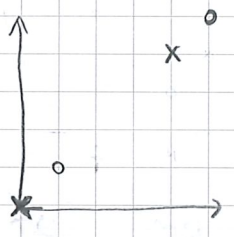
1) a) i Pam - algorithm

1. Define k arbitary points as your initial medoids.
2. Assign each non-medoid point to the closest medoid to form k clusters.
3. Calculate the total cost for the current medoid.
4. Calculate the total cost for other possible medoids.
5. Calculate the swapping cost New cost - Old cost
6. If at least one swapping cost is lower than 0 we swich to the medoids that corresponds to the lowest swapping cost.
7. repeat steps 3-6 until nothing changes

ii Swapping cost = $TC_{new} - TC_{old}$

iii Example of when swapping cost is 0:



Euclidean distance is used

	(5,5)	(1,1)
(0,0)	$\sqrt{25}$	$\sqrt{2}$
(4,4)	$\sqrt{2}$	$\sqrt{9}$

$TC = \sqrt{2} + \sqrt{2}$

our initial medoids are (5,5) and (1,1)

calculate the total cost

other possible medoids: (4,4) and (1,1)

calculate the total cost for (4,4) and (1,1)

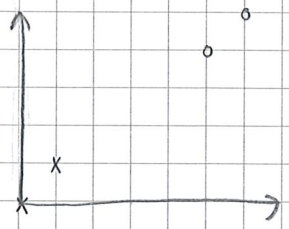
	(4,4)	(1,1)
(0,0)	$\sqrt{16}$	$\sqrt{2}$
(5,5)	$\sqrt{2}$	$\sqrt{16}$

$TC = \sqrt{2} + \sqrt{2}$

The swapping cost is the new cost - old cost

$(\sqrt{2} + \sqrt{2}) - (\sqrt{2} - \sqrt{2}) = 0$

Example of when swapping cost is strictly negative:



o initial medoids

	(4,4)	(5,5)
(1,1)	$\sqrt{9}$	$\sqrt{16}$
(0,0)	$\sqrt{16}$	$\sqrt{25}$

TC = 8

Here we check all neighbours to see if they all result in negative TC

	(4,4)	(1,1)
(0,0)	$\sqrt{16}$	$\sqrt{2}$
(1,1)	$\sqrt{9}$	0
(4,4)	0	$\sqrt{9}$
(5,5)	$\sqrt{2}$	$\sqrt{16}$

$TC = \sqrt{2} + \sqrt{2}$

	(4,4)	(0,0)
(0,0)	$\sqrt{16}$	0
(1,1)	$\sqrt{9}$	$\sqrt{2}$
(4,4)	0	$\sqrt{16}$
(5,5)	$\sqrt{2}$	$\sqrt{25}$

$TC = \sqrt{2} + \sqrt{2}$

	(5,5)	(1,1)
(0,0)	$\sqrt{25}$	$\sqrt{2}$
(1,1)	$\sqrt{16}$	0
(4,4)	$\sqrt{2}$	$\sqrt{9}$
(5,5)	0	$\sqrt{16}$

$TC = \sqrt{2} + \sqrt{2}$

	(5,5)	(0,0)
(0,0)	$\sqrt{25}$	0
(1,1)	$\sqrt{16}$	$\sqrt{2}$
(4,4)	$\sqrt{2}$	$\sqrt{16}$
(5,5)	0	$\sqrt{25}$

$TC = \sqrt{2} + \sqrt{2}$

all of the total cost for the neighbours are smaller than 8.

This will lead to all swapping costs being negative (new cost - old cost)

There for in this example the swapping cost is strictly negative.

b) PAM, CLARANS 0

c) None 1

2)

Dissimilarity matrix

	1	2	3	4	5
1	0				
2	8	0			
3	3	4	0		
4	1	7	9	0	
5	10	2	6	5	0

Agglomorative Clustering

Hierarchial clustering when we start with all the objects in separate clusters and merge dusters that are most similar until all objects are in one cluster.

We use compleate-link clustering, which means that the distance between two clusters is the maximum distance between two points in those clusters. Therefore, when we calculate the distance between clusters we use the maximum distance.

First we mearge 1 and 4 since they have the shortest distance.

{1,4} 2 3 5

	{1,4}	2	3
{1,4}	0		
2	8	0	
3	9	4	0
5	10	2	6

$d_{(1,4),2} = \max(d_{1,2}, d_{4,2}) = \max(8, 7) = 8$

$d_{(1,4),3} = \max(d_{1,3}, d_{4,3}) = \max(3, 9) = 9$

$d_{(1,4),5} = \max(d_{1,5}, d_{4,5}) = \max(10, 5) = 10$ threshold

Then we merge 2 and 5 since they have the shortest distance now.

{1,4} 3 {2,5}

	{1,4}	3
{1,4}	0	
3	9	0
{2,5}	10	6

$d_{(2,5),3} = \max(d_{2,3}, d_{5,3}) = \max(4, 6) = 6$

$d_{(2,5),(1,4)} = \max(d_{2,(1,4)}, d_{5,(1,4)}) = \max(8, 10) = 10$

2, 5

Then we merge 3 and {2,5}

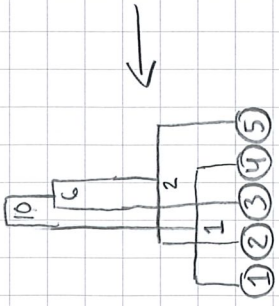
{1,4} {2,3,5}

	{1,4}
{1,4}	0
{2,3,5}	10

$d_{(1,4),(2,3,5)} = \max(d_{(1,4),3}, d_{(1,4),(2,5)}) = \max(9, 10) = 10$

Then we merge {1,4} and {2,3,5}

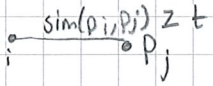
Dendogram



categorical data

3) Rock-algorithm

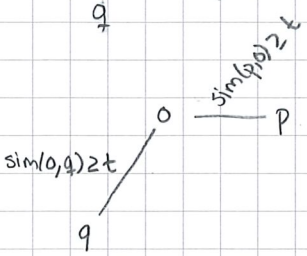
Rock uses a link measure to create clusters.
We take in a threshold describing the similarity we want for to points to be neighbours



points p and j are neighbours if the sim between the two points are higher or equal to some threshold t.

points p and q are common neighbours if they have a neighbour o that are neighbour to both p and q

o is the common neighbour



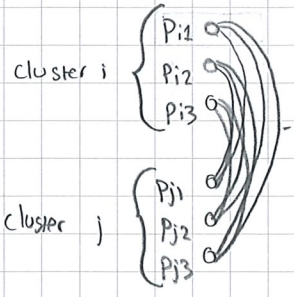
link for objects

the link measure is the number of common neighbours between two objects. the link between q and p are the number of points that are neigbours to both q and p.

link for clusters

$$\sum_{p_i \in C_i,\ p_j \in C_j} link(p_i, p_j)$$

this is the sum of all links where point p_i is in cluster i and p_j is in cluster j.



→ sum of all these common neighbours are the link for clusters

3) G = Goodness measure

$$G = \frac{\text{links for clusters}}{\text{Expected number of links for clusters}} = \frac{\sum_{i \in C_i, j \in C_j} (p_i, p_j)}{E\left(\sum_{i \in C_i, j \in C_j} (p_i, p_j)\right)}$$

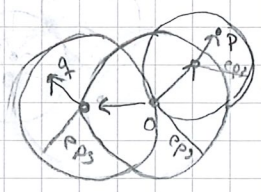
This is good for example when two clusters consist of different number of points.

It is quite likely that points in cluster A will find common neighbours in cluster B. Therefor the links between those clusters will be more than links between two small clusters of equal sizes. Therefor we use the goodness measure.

3

4) a) DBSCAN

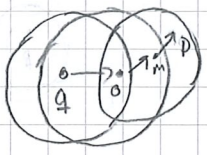
p and q are desity-connected if there exist a point o from where both p and q are density reachable



Each circle has radius eps and points ≥ minpts. Each circle has a core point.

Here p is not density-reachable from q since q is not a core-point.

a point is density-reachable from a core-point if that corepoint has a chain of directly-density reachable points to that point.



here p is density reachable from q.

In summary, p does not have to be density-reachable from q when they are density-connected because q doesn't have to be a core point for them to be density connected. However, q has to be a core point if p is density-reachable from q.

b) OPTICS is ran before DBSCAN in order to find a good eps. This is good since DBScan is sensitive to the eps parameter.

5)

	A	B	C	D	E	F	G
item K	gold	(0,0)	Y	N	Y	N	silver
item L	bronze	(1,1)	N	N	N	N	missing

keep all dummies

Distance A: ordinal variable. Transform to assymetric binary (like dummies)

	item K: gold	0	item K: silver	0	item K: bronze	0
item L: gold	0	0	silver	0	0	bronze	0	1
item L: 0	1	0	0	0	1	0	0	0

$\frac{b+c}{a+b+c} = \frac{1}{1} = 1$ 0 1

$distance = \frac{\sum \partial d}{\sum \partial}$ where $\partial = 0$ if one value is missing or if variable is assymetric binary and 0, and 1 otherwise

$distance = \frac{1+1}{2} = 1$

Distance B: manhattan distance $|x_1 - y_1| + |x_2 - y_2| \cdots |x_n - y_n|$

$distance = 1 + 1 = 2$

Distance C: binary symmetric →

	item i: 1	item i: 0
item j: 1	a	b
item j: 0	c	d

$distance = \frac{b+c}{a+b+c+d}$

	item K: Yes	item K: No
item L: Yes	0	0
item L: No	1	0

$distance = \frac{0+1}{0+0+1+0} = 1$

Distance D: binary symmetric

	item K: Yes	item K: No
item L: Yes	0	0
item L: No	0	1

$distance = \frac{0+0}{0+0+0+1} = 0$

Distance E: binary assymetric →

	item i: 1	item i: 0
item j: 1	a	b
item j: 0	c	d

$distance = \frac{b+c}{a+b+c}$

	item k: Yes	item k: No
item l: Yes	0	0
item l: No	1	0

$distance = \frac{1+0}{0+0+1} = 1$

Distance F: binary assymetric

		item k	
		Yes	No
item l	Yes	0	0
	No	0	1

$$\text{distance} = \frac{0+0}{0+0+0}$$

Distance G: missing value

$$\partial = \begin{cases} 0 & \text{if missing value or assymetric binary and 0} \\ 1 & \text{other wise} \end{cases}$$

$$\text{distance between item k and item l} = \frac{\sum \partial d}{\sum \partial}$$

$\sum \partial$ ← number of ∂

$$= \frac{1 \cdot 1 + 1 \cdot 2 + 1 \cdot 1 + 1 \cdot 0 + 1 \cdot 1 + 0 + 0}{1+1+1+1+1+0+0} = \frac{5}{5} = 1$$

The distance between the items is 1.

2

6 a) Apriori property

Subsets of frequent itemsets are also frequent. 1

b) Candidates C_k are produceeed by self-joining L_{k-1} and prune itemsets that have subsets that are not in L_{k-1}. 1

C_1 consist of the 1-itemssets in the database

c) An antimonotone constraint is checked for when minsup is checked. If an itemset in C_k does not fulfill the constrained it is not in L_k or the output. 1
The reason we can prune it altogether is because If an itemset does not fulfill an antimonotone constraint neither will a superset of that itemset. So we don't have to check again.

6) d) Apriori algorithm is proved by induction

1. $L_k \subseteq C_k$ is true for $k = 1$ (seen by 1st line in pseudocode)
2. We assume $L_{k-1} \subseteq C_{k-1}$ (induction hypothesis)
3. We prove for $L_k \subseteq C_k$

   i. assume contradiction: there exists an itemset I such that $I \in L_k$, $I \notin C_k$ (I is in L_k, but not in C_k)

   ii. induction hypothesis $L_{k-1} \subseteq C_{k-1}$

   iii. C_k is made from self-joining L_{k-1} (apriori-gen)

   since I is a large itemset $\in L_k$ it was not pruned from C_k, therfore $I \in C_k$

   This is a contradiction to our assumption in i, and therefor apriori is correct for $L_k \subseteq C_k$.

2

7)

Tid	Items
1	C B A
2	D C A
3	A B
4	A B
5	A D
6	A D

minsup = 1

Scan the database to find frequent 1-itemsets

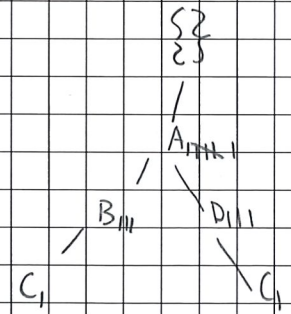
	sup
C	2
B	3
A	6
D	3

order: A, B, D, C
all have sup ≥ minsup all go in the output

Reorder the transactions in sup descending order

Tid	Items
1	A B C
2	A D C
3	A B
4	A B
5	A D
6	A D

construct Fp-tree



Conditional data bases

A:	—
B:	A:3
C:	A B:1 A D:1
D:	A:3

B - conditional database

items A
support 3
support ≥ minsup ok!
output: A B

{}
|
A_3

AB - conditional database empty, we don't need to check further

C - conditional database

items A, B, D
support 2, 1, 1 dont need to reorder
support ≥ minsup for all items
output: AC, BC, DC

{}
|
A_2
B_1 D_1

AC - conditional database
empty, we dont need to check further

BC - conditional database
items A
support 1
support ≥ minsup ok!
output: A B C

{}
|
A_1

ABC - conditional database
empty, we dont need to check further

DC - conditional database
items A
support 1
support ≥ minsup ok!
output: A DC

{}
|
A_1

ADC - conditional database
empty, we don't need to check further

D - conditional database
items A
support 3
support ≥ minsup ok!
output: A D

{}
|
A_3

AD - conditional database
empty, we don't need to check further

Output: {A, B, C, D, AB, AC, BC, DC, ABC, ADC, AD}

8) a) Monotone constraint: non-negative prices

sum(prices) ≥ 5

Antimonotone constraint:

sum(prices) ≤ 5

Convertible monotone, but not monotone, constraint:

avg(prices) ≥ 5 with respect to decreasing price order

Convertible antimonotone, but not antimonotone, constraint:

avg(prices) ≥ 5 with respect to increasing price order

2

b) ABC rules, confidence ≥ 50%

$c(AB \rightarrow C) = \frac{sup(ABC)}{sup(AB)} = \frac{1}{3} \approx 33\%$ not a rule, we don't have to check further

$c(AC \rightarrow B) = \frac{sup(ABC)}{sup(AC)} = \frac{1}{1} = 100\%$ RULE

$c(BC \rightarrow A) = \frac{sup(ABC)}{sup(BC)} = \frac{1}{1} = 100\%$ RULE

2

$c(A \rightarrow CB) = \frac{sup(ABC)}{sup(A)} = \frac{1}{6}$ not a rule

$c(C \rightarrow AB) = \frac{sup(ABC)}{sup(C)} = \frac{1}{2} = 50\%$ RULE

$c(B \rightarrow CA) = \frac{sup(ABC)}{sup(B)} = \frac{1}{3} \approx 33\%$ not a rule

$c(C \rightarrow AB) = \frac{sup(ABC)}{sup(C)} = \frac{1}{2} = 50\%$ RULE

RULES: AC → B, BC → A, C → AB

c) given that X and Y are dependent,

$X \rightarrow Y$ $Y \rightarrow X$ is a causal rule

0