Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Thomas Seidl
Max Berrendorf, Julian Busch

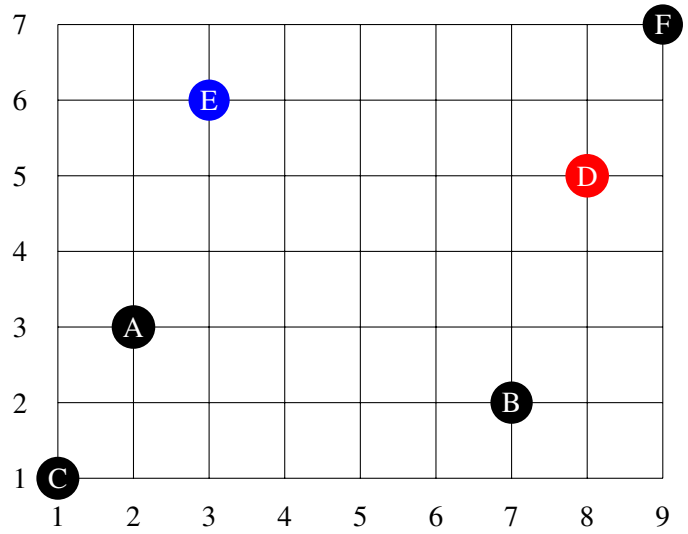
Knowledge Discovery and Data Mining I
WS 2018/19

Exercise 6: k-Medoid, EM, DBSCAN

Exercise 6-1 K-Medoid (PAM)

Consider the following 2-dimensional data set:

	A	B	C	D	E	F
x_1	2	7	1	8	3	9
x_2	3	2	1	5	6	7



(a) Perform the first loop of the PAM algorithm ($k = 2$) using the Manhattan distance. Select D and E (highlighted in the plot) as initial medoids and compute the resulting medoids and clusters.
Hint: When $C(m)$ denotes the cluster of medoid m, and M denotes the set of medoids, then the total distance TD may be computed as

$$TD = \sum_{m \in M} \sum_{o \in C(m)} d(m, o)$$

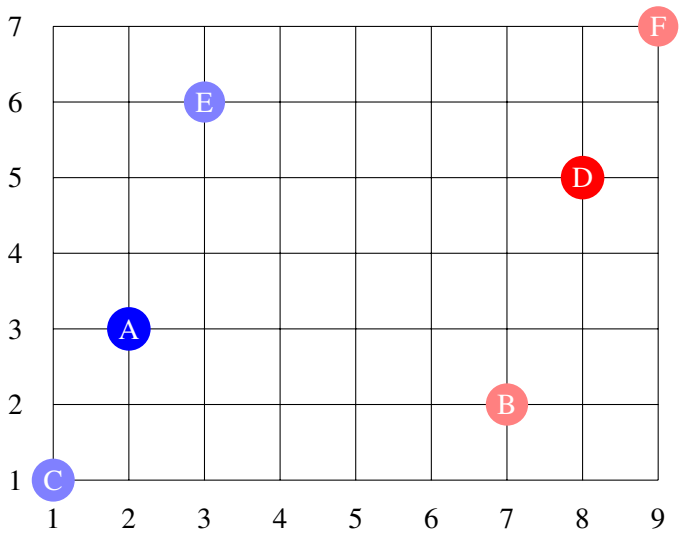
We have the following distance values (values which are clear by symmetry and reflexivity are left out):

	B	C	D	E	F
A	6	3	8	4	11
B		7	4	8	7
C			11	7	14
D				6	3
E					7

The following table shows assignments and $TD_{m \leftrightarrow n}$ value for each pair $(m, n) \in M \times N$ with $M = \{D, E\}$ and $N = \{A, B, C, F\}$.

Medoids		Assignment						TD
m_1	m_2	A	B	C	D	E	F	
D	E	1	0	1	0	1	0	**18**
D	A	1	0	1	0	1	0	**14**
D	B	1	1	1	0	0	0	22
D	C	1	0	1	0	0	0	16
D	F	0	0	0	0	0	1	29
E	A	1	1	1	0	0	0	22
E	B	0	1	0	1	0	0	22
E	C	1	1	1	0	0	0	23
E	F	0	1	0	1	0	1	21

The table shows that swapping E and A yields the largest improvement in terms of TD. The updated clustering after the first iteration is shown in the following figure.



(b) How can the clustering result $C_1 = \{A, B, C\}, C_2 = \{D, E, F\}$ be obtained with the PAM algorithm ($k = 2$) using the weighted Manhattan distance

$$d(x, y) = w_1 \cdot |x_1 - y_1| + w_2 \cdot |x_2 - y_2|?$$

Assume that B and E are the initial medoids and give values for the weights w_1 and w_2 for the first and second dimension respectively.

Consider $(w_1, w_2) = (0, 1)$, i.e. the distance is computed solely based on the second dimension. Then, we can use the reduced data set:

	A	B	C	D	E	F
x_2	3	2	1	5	6	7

Hence, we have the following distance values (values which are clear by symmetry and reflexivity are left out):

	B	C	D	E	F
A	1	2	2	3	4
B		1	3	4	5
C			4	5	6
D				1	2
E					1

The following tables shows assignments and TD values for the initial setting as well as for all $m \leftrightarrow n$ for $(m, n) \in M \times N$ with $M = \{B, E\}$, and $N = \{A, C, D, F\}$:

Medoids		Assignment						TD
m_1	m_2	A	B	C	D	E	F	
B	E	0	0	0	1	1	1	**4**
B	A	1	0	0	1	1	1	10
B	C	0	0	1	0	0	0	13
B	D	0	0	0	1	1	1	5
B	F	0	0	0	1	1	1	5
A	E	1	1	1	0	0	0	5
C	E	1	1	1	0	0	0	5
D	E	1	1	1	1	0	0	10
F	E	0	0	0	0	0	1	13

Hence, these medoids are the final ones and the partitioning stays stable.

Exercise 6-2 Convergence of PAM

Show that the algorithm PAM converges.

Let:

- M_i = Model (= Set of medoids) after the i-th iteration
- $TD(M_i)$ = Error of the model M_i.

Proof by contradiction:
Assume that PAM does not converge, i.e., starting from an arbitrary initialization, the algorithm performs an unbounded number of iterations $i \in \mathbb{N}$, such that TD monotonically decreases:

$$\forall i \in \mathbb{N} : TD(M_i) > TD(M_{i+1}).$$

However, the number of possible models in bounded by the number of possibilities to choose k medoids from the n objects. There are exactly $\binom{n}{k}$ such models. Since naturally $|\mathbb{N}| > \binom{n}{k}$, it follows that there exist iterations $t_1, t_2 \in \mathbb{N}$ with $t_1 < t_2$ and $M_{t_1} = M_{t_2}$.
Then $TD(M_{t_1}) = TD(M_{t_2})$, in contradiction to our assumption.
Thus, PAM converges at the very latest after all $\binom{n}{k}$ models have been tested (in which case the global optimum would have been found). Naturally, PAM tries to prevent this *complete enumeration* of all possible solutions for performance reasons.

Exercise 6-3 Assignments in EM-Algorithm

Given a data set with 100 points consisting of three Gaussian clusters A, B and C and the point p.

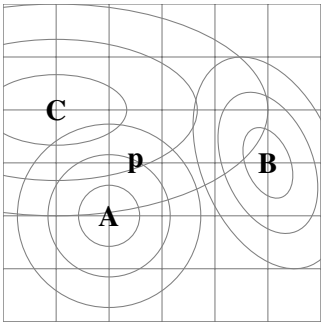
The cluster A contains 30% of all objects and is represented using the mean of all his points $\mu_A = (2, 2)$ and the covariance matrix $\Sigma_A = \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix}$.

The cluster B contains 20% of all objects and is represented using the mean of all his points $\mu_B = (5, 3)$ and the covariance matrix $\Sigma_B = \begin{pmatrix} 2 & 1 \\ 1 & 4 \end{pmatrix}$.

The cluster C contains 50% of all objects and is represented using the mean of all his points $\mu_C = (1, 4)$ and the covariance matrix $\Sigma_C = \begin{pmatrix} 16 & 0 \\ 0 & 4 \end{pmatrix}$.

The point p is given by the coordinates $(2.5, 3.0)$.
Compute the three probabilities of p belonging to the clusters A, B and C.

The following sketch is not exact, and only gives a rough idea of the cluster locations:



We have

$$\gamma_i(p) := \pi_i \cdot \mathcal{N}(p \mid \mu_i, \Sigma_i) = \pi_i \cdot \frac{1}{\sqrt{(2\pi)^2 \det(\Sigma_i)}} \exp\left(-\frac{1}{2}(p - \mu_i)^T \Sigma_i^{-1} (p - \mu_i)\right) \tag{1}$$

Substituting π_i, μ_i, Σ_i by the given parameters for each cluster yields:

Cluster A For A we have $\pi_A = \frac{3}{10}$, $\Sigma_A = \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix}$, and $\mu_A = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$. First, we compute

$$\det(\Sigma_A) = \det \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix} = 3^2 = 9.$$

Hence, we have

$$\Sigma_A^{-1} = \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix}^{-1} = \frac{1}{9} \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix} = \frac{1}{3} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Furthermore,

$$p - \mu_A = \begin{pmatrix} 2.5 \\ 3 \end{pmatrix} - \begin{pmatrix} 2 \\ 2 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 \\ 2 \end{pmatrix},$$

and moreover

$$(p - \mu_A)^T \Sigma_A^{-1} (p - \mu_A) = \frac{1}{2 \cdot 3 \cdot 2} \begin{pmatrix} 1 \\ 2 \end{pmatrix}^T \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \frac{1^2 + 2^2}{12} = \frac{5}{12}.$$

Finally, we can use these values together with (1) to obtain

$$\begin{aligned} \gamma_A(p) &= \pi_A \cdot \frac{1}{\sqrt{(2\pi)^2 \det(\Sigma_A)}} \exp\left(-\frac{1}{2}(p - \mu_A)^T \Sigma_A^{-1} (p - \mu_A)\right) \\ &= \frac{1}{20\pi} \exp\left(-\frac{5}{24}\right) \\ &\approx 0.0129223682965846 \end{aligned}$$

Cluster B For B we have $\pi_B = \frac{1}{5}$, $\Sigma_B = \begin{pmatrix} 2 & 1 \\ 1 & 4 \end{pmatrix}$, and $\mu_B = \begin{pmatrix} 5 \\ 3 \end{pmatrix}$. First, we compute

$$\det(\Sigma_B) = \det \begin{pmatrix} 2 & 1 \\ 1 & 4 \end{pmatrix} = 2 \cdot 4 - 1 \cdot 1 = 7.$$

Hence, we have

$$\Sigma_B^{-1} = \begin{pmatrix} 2 & 1 \\ 1 & 4 \end{pmatrix}^{-1} = \frac{1}{7} \begin{pmatrix} 4 & -1 \\ -1 & 2 \end{pmatrix}.$$

Furthermore,

$$p - \mu_B = \begin{pmatrix} 2.5 \\ 3 \end{pmatrix} - \begin{pmatrix} 5 \\ 3 \end{pmatrix} = -\frac{1}{2} \begin{pmatrix} 5 \\ 0 \end{pmatrix},$$

and moreover,

$$(p - \mu_B)^T \Sigma_B^{-1} (p - \mu_B) = \frac{1}{2 \cdot 7 \cdot 2} \begin{pmatrix} 5 \\ 0 \end{pmatrix}^T \begin{pmatrix} 4 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} 5 \\ 0 \end{pmatrix} = \frac{1}{28} \begin{pmatrix} 5 \\ 0 \end{pmatrix}^T \begin{pmatrix} 20 \\ -5 \end{pmatrix} = \frac{100}{28} = \frac{25}{7}.$$

Finally, we can use these values together with (1) to obtain

$$\begin{aligned} \gamma_B(p) &= \pi_B \cdot \frac{1}{\sqrt{(2\pi)^2 \det(\Sigma_B)}} \exp\left(-\frac{1}{2} (p - \mu_B)^T \Sigma_B^{-1} (p - \mu_B) \right) \\ &= \frac{1}{10\sqrt{7}\pi} \exp\left(-\frac{25}{14} \right) \\ &\approx 0.00201732210214117 \end{aligned}$$

Cluster C For C we have $\pi_C = \frac{1}{2}$, $\Sigma_C = \begin{pmatrix} 16 & 0 \\ 0 & 4 \end{pmatrix}$, and $\mu_C = \begin{pmatrix} 1 \\ 4 \end{pmatrix}$. First, we compute

$$\det(\Sigma_C) = \det \begin{pmatrix} 16 & 0 \\ 0 & 4 \end{pmatrix} = 16 \cdot 4 = 64.$$

Hence, we have

$$\Sigma_C^{-1} = \begin{pmatrix} 16 & 0 \\ 0 & 4 \end{pmatrix}^{-1} = \frac{1}{64} \begin{pmatrix} 4 & 0 \\ 0 & 16 \end{pmatrix} = \frac{1}{16} \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}.$$

Furthermore,

$$p - \mu_C = \begin{pmatrix} 2.5 \\ 3 \end{pmatrix} - \begin{pmatrix} 1 \\ 4 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 3 \\ -2 \end{pmatrix},$$

and moreover,

$$(p - \mu_C)^T \Sigma_C^{-1} (p - \mu_C) = \frac{1}{2 \cdot 16 \cdot 2} \begin{pmatrix} 3 \\ -2 \end{pmatrix}^T \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix} \begin{pmatrix} 3 \\ -2 \end{pmatrix} = \frac{1}{64} \begin{pmatrix} 3 \\ -2 \end{pmatrix}^T \begin{pmatrix} 3 \\ -8 \end{pmatrix} = \frac{25}{64}.$$

Finally, we can use these values together with (1) to obtain

$$\begin{aligned} \gamma_C(p) &= \pi_C \cdot \frac{1}{\sqrt{(2\pi)^2 \det(\Sigma_C)}} \exp\left(-\frac{1}{2} (p - \mu_C)^T \Sigma_C^{-1} (p - \mu_C) \right) \\ &= \frac{1}{32\pi} \exp\left(-\frac{25}{128} \right) \\ &\approx 0.00818233032076434 \end{aligned}$$

Given that the point was generated by the model, these probabilities are divided by

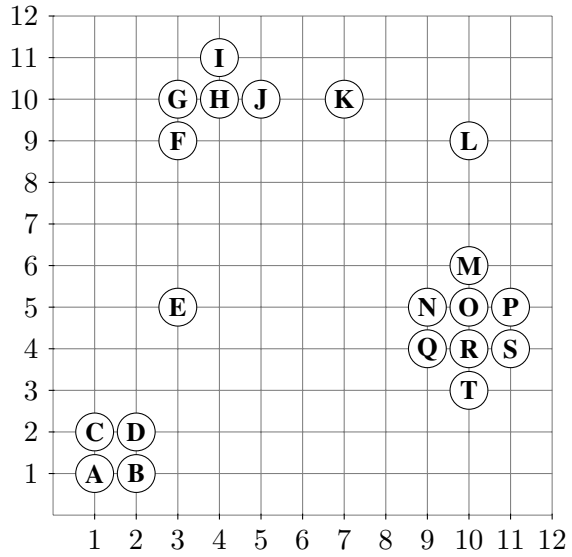
$$\gamma = \sum_{c \in \{A,B,C\}} \gamma_c \approx 0.0231220207194901,$$

yielding:

$$\begin{aligned}
\gamma'_A &= \frac{\gamma_A}{\gamma} \approx \frac{0.0129223682965846}{0.0231220207194901} \approx 55.89\% \\
\gamma'_B &= \frac{\gamma_B}{\gamma} \approx \frac{0.00201732210214117}{0.0231220207194901} \approx 8.72\% \\
\gamma'_C &= \frac{\gamma_C}{\gamma} \approx \frac{0.00818233032076434}{0.0231220207194901} \approx 35.39\%
\end{aligned}$$

Exercise 6-4 DBSCAN

Given the following data set:



As distance function, use Manhattan Distance:

$$L_1(x, y) = |x_1 - y_1| + |x_2 - y_2|$$

Compute DBSCAN and indicate which points are core points, border points and noise points.

Use the following parameter settings:

- Radius $\varepsilon = 1.1$ and *minPts* $= 2$
- Radius $\varepsilon = 1.1$ and *minPts* $= 3$
- Radius $\varepsilon = 1.1$ and *minPts* $= 4$
- Radius $\varepsilon = 2.1$ and *minPts* $= 4$
- Radius $\varepsilon = 4.1$ and *minPts* $= 5$
- Radius $\varepsilon = 4.1$ and *minPts* $= 4$

See tutorial slides.

Exercise 6-5 Properties of DBSCAN

Discuss the following questions/propositions about DBSCAN:

- Using *minPts* $= 2$, what happens to the border points?

  There are no border points: A border point must be a core points itself, since there must be at least one further object in its ϵ-neighborhood from which it is directly density reachable (otherwise it would not be connected to a cluster).

- The result of DBSCAN is deterministic w.r.t. the core and noise points but not w.r.t. the border points.

  If a border point is density-reachable from two clusters, it depends on the processing order and implementation, to which cluster it will be assigned.

- A cluster found by DBSCAN cannot consist of less than *minPts* points.

  Depends on the above case. It can happen that a border point will be assigned to another cluster, resulting in a cluster with less than $minPts$ points.

- If the dataset consists of n objects, DBSCAN will evaluate exactly n ϵ-range queries.

  Correct. This is not completely obvious from the pseudo-code presented in the lecture, but from each object, a single range query is executed to determine whether the object is a core object or not. If it is not a core object, it is a border object if it was discovered in a recursive call from another core object. Else, it is classified as a noise object until it is discovered from another core point, in which case it will be classified as a border object. In total, a chain of exactly one range query per object is performed.

  Therefore, a naive implementation will require $\mathcal{O}(n^2)$ time, since evaluating a range query with a sequential scan takes time $\mathcal{O}(n)$. Index-accelerated implementations typically runs in $\mathcal{O}(n \log n)$, since appropriate index structures are able to answer a range query in time $\mathcal{O}(\log n)$.

- On uniformly distributed data, DBSCAN will usually either assign all points to a single cluster or classify every point as noise. k-means on the other hand will partition the data into approximately equally sized partitions.

  Correct. Depending on the density threshold, DBSCAN will classify either all objects or no object as core objects. (By choosing e.g. $\varepsilon = \min_{o \in D} 10\text{-dist}(o)$ and *minPts* $= 10$, it can be provoked that a few single core points will be detected. However, finding such unfavorable parametrizations becomes more difficult with increasing dataset size.)
  For k-means on the other hand, solutions are (locally) optimal if all clusters are almost equally sized (at least if $k \cdot d \ll n$).
  However, solutions found in multiple k-means runs can be quite different.