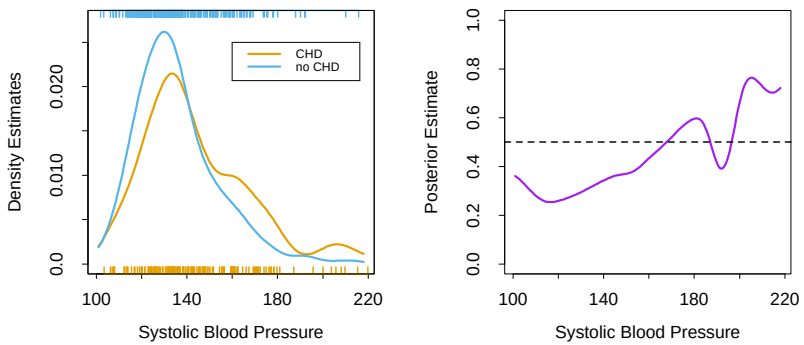
- We may estimate Kernel Density for each target class $k = 1, \ldots, K$, estimate class priors π_k and use Bayes' theorem:

$$\hat{Pr}(G = k|X = x_0) = \frac{\pi_k \hat{f}_k(x_0)}{\sum_{j=1}^{K} \pi_j \hat{f}_k(x_0)}.$$

- by the Parzen kernel estimate

$$\hat{f}_X(x_0) = \frac{1}{N\lambda} \sum_{i=1}^{N} K_\lambda(x_0, x_i),$$

- Popular choice for K_λ is the Gaussian kernel density ϕ_λ.



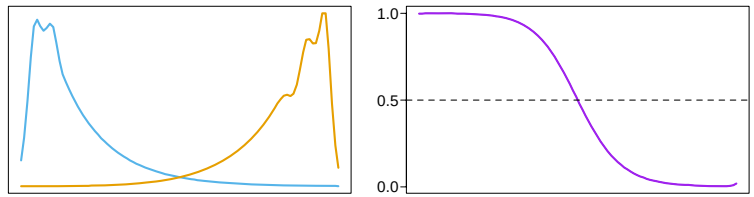


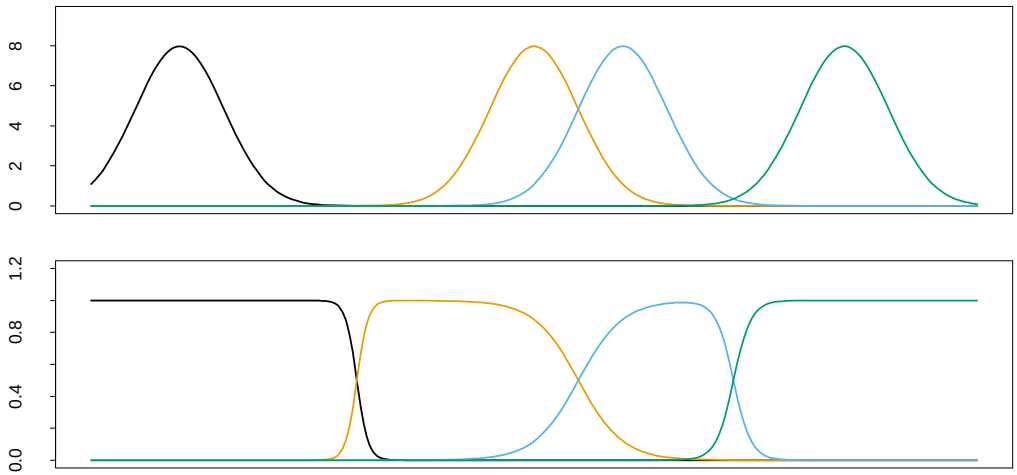
FIGURE 6.15. *The population class densities may have interesting structure (left) that disappears when the posterior probabilities are formed (right).*

Radial Basis Functions and Kernels for Regression

- The kernels do not have to be placed at all observation points.
- We may select (fit) prototype parameters ξ_j and scale patameters λ_j to place pre-defined number of kernels $K_{\lambda_j}(\xi_j, x)$, $j \in 1, \ldots, M, \lambda_j \in \mathbb{R}, \xi_j \in X$.
- and then fit the density as a linear function of kernels as basis

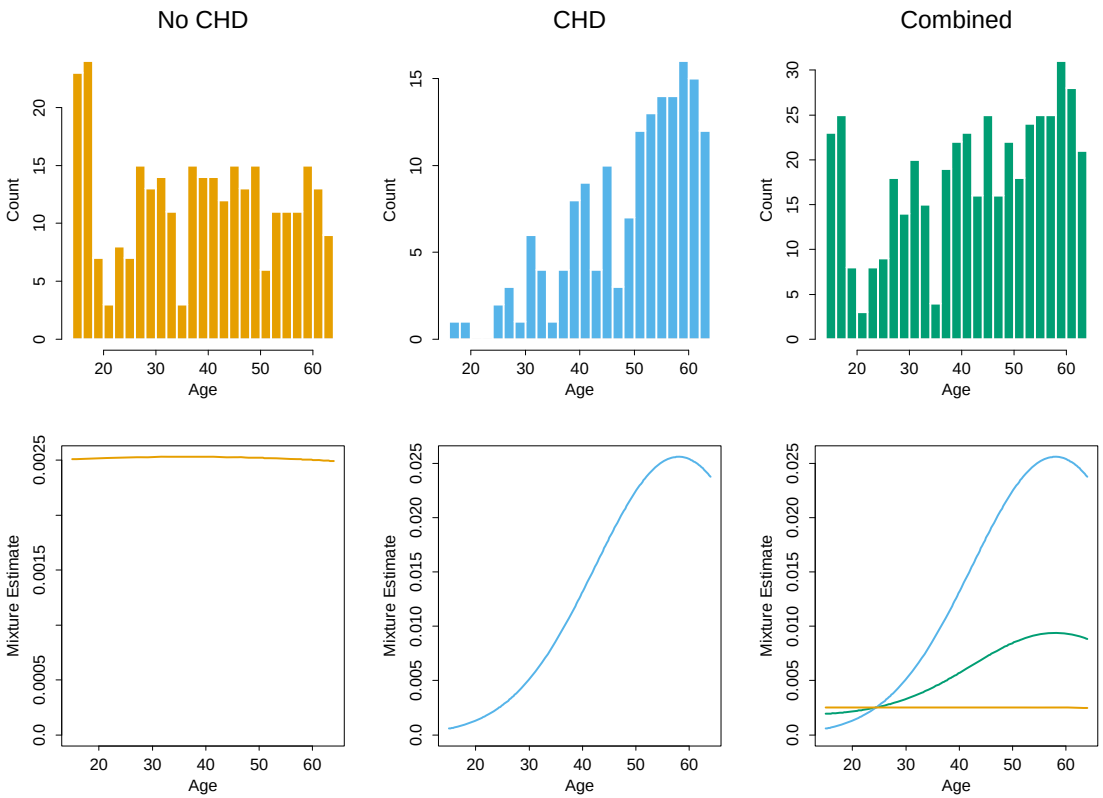
$$f(x) = \sum_{j=1}^{M} K_{\lambda_j}(\xi_j, x)\beta_j,$$

- We should either fit the lengthscale parameters λ_j or re-normalize the radial basis functions. Otherwise, the RBF can leave holes (upper figure, re-normalized down).



Mixuture Models for Density Estimation and Classification

- One RBF kernel was fitted for each class
- The data sample is classified according the more probable label (let kernels vote).
- If the covariance matrices are constrained to be scalar $\Sigma_m = \sigma_m I$, we actually fit the naive Bayes model.
- In this case, this method was as good as logistic regression.



Summary

- K-means clustering - the basic one
  - the number of clusters:
  - GAP
  - Silhouette
- The distance is crucial.
  - Consider standardization or weighting the features.
- K-medoids - does need metric, just a distance
- hierarchical clustering
  - different distance measures
  - dendrogram
- other approaches (mean shift clustering, Self Organizing Maps, Spectral Clustering).

Frequent itemsets, Association Rules

Unsupervised learning

- No goal class (either Y nor G).
- Usually binary data $X_{ij} \in \{0,1\}^{N \times p}$
- Value $= 1$ is our interest; for example purchase.
- p may be very large; for example the size of the range of goods in an market.
- Popular application: Market basket analysis.
- Generally: We look for L prototypes $v_1, \ldots, v_L \in X^p$ such that $P(v_\ell)$ is relatively large.
- With large p, we do not have enough data to estimate $P(v_\ell)$ since number of observations with $P(X = v_\ell)$ is too small.
- We seek for regions where $P(x)$ is large, that can be written as conjunctive rule on dimension conditions $\bigcap_{j=1}^{p}(X_j \in s_j)$ where s_j are selected values of the feature X_j.

Frequent itemsets, Association Rules

Unsupervised learning

- No goal class (either Y nor G).
- Usually binary data $X_{ij} \in \{0,1\}^{N \times p}$
- Value $= 1$ is our interest; for example purchase.
- p may be very large; for example the size of the range of goods in an market.
- Popular application: Market basket analysis.
- Generally: We look for L prototypes $v_1, \ldots, v_L \in X^p$ such that $P(v_\ell)$ is relatively large.
- With large p, we do not have enough data to estimate $P(v_\ell)$ since number of observations with $P(X = v_\ell)$ is too small.
- We seek for regions where $P(x)$ is large, that can be written as conjunctive rule on dimension conditions $\bigcap_{j=1}^{p}(X_j \in s_j)$ where s_j are selected values of the feature X_j.

Frequent itemsets, Association Rules

Unsupervised learning

- No goal class (either Y nor G).
- Usually binary data $X_{ij} \in \{0,1\}^{N \times p}$
- Value $= 1$ is our interest; for example purchase.
- p may be very large; for example the size of the range of goods in an market.
- Popular application: Market basket analysis.
- Generally: We look for L prototypes $v_1, \ldots, v_L \in X^p$ such that $P(v_\ell)$ is relatively large.
- With large p, we do not have enough data to estimate $P(v_\ell)$ since number of observations with $P(X = v_\ell)$ is too small.
- We seek for regions where $P(x)$ is large, that can be written as conjunctive rule on dimension conditions $\bigcap_{j=1}^{p}(X_j \in s_j)$ where s_j are selected values of the feature X_j.

Frequent itemsets, Association Rules

Unsupervised learning

- No goal class (either Y nor G).
- Usually binary data $X_{ij} \in \{0,1\}^{N \times p}$
- Value $= 1$ is our interest; for example purchase.
- p may be very large; for example the size of the range of goods in an market.
- Popular application: Market basket analysis.
- Generally: We look for L prototypes $v_1, \ldots, v_L \in X^p$ such that $P(v_\ell)$ is relatively large.
- With large p, we do not have enough data to estimate $P(v_\ell)$ since number of observations with $P(X = v_\ell)$ is too small.
- We seek for regions where $P(x)$ is large, that can be written as conjunctive rule on dimension conditions $\bigcap_{j=1}^{p}(X_j \in s_j)$ where s_j are selected values of the feature X_j.

Frequent itemsets, Association Rules

Unsupervised learning

- No goal class (either Y nor G).
- Usually binary data $X_{ij} \in \{0,1\}^{N \times p}$
- Value $= 1$ is our interest; for example purchase.
- p may be very large; for example the size of the range of goods in an market.
- Popular application: Market basket analysis.
- Generally: We look for L prototypes $v_1, \ldots, v_L \in X^p$ such that $P(v_\ell)$ is relatively large.
- With large p, we do not have enough data to estimate $P(v_\ell)$ since number of observations with $P(X = v_\ell)$ is too small.
- We seek for regions where $P(x)$ is large, that can be written as conjunctive rule on dimension conditions $\bigcap_{j=1}^{p}(X_j \in s_j)$ where s_j are selected values of the feature X_j.

Frequent itemsets, Association Rules

Unsupervised learning

- No goal class (either Y nor G).
- Usually binary data $X_{ij} \in \{0,1\}^{N \times p}$
- Value $= 1$ is our interest; for example purchase.
- p may be very large; for example the size of the range of goods in an market.
- Popular application: Market basket analysis.
- Generally: We look for L prototypes $v_1, \ldots, v_L \in X^p$ such that $P(v_\ell)$ is relatively large.
- With large p, we do not have enough data to estimate $P(v_\ell)$ since number of observations with $P(X = v_\ell)$ is too small.
- We seek for regions where $P(x)$ is large, that can be written as conjunctive rule on dimension conditions $\bigcap_{j=1}^{p}(X_j \in s_j)$ where s_j are selected values of the feature X_j.

Frequent itemsets, Association Rules

Unsupervised learning

- No goal class (either Y nor G).
- Usually binary data $X_{ij} \in \{0,1\}^{N \times p}$
- Value $= 1$ is our interest; for example purchase.
- p may be very large; for example the size of the range of goods in an market.
- Popular application: Market basket analysis.
- Generally: We look for L prototypes $v_1, \ldots, v_L \in X^p$ such that $P(v_\ell)$ is relatively large.
- With large p, we do not have enough data to estimate $P(v_\ell)$ since number of observations with $P(X = v_\ell)$ is too small.
- We seek for regions where $P(x)$ is large, that can be written as conjunctive rule on dimension conditions $\bigcap_{j=1}^{p}(X_j \in s_j)$ where s_j are selected values of the feature X_j.

Frequent itemsets, Association Rules

Unsupervised learning

- No goal class (either Y nor G).
- Usually binary data $X_{ij} \in \{0,1\}^{N \times p}$
- Value $= 1$ is our interest; for example purchase.
- p may be very large; for example the size of the range of goods in an market.
- Popular application: Market basket analysis.
- Generally: We look for L prototypes $v_1, \ldots, v_L \in X^p$ such that $P(v_\ell)$ is relatively large.
- With large p, we do not have enough data to estimate $P(v_\ell)$ since number of observations with $P(X = v_\ell)$ is too small.
- We seek for regions where $P(x)$ is large, that can be written as conjunctive rule on dimension conditions $\bigcap_{j=1}^{p}(X_j \in s_j)$ where s_j are selected values of the feature X_j.

Hypothesis space for Apriori

ESL book Figure:

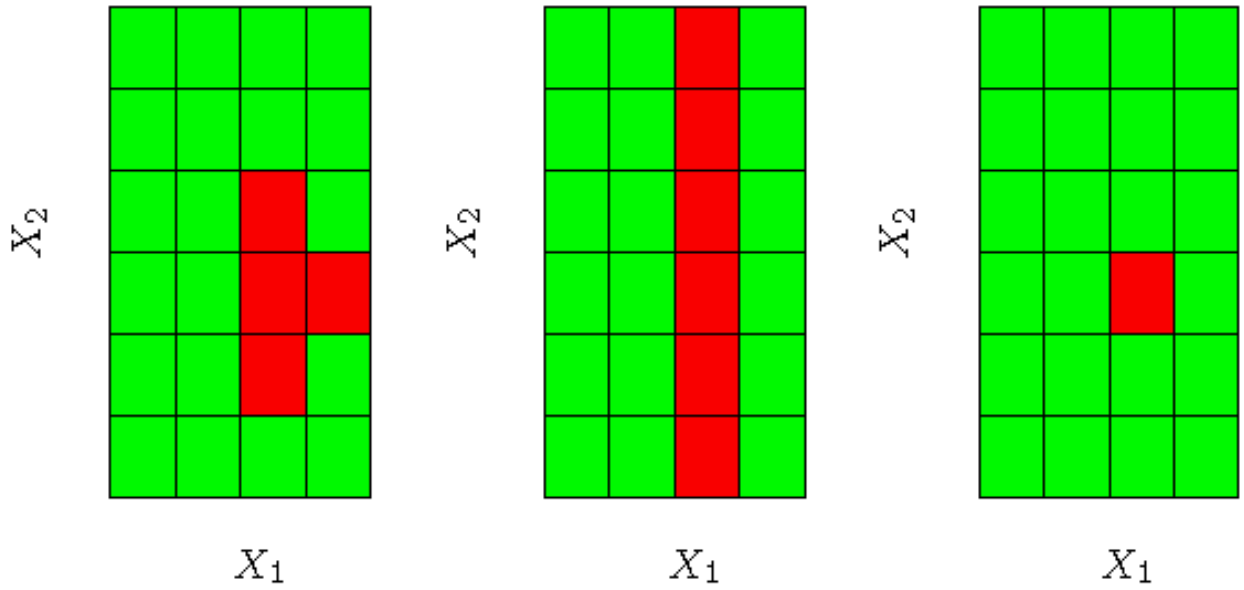


FIGURE 14.1. *Simplifications for association rules. Here there are two inputs X_1 and X_2, taking four and six distinct values, respectively. The red squares indicate areas of high density. To simplify the computations, we assume that the derived subset corresponds to either a single value of an input or all values. With this assumption we could find either the middle or right pattern, but not the left one.*

Market Basket Analysis

- For very large datasets, $p \approx 10^4$, $N \approx 10^8$; in unit ball is the distance to the nearest neighbour ≈ 0.9981.
- Simplifications: Test on feature X_j either equal to a specific value or no restriction at all,
- I select combinations of items with higher number of occurences (**support**) than predefined threshold t.
- I select all combinations fulfilling conditions above.
- Categorical variables may be codded by dummy variables in advance (if not too many).
  - OneHotEncoder for each class p_j a new variable $X_{p_j} \in \{0, 1\}$.

Market Basket Analysis

- For very large datasets, $p \approx 10^4, N \approx 10^8$; in unit ball is the distance to the nearest neighbour ≈ 0.9981.
- Simplifications: Test on feature X_j either equal to a specific value or no restriction at all,
- I select combinations of items with higher number of occurences (**support**) than predefined threshold t.
- I select all combinations fulfilling conditions above.
- Categorical variables may be codded by dummy variables in advance (if not too many).
- OneHotEncoder for each class p_j a new variable $X_{p_j} \in \{0, 1\}$.

Market Basket Analysis

- For very large datasets, $p \approx 10^4, N \approx 10^8$; in unit ball is the distance to the nearest neighbour ≈ 0.9981.
- Simplifications: Test on feature X_j either equal to a specific value or no restriction at all,
- I select combinations of items with higher number of occurences (**support**) than predefined threshold t.
- I select all combinations fulfilling conditions above.
- Categorical variables may be codded by dummy variables in advance (if not too many).
  - One-hot encoder for each class p_j a new variable $X_{p_j} = 1$ or $= 0$.

Market Basket Analysis

- For very large datasets, $p \approx 10^4, N \approx 10^8$; in unit ball is the distance to the nearest neighbour ≈ 0.9981.
- Simplifications: Test on feature X_j either equal to a specific value or no restriction at all,
- I select combinations of items with higher number of occurences (**support**) than predefined threshold t.
- I select all combinations fulfilling conditions above.
- Categorical variables may be codded by dummy variables in advance (if not too many).
  - **OneHotEncoder** for each class g, a new variable $X_g = [X === g]$.

Market Basket Analysis

- For very large datasets, $p \approx 10^4, N \approx 10^8$; in unit ball is the distance to the nearest neighbour ≈ 0.9981.
- Simplifications: Test on feature X_j either equal to a specific value or no restriction at all,
- I select combinations of items with higher number of occurences (**support**) than predefined threshold t.
- I select all combinations fulfilling conditions above.
- Categorical variables may be codded by dummy variables in advance (if not too many).
  - **OneHotEncoder** for each class g, a new variable $X_g = [X == g]$.

Market Basket Analysis

- For very large datasets, $p \approx 10^4, N \approx 10^8$; in unit ball is the distance to the nearest neighbour ≈ 0.9981.
- Simplifications: Test on feature X_j either equal to a specific value or no restriction at all,
- I select combinations of items with higher number of occurences (**support**) than predefined threshold t.
- I select all combinations fulfilling conditions above.
- Categorical variables may be codded by dummy variables in advance (if not too many).
  - **OneHotEncoder** for each class g, a new variable $X_g = [X == g]$.

Apriori Algorithm!

```
1: procedure APRIORI:(X dataset, t threshold for support )
2:     i ← 0
3:     Generate list of candidates of the length i
4:     while Candidate set not empty do
5:         for each data sample do
6:             for each candidate do
7:                 if all items of candidate appear in the data sample then
8:                     increase the candidate counter by 1
9:                 end if
10:            end for
11:        end for
12:        i ← i + 1
13:        Discard candidates with support less than t.
14:        Generate list of candidates of the length i
15:            Join any two candidates from previous step having i − 2
   elements common. (More pruning possible.)
16:    end while
17: end procedure
```

Example: Apriori Algorithm

- $t = 0.2$
- $t * N = 2 = 0.20 * 10$
- Data

a b c e f o
a c g
e i
a c d e g
a c e g l
e j
a b c e f p
a c d
a c e g m
a c e g n

i=1
a=8
b=2
c=8
d=2
e=8
f=2
g=5
i=j=l=o=1
p=m=n=1

i=2
ab=2
ac=8
ad=2
ae=6
af=2
ag=5
bc=2
bd=0
be=2
bf=2
bg=0
cd=2
ce=6
cf=2
cg=5
de=1
df=0
dg=1
ef=2

i=3
abc=2
abd=0
abe=2
abf=2
abg=0
acd=2
ace=6
acf=2
acg=5
ade=1
adf=0
adg=1
aeg=4
...

i=4 ...
abce=2
abcf=2
abef=2
abeg=0
acef=2
aceg=4
adeg=1
aefg=0
...

Properties of the Apriori Algorithm

- Applicable for very large data (with high threshold t).
- The key idea:
  - Only few of 2^K combinations have high support $> t$,
  - **subset of high–support combination has also high support**.
- The number of passes through the data is equal to the size of the longest supported combination. The data does not to be in memory simultaneously.
- *FPgrowth* algorithm needs only two passes through the data.

Properties of the Apriori Algorithm

- Applicable for very large data (with high threshold t).
- The key idea:
  - Only few of 2^K combinations have high support $> t$,
  - **subset of high–support combination has also high support**.
- The number of passes through the data is equal to the size of the longest supported combination. The data does not to be in memory simultaneously.
- *FPgrowth* algorithm needs only two passes through the data.

Properties of the Apriori Algorithm

- Applicable for very large data (with high threshold t).
- The key idea:
  - Only few of 2^K combinations have high support $> t$,
  - **subset of high–support combination has also high support**.
- The number of passes through the data is equal to the size of the longest supported combination. The data does not to be in memory simultaneously.
- *FPgrowth* algorithm needs only two passes through the data.

Properties of the Apriori Algorithm

- Applicable for very large data (with high threshold t).
- The key idea:
  - Only few of 2^K combinations have high support $> t$,
  - **subset of high–support combination has also high support**.
- The number of passes through the data is equal to the size of the longest supported combination. The data does not to be in memory simultaneously.
- *FPgrowth* algorithm needs only two passes through the data.

Association Rules !

- From each supported itemset $\mathcal{K}$ found by Apriori algorithm we create a list of **association rules**, implications of the form $A \Rightarrow B$ where:
  - A, B are disjoint and $A \cup B = \mathcal{K}$
  - A is called **antecedent**
  - B is called **consequent**.
- Support of the rule $T(A \Rightarrow B)$ is defined as normalized support of the itemset $\mathcal{K}$, that is normalized support of the conjunction $A\&B$.

$$T(\mathcal{K}) = \frac{|data_{\mathcal{K}}|}{|data|}$$

$$T(A \Rightarrow B) = \frac{|data_{A\&B}|}{|data|}$$

Association Rules !

- From each supported itemset $\mathcal{K}$ found by Apriori algorithm we create a list of **association rules**, implications of the form $A \Rightarrow B$ where:
  - A, B are disjoint and $A \cup B = \mathcal{K}$
  - A is called **antecedent**
  - B is called **consequent**.
- Support of the rule $T(A \Rightarrow B)$ is defined as normalized support of the itemset $\mathcal{K}$, that is normalized support of the conjunction $A\&B$.

$$T(\mathcal{K}) = \frac{|data_{\mathcal{K}}|}{|data|}$$

$$T(A \Rightarrow B) = \frac{|data_{A\&B}|}{|data|}$$

Rule Confidence and Lift

There are two important measures for a rule $A \Rightarrow B$:

- **Confidence** (predictability, přesnost)

$$C(A \Rightarrow B) = \frac{T(A \Rightarrow B)}{T(A)}$$

  that is an estimate of $P(B|A)$,

- Support $T(B)$ is an estimate of $P(B)$,
- **Lift** is the ration of confidence and expected precision:

$$L(A \Rightarrow B) = \frac{C(A \Rightarrow B)}{T(B)}$$

  that is an estimate of $\frac{P(A\&B)}{P(A) \cdot P(B)}$.

- **Leverage** is the difference of supports:

$$leverage(A \Rightarrow B) = T(A \Rightarrow B) - T(A) \cdot T(B)$$

- **Conviction** is the ratio:

$$conviction(A \Rightarrow B) = \frac{1 - T(B)}{1 - C(A \Rightarrow B)}.$$

Rule Confidence and Lift

There are two important measures for a rule $A \Rightarrow B$:

- **Confidence** (predictability, přesnost)

$$C(A \Rightarrow B) = \frac{T(A \Rightarrow B)}{T(A)}$$

  that is an estimate of $P(B|A)$,
- Support $T(B)$ is an estimate of $P(B)$,
- **Lift** is the ration of confidence and expected precision:

$$L(A \Rightarrow B) = \frac{C(A \Rightarrow B)}{T(B)}$$

  that is an estimate of $\frac{P(A\&B)}{P(A) \cdot P(B)}$.
- **Leverage** is the difference of supports:

$$leverage(A \Rightarrow B) = T(A \Rightarrow B) - T(A) \cdot T(B)$$

- **Conviction** is the ratio:

$$conviction(A \Rightarrow B) = \frac{1 - T(B)}{1 - C(A \Rightarrow B)}.$$

Rule Confidence and Lift

There are two important measures for a rule $A \Rightarrow B$:

- **Confidence** (predictability, přesnost)

$$C(A \Rightarrow B) = \frac{T(A \Rightarrow B)}{T(A)}$$

  that is an estimate of $P(B|A)$,
- Support $T(B)$ is an estimate of $P(B)$,
- **Lift** is the ration of confidence and expected precision:

$$L(A \Rightarrow B) = \frac{C(A \Rightarrow B)}{T(B)}$$

  that is an estimate of $\frac{P(A\&B)}{P(A) \cdot P(B)}$.
- **Leverage** is the difference of supports:

$$\mathit{leverage}(A \Rightarrow B) = T(A \Rightarrow B) - T(A) \cdot T(B)$$

- **Conviction** is the ratio:

$$\mathit{conviction}(A \Rightarrow B) = \frac{1 - T(B)}{1 - C(A \Rightarrow B)}.$$

Association Rule Example

ESL book example:

Association rule 2: Support 13.4%, confidence 80.8%, and lift 2.13.

$$\left[\begin{array}{rcl} \text{language in home} & = & \textit{English} \\ \text{householder status} & = & \textit{own} \\ \text{occupation} & = & \{\textit{professional/managerial}\} \end{array}\right] \Downarrow \text{income} \geq \textit{\$40,000}$$

- $\mathcal{K} = \{$English, own, prof/man, income>\$40000$\}$,
- 13.4% people has all four properties,
- 80.8% of people with {English, own, prof/man} have income> \$40000,
- $T(\text{income} > \$40000) = 37.94\%$, therefore $\textit{Lift} = 2.13$.

The Goal of Apriori Algorithm !

- Apriori finds all rules with high support.
- Frequently, it finds many of rules.
- We usually select lower threshold c on confidence, that is we select rules with $T(A \Rightarrow B) > t$ and $C(A \Rightarrow B) > c$.
- Conversion of itemsets to rules is usually relatively fast compared to search of itemsets.
- See lispMiner for user interface and a lot of more.
- Python Apriori library:

```
from mlxtend.preprocessing import TransactionEncoder

from mlxtend.frequent_patterns import apriori, association_rules
from mlxtend.frequent_patterns import fpgrowth,fpmax

from mlxtend.frequent_patterns import hmine
```

Demographical Data ESL Example

Feature	Demographic	# Values	Type
1	Sex	2	Categorical
2	Marital status	5	Categorical
3	Age	7	Ordinal
4	Education	6	Ordinal
5	Occupation	9	Categorical
6	Income	9	Ordinal
7	Years in Bay Area	5	Ordinal
8	Dual incomes	3	Categorical
9	Number in household	9	Ordinal
10	Number of children	9	Ordinal
11	Householder status	3	Categorical
12	Type of home	5	Categorical
13	Ethnic classification	8	Categorical
14	Language in home	3	Categorical

Demographical Example – Continuing

- $N = 9409$ questionnaires, the ESL authors selected 14 questions.
- Preprocessing:
  - `na.omit() remove records with missing values,`
  - `ordinal features cut by median to binary,`
  - `for categorical create dummy variable for each category.`
- Apriori input was matrix 6876×50.
- Output: 6288 association rules
  - with max. 5 elements
  - with support at least 10%.

Negated Literals – Useful, Problematic

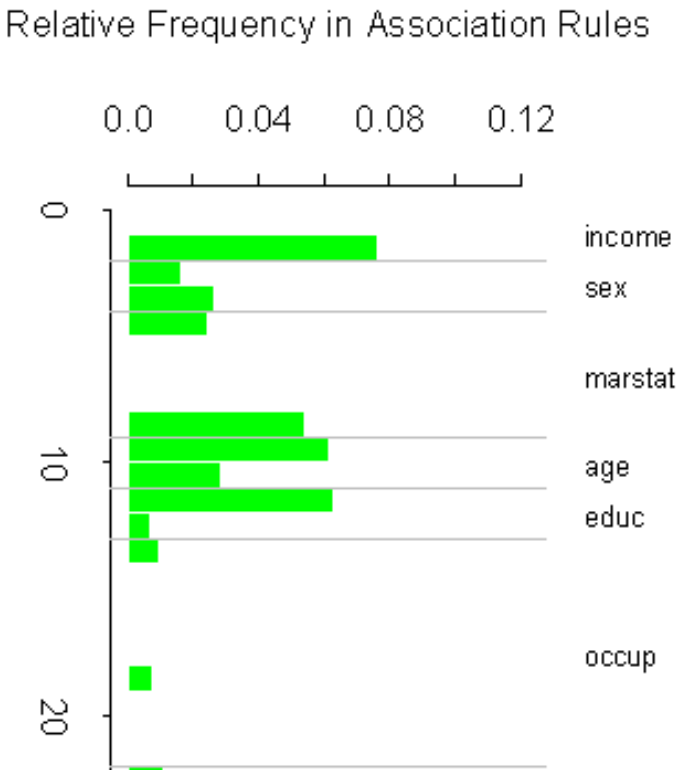
Association rule 3: Support 26.5%, confidence 82.8% and lift 2.15.

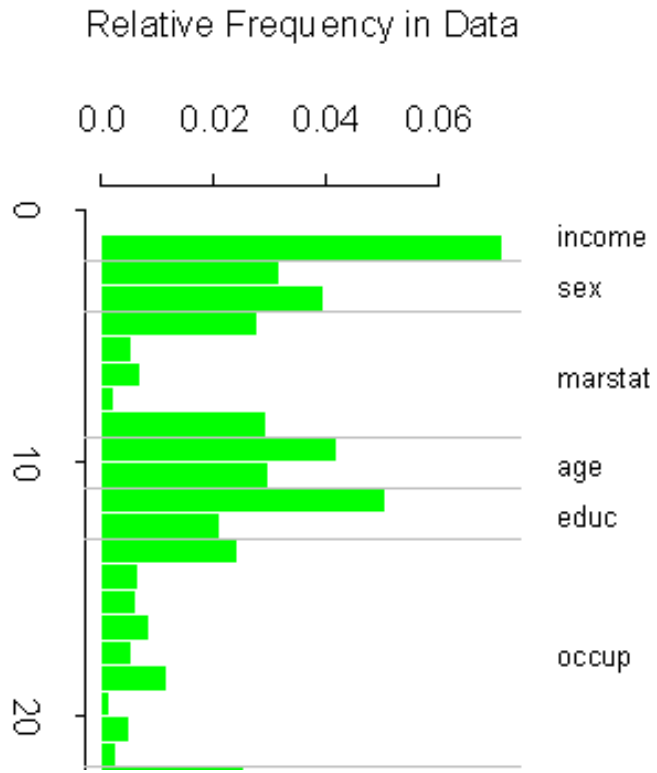
$$\left[\begin{array}{rcl} \text{language in home} & = & \textit{English} \\ \text{income} & < & \textit{\$40,000} \\ \text{marital status} & = & \textit{not married} \\ \text{number of children} & = & \textit{0} \end{array}\right]$$

$$\Downarrow$$

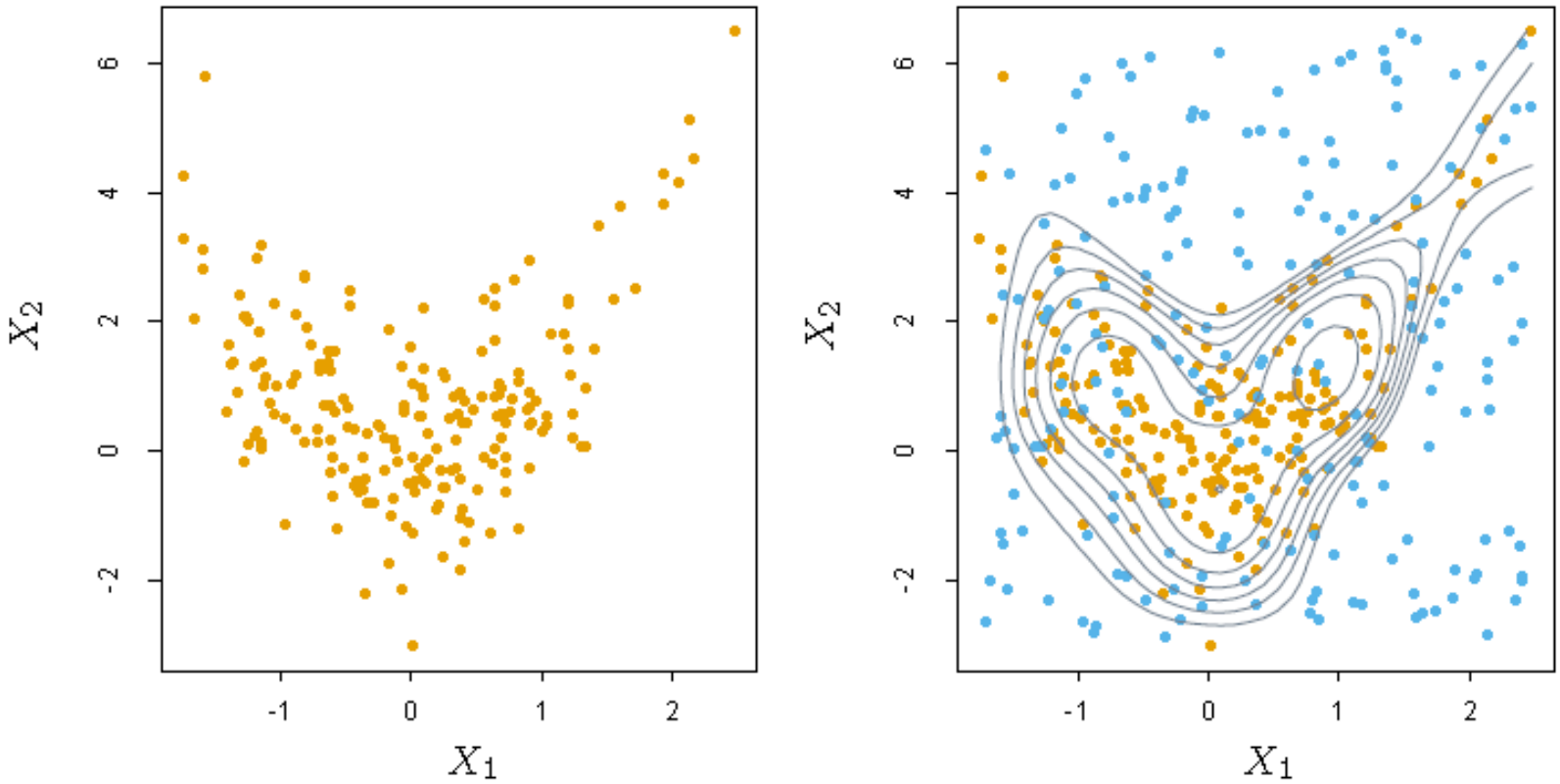
$$\text{education} \notin \{\textit{college graduate}, \textit{graduate study}\}$$

Non–frequent Values Dissapear





Unsupervised Learning as Supervised Learning



- We add additional attribute Y_G.
- $Y_G = 1$ for all our data.
- We generate randomly a dataset of similar size with uniform distribution, set $Y_G = 0$ for this artificial data.
- The task is to separate $Y_G = 1$ and $Y_G = 0$.

Generalize Association Rules

- We search for high lift, where probability of conjunction is greater than expected.
- Hypothesis is specified by column indexes j and subsets of values s_j corresponding features X_j. We aim:

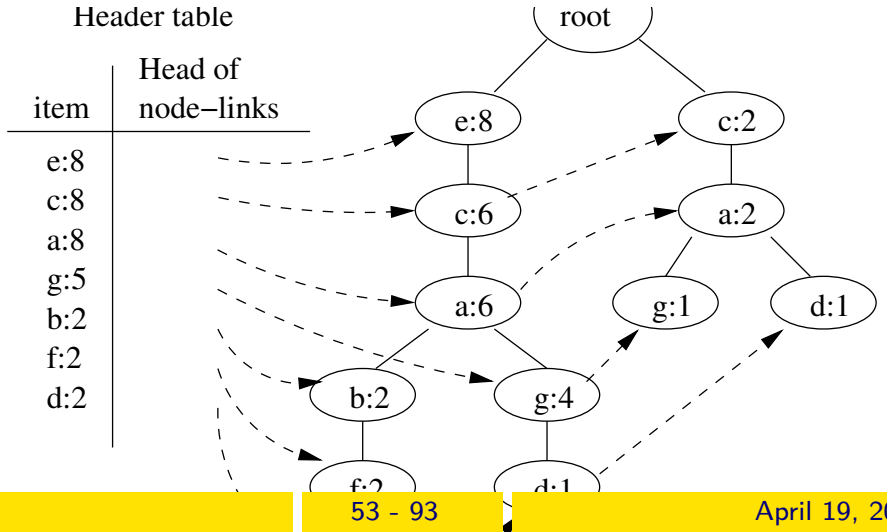
$$\hat{P}\left(\bigcap_{j\in\mathcal{J}}(X_j \in s_j)\right) = \frac{1}{N}\sum_{1}^{N} I\left(\bigcap_{j\in\mathcal{J}}(x_{ij} \in s_j)\right) >> \Pi_{j\in\mathcal{J}}\hat{P}(X_j \in s_j)$$

- On the data from previous slide, CART (decision tree alg.) or PRIM ('bump hunting') may be used.
- Figure on previous slide: Logistic regression on tensor product of natural splines.
- Other methods may be used. All are heuristics compared to full evaluation by Apriori.

FP–tree

```
1: procedure FP–TREE:(Data )
2:     Calculate counts of items (singletons)
3:     Create table header ordered by decreasing item count
4:     for each data sample do
5:         order items according to header
6:         insert branch into the tree
7:         increase all counters on the inserted branch
8:     end for
9:     return the tree
10: end procedure
```

Data	ordered
a b c e f o	e c a b f
a c g	c a g
e i	e
a c d e g	e c a g d
a c e g l	e c a g
e j	
a b c e f p	
a c d	



Frequent Itemsets with only 2 pass through data

- Build an internal structure called FP-tree
- Call FP-growth to generate frequent itemsets
  - Each construction of a conditional tree needs 2 pass through the parent tree
  - an optimized version with only 1 pass is presented. (It needs an additional data structure array.)
- FP-max to find maximal itemsets
  - non of immediate supersets is frequent
- FP-close to find close itemsets
  - non of immediate supersets has the same support.

Frequent Itemsets with only 2 pass through data

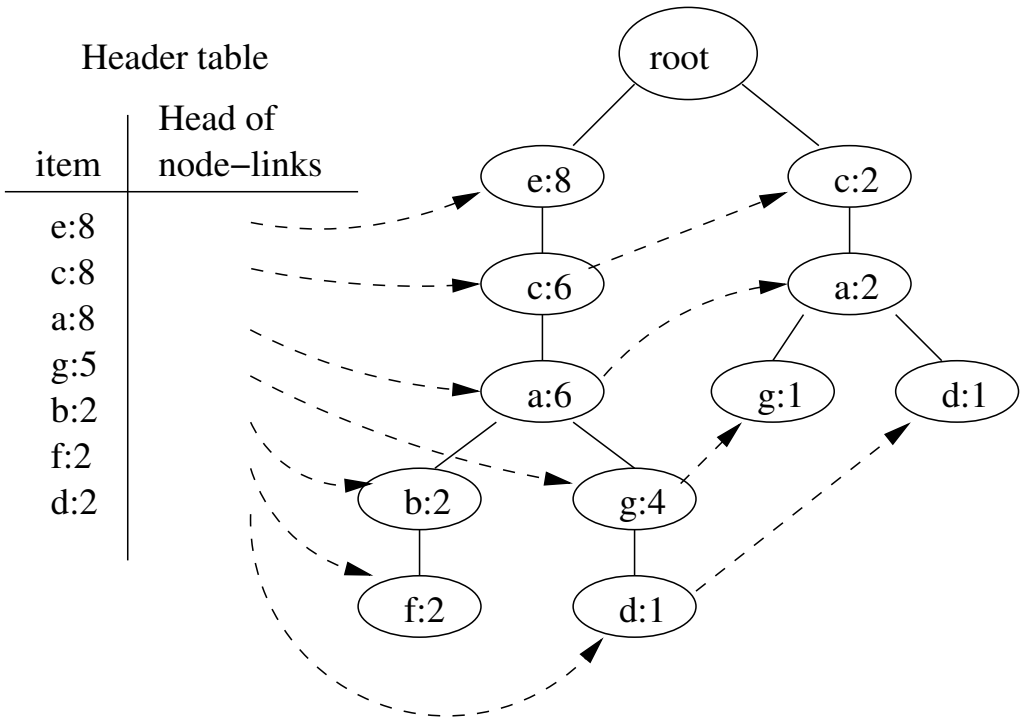
- Build an internal structure called FP-tree
- Call FP-growth to generate frequent itemsets
  - Each construction of a conditional tree needs 2 pass through the parent tree
  - an optimized version with only 1 pass is presented. (It needs an additional data structure array.)
- FP-max to find maximal itemsets
  - non of immediate supersets is frequent
- FP-close to find close itemsets
  - non of immediate supersets has the same support.

Frequent Itemsets with only 2 pass through data

- Build an internal structure called FP-tree
- Call FP-growth to generate frequent itemsets
  - Each construction of a conditional tree needs 2 pass through the parent tree
  - an optimized version with only 1 pass is presented. (It needs an additional data structure array.)
- FP-max to find maximal itemsets
  - non of immediate supersets is frequent
- FP-close to find close itemsets
  - non of immediate supersets has the same support.

FP-tree

- **FP-tree** contains all frequency information of the database.
- Principle: If X and Y are two itemsets, the count of itemsets $X \cup Y$ in the database is exactly that of Y in the restriction of the database to those transactions containing X.



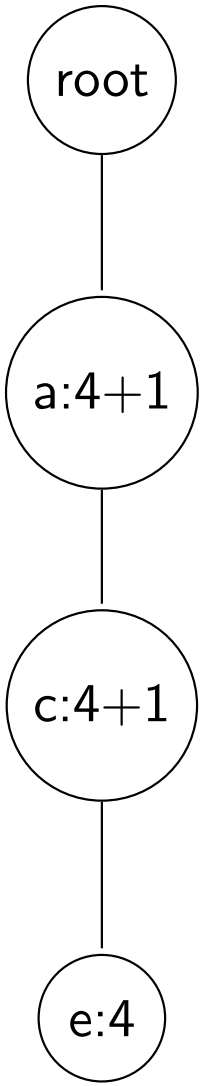
i=3
abc=2
abd=0
abe=2
abf=2
abg=0
acd=2
ace=6
acf=2
acg=5
ade=1
adf=0
adg=1
aeg=4
...

FPgrowth*

```
1: procedure FPgrowth*:(T a conditional FP-tree )
2:     if T only contains a single path P then
3:         for each subpath Y of P do
4:             output pattern Y ∪ T.base with
5:                 count = smallest count of nodes in Y
6:         end for
7:     else
8:         for each i in T.header do
9:             Y ← T.base ∪ {i} with i.count
10:            if T.array is not NULL then
11:                construct a new header table for Y's FP-tree from T.array
12:            else
13:                construct a new header table for Y's from T
14:            end if
15:            construct Y's conditional FP-tree T_Y and its array A_Y;
16:            if T_Y ≠ ∅ then
17:                call FPgrowth*(T_Y)
18:            end if
19:        end for
20:    end if
```

$$t = 2$$

$$T_{\{g\}} = [a:5, c:5, e:4]$$



	e	c	a	g	b	f
c	6					
a	6	8				
g	4	5	5			
b	2	2	2	0		
f	2	2	2	0	2	
d	1	2	2	1	0	0

(a) $A_{\emptyset}$

	a	c
c	5	
e	4	4

(b) $A_{\{g\}}$

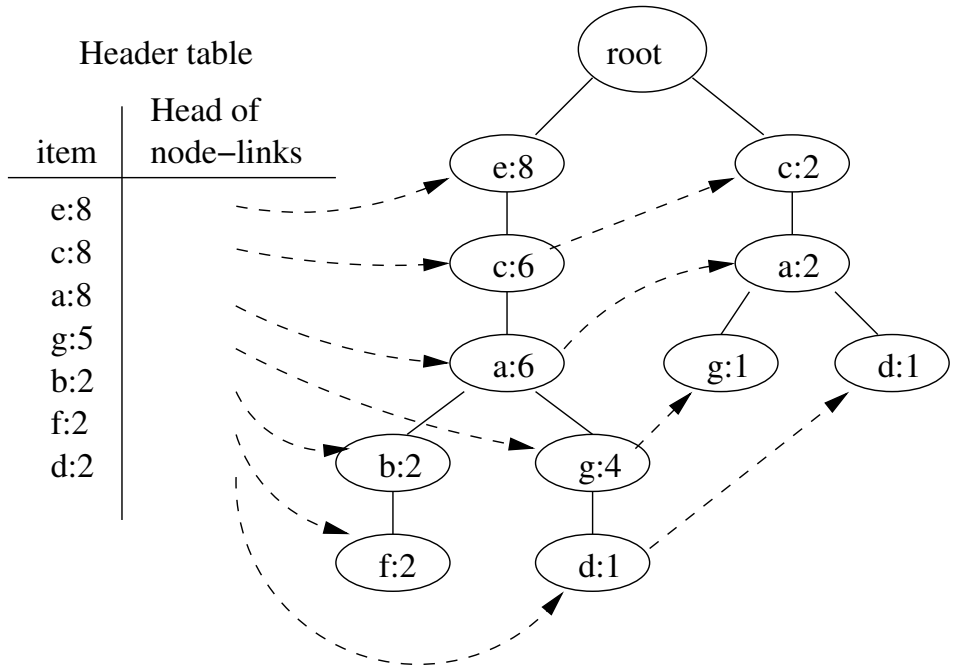


Table of Contens