Programming for Data Science – Unsupervised Learning

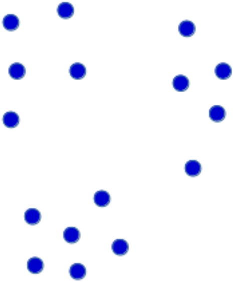
Henrik Boström

Prof. of Computer Science - Data Science Systems
Division of Software and Computer Systems
Department of Computer and Systems Sciences
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology
bostromh@kth.se

Outline

- Clustering
  - The k-means algorithm
  - Distance metrics
  - Cluster evaluation
  - Probability-based clustering
  - Hierarchical clustering
- Frequent itemset and association rule mining
  - The Apriori algorithm
  - Association rules

Clustering

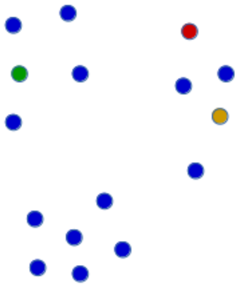


The k-means algorithm

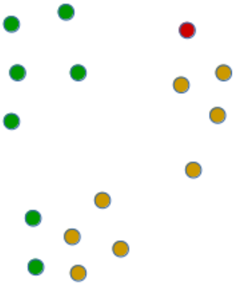
```
Input: instances X = {X1, ..., Xn}, number of clusters k
Output: a set of clusters {C1, ..., Ck}

{C1, ..., Ck} = a randomized partitioning of X

repeat

   c1, ..., ck = centroids of C1, ..., Ck

   move each instance Xi to the cluster Cj,
   which corresponds to the closest centroid cj

until no instance was moved to another cluster
```

The k-means algorithm (example)

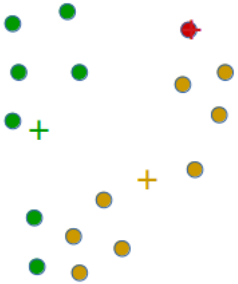
The k-means algorithm (example)



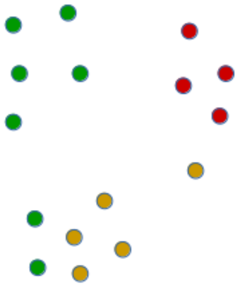
The k-means algorithm (example)



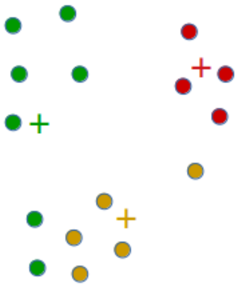
The k-means algorithm (example)



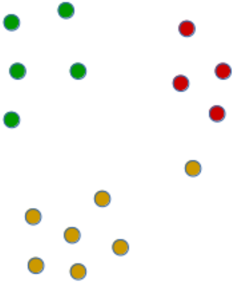
The k-means algorithm (example)



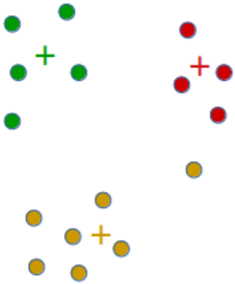
The k-means algorithm (example)



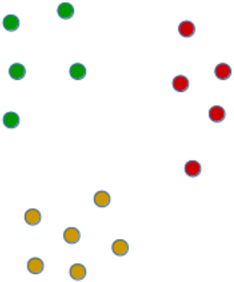
The k-means algorithm (example)



The k-means algorithm (example)



The k-means algorithm (example)



Distance metrics

- Euclidean distance

$$||a-b||_2 = \sqrt{\sum_i (a_i - b_i)^2}$$

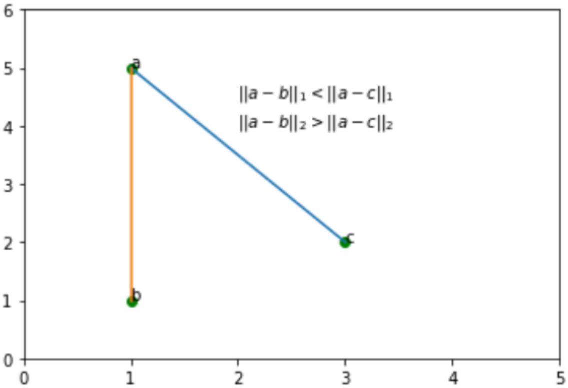
- Manhattan distance

$$||a-b||_1 = \sum_i |a_i - b_i|$$

- Hamming distance

$$H(a,b) = \sum_i \mathbf{1}(a_i \neq b_i)$$

Distance metrics (example)



Evaluation metrics

- Sum-of-squared-error

$$SSE(C) = \sum_{C_j \in C} \sum_{o \in C_j} (o - cent_j)^2$$

- Silhouette value

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

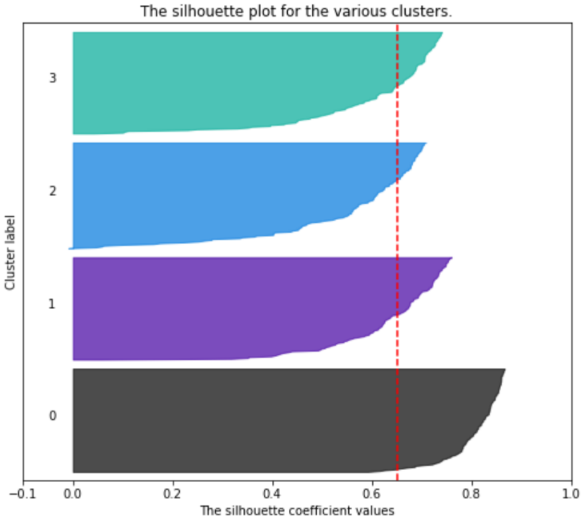
where $a(o)$ is the average distance to objects in the cluster of o and $b(o)$ is the average distance to objects in the nearest cluster not including o

- Rand index

$$R(C_1, C_2) = \frac{a + b}{\binom{n}{2}}$$

where a is the number of pairs of objects in the same cluster in both C_1 and C_2 and b is the number of pairs of objects in different clusters in both C_1 and C_2

Silhouette value (example)



Rand index (example)

```
C_1 = {{a,b,c},{d}}
C_2 = {{a,b},{c,d}}

Pair Same or different for both C_1 and C_2?
a b  1
a c  0
a d  1
b c  0
b d  1
c d  0

R(C_1,C_2) = 3/6
```

Probability-based clustering

```
Input: instances {X_1, ..., X_n}, number of clusters k
Output: models and probabilities {(M_1,P_1), ..., (M_k,P_k)}

M_1, ..., M_k = (randomly) initialized model
P_1, ..., P_k = 1/k

repeat
   for each instance X_i:               # Expectation step
      calculate weights w_i1, ..., w_ik,
      where w_ij = P(M_j|X_i) = P_j*P(X_i|M_j)/P(X_i)

   update each (M_j,P_j),                # Maximization step
     using (X_1, w_1j), ..., (X_n,w_nj)

until likelihood does not increase more than epsilon
```

Probability-based clustering

- Gaussian Mixture Models; each model is a normal distribution function, e.g., for the univariate case:

$$f(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where

$$\mu = \frac{w_1 x_1 + \ldots + w_n x_n}{w_1 + \ldots + w_n}$$

and

$$\sigma^2 = \frac{w_1 (x_1 - \mu)^2 + \ldots + w_n (x_n - \mu)^2}{w_1 + \ldots + w_n}$$

- naïve Bayes; each cluster is handled as a class, and $P(x_1 \& \ldots \& x_m | c) = P(x_1|c) \cdots P(x_m|c)$

Top-down (divisive) hierarchical clustering

```
Input: instances X
Output: a binary hierarchical cluster H

if |X| = 1 then return H = I

{C1, C2} = a partition of X

H1 = TopDownClustering(C1)
H2 = TopDownClustering(C2)

H = {H1,H2}
```

Bottom-up (agglomerative) clustering

```
Input: instances X1, ..., Xn
Output: a binary hierarchical cluster H

H = {{X1}, ..., {Xn}}

for i = 1 to n-1:
   select two elements H1 and H2 in H
   H = H \ {H1,H2} U {{H1,H2}}
```

Bottom-up (agglomerative) clustering (cont.)

Selecting the two nearest clusters (A and B) to merge based on:

- Complete-linkage

$$max\{d(a,b) : a \in A, b \in B\}$$

- Single-linkage

$$min\{d(a,b) : a \in A, b \in B\}$$

- Average-linkage

$$\frac{1}{|A| \cdot |B|} \sum_{a \in A} \sum_{b \in B} d(a,b)$$

- Ward's minimum variance criterion

$$SSE(\{\{A,B\}\}) - SSE(\{A,B\})$$

Agglomerative clustering (example)

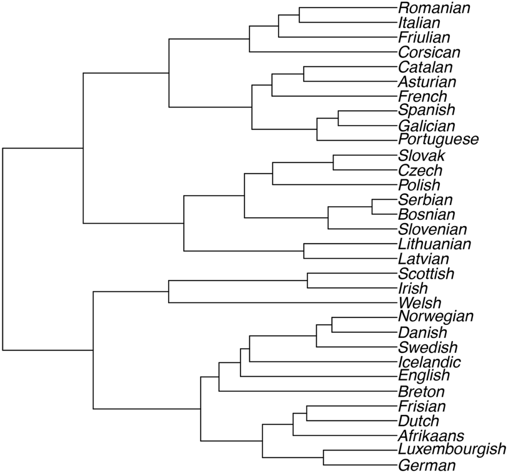
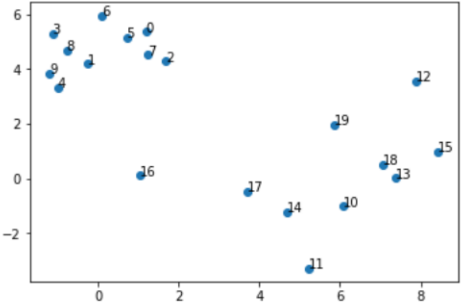


Figure taken from: Strauss T, von Maltitz MJ (2017) Generalising Wards Method for Use with Manhattan Distances. PLoS ONE 12(1): e0168288. https://doi.org/10.1371/journal.pone.0168288

Agglomerative clustering using SciPy

```python
import numpy as np
from matplotlib import pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
```

```python
a = np.random.multivariate_normal([0, 5], [[2, 1], [1, 3]], size=10)
b = np.random.multivariate_normal([5, 0], [[3, 1], [1, 4]], size=10)
X = np.concatenate((a,b))
plt.scatter(X[:,0], X[:,1])
for i in range(X.shape[0]):
    plt.text(X[i,0], X[i,1], str(i))
```



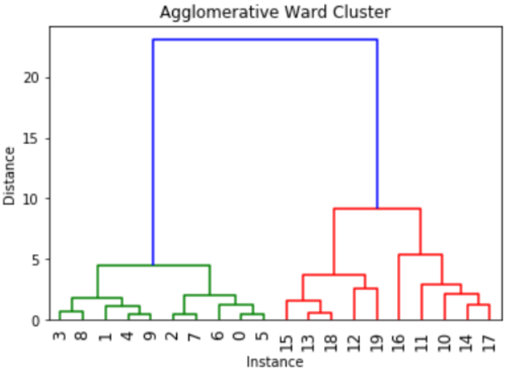
Agglomerative clustering using SciPy (cont.)

```python
Z = linkage(X,"ward")
Z
```

```
array([[ 2.        ,  7.        ,  0.49923011,  2.        ],
       [ 0.        ,  5.        ,  0.51911866,  2.        ],
       [ 4.        ,  9.        ,  0.54636394,  2.        ],
       [13.        , 18.        ,  0.58065412,  2.        ],
       [ 3.        ,  8.        ,  0.69904001,  2.        ],
       [ 1.        , 22.        ,  1.20402807,  3.        ],
       [14.        , 17.        ,  1.23777125,  2.        ],
       [ 6.        , 21.        ,  1.28347217,  3.        ],
       [15.        , 23.        ,  1.62659978,  3.        ],
       [24.        , 25.        ,  1.87422307,  5.        ],
       [20.        , 27.        ,  2.09445922,  5.        ],
       [10.        , 26.        ,  2.18769743,  3.        ],
       [12.        , 19.        ,  2.58535726,  2.        ],
       [11.        , 31.        ,  2.95339507,  4.        ],
       [28.        , 32.        ,  3.68845284,  5.        ],
       [29.        , 30.        ,  4.51138975, 10.        ],
       [16.        , 33.        ,  5.32942192,  5.        ],
       [34.        , 36.        ,  9.1656    , 10.        ],
       [35.        , 37.        , 23.0267602 , 20.        ]])
```

Agglomerative clustering using SciPy (cont.)

```python
plt.title('Agglomerative Ward Cluster')
plt.xlabel('Instance')
plt.ylabel('Distance')
d = dendrogram(Z,leaf_rotation=90)
```



Frequent itemset and association rule mining

- An *itemset* is an (unordered) set of items
- A *database* is a multiset of itemsets
- The (absolute) support of an itemset S, given a database D, is:

$$sup(S, D) = |\{I : I \in D \;\&\; S \subseteq I\}|$$

- An association rule is a rule on the form $A \rightarrow B$, where A and B are itemsets
- The confidence of an association rule $A \rightarrow B$, given a database D, is

$$conf(A \rightarrow B, D) = \frac{sup(A \cup B, D)}{sup(A, D)}$$

The Apriori algorithm

```
Input: a database D and minimum support s
Output: frequent itemsets F

L_1 = all items i ∈ D such that sup({i},D) ≥ s
k=2
while L_k-1 ≠ {}:
   C_k = {a ∪ {b} : a ∈ L_k-1 & b ∉ a} \
        {c : {s: s ⊆ c & s = |k-1|} ⊄ of L_k-1}
   for i in D:
      D_i = {c : c in C_k and c ⊆ i}
      for c in D_i:
         count[c] += 1
   L_k = {c : c in C_k & count[c] ≥ s}
   k = k+1
return F = L_1 ∪ ... ∪ L_k-1
```

The Apriori algorithm (example)

```
D = {{a,b},{a,c},{a,b,c},{a,b,c,d},{c,d,e}}
s = 2

L_1 = {{a},{b},{c},{d}}

C_2 = {{a,b},{a,c},{a,d},{b,c},{b,d},{c,d}}
L_2 = {{a,b},{a,c},{b,c},{c,d}}

C_3 = {{a,b,c}}
L_3 = {{a,b,c}}

C_4 = {}
L_4 = {}
```

Find association rules

```
Input: frequent itemsets F, database D,
       confidence c
Output: a set of association rules A

A = {}
for each itemset f in F:
    A += {a → b: ∅ ⊂ a ⊂ f & b = f \ a &
                conf(a → b,D) ≥ c}
```

Find association rules (example)

```
F = {{a},{b},{c},{d},
     {a,b},{a,c},{b,c},{c,d},
     {a,b,c}}
D = {{a,b},{a,c},{a,b,c},{a,b,c,d},{c,d,e}}
c = 1

A:
{b} -> {a}
{b,c} -> {a}
```

Frequent itemset mining using MLxtend

```python
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
```

```python
df = pd.read_csv("tic-tac-toe.txt")
transactions = [[col+"="+row[col] for col in df.columns] for _,row in df.iterrows()]
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
database = pd.DataFrame(te_ary, columns=te.columns_)
database
```

	CLASS=negative	CLASS=positive	bottom-left-square=b	bottom-left-square=o	bottom-left-square=x	bottom-middle-square=b	bottom-middle-square=o	bottom-middle-square=x	bottom-right-square=b	bottom-right-square=o
0	False	True	False	False	True	False	True	False	False	True
1	False	True	False	True	False	False	False	True	False	True
2	False	True	False	True	False	False	True	False	False	False
3	False	True	False	True	False	True	False	False	True	False
4	False	True	True	False	False	False	True	False	True	False
5	False	True	True	False	False	True	False	False	False	True

Frequent itemset mining using MLxtend (cont.)

```
frequent_itemsets = apriori(database, min_support=0.05,use_colnames=True)
frequent_itemsets
```

	support	itemsets
0	0.346555	(CLASS=negative)
1	0.653445	(CLASS=positive)
2	0.213987	(bottom-left-square=b)
3	0.349687	(bottom-left-square=o)
4	0.436326	(bottom-left-square=x)
5	0.260960	(bottom-middle-square=b)

Frequent itemset mining using MLxtend (cont.)

```python
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=1.0)
for _,rule in rules.iterrows():
    print("{} -> \n{} \n".format(list(rule["antecedents"]),list(rule["consequents"])))

```

```
['middle-middle-square=o', 'top-right-square=o', 'bottom-left-square=o'] ->
['CLASS=negative']

['middle-middle-square=o', 'top-left-square=o', 'bottom-right-square=o'] ->
['CLASS=negative']

['bottom-left-square=o', 'CLASS=positive', 'top-right-square=o'] ->
['middle-middle-square=x']

['bottom-middle-square=x', 'bottom-left-square=x', 'bottom-right-square=x'] ->
['CLASS=positive']

['bottom-left-square=x', 'middle-left-square=x', 'top-left-square=x'] ->
['CLASS=positive']
```

Concluding remarks

- We have considered some approaches for unsupervised learning; clustering and frequent itemset mining
- The output of some of the clustering algorithms not only depends on the choice of parameters, but also to a large extent on random initializations; the algorithms typically need to be re-run several times
- Some cluster evaluation metrics rely on ground truth (cluster labels); however, if the task actually concerns labeling, then supervised learning should be employed instead
- It should be noted that most of the considered unsupervised learning techniques are computationally very costly