K-means Clustering and its Generalization

lecturer: J. Matas

authors: J. Matas, T. Werner, O. Drbohlav

Formulation of the Least-Squares Clustering Problem

Given:	$\mathcal{T} = \{\mathbf{x}_l\}_{l=1}^{L}$,	the set of observations
	K	the number of desired cluster prototypes
Output:	$(\mathbf{c}_k)_{k=1}^{K}$,	the set of cluster prototypes (etalons)
	$\{\mathcal{T}_k\}_{k=1}^{K}$	the clustering (partitioning) of the data
		$\cup_{k=1}^{K} \mathcal{T}_k = \{x_l\}_{l=1}^{L}$, $\mathcal{T}_i \cap \mathcal{T}_j = \emptyset$ for $i \neq j$

The result is obtained by solving the following optimization problem:

$$(\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_K; \mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_K) = \underset{\text{all } \mathbf{c}'_k, \mathcal{T}'_k}{\operatorname{argmin}} J(\mathbf{c}'_1, \mathbf{c}'_2, ..., \mathbf{c}'_K; \mathcal{T}'_1, \mathcal{T}'_2, ..., \mathcal{T}'_K)\,,$$

where

$$J(\mathbf{c}'_1, \mathbf{c}'_2, ..., \mathbf{c}'_K; \mathcal{T}'_1, \mathcal{T}'_2, ..., \mathcal{T}'_K) = \sum_{k=1}^{K} \sum_{\mathbf{x} \in \mathcal{T}'_k} \|\mathbf{x} - \mathbf{c}'_k\|^2$$

over all clusters

over data in cluster k

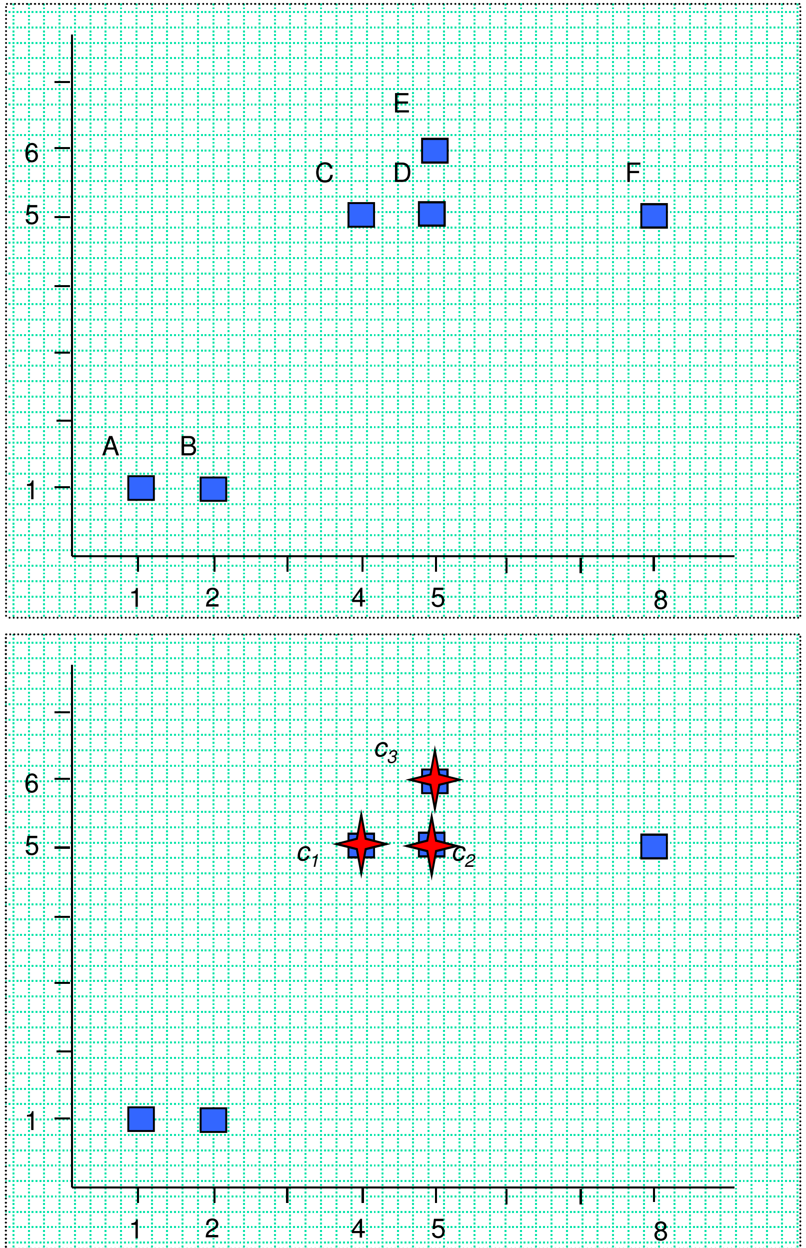
Squared Euclidean distance of data point from its etalon

K-means: Algorithm for the LS clustering problem

Given:	$\mathcal{T} = \{\mathbf{x}_l\}_{l=1}^{L}$,	the set of observations
	K	the number of desired cluster prototypes
Output:	$(\mathbf{c}_k)_{k=1}^{K}$,	the set of cluster prototypes (etalons)
	$\{\mathcal{T}_k\}_{k=1}^{K}$	the clustering (partitioning) of the data
		$\cup_{k=1}^{K} \mathcal{T}_k = \{x_l\}_{l=1}^{L}$, $\mathcal{T}_i \cap \mathcal{T}_j = \emptyset$ for $i \neq j$

1. Initialize $\mathbf{c}_k$ (e.g. by assigning random $\mathbf{x}_l$ to $\mathbf{c}_k$)
2. Assignment optimization:
   $\mathcal{T}_k = \{\mathbf{x} \in \mathcal{T} : \forall j, ||\mathbf{x} - \mathbf{c}_k||_2^2 \leq ||\mathbf{x} - \mathbf{c}_j||_2^2\}$
3. Prototype optimization:
   $\mathbf{c}_k = \frac{1}{|\mathcal{T}_k|} \sum_{\mathbf{x} \in \mathcal{T}_k} \mathbf{x}$
4. Terminate if $\mathcal{T}_k^{t+1} = \mathcal{T}_k^{t}, \forall k$; else go to 2

K-means: an Example (1/1)

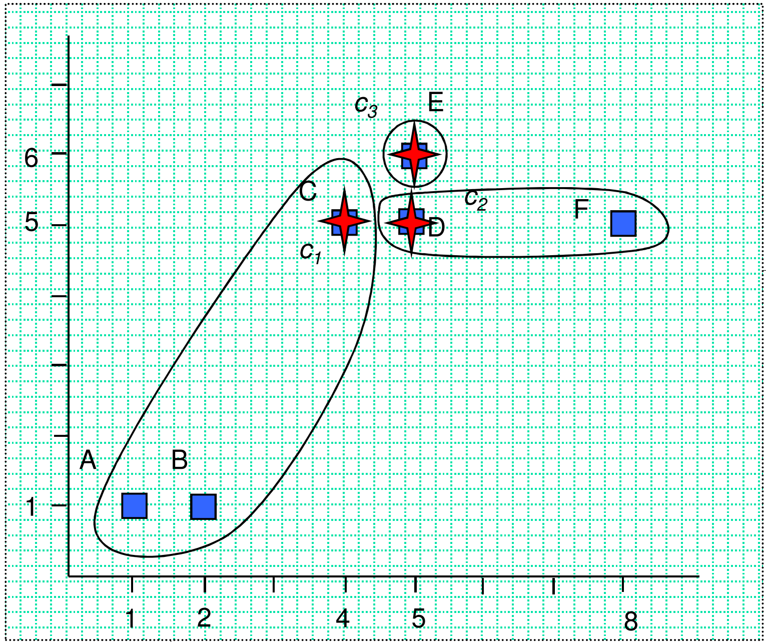


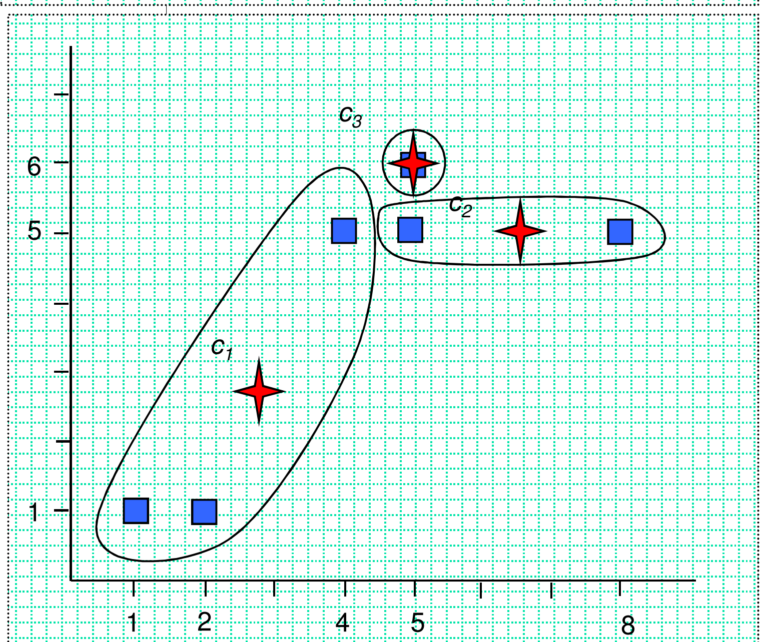
Number of clusters $K=3$

Initialization:

$\mathbf{c}_k$ = random $\mathbf{x}_l$,
without replacement

K-means: an Example (2/4)





Optimizing partitions:

Euclidean Distances

	A	B	C	D	E	F
c_1	***5***	***4,5***	***0***	1	1,4	4
c_2	5,7	5	1	***0***	1	***3***
c_3	6,4	5,8	1,4	1	***0***	3,2

Sum of squares $= J^1(.) = 9.0$

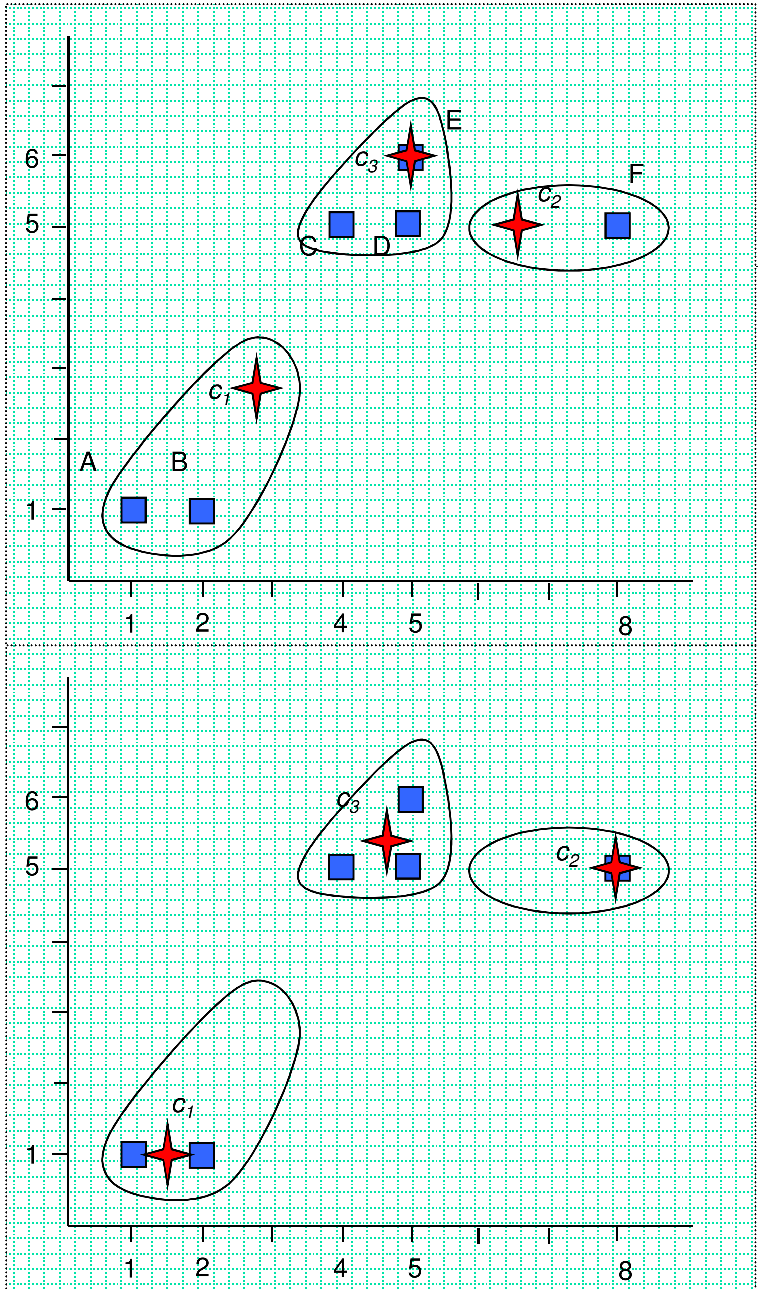
Optimizing prototypes:

$$c_1 = \left(\frac{1+2+4}{3}, \frac{1+1+5}{3}\right) = (2.3, 2.3)$$

$$c_2 = \left(\frac{5+8}{2}, \frac{5+5}{2}\right) = (6.5, 5)$$

$$c_3 = (5,6)$$

K-means: an Example (3/4)



Optimizing partitions:

Euclidean Distances

	A	B	C	D	E	F
c_1	***1,9***	***1,4***	3,1	3,8	4,5	6,3
c_2	6,8	6	2,5	1,5	1,8	***1,5***
c_3	6,4	5,8	***1,4***	***1***	***0***	3,2

Sum of squares = $J^2(.)$ = 1.78

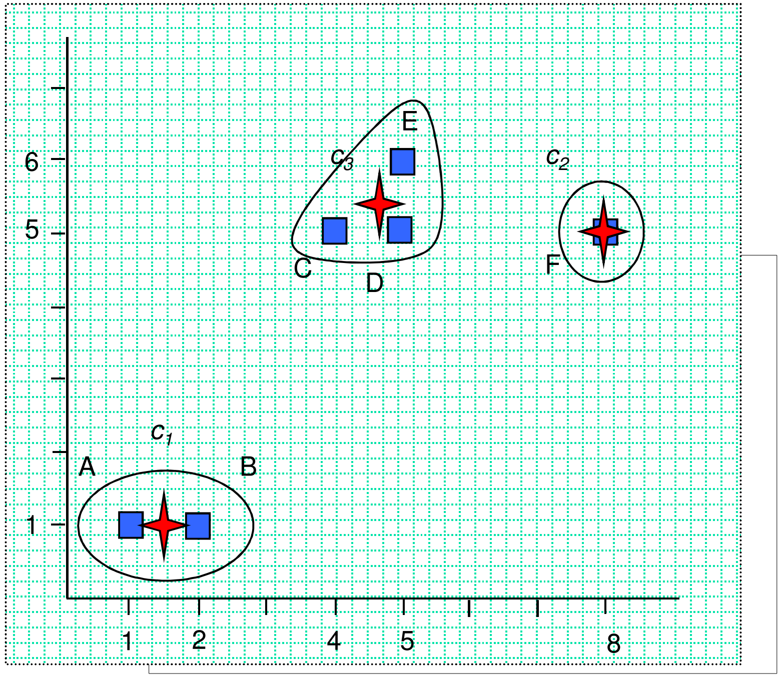
Optimizing prototypes:

$$c_1 = \left(\frac{1+2}{2}, \frac{1+1}{2}\right) = (1.5,1)$$

$$c_2 = (8,5)$$

$$c_3 = \left(\frac{4+5+5}{3}, \frac{5+5+6}{3}\right) = (4.7,5.3)$$

K-means: an Example (4/4)



Optimizing partitions:

Euclidean Distances

	A	B	C	D	E	F
c_1	***0,5***	***0,5***	4,7	5,3	6,1	7,6
c_2	8,1	7,2	4	3	3,2	***0***
c_3	5,7	5,1	***0,7***	***0,5***	***0,7***	3,3

Sum of squares = $J^3(.) = 0.31$

Assignment unchanged ⇒ terminate

K-means: Convergence Properties

- If neither Step 3 nor Step 2 changes $J(\cdot)$, the algorithm terminates.
- Step 3 (cluster centre optimization) reduces $J(\cdot)$, because for a fixed assignment $\mathcal{T}_k$, the mean over the data points in $\mathcal{T}_k$ is the optimal solution for the squared error.
- Step 2 (assignment optimization) reduces $J(\cdot)$ because for *every* $\mathbf{x}_l$, the contribution to the cost function either stays the same, or gets lower.
- The fact that $J(\cdot)$ is reduced implies that no assignment is repeated during the run of the algorithm.
- Since there is a finite number of assignmens (how many?) *the k-means algorithm converges, in a finite number of steps*, to a *local* minimum.

K-means: Notes

- Alternatively, $\mathcal{T}_{\mathbf{k}}$ is initialised, and steps 2. and 3. are swapped
- The k-means algorithm is not a guaranteed global minimum optimizer. This is easily proved by a counter-example.
- Efficiency. The complexity of Step 2. (assignment optimization) dominates, as for every observation the nearest prototype is sought. Trivially implemented, this requires $L \times K$ operations. Any idea for a speed-up?

K-means Generalization

In:	$\mathcal{T} = \{\mathbf{x_l}\}_{l=1}^{L}$,	the set of observations
	$d(.,.)$	"distance function" (may not be a metric)
Out:	$(\mathbf{c_k})_{k=1}^{K}$,	the set of cluster prototypes (etalons)
	$\{\mathcal{T}_k\}_{k=1}^{K}$	the clustering (partitioning) of the data

1. Initialize $\mathbf{c_k}$ (e.g. by assigning random $\mathbf{x_l}$ to $\mathbf{c_k}$)
2. Assignment optimization:
   $\mathcal{T}_k = \{\mathbf{x} \in \mathcal{T} : \forall j, d(\mathbf{x}, \mathbf{c_k}) \leq d(\mathbf{x}, \mathbf{c_j})\}$
3. Prototype optimization:
   $\mathbf{c_k} = \arg\min_{\mathbf{c}} \sum_{\mathbf{x} \in \mathcal{T}_k} d(\mathbf{x}, \mathbf{c})$
4. Terminate If $\mathcal{T}_k^{t+1} = \mathcal{T}_k^{t}, \forall k$; else go to 2

K-means Generalization: K-medians

In:	$\mathcal{T} = \{\mathbf{x}_l\}_{l=1}^{L}$,	the set of observations
	$d(.,.)$	$\|\|\mathbf{c} - \mathbf{x}\|\|_1$, ie. $d(.,.)$ is the L1-metric
Out:	$(\mathbf{c}_k)_{k=1}^{K}$,	the set of cluster prototypes (etalons)
	$\{\mathcal{T}_k\}_{k=1}^{K}$	the clustering (partitioning) of the data

1. Initialize $\mathbf{c}_k$ (e.g. by assigning random $\mathbf{x}_l$ to $\mathbf{c}_k$)

2. Assignment optimization:
   $\mathcal{T}_k = \{\mathbf{x} \in \mathcal{T} : \forall j, d(\mathbf{x}, \mathbf{c}_k) \leq d(\mathbf{x}, \mathbf{c}_j)\}$

3. Prototype optimization:
   $\mathbf{c}_k = \text{median}\{\mathcal{T}_k\}$

4. Terminate If $\mathcal{T}_k^{t+1} = \mathcal{T}_k^{t}, \forall k$; else go to 2

Median is the minimizer of the L1-norm in a cluster, ie.
$\text{median}\{\mathcal{T}_k\} = \mathbf{c}_\mathbf{k}^{\star} = \arg\min_{\mathbf{c}} \sum_{\mathbf{x} \in \mathcal{T}_k} ||\mathbf{x} - \mathbf{c}_\mathbf{k}||_1$

K-means Generalization: Clustering Strings

In: $\mathcal{T} = \{\mathbf{x}_l\}_{l=1}^{L}$, observations $\mathbf{x_l}$ are strings
$d(s_1, s_2)$ is the Levenshtein distance, ie. the number of edit operations to transform s_1 into s_2

Out: $(\mathbf{c}_k)_{k=1}^{K}$, the set of cluster prototypes, $\mathbf{c}_k$ are strings
$\{\mathcal{T}_k\}_{k=1}^{K}$ the clustering (partitioning) of the data

1. Initialize $\mathbf{c}_k$
2. Assignment optimization:
   $\mathcal{T}_k = \{\mathbf{x} \in \mathcal{T} : \forall j, d(\mathbf{x}, \mathbf{c}_k) \leq d(\mathbf{x}, \mathbf{c}_j)\}$
3. Prototype optimization:
   $\mathbf{c}_k = \arg\min_{\mathbf{c}} \sum_{\mathbf{x} \in \mathcal{T}_k} d(\mathbf{x}, \mathbf{c})$
4. Terminate If $\mathcal{T}_k^{t+1} = \mathcal{T}_k^{t}, \forall k$; else go to 2

.

K-means Generalization: Clustering Strings: Notes

- the calculation of $d(.,.)$ might be non-trivial
- It might be very hard to minimize $\sum_{\mathbf{x} \in \mathcal{T}_k} d(\mathbf{x}, \mathbf{c})$ over the space of all strings.
  The minimization can be restricted to $\mathbf{c} \in \mathcal{T}$.
- Is the algorithm guaranteed to terminate if Step 2. (Step 3.) is only improving $J(\cdot)$, not finding the minimimum (given fixed $\mathcal{T}$ or $\mathbf{c}_k$ respectively)?

K-means Generalization: Euclidean Clustering

Given:	$\mathcal{T} = \{\mathbf{x}_l\}_{l=1}^{L}$,	the set of observations
	K	the number of desired cluster prototypes
Output:	$(\mathbf{c}_k)_{k=1}^{K}$,	the set of cluster prototypes (etalons)
	$\{\mathcal{T}_k\}_{k=1}^{K}$	the clustering (partitioning) of the data $\cup_{k=1}^{K} \mathcal{T}_k = \{x_l\}_{l=1}^{L}$, $\mathcal{T}_i \cap \mathcal{T}_j = \emptyset$ for $i \neq j$

1. Initialize $\mathbf{c}_k$ (e.g. by assigning random $\mathbf{x}_l$ to $\mathbf{c}_k$)
2. Assignment optimization:
   $\mathcal{T}_k = \{\mathbf{x} \in \mathcal{T} : \forall j, ||\mathbf{x} - \mathbf{c}_k||_2 \leq ||\mathbf{x} - \mathbf{c}_j||_2\}$
3. Prototype optimization: no closed-form solution for *geometric median*. Use e.g. iterative Weiszfeld's algorithm.
   $\mathbf{c}_k = \mathrm{argmin}_{\mathbf{c}} \sum_{x \in \mathcal{T}_k} \|\mathbf{x} - \mathbf{c}\|_2$
4. Terminate if $\mathcal{T}_k^{t+1} = \mathcal{T}_k^{t}, \forall k$; else go to 2

macros_rpz.tex
sfmath.sty
cmpitemize.tex

Thank you for your attention.