Ensemble Clustering

Seminar im Fachgebiet Wissensverarbeitung

Malek Alsalamat

Universität Kassel

July 5, 2022

Inhalt der Präsentation

In der Präsentation werden wir folgendes behandeln:

1. Was ist Clustering
2. K-Means
3. Ensemble Clustering und das Ensemble Problem
4. Zielfunktion
5. Effiziente consensus Funktion
6. Cluster-based Similarity Partitioning Algorithm
7. Anwendungen

Was ist Clustering

Definition von Clustering

Clustering ist die Methode im maschinellen Lernen und genauer gesagt "im unüberwachten Lernen", Datenpunkte in Gruppen (Clusters) ohne vorhandenes Wissen zu ordnen. Um das zu schaffen, wird die Ähnlichkeit zwischen die Daten berechnet.

Ziel von Clustering:

- Ähnliche Datenpunkte (Objekte) sollen im selben Cluster sein.
- Datenpunkte im verschiedenen Cluster soll möglichst unähnlich sein.

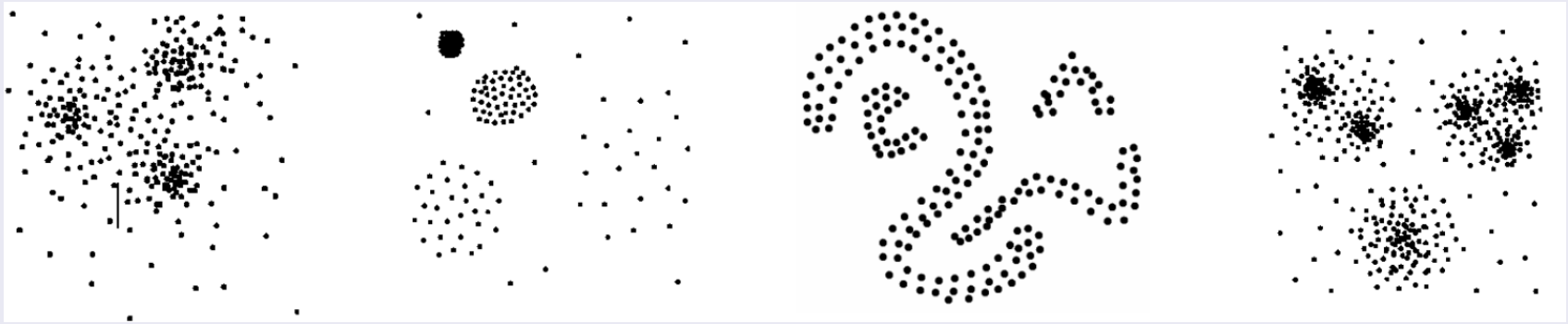
Was ist Clustering

Definition von Clustering

Clustering ist die Methode im maschinellen Lernen und genauer gesagt "im unüberwachten Lernen", Datenpunkte in Gruppen (Clusters) ohne vorhandenes Wissen zu ordnen. Um das zu schaffen, wird die Ähnlichkeit zwischen die Daten berechnet.

Beispiel.1

Cluster unterschiedlicher Größe, Form und Dichte bzw. hierarchische Cluster



K-Means als Beispiel für partitionierende Verfahren

Im partitionierenden Verfahren wird ein Clustering mit k Cluster mit minimalen Kosten gesucht. Und bei K-Means wie der Name bereits sagt, sucht das Algorithmus für jeden Cluster einen zentralen Punkt, bei dem die Varianz zu allen umliegenden Punkten möglichst gering ist. Das Ganze passiert in einem iterativen Verfahren wie folgendes:

1. Initialisierung: Zufällige Auswahl von K Zentren.
2. Zuweisung aller Datenpunkten zum nächstliegenden Zentrum, gemessen an einer Distanzmetrik. Distanzmetrik berechnet hier die Ähnlichkeit.
3. Verschieben der Zentren in den Mittelpunkt aller zugeteilten Datenpunkte.
4. Gehe zu 2), außer ein Abbruchkriterium ist erreicht. Der Abbruchkriterium ist, wenn die Zentren sich nicht mehr ändern.

Ensemble Clustering

Definition

Im Zusammenhang mit maschinellem Lernen ist Ensemble im Allgemein definiert als ein maschinelles Lernsystem, das mit einer Reihe von parallel arbeitenden einzelnen Modellen konstruiert wird. Und die Ergebnisse von diesen Modellen werden mit einer Entscheidungsstrategie kombiniert, um einzige Lösung für das gegebenen Problem zu liefern.

Ensemble Clustering

Definition

Im Zusammenhang mit maschinellem Lernen ist Ensemble im Allgemein definiert als ein maschinelles Lernsystem, das mit einer Reihe von parallel arbeitenden einzelnen Modellen konstruiert wird. Und die Ergebnisse von diesen Modellen werden mit einer Entscheidungsstrategie kombiniert, um einzige Lösung für das gegebenen Problem zu liefern.

Die Gründe für die Anwendung der Ensemble Methode auf Clustering Probleme sind:

1. Es gibt kein vorhandenes Wissen über die zugrundeliegende Struktur
2. Es gibt keinen einzelnen Clustering-Algorithmus, der für verschiedene Probleme konsistent gute Leistungen erbringen kann.

Das Ensemble Problem

Gegeben

Mehrere Clusterings aus verschiedenen Algorithmen von derselben Datenmenge.

Gesucht

Ein kombiniertes Clustering, welches so viele Informationen wie möglich mit den gegebenen Clusterings teilt.

Notation: Sei $X = \{x_1, x_2, \ldots, x_n\}$ eine Menge von Objekten (Datenpunkte). Eine Partitionierung dieser n Objekten in k Clusters kann als eine Menge von k Mengen von Objekten $\{C_l \mid l = 1, \ldots, k\}$ oder als ein Label Vektor $\lambda \in N^n$ dargestellt werden. Ein Clusterer Φ ist eine Funktion, welche einen Label Vektor bei gegebene Objekten liefert. Eine Mengen von r Clusterings (Labelings) $\lambda^{(1,\ldots,r)}$ kann zu einem Clustering anhand einer Consensus Funktion kombiniert werden.

Das Ensemble Problem

Sei $X = \{x_1, x_2, \ldots, x_7\}$ eine Menge von Objekte und 3 Clusters $\{C_l, C_2, C_3\}$ und die Clustering $\lambda^{(1,\ldots,r)}$ wie in der Tabelle gezeigt.

	$\lambda^{(1)}$	$\lambda^{(2)}$	$\lambda^{(3)}$	$\lambda^{(4)}$
x_1	1	2	1	1
x_2	1	2	1	2
x_3	1	2	2	?
x_4	2	3	2	1
x_5	2	3	3	2
x_6	3	1	3	?
x_7	3	1	3	?

Das Ensemble Problem

Sei $X = \{x_1, x_2, \ldots, x_7\}$ eine Menge von Objekte und 3 Clusters $\{C_l, C_2, C_3\}$ und die Clustering $\lambda^{(1,\ldots,r)}$ wie in der Tabelle gezeigt.

	$\lambda^{(1)}$	$\lambda^{(2)}$	$\lambda^{(3)}$	$\lambda^{(4)}$
x_1	1	2	1	1
x_2	1	2	1	2
x_3	1	2	2	?
x_4	2	3	2	1
x_5	2	3	3	2
x_6	3	1	3	?
x_7	3	1	3	?

Das Ensemble Problem

Sei $X = \{x_1, x_2, \ldots, x_7\}$ eine Menge von Objekte und 3 Clusters $\{C_l, C_2, C_3\}$ und die Clustering $\lambda^{(1,\ldots,r)}$ wie in der Tabelle gezeigt.

	$\lambda^{(1)}$	$\lambda^{(2)}$	$\lambda^{(3)}$	$\lambda^{(4)}$
x_1	1	2	1	1
x_2	1	2	1	2
x_3	1	2	2	?
x_4	2	3	2	1
x_5	2	3	3	2
x_6	3	1	3	?
x_7	3	1	3	?

Das Ensemble Problem

Sei $X = \{x_1, x_2, \ldots, x_7\}$ eine Menge von Objekte und 3 Clusters $\{C_l, C_2, C_3\}$ und die Clustering $\lambda^{(1,\ldots,r)}$ wie in der Tabelle gezeigt.

	$\lambda^{(1)}$	$\lambda^{(2)}$	$\lambda^{(3)}$	$\lambda^{(4)}$
x_1	1	2	1	1
x_2	1	2	1	2
x_3	1	2	2	?
x_4	2	3	2	1
x_5	2	3	3	2
x_6	3	1	3	?
x_7	3	1	3	?

Das Ensemble Problem

Sei $X = \{x_1, x_2, \ldots, x_7\}$ eine Menge von Objekte und 3 Clusters $\{C_l, C_2, C_3\}$ und die Clustering $\lambda^{(1,\ldots,r)}$ wie in der Tabelle gezeigt.

	$\lambda^{(1)}$	$\lambda^{(2)}$	$\lambda^{(3)}$	$\lambda^{(4)}$
x_1	1	2	1	1
x_2	1	2	1	2
x_3	1	2	2	?
x_4	2	3	2	1
x_5	2	3	3	2
x_6	3	1	3	?
x_7	3	1	3	?

Gesucht ist jetzt ein kombiniertes Clustering, welches so viele Informationen wie möglich mit den vier gegebenen Clusterings teilt. Intuitiv, ein gutes kombiniertes Clustering in diesem Fall ist $\lambda^{(1)}$ oder auch $\lambda^{(2)}$.

Zielfunktion für Ensemble Clustering

Definition

Gegeben nun r Gruppierungen mit $\lambda^{(i)}$ ist die i-te Gruppierung, welche k^i Clusters hat. Eine *consensus* Funktion ist wie folgt definiert $\Gamma : \mathbb{N}^{n \times r} \rightarrow \mathbb{N}^n$.

$$\Gamma : \{\lambda^i \mid i \in 1, \ldots, r\} \rightarrow \lambda. \tag{1}$$

Eine sinnvolle Ziel für die *consensus* Funktion ist, ein Clustering zu suchen, welches die meisten Informationen mit den ursprünglichen **Clusterings** teilt.

Zielfunktion für Ensemble Clustering

Sei $n_h^{(a)}$ die Anzahl von Objekten in Cluster C_h anhand das Clustering $\lambda^{(a)}$ und $n_l^{(b)}$ die Anzahl von Objekten in Cluster C_l anhand das Clustering $\lambda^{(b)}$. Sei $n_{h,l}$ die Anzahl der gemeinsamen Objekten zwischen Cluster h anhand $\lambda^{(a)}$ und Cluster l anhand $\lambda^{(b)}$. Dann ist:

$$\phi^{(NMI)}(\lambda^{(a)}, \lambda^{(b)}) = \frac{\sum_{h=1}^{k^{(a)}} \sum_{l=1}^{k^{(b)}} n_{h,l} \cdot \log\left(\frac{n \cdot n_{h,l}}{n_h^{(a)} \cdot n_l^{(b)}}\right)}{\sqrt{\left(\sum_{h=1}^{k^{(a)}} n_h^{(a)} \log \frac{n_h^{(a)}}{n}\right)\left(\sum_{l=1}^{k^{(b)}} n_l^{(b)} \cdot \log \frac{n_l^{(b)}}{n}\right)}}. \quad (2)$$

Zielfunktion für Ensemble Clustering

Von Gleichung (2) wird einen Maß zwischen einer Menge Λ und einem einzelnen Clustering $\hat{\lambda}$ als die durchschnittliche normalisierte gegenseitige Information definiert (ANMI):

$$\phi^{(ANMI)}(\Lambda, \hat{\lambda}) = \frac{1}{r} \sum_{q=1}^{r} \phi^{(NMI)}(\hat{\lambda}, \lambda^{(q)}). \tag{3}$$

Zielfunktion für Ensemble Clustering

Von Gleichung (2) wird einen Maß zwischen einer Menge Λ und einem einzelnen Clustering $\hat{\lambda}$ als die durchschnittliche normalisierte gegenseitige Information definiert (ANMI):

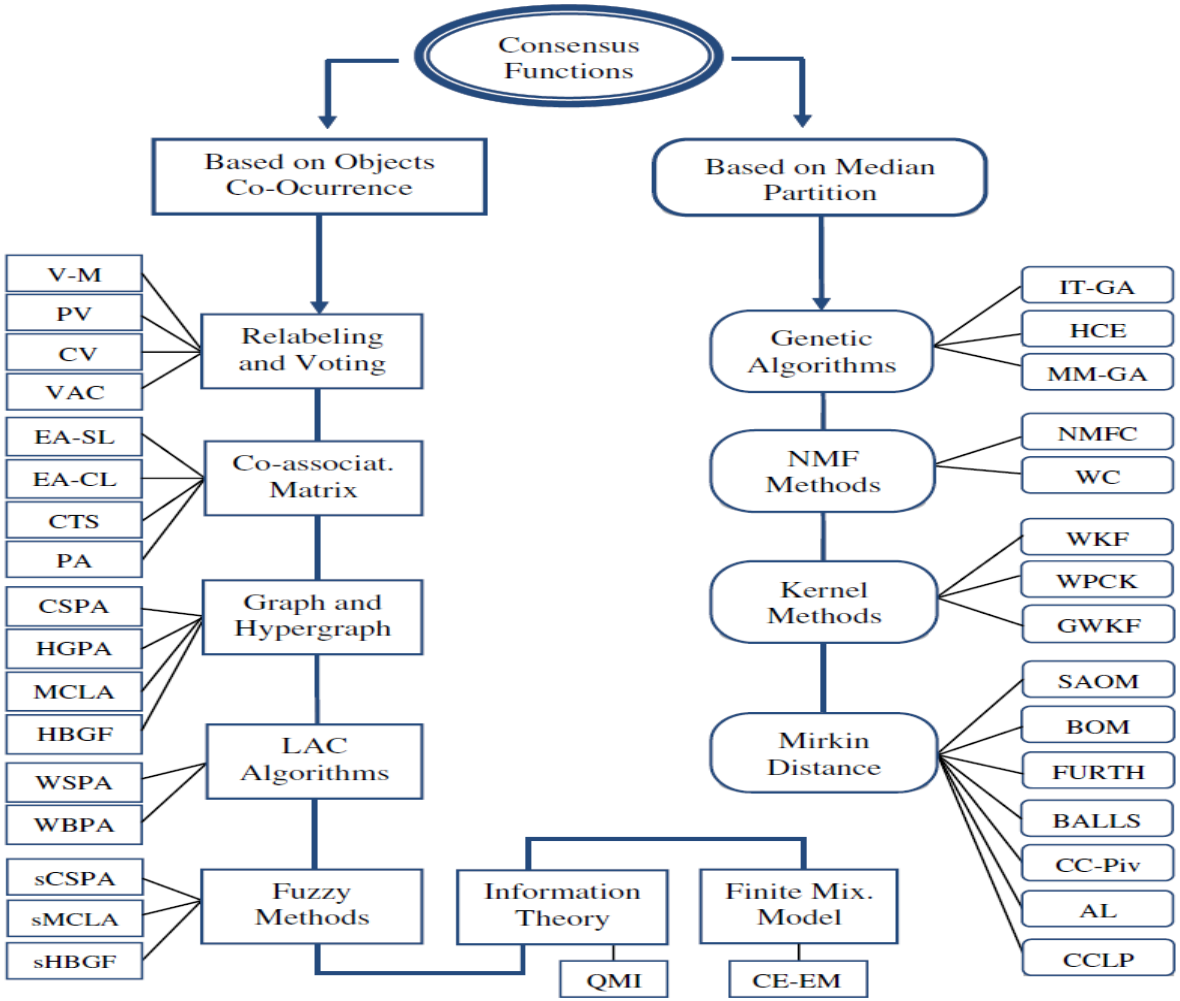
$$\phi^{(ANMI)}(\Lambda, \hat{\lambda}) = \frac{1}{r} \sum_{q=1}^{r} \phi^{(NMI)}(\hat{\lambda}, \lambda^{(q)}). \tag{3}$$

Das optimale kombinierte Clustering $\lambda^{(k-opt)}$ ist das Clustering mit der maximalen durchschnittlichen gegenseitigen Information, wobei k ist die Anzahl von *consensus* Clusters. In anderen Worten, $\phi^{(ANMI)}$ ist unsere Zielfunktion und $\lambda^{(k-opt)}$ ist:

$$\lambda^{(k-opt)} = \arg\max_{\hat{\lambda}} \sum_{q=1}^{r} \phi^{(NMI)}(\hat{\lambda}, \lambda^{(q)}), \tag{4}$$

wobei $\hat{\lambda}$ durch alle mögliche $k - Partitionierungen$ läuft.

Effiziente consensus Funktion



Cluster-based Similarity Partitioning Algorithm (CSPA)

Hier wird aus dem Hypergraph eine $n \times n$ Ähnlichkeitsmatrix (die Co-Assoziationsmatrix) konstruiert. Dies kann als eine Adjazenzmatrix eines zusammenhängenden Graphen angesehen werden, wobei die Knoten die Elemente der Datenmenge X sind und die kanten zwischen zwei Objekten ein zugeordnetes Gewicht hat. Das Gewicht entspricht, wie oft sich die Objekte im selben Cluster befinden. Danach wird das Graph-Partitionierungsalgorithmus MEITS verwendet, um die *consensus* Partitionierung zu erhalten.

HyperGraphs Partitioning Algorithm (HGPA)

Hier wird den Hypergraphen direkt partitioniert, indem die minimale Anzahl von Hyperkanten eliminiert wird. Außerdem wird es davon ausgegangen, dass alle Hyperkanten das gleiche Gewicht haben. Und es wird nach der kleinstmöglichen Anzahl von Hyperkanten gesucht, die den Hypergraphen in k zusammenhängende Komponenten von annähernd gleicher Dimension unterteilen. Für die Implementierung dieser Methode wird die Hypergraphs-Partitionierungspaket *HMETIS* verwendet.

Hybrid Bipartite Graph Formulation (HBGF)

In diesem letzten Algorithmus werden die Clusters und Objekten zusammen in derselben Graph modelliert. Bei dieser Methode wird der bipartite-Graph so gebildet, dass es keine Kanten zwischen Knoten gibt, falls sie beide entweder Objekte oder Clusters sind. Also es gibt nur Kanten zwischen zwei Knoten, falls ein Knoten einem Cluster repräsentiert und der Zweite einem Objekt, der zu diesem Cluster gehört, repräsentiert. Die *consensus* Partitionierung wird unter der Verwendung von METIS oder *Spectral – Clustering* erhalten.

Darstellung von Gruppen von Clusterings als Hypergraph

Für jeden Label Vektor (*Clustering*) $\lambda^{(q)} \in \mathbb{N}^n$ konstruieren wir die binäre Zugehörigkeitsindikatormatrix $H^{(q)} \in \mathbb{N}^{n \times k^{(q)}}$, wobei jeder Cluster als Hyperkante (*Spalte*) dargestellt wird (Siehe Abb.2). Alle Einträge einer Zeile in der binären Zugehörigkeitsindikatormatrix $H^{(q)}$ werden zu 1 addiert, falls die Zeile mit einem Objekt, welches sein Cluster bekannt ist, übereinstimmt. und Zeilen für Objekte mit unbekanntem Cluster sind mit Null ausgefüllt.
Die Blockmatrix $H = (H^{(1)} \dots H^{(r)})$ definiert die Adjazenzmatrix eines Hypergraphen mit n Knoten und $\sum_{q=1}^{r} k^{(q)}$ Hyperkanten. Jeder Spaltenvektor $\boldsymbol{h_a}$ spezifiziert eine Hyperkante h_a, wobei 1 anzeigt, dass der Knoten mit der entsprechenden Zeile zu der Hyperkante gehört und 0 gibt an, dass dies nicht der Fall ist. Somit haben wir jeder Cluster zu einem Hyperkante und die Menge von Clusterings zu einem Hypergraph abgebildet.

Darstellung von Gruppen von Clusterings als Hypergraph

Beispiel für die Umwandlung von Clusterings zu einem Hypergraph:

	$\lambda^{(1)}$	$\lambda^{(2)}$	$\lambda^{(3)}$	$\lambda^{(4)}$
x_1	1	2	1	1
x_2	1	2	1	2
x_3	1	2	2	?
x_4	2	3	2	1
x_5	2	3	3	2
x_6	3	1	3	?
x_7	3	1	3	?

$\Leftrightarrow$

	$\mathbf{H}^{(1)}$			$\mathbf{H}^{(2)}$			$\mathbf{H}^{(3)}$			$\mathbf{H}^{(4)}$	
	$\mathbf{h}_1$	$\mathbf{h}_2$	$\mathbf{h}_3$	$\mathbf{h}_4$	$\mathbf{h}_5$	$\mathbf{h}_6$	$\mathbf{h}_7$	$\mathbf{h}_8$	$\mathbf{h}_9$	$\mathbf{h}_{10}$	$\mathbf{h}_{11}$
v_1	1	0	0	0	1	0	1	0	0	1	0
v_2	1	0	0	0	1	0	1	0	0	0	1
v_3	1	0	0	0	1	0	0	1	0	0	0
v_4	0	1	0	0	0	1	0	1	0	1	0
v_5	0	1	0	0	0	1	0	0	1	0	1
v_6	0	0	1	1	0	0	0	0	1	0	0
v_7	0	0	1	1	0	0	0	0	1	0	0

Cluster-based Similarity Partitioning Algorithm

1. Der eingangsbezogene Durchschnitt von r solchen Matrizen, die die r Gruppen darstellen, ergibt eine Gesamtähnlichkeitsmatrix S mit einer feineren Auflösung. Einträge von S bezeichnen den Anteil der Gruppierungen, in denen sich zwei Objekte im selben Cluster befinden, und können mit einer einzigen dünnbesetzten Matrixmultiplikation ($sparse - matrix$) berechnet werden $S = \frac{1}{r} HH^T$.

Cluster-based Similarity Partitioning Algorithm

1. Der eingangsbezogene Durchschnitt von r solchen Matrizen, die die r Gruppen darstellen, ergibt eine Gesamtähnlichkeitsmatrix S mit einer feineren Auflösung. Einträge von S bezeichnen den Anteil der Gruppierungen, in denen sich zwei Objekte im selben Cluster befinden, und können mit einer einzigen dünnbesetzten Matrixmultiplikation ($sparse - matrix$) berechnet werden $S = \frac{1}{r} HH^T$.
2. Nun können wir die Ähnlichkeitsmatrix verwenden, um die Objekte mit einem beliebigen sinnvollen Clustering-Algorithmus auf der Basis von Ähnlichkeit zu *reclustern*. In unserem Fall entscheiden wir uns für eine Partitionierung des induzierten Ähnlichkeitsgraphen (Knote = Objekt, Kantengewicht = Ähnlichkeit) mit METIS

Cluster-based Similarity Partitioning Algorithm

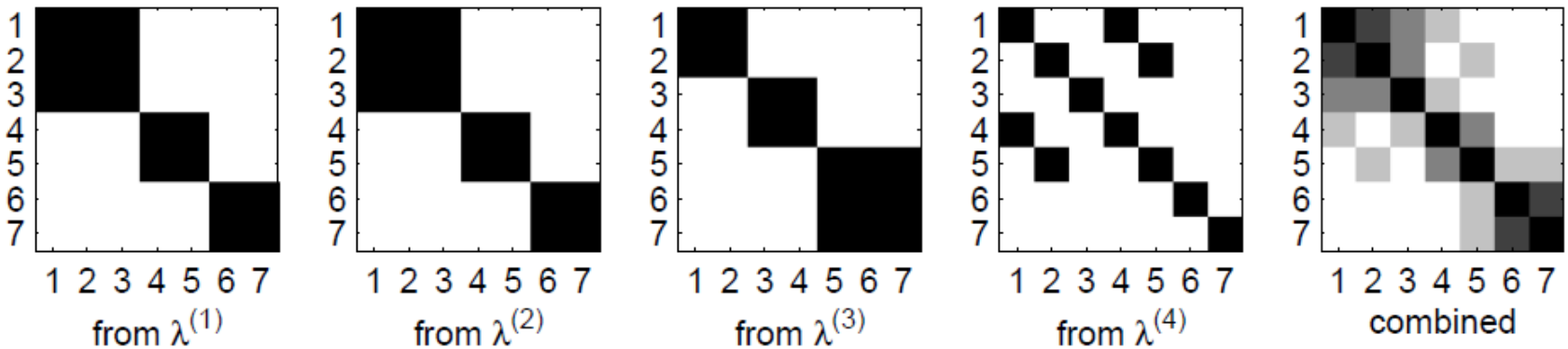
Berechnung von S für das Ensemble Problem:

$r \cdot S =$

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	4	3	2	1	0	0	0
x_2	3	4	2	0	1	0	0
x_3	2	2	3	1	0	0	0
x_4	1	0	1	4	2	0	0
x_5	0	1	2	4	1	1	1
x_6	0	0	0	0	1	3	3
x_7	0	0	0	0	1	3	3

Cluster-based Similarity Partitioning Algorithm

Veranschaulichung von (CSPA) für das in Abb.2 angegebene Cluster-Ensemble-Beispielproblem. Jedes Clustering besitzt eine Ähnlichkeitsmatrix. Die Matrixeinträge werden durch Dunkelheit proportional zur Ähnlichkeit dargestellt. Ihr Durchschnitt wird dann verwendet, um die Objekte neu zu clustern und einen Konsens zu erhalten.



Mehrstufige Graphenbisektion

Ein mehrstufiger Graphbisektionsalgorithmus funktioniert wie folgt: betrachte einen gewichteten Graphen $G_0 = (V_0, E_0)$ mit Gewichten sowohl aus Knoten als auch auf Kanten. Dieser Algorithmus besteht aus den folgenden drei Phasen:

1. Vergröberungsphase *Coarsening*: Der Graph G_0 wird in ein Folge von kleineren Graphen $G_1, G_2, \ldots, G_m$ umgewandelt, so dass $|V_0| > |V_1| > \ldots > |V_m|$.

Mehrstufige Graphenbisektion

Ein mehrstufiger Graphbisektionsalgorithmus funktioniert wie folgt: betrachte einen gewichteten Graphen $G_0 = (V_0, E_0)$ mit Gewichten sowohl aus Knoten als auch auf Kanten. Dieser Algorithmus besteht aus den folgenden drei Phasen:

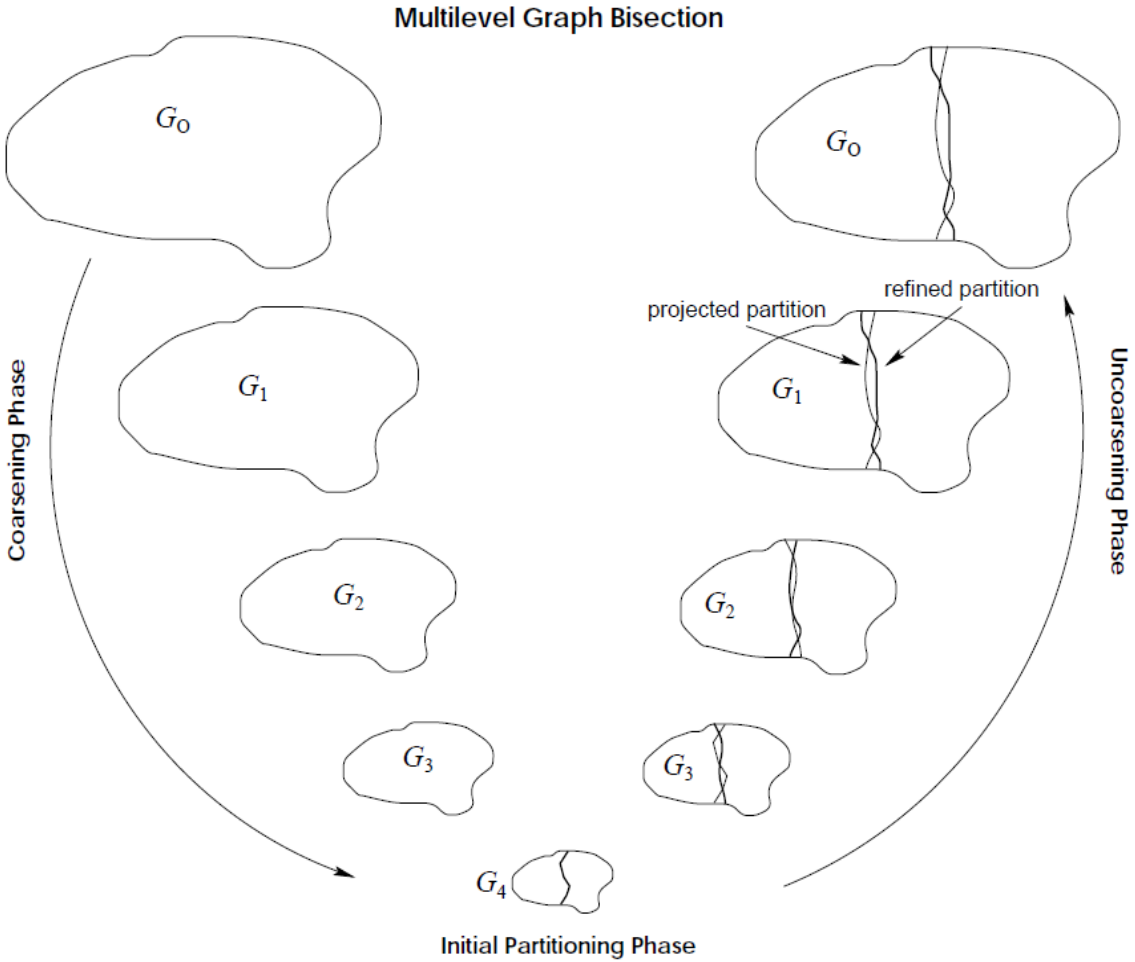
1. Vergröberungsphase *Coarsening*: Der Graph G_0 wird in ein Folge von kleineren Graphen $G_1, G_2, \ldots, G_m$ umgewandelt, so dass $|V_0| > |V_1| > \ldots > |V_m|$.
2. Partitionierungsphase: Eine 2-fache-Partition P_m des Graphen $G_m = (V_m, E_m)$ wird berechnet, die V_m in zwei Teile unterteilt, die jeweils die Hälfte der Knoten von G_0 enthält.

Mehrstufige Graphenbisektion

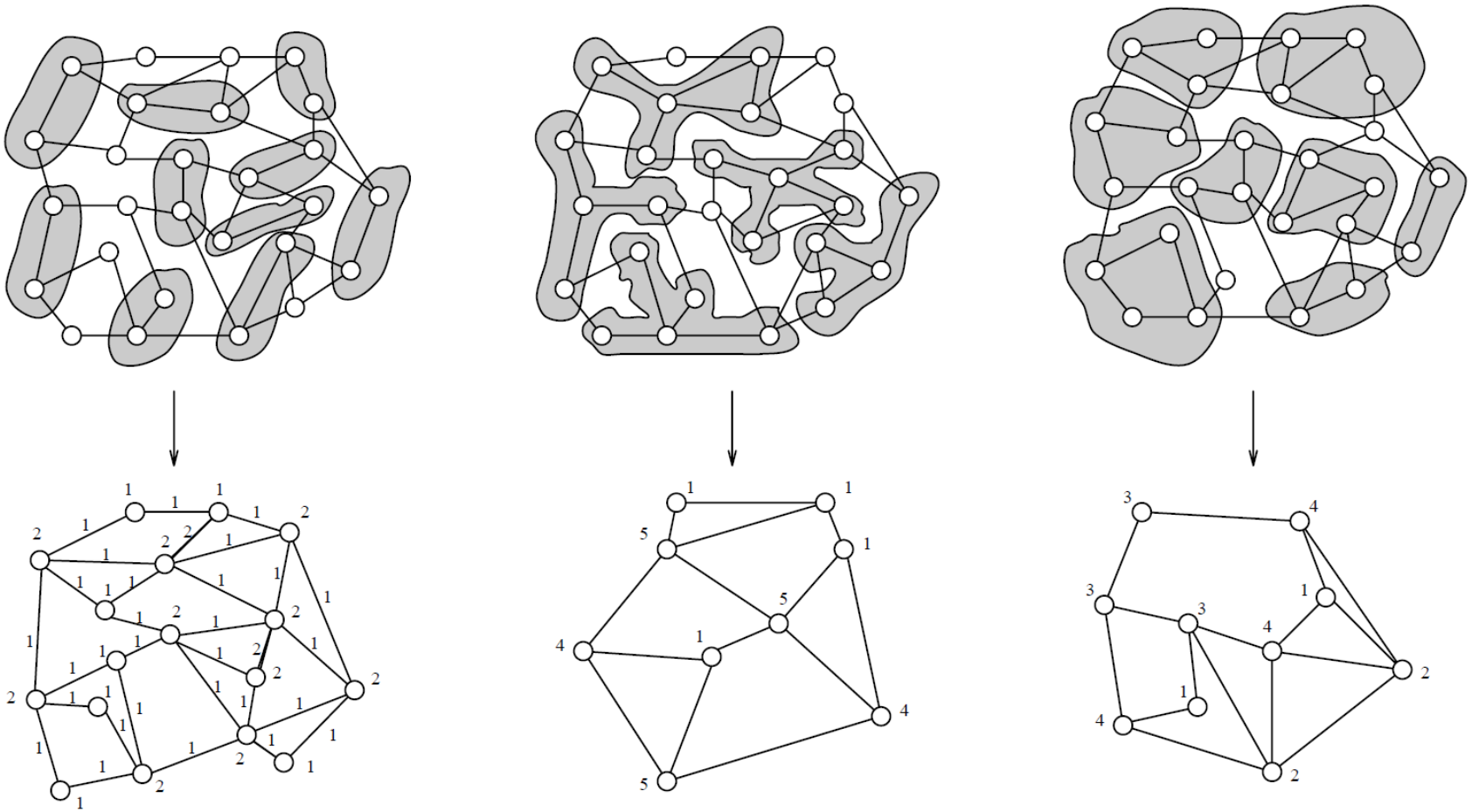
Ein mehrstufiger Graphbisektionsalgorithmus funktioniert wie folgt: betrachte einen gewichteten Graphen $G_0 = (V_0, E_0)$ mit Gewichten sowohl aus Knoten als auch auf Kanten. Dieser Algorithmus besteht aus den folgenden drei Phasen:

1. Vergröberungsphase *Coarsening*: Der Graph G_0 wird in ein Folge von kleineren Graphen $G_1, G_2, \ldots, G_m$ umgewandelt, so dass $|V_0| > |V_1| > \ldots > |V_m|$.
2. Partitionierungsphase: Eine 2-fache-Partition P_m des Graphen $G_m = (V_m, E_m)$ wird berechnet, die V_m in zwei Teile unterteilt, die jeweils die Hälfte der Knoten von G_0 enthält.
3. Entgröberungsphase *Uncoarsening*: Die Partition P_m von G_m wird auf G_0 zurückprojiziert, indem man durch die Zwischenpartitionen $P_{m-1}, P_{m-2}, \ldots, P_1, P_0$ geht.

Mehrstufige Graphenbisektion



Vergröberungsphase (*Coarsening*)



Vergröberungsphase (*Coarsening*)

Random matching (RM)

Ein maximales Matching kann mit einen randomisierten Algorithmus generiert werden. Die Knoten sind dann in zufälliger Reihenfolge besucht. Wenn ein Knoten u noch nicht ausgewählt wurde, dann wählen wir einen seiner nicht ausgewählter Nachbarknoten aus. Wenn es solcher Knoten v existiert, wird die Kante (u, v) zum Matching gefügt und markieren wir die Knoten u und v als besucht. Wenn es keinen solchen Konten v gibt, dann bleibt u wie es ist in der RM. Die Komplexität von diesem Algorithmus liegt in $O(|E|)$.

Partitionierungsphase

Ziel von der Partitionierungsphase ist den Graphen in zwei Teile zu schneiden so, dass jeder Teil ungefähr die Hälfte des Knotengewichts des ursprünglichen Graphen enthält. Eine Partition von G_m kann mit verschiedenen Algorithmen erhalten werden, wie z.B:

1. spektrale Bisektion *SB*

Partitionierungsphase

Ziel von der Partitionierungsphase ist den Graphen in zwei Teile zu schneiden so, dass jeder Teil ungefähr die Hälfte des Knotengewichts des ursprünglichen Graphen enthält. Eine Partition von G_m kann mit verschiedenen Algorithmen erhalten werden, wie z.B:

1. spektrale Bisektion SB
2. geometrische Bisektion (wenn Koordinaten verfügbar sind)

Partitionierungsphase

Ziel von der Partitionierungsphase ist den Graphen in zwei Teile zu schneiden so, dass jeder Teil ungefähr die Hälfte des Knotengewichts des ursprünglichen Graphen enthält. Eine Partition von G_m kann mit verschiedenen Algorithmen erhalten werden, wie z.B:

1. spektrale Bisektion SB
2. geometrische Bisektion (wenn Koordinaten verfügbar sind)
3. kombinatorische Methoden

Graph growing partitioning algorithm (GGP)

GGP

Eine einfache Methode zur Bisektion des Graphen und sie funktioniert wir folgt:
Man startet bei einem beliebigen Knoten und bildet eine Region um ihn herum, bis die Hälfte der Knoten oder die Hälfte des gesamten Knotengewichtes einbezogen ist. Die Qualität des *GGP* ist abhängig von der Wahl eines Knotens, von dem aus das Wachstum des Graphen beginnt, und verschiedene Startknoten ergeben unterschiedliche Kantenschnitte. Um dieses Problem teilweise zu lösen, wählen wir zufällig 10 Eckpunkte und lassen 10 verschiedene Regionen wachsen. Der Versuch mit dem kleineren Kantenschnitt wird als die Partition ausgewählt.

Entgröberungsphase (*Uncoarsening*)

Inhalt...

Segmentierungskombination

Der Random-Walker-Algorithmus wird für einen ungerichteten Graphen $G = (V, E, w)$ formuliert, in dem jedes Pixel p_i einen entsprechenden Knoten $v_i \in V$ hat. Jede Kante $e_{ij} \in E$ hat ein Gewicht w_{ij}, das die Ähnlichkeit zwischen den benachbarten Pixeln v_i und v_j (in 4-Nachbarschaft). In diesem *Framework* wird der Anzahl der Regionen in einem Bild als K bezeichnet. Der tatsächliche Anzahl ist eigentlich nicht bekannt. Bei der Berechnung einer Reihe von Kombinationssegmentierungen mit $K \in [K_{min}, K_{max}]$ Seed-Regionen, brauchen wir ein Verfahren um die optimale Region auszuwählen. In[7] wird die optimale Segmentationskombination S^{kopt} so ausgewählt, dass es die eine mit maximale durchschnittliche gegenseitige Information zwischen alle Segmentastionen S_q in Λ, wobei $\Lambda = \{S_1, \ldots, S_N\}$ ist. Die optimale S^{kopt} wird nach der Gleichung (5) berechnet, wobei $\hat{\lambda}$ bedeckt all mögliche $K \in [K_{min}, K_{max}]$ Segmentastionen.

Anwendungen

(a) ANMI = 0.6083 (b) ANMI = 0.6577 (c) ANMI = 0.6997

(d) k=21; ANMI=0.7102 (e) k=20; ANMI=0.7172

(a)-(c) zeigen drei Eingabesegmentierungen mit dem schlechtesten, dem mittleren und dem besten ANMI. (d) Kombinierte Segmentierung mit dem vorgeschlagenen Algorithmus und (e) Kombinierte Segmentierung mit optimalem K.