Script generated by TTT

Title: groh: profile1 (27.05.2015)

Date: Wed May 27 08:22:56 CEST 2015

Duration: 82:49 min

Pages: 72

Social Gaming / Social Computing SS 2015

PD Dr. Georg Groh

Applied Informatics / Cooperative Systems AICOS

Technische Universität München

Finding Clusters in Profiles

Examples for profile elements that can be embedded in metric spaces:

- Location & Velocity: Metric space: ($\mathbb{R}^3$, || . ||)
- Text describing Interests: Metric space: ($\mathbb{R}^{|Voc|}$, || . ||) where Voc denotes the Vocabulary of the text.

"I like to dance samba, bake pizza, watch tv and plant trees in the garden. I also like to bake cakes." ⇒

I	2
like	2
to	2
dance	1
samba	1
bake	2
pizza	1
watch	1
tv	1
and	1
plant	1
trees	1
in	1
the	1
garden	1
also	1
cakes	1

*Often: Instead of term-frequency (tf) alone: use term-frequency * inverse document frequency (idf); idf = log (#of docs where t occurs / #of docs)*

Finding Clusters in Profiles

- How do we compute clusters in metric spaces?
- Group models: How do we compute socially meaningful clusters in metric spaces (and thus avoid quasi-groups)?
- First some notations / basics:
  - In graph clustering we had: A graph clustering **C**={C_1, C_2, ..., C_K} is a partion of V into non-empty subsets C_k
  - Now: clustering $\mathscr{C} : \mathcal{X} \to \mathcal{I}$: mapping of a metric value space $\mathcal{X}$ to a set of cluster indices $\mathcal{I}$
  - Clusterings can be:
    - exclusive or non-exclusive
    - crisp or fuzzy
    - hierarchical or non-hierarchical

Finding Clusters in Profiles

- How do we compute clusters in metric spaces?
- Group models: How do we compute socially meaningful clusters in metric spaces (and thus avoid quasi-groups)?
- First some notations / basics:
  - In graph clustering we had: A graph clustering **C**={C_1, C_2, ..., C_K} is a partion of V into non-empty subsets C_k
  - Now: clustering $\mathscr{C} : \mathcal{X} \to \mathcal{I}$: mapping of a metric value space X to a set of cluster indices I
  - Clusterings can be:
    - exclusive or non-exclusive
    - crisp or fuzzy
    - hierarchical or non-hierarchical

Finding Clusters in Profiles

- How do we compute clusters in metric spaces?
- Group models: How do we compute socially meaningful clusters in metric spaces (and thus avoid quasi-groups)?
- First some notations / basics:
  - In graph clustering we had: A graph clustering **C**={C_1, C_2, ..., C_K} is a partion of V into non-empty subsets C_k
  - Now: clustering $\mathscr{C} : \mathcal{X} \to \mathcal{I}$: mapping of a metric value space X to a set of cluster indices I
  - Clusterings can be:
    - exclusive or non-exclusive
    - crisp or fuzzy
    - hierarchical or non-hierarchical

Finding Clusters in Profiles

- Exclusive → non overlapping clusters; non-exclusive → overlapping clusters
- Hierarchical clustering → imposes a tree structure (Dendrogram) on the C_k where an edge C_i → C'_j implies C_i ⊂ C'_j;
- Crisp clusterings: Conventional characteristic functions α_k for each Cluster C_k

$$\alpha_k : \mathcal{X} \to \{0,1\} \text{ with } \alpha_k(x \in \mathcal{X}) = \begin{cases} 1 & x \in \mathcal{C}_k \\ 0 & x \notin \mathcal{C}_k \end{cases}$$

- Fuzzy clusterings: fuzzy membership function α _k for each Cluster C_k

$$\alpha_k : \mathcal{X} \to [0,1]$$

Finding Clusters in Profiles

- Exclusive → non overlapping clusters; non-exclusive → overlapping clusters
- Hierarchical clustering → imposes a tree structure (Dendrogram) on the C_k where an edge C_i → C'_j implies C_i ⊂ C'_j;
- Crisp clusterings: Conventional characteristic functions α_k for each Cluster C_k

$$\alpha_k : \mathcal{X} \to \{0,1\} \text{ with } \alpha_k(x \in \mathcal{X}) = \begin{cases} 1 & x \in \mathcal{C}_k \\ 0 & x \notin \mathcal{C}_k \end{cases}$$

- Fuzzy clusterings: fuzzy membership function α _k for each Cluster C_k

$$\alpha_k : \mathcal{X} \to [0,1]$$

Finding Clusters in Profiles

- Exclusive → non overlapping clusters; non-exclusive → overlapping clusters
- Hierarchical clustering → imposes a tree structure (Dendrogram) on the C_k where an edge C_i → C'_j implies C_i ⊂ C'_j;
- Crisp clusterings: Conventional characteristic functions α_k for each Cluster C_k

$$\alpha_k : \mathcal{X} \rightarrow \{0,1\} \text{ with } \alpha_k(x \in \mathcal{X}) = \begin{cases} 1 & x \in \mathcal{C}_k \\ 0 & x \notin \mathcal{C}_k \end{cases}$$

- Fuzzy clusterings: fuzzy membership function α _k for each Cluster C_k

$$\alpha_k : \mathcal{X} \rightarrow [0,1]$$

Metric variant of Single / Complete link clustering

- Metric variant of Single / Complete link clustering: Hierarchical, crisp, non-overlapping
- Completely analogous to graph clustering case: Start with singletons and on each level of the dendrogram merge two clusters with minimal distance (cost)
  - Single link:

$$d(\mathcal{C}_{k_1}, \mathcal{C}_{k_2}) = \min_{\{n_1, n_2 | x_{n_1} \in \mathcal{C}_{k_1} \wedge x_{n_2} \in \mathcal{C}_{k_2}\}} ||x_{n_1} - x_{n_2}||$$

  - Complete link:

$$d(\mathcal{C}_{k_1}, \mathcal{C}_{k_2}) = \max_{\{n_1, n_2 | x_{n_1} \in \mathcal{C}_{k_1} \wedge x_{n_2} \in \mathcal{C}_{k_2}\}} ||x_{n_1} - x_{n_2}||$$

Metric variant of Single / Complete link clustering

- Metric variant of Single / Complete link clustering: Hierarchical, crisp, non-overlapping
- Completely analogous to graph clustering case: Start with singletons and on each level of the dendrogram merge two clusters with minimal distance (cost)
  - Single link:

$$d(\mathcal{C}_{k_1}, \mathcal{C}_{k_2}) = \min_{\{n_1, n_2 | x_{n_1} \in \mathcal{C}_{k_1} \wedge x_{n_2} \in \mathcal{C}_{k_2}\}} ||x_{n_1} - x_{n_2}||$$

  - Complete link:

$$d(\mathcal{C}_{k_1}, \mathcal{C}_{k_2}) = \max_{\{n_1, n_2 | x_{n_1} \in \mathcal{C}_{k_1} \wedge x_{n_2} \in \mathcal{C}_{k_2}\}} ||x_{n_1} - x_{n_2}||$$

Metric variant of Single / Complete link clustering

- Metric variant of Single / Complete link clustering: Hierarchical, crisp, non-overlapping
- Completely analogous to graph clustering case: Start with singletons and on each level of the dendrogram merge two clusters with minimal distance (cost)
  - Single link:

$$d(\mathcal{C}_{k_1}, \mathcal{C}_{k_2}) = \min_{\{n_1, n_2 | x_{n_1} \in \mathcal{C}_{k_1} \wedge x_{n_2} \in \mathcal{C}_{k_2}\}} ||x_{n_1} - x_{n_2}||$$

  - Complete link:

$$d(\mathcal{C}_{k_1}, \mathcal{C}_{k_2}) = \max_{\{n_1, n_2 | x_{n_1} \in \mathcal{C}_{k_1} \wedge x_{n_2} \in \mathcal{C}_{k_2}\}} ||x_{n_1} - x_{n_2}||$$

K-Means Clustering

- General idea (also valid in graph clustering): Optimize objective function that formalizes clustering paradigm.
- K-Means: Optimize intra cluster coherence:
  - Describe cluster C_k by prototype μ_k; prototype need not be an actual pattern (If so, algorithm works with slight modifications as well)
  - Determine cluster for each pattern x_n by nearest neighbour rule:

$$\mathscr{C}(x_n) = k_a \leftrightarrow ||x_n - \mu_{k_a}|| = \min_i ||x_n - \mu_k||$$

K-Means Clustering

- General idea (also valid in graph clustering): Optimize objective function that formalizes clustering paradigm.
- K-Means: Optimize intra cluster coherence:
  - Describe cluster C_k by prototype μ_k; prototype need not be an actual pattern (If so, algorithm works with slight modifications as well)
  - Determine cluster for each pattern x_n by nearest neighbour rule:

$$\mathscr{C}(x_n) = k_a \leftrightarrow ||x_n - \mu_{k_a}|| = \min_i ||x_n - \mu_k||$$

K-Means Clustering

- K-Means: Optimize intra cluster coherence:
  - Find prototypes by optimizing objective function modeling intra cluster coherence as mean square error

$$J_{\text{SQE}} = \sum_{k=1}^{K} \sum_{\{n | x_n \in \mathcal{C}_k\}} ||x_n - \mu_k||^2$$

$$\frac{\mathrm{d}J_{\text{SQE}}}{\mathrm{d}\mu_k} \stackrel{!}{=} 0 \quad \Longrightarrow \quad \mu^k = \frac{1}{|\mathcal{C}_k|} \sum_{\{n | x_n \in \mathcal{C}_k\}} x_n$$

  - → cluster prototypes are barycenters („centers of gravity“) of their clusters.

K-Means Clustering

- K-Means: Optimize intra cluster coherence:
  - Find prototypes by optimizing objective function modeling intra cluster coherence as mean square error

$$J_{\text{SQE}} = \sum_{k=1}^{K} \sum_{\{n | x_n \in \mathcal{C}_k\}} ||x_n - \mu_k||^2$$

$$\frac{\mathrm{d}J_{\text{SQE}}}{\mathrm{d}\mu_k} \stackrel{!}{=} 0 \quad \Longrightarrow \quad \mu^k = \frac{1}{|\mathcal{C}_k|} \sum_{\{n | x_n \in \mathcal{C}_k\}} x_n$$

  - → cluster prototypes are barycenters („centers of gravity“) of their clusters.

K-Means Clustering

- K-Means: Optimize intra cluster coherence:
  - Find prototypes by optimizing objective function modeling intra cluster coherence as mean square error

$$J_{\mathrm{SQE}} = \sum_{k=1}^{K} \sum_{\{n|x_n \in \mathcal{C}_k\}} ||x_n - \mu_k||^2$$

$$\frac{\mathrm{d}J_{\mathrm{SQE}}}{\mathrm{d}\mu_k} \overset{!}{=} 0 \implies \mu^k = \frac{1}{|\mathcal{C}_k|} \sum_{\{n|x_n \in \mathcal{C}_k\}} x_n$$

  - → cluster prototypes are barycenters („centers of gravity“) of their clusters.

K-Means Clustering

- K-Means: Optimize intra cluster coherence:
  - Find prototypes by optimizing objective function modeling intra cluster coherence as mean square error

$$J_{\mathrm{SQE}} = \sum_{k=1}^{K} \sum_{\{n|x_n \in \mathcal{C}_k\}} ||x_n - \mu_k||^2$$

$$\frac{\mathrm{d}J_{\mathrm{SQE}}}{\mathrm{d}\mu_k} \overset{!}{=} 0 \implies \mu^k = \frac{1}{|\mathcal{C}_k|} \sum_{\{n|x_n \in \mathcal{C}_k\}} x_n$$

  - → cluster prototypes are barycenters („centers of gravity“) of their clusters.

K-Means Clustering

- K-Means: Optimize intra cluster coherence:
  - Find prototypes by optimizing objective function modeling intra cluster coherence as mean square error

$$J_{\mathrm{SQE}} = \sum_{k=1}^{K} \sum_{\{n|x_n \in \mathcal{C}_k\}} ||x_n - \mu_k||^2$$

$$\frac{\mathrm{d}J_{\mathrm{SQE}}}{\mathrm{d}\mu_k} \overset{!}{=} 0 \implies \mu^k = \frac{1}{|\mathcal{C}_k|} \sum_{\{n|x_n \in \mathcal{C}_k\}} x_n$$

  - → cluster prototypes are barycenters („centers of gravity“) of their clusters.

K-Means Clustering

- K-Means: Optimize intra cluster coherence:
  - Find prototypes by optimizing objective function modeling intra cluster coherence as mean square error

$$J_{\mathrm{SQE}} = \sum_{k=1}^{K} \sum_{\{n|x_n \in \mathcal{C}_k\}} ||x_n - \mu_k||^2$$

$$\frac{\mathrm{d}J_{\mathrm{SQE}}}{\mathrm{d}\mu_k} \overset{!}{=} 0 \implies \mu^k = \frac{1}{|\mathcal{C}_k|} \sum_{\{n|x_n \in \mathcal{C}_k\}} x_n$$

  - → cluster prototypes are barycenters („centers of gravity“) of their clusters.

K-Means Clustering

- K-Means: Optimize intra cluster coherence:
  - Find prototypes by optimizing objective function modeling intra cluster coherence as mean square error

$$J_{\mathrm{SQE}} = \sum_{k=1}^{K} \sum_{\{n|x_n \in \mathcal{C}_k\}} ||x_n - \mu_k||^2$$

$$\frac{\mathrm{d}J_{\mathrm{SQE}}}{\mathrm{d}\mu_k} \stackrel{!}{=} 0 \quad \Longrightarrow \quad \mu^k = \frac{1}{|\mathcal{C}_k|} \sum_{\{n|x_n \in \mathcal{C}_k\}} x_n$$

  - → cluster prototypes are barycenters („centers of gravity") of their clusters.

K-Means Clustering

- K-Means: Optimize intra cluster coherence:
  - Find prototypes by optimizing objective function modeling intra cluster coherence as mean square error

$$J_{\mathrm{SQE}} = \sum_{k=1}^{K} \sum_{\{n|x_n \in \mathcal{C}_k\}} ||x_n - \mu_k||^2$$

$$\frac{\mathrm{d}J_{\mathrm{SQE}}}{\mathrm{d}\mu_k} \stackrel{!}{=} 0 \quad \Longrightarrow \quad \mu^k = \frac{1}{|\mathcal{C}_k|} \sum_{\{n|x_n \in \mathcal{C}_k\}} x_n$$

  - → cluster prototypes are barycenters („centers of gravity") of their clusters.

K-Means Clustering

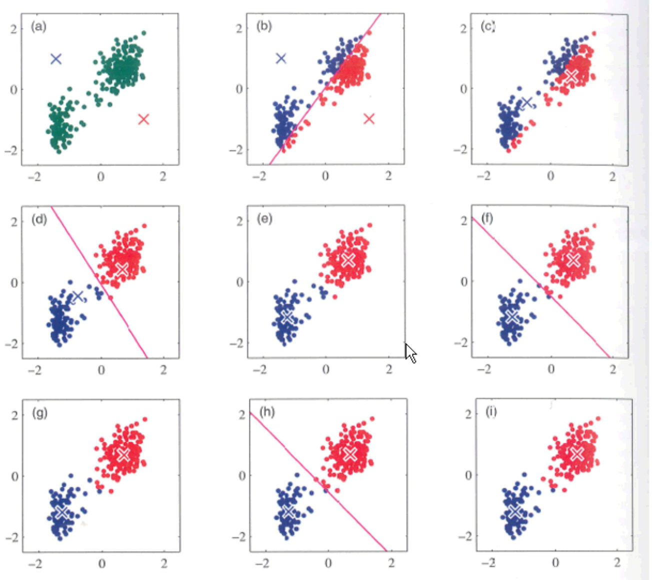
- K-Means: Optimize intra cluster coherence:
  - Find prototypes by optimizing objective function modeling intra cluster coherence as mean square error

$$J_{\mathrm{SQE}} = \sum_{k=1}^{K} \sum_{\{n|x_n \in \mathcal{C}_k\}} ||x_n - \mu_k||^2$$

$$\frac{\mathrm{d}J_{\mathrm{SQE}}}{\mathrm{d}\mu_k} \stackrel{!}{=} 0 \quad \Longrightarrow \quad \mu^k = \frac{1}{|\mathcal{C}_k|} \sum_{\{n|x_n \in \mathcal{C}_k\}} x_n$$

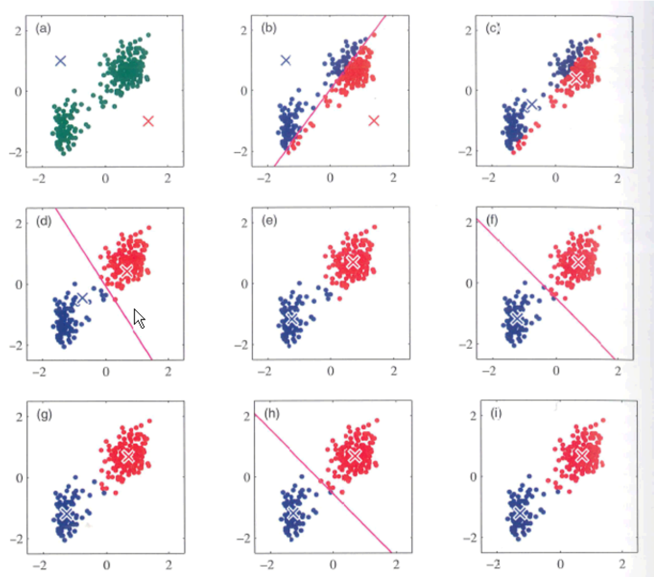
  - → cluster prototypes are barycenters („centers of gravity") of their clusters.

K-Means Clustering



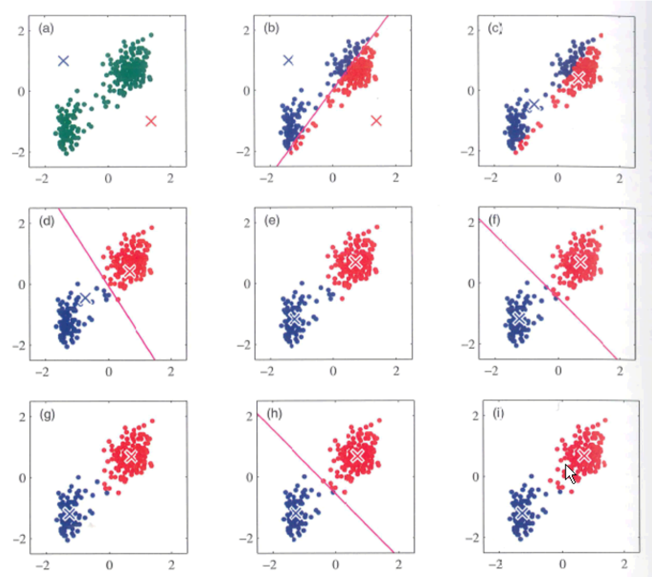
[3]

K-Means Clustering



[3]

K-Means Clustering


[3]

K-Means Clustering

- Dunn Index:

$$D = \min_{k_1 \in [1,K]} \left(\min_{k_2 \in [1,K]} \left(\frac{d_1(\mathcal{C}_{k_1}, \mathcal{C}_{k_2})}{\max_{k_3 \in [1,K]} d_2(\mathcal{C}_{k_3})} \right) \right)$$

where $d_1(\mathcal{C}_{k_1}, \mathcal{C}_{k_2})$ is the distance function between two clusters defined by

$$d_1(\mathcal{C}_{k_1}, \mathcal{C}_{k_2}) = \min_{\{(n_1,n_2) | x_{n_1} \in \mathcal{C}_{k_1} \wedge x_{n_2} \in \mathcal{C}_{k_2}\}} ||x_{n_1} - x_{n_2}||$$

(that is the single link distance from SAHN).
The “diameter” d_2 of the clusters is defined by

$$d_2(\mathcal{C}_i) = \max_{\{(n_1,n_2) | x_{n_1} \in \mathcal{C}_i \wedge x_{n_2} \in \mathcal{C}_i\}} ||x_{n_1} - x_{n_2}||$$

[7]

K-Means Clustering

- Dunn Index:

$$D = \min_{k_1 \in [1,K]} \left(\min_{k_2 \in [1,K]} \left(\frac{d_1(\mathcal{C}_{k_1}, \mathcal{C}_{k_2})}{\max_{k_3 \in [1,K]} d_2(\mathcal{C}_{k_3})} \right) \right)$$

where $d_1(\mathcal{C}_{k_1}, \mathcal{C}_{k_2})$ is the distance function between two clusters defined by

$$d_1(\mathcal{C}_{k_1}, \mathcal{C}_{k_2}) = \min_{\{(n_1,n_2) | x_{n_1} \in \mathcal{C}_{k_1} \wedge x_{n_2} \in \mathcal{C}_{k_2}\}} ||x_{n_1} - x_{n_2}||$$

(that is the single link distance from SAHN).
The “diameter” d_2 of the clusters is defined by

$$d_2(\mathcal{C}_i) = \max_{\{(n_1,n_2) | x_{n_1} \in \mathcal{C}_i \wedge x_{n_2} \in \mathcal{C}_i\}} ||x_{n_1} - x_{n_2}||$$

[7]

K-Means Clustering

- Dunn Index:

$$D = \min_{k_1 \in [1,K]} \left(\min_{k_2 \in [1,K]} \left(\frac{d_1(\mathcal{C}_{k_1}, \mathcal{C}_{k_2})}{\max_{k_3 \in [1,K]} d_2(\mathcal{C}_{k_3})} \right) \right)$$

where $d_1(\mathcal{C}_{k_1}, \mathcal{C}_{k_2})$ is the distance function between two clusters defined by

$$d_1(\mathcal{C}_{k_1}, \mathcal{C}_{k_2}) = \min_{\{(n_1,n_2) | x_{n_1} \in \mathcal{C}_{k_1} \wedge x_{n_2} \in \mathcal{C}_{k_2}\}} ||x_{n_1} - x_{n_2}||$$

(that is the single link distance from SAHN).
The "diameter" d_2 of the clusters is defined by

$$d_2(\mathcal{C}_i) = \max_{\{(n_1,n_2) | x_{n_1} \in \mathcal{C}_i \wedge x_{n_2} \in \mathcal{C}_i\}} ||x_{n_1} - x_{n_2}||$$

[7]

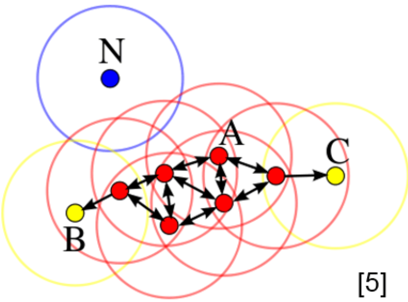
K-Means Clustering

Example Application: Clustering locations

- Problem: How do we distinguish socially relevant clusters (candidates for groups) from quasi groups?
  - Compute clusterings over period of time: Good candidates: clusters that appear over and over again, clusters that appear periodically
  - Establish threshold for distance in clusters: Human "social distance": A few meters (if groups are very small); few tens of meters (if groups are medium sized)
  - Include velocities: If divergent → no group

DBSCAN

- K-Means is „OK" as cluster algorithm, but has certain disadvantages:
  - favors spherical clusters
  - need to know K
  - no notion of noise

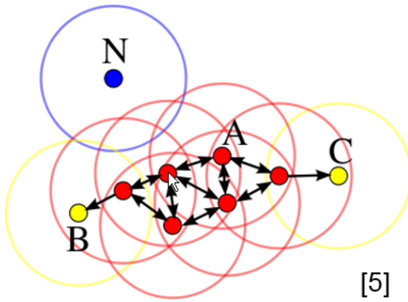


[5]

- Alternative → DBSCAN [4]
  (de facto state of the art):
  - Idea: Two parameters: minPt, ε
  - Rough idea: iterate:
    visit previously unseen pattern x:
    if in ε-neighborhood {x'} of x: |{x'}|≥ minPt then
    start new cluster: include x and {x'} and those of their ε-neighborhoods {x''} that are dense enough (|{x''}|≥ minPt), etc.
    else: x is noise

DBSCAN

- K-Means is „OK" as cluster algorithm, but has certain disadvantages:
  - favors spherical clusters
  - need to know K
  - no notion of noise

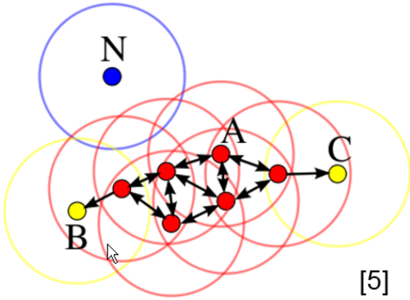


[5]

- Alternative → DBSCAN [4]
  (de facto state of the art):
  - Idea: Two parameters: minPt, ε
  - Rough idea: iterate:
    visit previously unseen pattern x:
    if in ε-neighborhood {x'} of x: |{x'}|≥ minPt then
    start new cluster: include x and {x'} and those of their ε-neighborhoods {x''} that are dense enough (|{x''}|≥ minPt), etc.
    else: x is noise

DBSCAN

- K-Means is „OK“ as cluster algorithm, but has certain disadvantages:
  - favors spherical clusters
  - need to know K
  - no notion of noise
- Alternative → DBSCAN [4]
  (de facto state of the art):
  - Idea: Two parameters: minPt, ε
  - Rough idea: iterate:
    visit previously unseen pattern x:
    if in ε-neighborhood {x‘} of x: |{x‘}|≥ minPt then
    start new cluster: include x and {x‘} and those of their ε-neighborhoods {x‘‘} that are dense enough (|{x‘‘}|≥ minPt), etc.
    else: x is noise



[5]

DBSCAN

- Advantages of DBSCAN:
  - We do not need to know K in advance
  - arbitrarily shaped clusters
  - notion of noise
- Disadvantages:
  - instead of having to know K, we need to „guess“ minPt and ε instead (can be a problem for high dimensional pattern spaces (→ curse of dimensionality))
  - original DBSCAN has fixed (minPt, ε) → problems when cluster density varies

K-Means Clustering

- Interesting aspect: How do we determine correct number k of clusters? (Same problem with graph clustering: where to cut dendrogram?)
- Answer: Compute for every k clusterings; chose the best clustering with a cluster quality measure
- Cluster quality measures for metric case: (countless variants exist in literature; for an overview: e.g. [2]) (Objective functions modeling clustering paradigm):
  - Dunn-Index
  - Entropy based indices
  - ....

Fuzzy C-Means Clustering

- K-Means was a crisp algorithm. Now: fuzzy variant
- Reformulate K-Means objective function with membership matrix r_{nk}: Membership of pattern x_n in class C_k

$$J_{SQE} = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2$$

- optimization criterion

$$\mathrm{d}J_{SQE}/\mathrm{d}\mu_k = 0$$

- together with non-overlaping constraint

$$\forall n (\exists k (r_{nk} = 1) \wedge ((k' \neq k) \rightarrow (r_{nk'} = 0)))$$

leads to well known K-Means

$$\mu_k = \sum_{n=1}^{N} r_{nk} x_n / \sum_{n=1}^{N} r_{nk} = (1/|\mathcal{C}_k|) \sum_{n | x_n \in \mathcal{C}_k} x_n$$

Fuzzy C-Means Clustering

- K-Means was a crisp algorithm. Now: fuzzy variant
- Reformulate K-Means objective function with membership matrix

r_{nk}: Membership of pattern x_n in class C_k

$$J_{SQE} = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2$$

- optimization criterion

$$\mathrm{d}J_{SQE}/\mathrm{d}\mu_k = 0$$

- together with non-overlaping constraint

$$\forall n (\exists k (r_{nk} = 1) \wedge ((k' \neq k) \rightarrow (r_{nk'} = 0)))$$

leads to well known K-Means

$$\mu_k = \sum_{n=1}^{N} r_{nk} x_n / \sum_{n=1}^{N} r_{nk} = (1/|\mathcal{C}_k|) \sum_{n | x_n \in \mathcal{C}_k} x_n$$

Fuzzy C-Means Clustering

- K-Means was a crisp algorithm. Now: fuzzy variant
- Reformulate K-Means objective function with membership matrix

r_{nk}: Membership of pattern x_n in class C_k

$$J_{SQE} = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2$$

- optimization criterion

$$\mathrm{d}J_{SQE}/\mathrm{d}\mu_k = 0$$

- together with non-overlaping constraint

$$\forall n (\exists k (r_{nk} = 1) \wedge ((k' \neq k) \rightarrow (r_{nk'} = 0)))$$

leads to well known K-Means

$$\mu_k = \sum_{n=1}^{N} r_{nk} x_n / \sum_{n=1}^{N} r_{nk} = (1/|\mathcal{C}_k|) \sum_{n | x_n \in \mathcal{C}_k} x_n$$

Fuzzy C-Means Clustering

- K-Means was a crisp algorithm. Now: fuzzy variant
- Reformulate K-Means objective function with membership matrix

r_{nk}: Membership of pattern x_n in class C_k

$$J_{SQE} = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2$$

- optimization criterion

$$\mathrm{d}J_{SQE}/\mathrm{d}\mu_k = 0$$

- together with non-overlaping constraint

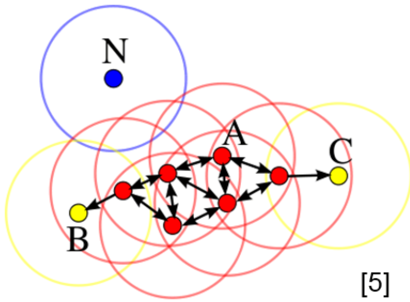
$$\forall n (\exists k (r_{nk} = 1) \wedge ((k' \neq k) \rightarrow (r_{nk'} = 0)))$$

leads to well known K-Means

$$\mu_k = \sum_{n=1}^{N} r_{nk} x_n / \sum_{n=1}^{N} r_{nk} = (1/|\mathcal{C}_k|) \sum_{n | x_n \in \mathcal{C}_k} x_n$$

DBSCAN

- K-Means is „OK“ as cluster algorithm, but has certain disadvantages:
  - favors spherical clusters
  - need to know K
  - no notion of noise



[5]

- Alternative → DBSCAN [4]
  (de facto state of the art):
  - Idea: Two parameters: minPt, ε
  - Rough idea: iterate:

    visit previously unseen pattern x:

    if in ε-neighborhood {x‘} of x: |{x‘}|≥ minPt then

    start new cluster: include x and {x‘} and those of their ε-neighborhoods {x‘‘} that are dense enough (|{x‘‘}|≥ minPt), etc.

    else: x is noise

K-Means Clustering

- Dunn Index:

$$D = \min_{k_1 \in [1,K]} \left(\min_{k_2 \in [1,K]} \left(\frac{d_1(\mathcal{C}_{k_1}, \mathcal{C}_{k_2})}{\max_{k_3 \in [1,K]} d_2(\mathcal{C}_{k_3})} \right) \right)$$

where $d_1(\mathcal{C}_{k_1}, \mathcal{C}_{k_2})$ is the distance function between two clusters defined by

$$d_1(\mathcal{C}_{k_1}, \mathcal{C}_{k_2}) = \min_{\{(n_1,n_2) | x_{n_1} \in \mathcal{C}_{k_1} \wedge x_{n_2} \in \mathcal{C}_{k_2}\}} ||x_{n_1} - x_{n_2}||$$

(that is the single link distance from SAHN).
The "diameter" d_2 of the clusters is defined by

$$d_2(\mathcal{C}_i) = \max_{\{(n_1,n_2) | x_{n_1} \in \mathcal{C}_i \wedge x_{n_2} \in \mathcal{C}_i\}} ||x_{n_1} - x_{n_2}||$$

[7]

Fuzzy C-Means Clustering

- K-Means was a crisp algorithm. Now: fuzzy variant
- Reformulate K-Means objective function with membership matrix

r_{nk}: Membership of pattern x_n in class C_k

$$J_{SQE} = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2$$

- optimization criterion

$$\mathrm{d}J_{SQE} / \mathrm{d}\mu_k = 0$$

- together with non-overlaping constraint

$$\forall n (\exists k (r_{nk} = 1) \wedge ((k' \neq k) \rightarrow (r_{nk'} = 0)))$$

leads to well known K-Means

$$\mu_k = \sum_{n=1}^{N} r_{nk} x_n / \sum_{n=1}^{N} r_{nk} = (1/|\mathcal{C}_k|) \sum_{n | x_n \in \mathcal{C}_k} x_n$$

Fuzzy C-Means Clustering

- K-Means was a crisp algorithm. Now: fuzzy variant
- Reformulate K-Means objective function with membership matrix

r_{nk}: Membership of pattern x_n in class C_k

$$J_{SQE} = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2$$

- optimization criterion

$$\mathrm{d}J_{SQE} / \mathrm{d}\mu_k = 0$$

- together with non-overlaping constraint

$$\forall n (\exists k (r_{nk} = 1) \wedge ((k' \neq k) \rightarrow (r_{nk'} = 0)))$$

leads to well known K-Means

$$\mu_k = \sum_{n=1}^{N} r_{nk} x_n / \sum_{n=1}^{N} r_{nk} = (1/|\mathcal{C}_k|) \sum_{n | x_n \in \mathcal{C}_k} x_n$$

Fuzzy C-Means Clustering

- Now modify objective function to:

$$J_{GSQE} = \sum_{n=1}^{N} \sum_{k=1}^{K} (r_{nk})^m ||x_n - \mu_k||^2$$

- Exponent m models degree of fuzzyness:
  - m → 1 : K-Means (crisp case);
  - m → ∞ : r_{nk} → 1/K (where K is the number of clusters)
- Optimize the obj. fct. under the conditions:

$$\forall x_n \quad : \quad \sum_{k=1}^{K} \alpha_k(x_n) = \sum_{k=1}^{K} r_{nk} = 1$$

$$\forall \mathcal{C}_k \quad : \quad \sum_{n=1}^{N} \alpha_k(x_n) = \sum_{n=1}^{N} r_{nk} > 0$$

Fuzzy C-Means Clustering

- Now modify objective function to:

$$J_{GSQE} = \sum_{n=1}^{N} \sum_{k=1}^{K} (r_{nk})^m ||x_n - \mu_k||^2$$

- Exponent m models degree of fuzzyness:
  m → 1 : K-Means (crisp case);
  m → ∞ : r_{nk} → 1/K (where K is the number of clusters)

- Optimize the obj. fct. under the conditions:

$$\forall\, x_n \quad : \quad \sum_{k=1}^{K} \alpha_k(x_n) = \sum_{k=1}^{K} r_{nk} = 1$$

$$\forall\, \mathcal{C}_k \quad : \quad \sum_{n=1}^{N} \alpha_k(x_n) = \sum_{n=1}^{N} r_{nk} > 0$$

Fuzzy C-Means Clustering

- Now modify objective function to:

$$J_{GSQE} = \sum_{n=1}^{N} \sum_{k=1}^{K} (r_{nk})^m ||x_n - \mu_k||^2$$

- Exponent m models degree of fuzzyness:
  m → 1 : K-Means (crisp case);
  m → ∞ : r_{nk} → 1/K (where K is the number of clusters)

- Optimize the obj. fct. under the conditions:

$$\forall\, x_n \quad : \quad \sum_{k=1}^{K} \alpha_k(x_n) = \sum_{k=1}^{K} r_{nk} = 1$$

$$\forall\, \mathcal{C}_k \quad : \quad \sum_{n=1}^{N} \alpha_k(x_n) = \sum_{n=1}^{N} r_{nk} > 0$$

Fuzzy C-Means Clustering

- Now modify objective function to:

$$J_{GSQE} = \sum_{n=1}^{N} \sum_{k=1}^{K} (r_{nk})^m ||x_n - \mu_k||^2$$

- Exponent m models degree of fuzzyness:
  m → 1 : K-Means (crisp case);
  m → ∞ : r_{nk} → 1/K (where K is the number of clusters)

- Optimize the obj. fct. under the conditions:

$$\forall\, x_n \quad : \quad \sum_{k=1}^{K} \alpha_k(x_n) = \sum_{k=1}^{K} r_{nk} = 1$$

$$\forall\, \mathcal{C}_k \quad : \quad \sum_{n=1}^{N} \alpha_k(x_n) = \sum_{n=1}^{N} r_{nk} > 0$$

Fuzzy C-Means Clustering

- Now modify objective function to:

$$J_{GSQE} = \sum_{n=1}^{N} \sum_{k=1}^{K} (r_{nk})^m ||x_n - \mu_k||^2$$

- Exponent m models degree of fuzzyness:
  m → 1 : K-Means (crisp case);
  m → ∞ : r_{nk} → 1/K (where K is the number of clusters)

- Optimize the obj. fct. under the conditions:

$$\forall\, x_n \quad : \quad \sum_{k=1}^{K} \alpha_k(x_n) = \sum_{k=1}^{K} r_{nk} = 1$$

$$\forall\, \mathcal{C}_k \quad : \quad \sum_{n=1}^{N} \alpha_k(x_n) = \sum_{n=1}^{N} r_{nk} > 0$$

Fuzzy C-Means Clustering

- K-Means was a crisp algorithm. Now: fuzzy variant
- Reformulate K-Means objective function with membership matrix

r_{nk}: Membership of pattern x_n in class C_k

$$J_{SQE} = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2$$

- optimization criterion

$$\mathrm{d}J_{SQE}/\mathrm{d}\mu_k = 0$$

- together with non-overlaping constraint

$$\forall n (\exists k (r_{nk} = 1) \wedge ((k' \neq k) \rightarrow (r_{nk'} = 0)))$$

leads to well known K-Means

$$\mu_k = \sum_{n=1}^{N} r_{nk} x_n / \sum_{n=1}^{N} r_{nk} = (1/|\mathcal{C}_k|) \sum_{n | x_n \in \mathcal{C}_k} x_n$$

Fuzzy C-Means Clustering

- Result:

$$r_{nk} = \left(\sum_{k'=1}^{K} \left(\frac{||x_n - \mu_k||}{||x_n - \mu_{k'}||} \right)^{\frac{2}{m-1}} \right)^{-1} \qquad (☺)$$

$$\mu_k = \frac{\sum_{n=1}^{N} r_{nk}^m x_n}{\sum_{n=1}^{N} r_{nk}} \qquad (☺☺)$$

- the result assumes that no patterns and prototypes coincide

$$\forall\, n, k : \quad ||x_n - \mu_k|| \neq 0$$

if they do coincide, set $r_{nk} = 1$ for $x_n = \mu_k$ and $r_{nk} = 0$ for $x_n \neq \mu_k$

Fuzzy C-Means Clustering

- Limit m → ∞ gives:

$$r_{nk} \xrightarrow{m \to \infty} \frac{1}{\sum_{k'=1}^{K} 1} = \frac{1}{K}$$

- Limit m → 1 we get the nearest neighbor rule (K-Means) because:

$$r_{nk} = 1/((\sum_{k' \neq k} (\frac{||x_n - \mu_k||}{||x_n - \mu_{k'}||})^{\frac{2}{m-1}}) + 1)$$

in the limit m→1 the first sum in the denominator becomes ∞ if

$$||x_n - \mu_k|| \neq \min_{1 \leq k' \leq K} ||x_n - \mu_{k'}||$$

and it becomes 0 if

$$||x_n - \mu_k|| = \min_{1 \leq k' \leq K} ||x_n - \mu_{k'}||$$

Fuzzy C-Means Clustering

- Limit m → ∞ gives:

$$r_{nk} \xrightarrow{m \to \infty} \frac{1}{\sum_{k'=1}^{K} 1} = \frac{1}{K}$$

- Limit m → 1 we get the nearest neighbor rule (K-Means) because:

$$r_{nk} = 1/((\sum_{k' \neq k} (\frac{||x_n - \mu_k||}{||x_n - \mu_{k'}||})^{\frac{2}{m-1}}) + 1)$$

in the limit m→1 the first sum in the denominator becomes ∞ if

$$||x_n - \mu_k|| \neq \min_{1 \leq k' \leq K} ||x_n - \mu_{k'}||$$

and it becomes 0 if

$$||x_n - \mu_k|| = \min_{1 \leq k' \leq K} ||x_n - \mu_{k'}||$$

Fuzzy C-Means Clustering

- Limit m → ∞ gives:

$$r_{nk} \stackrel{m\to\infty}{\longrightarrow} \frac{1}{\sum_{k'=1}^{K} 1} = \frac{1}{K}$$

- Limit m → 1 we get the nearest neighbor rule (K-Means) because:

$$r_{nk} = 1/((\sum_{k' \neq k}(\frac{||x_n-\mu_k||}{||x_n-\mu_{k'}||})^{\frac{2}{m-1}}) + 1)$$

  in the limit m→1 the first sum in the denominator becomes ∞ if

$$||x_n - \mu_k|| \neq \min_{1\le k' \le K} ||x_n - \mu_{k'}||$$

  and it becomes 0 if

$$||x_n - \mu_k|| = \min_{1\le k' \le K} ||x_n - \mu_{k'}||$$

Fuzzy C-Means Clustering

- Limit m → ∞ gives:

$$r_{nk} \stackrel{m\to\infty}{\longrightarrow} \frac{1}{\sum_{k'=1}^{K} 1} = \frac{1}{K}$$

- Limit m → 1 we get the nearest neighbor rule (K-Means) because:

$$r_{nk} = 1/((\sum_{k' \neq k}(\frac{||x_n-\mu_k||}{||x_n-\mu_{k'}||})^{\frac{2}{m-1}}) + 1)$$

  in the limit m→1 the first sum in the denominator becomes ∞ if

$$||x_n - \mu_k|| \neq \min_{1\le k' \le K} ||x_n - \mu_{k'}||$$

  and it becomes 0 if

$$||x_n - \mu_k|| = \min_{1\le k' \le K} ||x_n - \mu_{k'}||$$

Fuzzy C-Means Clustering

- Limit m → ∞ gives:

$$r_{nk} \stackrel{m\to\infty}{\longrightarrow} \frac{1}{\sum_{k'=1}^{K} 1} = \frac{1}{K}$$

- Limit m → 1 we get the nearest neighbor rule (K-Means) because:

$$r_{nk} = 1/((\sum_{k' \neq k}(\frac{||x_n-\mu_k||}{||x_n-\mu_{k'}||})^{\frac{2}{m-1}}) + 1)$$

  in the limit m→1 the first sum in the denominator becomes ∞ if

$$||x_n - \mu_k|| \neq \min_{1\le k' \le K} ||x_n - \mu_{k'}||$$

  and it becomes 0 if

$$||x_n - \mu_k|| = \min_{1\le k' \le K} ||x_n - \mu_{k'}||$$

Fuzzy C-Means Clustering

- Now modify objective function to:

$$J_{GSQE} = \sum_{n=1}^{N}\sum_{k=1}^{K} (r_{nk})^m ||x_n - \mu_k||^2$$

- Exponent m models degree of fuzzyness:
  m → 1 : K-Means (crisp case);
  m → ∞ : r_{nk} → 1/K (where K is the number of clusters)

- Optimize the obj. fct. under the conditions:

$$\forall\, x_n \; : \; \sum_{k=1}^{K} \alpha_k(x_n) = \sum_{k=1}^{K} r_{nk} = 1$$

$$\forall\, \mathcal{C}_k \; : \; \sum_{n=1}^{N} \alpha_k(x_n) = \sum_{n=1}^{N} r_{nk} > 0$$

Gaussian Mixture Models

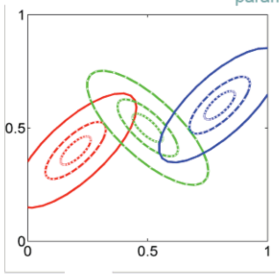
- Fuzzy C-Means is “OK” as a non-crisp clustering alg. but (as K-Means) favors spherical clusters → better approaches

- Example: Gaussian Mixture Models (GMM)

  - Linear combination of Gaussians

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \quad \text{where} \quad \sum_{k=1}^{K} \pi_k = 1, \quad 0 \leqslant \pi_k \leqslant 1$$

parameters to be estimated



[6]

Gaussian Mixture Models

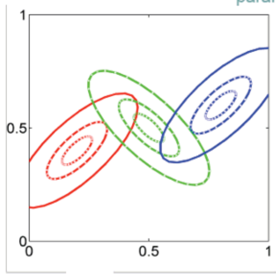
- Fuzzy C-Means is “OK” as a non-crisp clustering alg. but (as K-Means) favors spherical clusters → better approaches

- Example: Gaussian Mixture Models (GMM)

  - Linear combination of Gaussians

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \quad \text{where} \quad \sum_{k=1}^{K} \pi_k = 1, \quad 0 \leqslant \pi_k \leqslant 1$$

parameters to be estimated



[6]

Gaussian Mixture Models

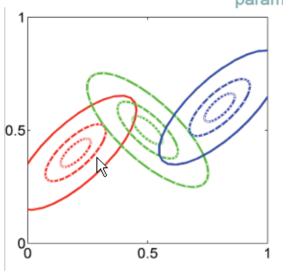
- Fuzzy C-Means is “OK” as a non-crisp clustering alg. but (as K-Means) favors spherical clusters → better approaches

- Example: Gaussian Mixture Models (GMM)

  - Linear combination of Gaussians

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \quad \text{where} \quad \sum_{k=1}^{K} \pi_k = 1, \quad 0 \leqslant \pi_k \leqslant 1$$

parameters to be estimated



[6]

Gaussian Mixture Models

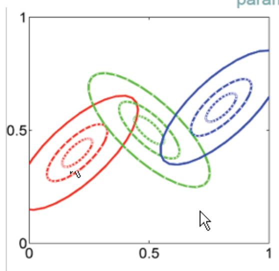
- Fuzzy C-Means is “OK” as a non-crisp clustering alg. but (as K-Means) favors spherical clusters → better approaches

- Example: Gaussian Mixture Models (GMM)

  - Linear combination of Gaussians

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \quad \text{where} \quad \sum_{k=1}^{K} \pi_k = 1, \quad 0 \leqslant \pi_k \leqslant 1$$

parameters to be estimated



[6]

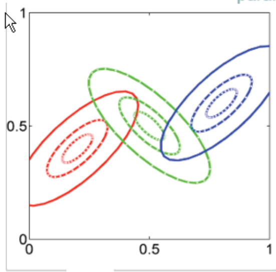
Gaussian Mixture Models

- Fuzzy C-Means is "OK" as a non-crisp clustering alg. but (as K-Means) favors spherical clusters → better approaches

- Example: Gaussian Mixture Models (GMM)
  - Linear combination of Gaussians

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \quad \text{where} \quad \sum_{k=1}^{K} \pi_k = 1, \quad 0 \leqslant \pi_k \leqslant 1$$

parameters to be estimated



[6]

Gaussian Mixture Models

- Fuzzy C-Means is "OK" as a non-crisp clustering alg. but (as K-Means) favors spherical clusters → better approaches

- Example: Gaussian Mixture Models (GMM)
  - Linear combination of Gaussians

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \quad \text{where} \quad \sum_{k=1}^{K} \pi_k = 1, \quad 0 \leqslant \pi_k \leqslant 1$$

this is usually written as $p(x|\Theta)$ *denoting the dependency on the parameters* $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{\{k \in \{1,2,\ldots,K\}\}}$

Writing this as a conditional probability makes sense in connection with Bayesian Machine Learning (see [8])

Gaussian Mixture Models

- Fuzzy C-Means is "OK" as a non-crisp clustering alg. but (as K-Means) favors spherical clusters → better approaches

- Example: Gaussian Mixture Models (GMM)
  - Linear combination of Gaussians

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \quad \text{where} \quad \sum_{k=1}^{K} \pi_k = 1, \quad 0 \leqslant \pi_k \leqslant 1$$

this is usually written as $p(x|\Theta)$ *denoting the dependency on the parameters* $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{\{k \in \{1,2,\ldots,K\}\}}$

Writing this as a conditional probability makes sense in connection with Bayesian Machine Learning (see [8])

Gaussian Mixture Models

- Fuzzy C-Means is "OK" as a non-crisp clustering alg. but (as K-Means) favors spherical clusters → better approaches

- Example: Gaussian Mixture Models (GMM)
  - Linear combination of Gaussians

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \quad \text{where} \quad \sum_{k=1}^{K} \pi_k = 1, \quad 0 \leqslant \pi_k \leqslant 1$$

this is usually written as $p(x|\Theta)$ *denoting the dependency on the parameters* $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{\{k \in \{1,2,\ldots,K\}\}}$

Writing this as a conditional probability makes sense in connection with Bayesian Machine Learning (see [8])

from here we follow [3], so citations for images etc. are omitted

Machine Learning

- For a distribution $p(x|\Theta)$ parametrized by a set of parameters Θ and iid data $X = \{x_1, x_2, \dots, x_N\}$, simple machine learning corresponds to finding the Θ that best explains the data
- iid: „identically independently drawn“ ⇨ $p(X|\Theta) = \prod_i p(x_i|\Theta)$
- $p(\mathrm{X}|\Theta)$ is called likelihood
- „finding the Θ that best explains the data“:
  Maximum Likelihood: $\Theta_{ML} = argmax_\Theta \; p(X|\Theta)$ ⇨ $\nabla_\Theta \, p(X|\Theta) \overset{!}{=} 0$
- convenient: use $\log \; p(X|\theta)$ instead of $p(X|\theta)$
  ⇨ $\log \; p(X|\Theta) = \sum_i \log \; p(x\_i|\Theta)$

Machine Learning

- For a distribution $p(x|\Theta)$ parametrized by a set of parameters Θ and iid data $X = \{x_1, x_2, \dots, x_N\}$, simple machine learning corresponds to finding the Θ that best explains the data
- iid: „identically independently drawn“ ⇨ $p(X|\Theta) = \prod_i p(x_i|\Theta)$
- $p(\mathrm{X}|\Theta)$ is called likelihood
- „finding the Θ that best explains the data“:
  Maximum Likelihood: $\Theta_{ML} = argmax_\Theta \; p(X|\Theta)$ ⇨ $\nabla_\Theta \, p(X|\Theta) \overset{!}{=} 0$
- convenient: use $\log \; p(X|\theta)$ instead of $p(X|\theta)$
  ⇨ $\log \; p(X|\Theta) = \sum_i \log \; p(x\_i|\Theta)$

Machine Learning

- For a distribution $p(x|\Theta)$ parametrized by a set of parameters Θ and iid data $X = \{x_1, x_2, \dots, x_N\}$, simple machine learning corresponds to finding the Θ that best explains the data
- iid: „identically independently drawn“ ⇨ $p(X|\Theta) = \prod_i p(x_i|\Theta)$
- $p(\mathrm{X}|\Theta)$ is called likelihood
- „finding the Θ that best explains the data“:
  Maximum Likelihood: $\Theta_{ML} = argmax_\Theta \; p(X|\Theta)$ ⇨ $\nabla_\Theta \, p(X|\Theta) \overset{!}{=} 0$
- convenient: use $\log \; p(X|\theta)$ instead of $p(X|\theta)$
  ⇨ $\log \; p(X|\Theta) = \sum_i \log \; p(x\_i|\Theta)$

Machine Learning

- For a distribution $p(x|\Theta)$ parametrized by a set of parameters Θ and iid data $X = \{x_1, x_2, \dots, x_N\}$, simple machine learning corresponds to finding the Θ that best explains the data
- iid: „identically independently drawn“ ⇨ $p(X|\Theta) = \prod_i p(x_i|\Theta)$
- $p(\mathrm{X}|\Theta)$ is called likelihood
- „finding the Θ that best explains the data“:
  Maximum Likelihood: $\Theta_{ML} = argmax_{\Theta}\ p(X|\Theta)$ ⇨ $\nabla_{\Theta}\, p(X|\Theta) \stackrel{!}{=} 0$
- convenient: use $\log\ p(X|\theta)$ instead of $p(X|\theta)$
  ⇨ $\log\ p(X|\Theta) = \sum_i \log\ p(x\_i|\Theta)$

Machine Learning

- For a distribution $p(x|\Theta)$ parametrized by a set of parameters Θ and iid data $X = \{x_1, x_2, \dots, x_N\}$, simple machine learning corresponds to finding the Θ that best explains the data
- iid: „identically independently drawn“ ⇨ $p(X|\Theta) = \prod_i p(x_i|\Theta)$
- $p(\mathrm{X}|\Theta)$ is called likelihood
- „finding the Θ that best explains the data“:
  Maximum Likelihood: $\Theta_{ML} = argmax_{\Theta}\ p(X|\Theta)$ ⇨ $\nabla_{\Theta}\, p(X|\Theta) \stackrel{!}{=} 0$
- convenient: use $\log\ p(X|\theta)$ instead of $p(X|\theta)$
  ⇨ $\log\ p(X|\Theta) = \sum_i \log\ p(x\_i|\Theta)$

Machine Learning

- For a distribution $p(x|\Theta)$ parametrized by a set of parameters Θ and iid data $X = \{x_1, x_2, \dots, x_N\}$, simple machine learning corresponds to finding the Θ that best explains the data
- iid: „identically independently drawn“ ⇨ $p(X|\Theta) = \prod_i p(x_i|\Theta)$
- $p(\mathrm{X}|\Theta)$ is called likelihood
- „finding the Θ that best explains the data“:
  Maximum Likelihood: $\Theta_{ML} = argmax_{\Theta}\ p(X|\Theta)$ ⇨ $\nabla_{\Theta}\, p(X|\Theta) \stackrel{!}{=} 0$
- convenient: use $\log\ p(X|\theta)$ instead of $p(X|\theta)$
  ⇨ $\log\ p(X|\Theta) = \sum_i \log\ p(x\_i|\Theta)$

Machine Learning

- For a distribution $p(x|\Theta)$ parametrized by a set of parameters Θ and iid data $X = \{x_1, x_2, \dots, x_N\}$, simple machine learning corresponds to finding the Θ that best explains the data
- iid: „identically independently drawn“ ⇨ $p(X|\Theta) = \prod_i p(x_i|\Theta)$
- $p(\mathrm{X}|\Theta)$ is called likelihood
- „finding the Θ that best explains the data“:
  Maximum Likelihood: $\Theta_{ML} = argmax_{\Theta}\ p(X|\Theta)$ ⇨ $\nabla_{\Theta}\, p(X|\Theta) \stackrel{!}{=} 0$
- convenient: use $\log\ p(X|\theta)$ instead of $p(X|\theta)$
  ⇨ $\log\ p(X|\Theta) = \sum_i \log\ p(x\_i|\Theta)$

- For a distribution $p(x|\Theta)$ parametrized by a set of parameters Θ and iid data $X = \{x_1, x_2, \dots, x_N\}$, simple machine learning corresponds to finding the Θ that best explains the data

- iid: „identically independently drawn" ⇨ $p(X|\Theta) = \prod_i p(x_i|\Theta)$

- $p(\mathrm{X}|\Theta)$ is called likelihood

- „finding the Θ that best explains the data":
  Maximum Likelihood: $\Theta_{ML} = argmax_{\Theta}\ p(X|\Theta)$ ⇨ $\nabla_{\Theta}\, p(X|\Theta) \stackrel{!}{=} 0$

- convenient: use $\log\ p(X|\theta)$ instead of $p(X|\theta)$
  ⇨ $\log\ p(X|\Theta) = \sum_i \log\ p(x\_i|\Theta)$