

Ensemble Clustering

Seminar im Fachgebiet Wissensverarbeitung

Malek Alsalamat

Universität Kassel

1 Einleitung

Menschen haben immer die Interesse, Sachen in Gruppen einzuordnen. Mit Hilfe ihren Sinnesorganen können sie das leicht machen. Der Mensch kann zwischen zwei Bildern von einem Hund und einer Katze sofort durch sein Sehen unterscheiden. Aber wie kann ein Computer das schaffen, das keine Sinnesorgane hat? Clustering und Klassifikation sind zwei Sachen, die das Computer benutzt, um Objekte in Gruppen oder Clusters anhand der Daten dieser Objekte einzuordnen.

Klassifikation: ist der Prozess, in dem entschieden wird, zu welchem Cluster das neue Objekt gehört. Aber hier wird darauf nicht eingegangen, sondern nur auf Clustering.

2 Was ist Clustering

Clustering ist die Methode im maschinellen Lernen und genauer gesagt "im überwachten 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.

Also Clustering führt zu der Verringerung von der Komplexität.

Die gefundenen Ähnlichkeitsgruppen können graphentheoretisch, hierarchisch, partitionierend oder optimierend sein. Inzwischen gibt es im Clustering eine große Vielfalt an Methoden, die eingesetzt werden können. Das hängt aber vom Anwendungsfall und der Größe der Datenmenge ab.

Was man aber dabei beachten soll, dass die Algorithmen verschiedene Ergebnisse liefern können. Was öfter der Fall ist.

→ Kurzzusammenfassung Aufbau der Arbeit

evtl. stehen?

das ist doch im Prinzip die Einleitung oder?

u

- 2- / das

1?

Klasse

→ Vorsicht: Klassifikation ordnet Objekte einer Klasse zu, also einer Menge von Objekten, bei denen eine ausgezeichnete Eigenschaft gleich ist.

entfernen was das bedeuten soll Quellen belegen?

Die klassische Methode ist

3 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:

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

4 Ensemble Clustering

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 einzigste Lösung vom gegebenen Problem zu liefern.

Die Ensemble Methode war zuerst nur im Bereich des überwachten Lernens benutzt und betrachtet, aber aufgrund seiner erfolgreichen Applikationen in Klassifizierungsaufgaben wurde es versucht, die Ensemble Methode auf unüberwachtes Lernen anzuwenden. Insbesondere auf Clustering Probleme. Die Gründe dafür sind erstens, dass es kein vorhandenes Wissen über die zugrundeliegende Struktur. Zweitens es gibt keinen einzelnen Clustering-Algorithmus, der für verschiedene Probleme konsistent gute Leistungen erbringen kann.

Ensemble Clustering zielt darauf ab, mehrere Clusteringsmodelle zu kombinieren, um ein besseres Ergebnis als die einzelnen Clustering-Algorithmen. Dies ist in vielen Kontexten sehr nützlich:

- Qualität und Robustheit
- Knowledge Reuse
- Distributed Computing

Man kann jetzt Ensemble Clustering als Problem sehen und zwar wie folgendes:

Gegeben: Mehrere Clusterings aus verschiedenen Algorithmen von derselben Datenmenge.

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

wo kommen die auf einmal her?

warum heißt es partitionierend?

1. wie für das

unglückliche Formulierung

Uff: Das E.C Problem lässt sich wie folgt beschreiben:

5 Das Ensemble Problem

Notation: Sei $X = \{x_1, x_2, \dots, x_n\}$ eine Menge von Objekten (Datenpunkte). Eine Partitionierung dieser n Objekten in k Clusters kann als eine Menge von k Mengen von Objekten $\{C_l \mid l = 1, \dots, k\}$ 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) $\lambda^{(1, \dots, r)}$ ist zu einem Clustering anhand einer Consensus Funktion kombiniert.

5.1 Beispiel

Sei $X = \{x_1, x_2, \dots, x_7\}$ eine Menge von Objekte und 4 Clusters $\{C_1, C_2, C_3, C_4\}$. Die folgende Label Vektoren spezifizieren vier Clusterings von X .

$$\begin{aligned} \lambda^{(1)} &= (C_1, C_1, C_1, C_2, C_2, C_3, C_3)^T & \lambda^{(2)} &= (C_2, C_2, C_2, C_3, C_3, C_1, C_1)^T \\ \lambda^{(3)} &= (C_1, C_1, C_2, C_2, C_3, C_3, C_3)^T & \lambda^{(4)} &= (C_1, C_2, ?, C_1, C_2, ?, ?)^T \end{aligned}$$

$\lambda^{(1)}$ wird so gelesen: $x_1, x_2, x_3 \in C_1$ usw.

Mit scharfem Sehen kann man erkennen, dass die Label Vektoren $\lambda^{(1)}$ und $\lambda^{(2)}$ logischerweise identisch sind. $\lambda^{(3)}$ liefert einen Unterschied von der Zugehörigkeit der Objekte x_3 und x_5 . $\lambda^{(4)}$ weicht sich von der Anderen sehr ab, außerdem $\lambda^{(4)}$ enthält *missing Data*.

Gesucht ist jetzt wie vorher gesprochen 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 $\lambda^{(2)}$.

5.2 Objektive Funktion für Ensemble Clustering

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

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

Wenn es grundsätzliche keine Informationen über die relative Bedeutung der einzelnen Gruppierungen gibt, dann ist eine sinnvolle Ziel für die *consensus* Antwort, ein Clustering zu suchen, welches die meisten Informationen mit dem ursprünglichen *BC* Clusterings teilt.

Im Folgenden brauchen wir zwei Definitionen und zwar von Entropie und gegenseitige Information.

Entropie: ist die minimale Beschreibungskomplexität einer Zufallsvariablen.

Gegenseitige Information: ist die Kommunikationsrate im Gegenwart von Rauschen.

Gegenseitige Information, die ein symmetrisches Maß zur Quantifizierung der statistischen Informationen zwischen zwei Distributionen ist, liefert einen Hinweis auf die gemeinsame Informationen zwischen einem Paar von Clusterings.

selbst
ausgedacht?
aus
Quelle?

klare

-?
↑ (consensus info)

Seien X und Y zwei Zufallsvariablen beschrieben durch das Clustering $\lambda^{(a)}$ und $\lambda^{(b)}$ mit $k^{(a)}$ und $k^{(b)}$ Clusters. $I(X, Y)$ beschreibt die gegenseitige Information zwischen X und Y , und $H(X)$ beschreibt die Entropie von X .

Zur Vereinfachung der Interpretation und Vergleichbarkeit ist eine normalisierte Version von $I(X, Y)$, welche zwischen 0 und 1 liegt, gewünscht. Dies folgt zur normalisierten gegenseitigen Information NMI:

$$NMI(X, Y) = \frac{I(X, Y)}{\sqrt{H(X)H(Y)}}. \quad (2)$$

Da $I(X, Y)$ eine Metrik ist, also es gilt $I(X, Y) \leq \min(H(X), H(Y))$, ist dann $NMI(X, X) = 1$ genauso wie gewünscht.

Die zweite Gleichung kann durch von *Clusterings* bereitgestellten Strichproben geschätzt werden. 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:

$$\phi^{(NMI)}(\lambda^{(a)}, \lambda^{(b)}) = \frac{\sum_{h=1}^{k^{(a)}} \sum_{l=1}^{k^{(b)}} n_{h,l} \cdot \log \left(\frac{n \cdot n_{h,l}}{n_h^{(a)} \cdot n_l^{(b)}} \right)}{\sqrt{\left(\sum_{h=1}^{k^{(a)}} n_h^{(a)} \cdot \log \frac{n_h^{(a)}}{n} \right) \left(\sum_{l=1}^{k^{(b)}} n_l^{(b)} \cdot \log \frac{n_l^{(b)}}{n} \right)}}. \quad (3)$$

Von Gleichung (3) wird ein Maß zwischen einer Menge Λ und einem einzelnen Clustering $\hat{\lambda}$ als die durchschnittliche normalisierte gegenseitige Information definiert (ANMI):

$$\phi^{(ANMI)}(\Lambda, \hat{\lambda}) = \frac{1}{r} \sum_{q=1}^r \phi^{(NMI)}(\hat{\lambda}, \lambda^{(q)}). \quad (4)$$

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 objektive Funktion und $\lambda^{(k-opt)}$ ist:

$$\lambda^{(k-opt)} = \arg \max_{\hat{\lambda}} \sum_{q=1}^r \phi^{(NMI)}(\hat{\lambda}, \lambda^{(q)}), \quad (5)$$

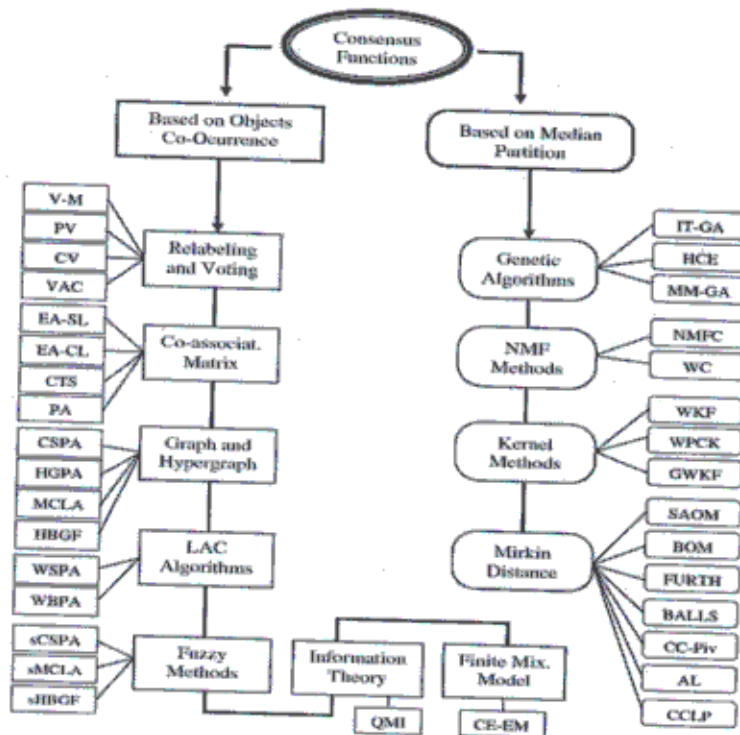
wobei $\hat{\lambda}$ durch alle mögliche k -Partitionierungen läuft.

In unserem Beispiel enthält das Clustering $\lambda^{(4)}$ *missingData*. Für solche Fälle kann das objektive *consensus* Clustering aus der Gleichung (5) durch Berechnung eines gewichteten Durchschnitts von der gegenseitigen Information mit allen bekannten Clusters verallgemeinert werden.

$$\lambda^{(k-opt)} = \arg \max_{\hat{\lambda}} \sum_{q=1}^r |\mathcal{T}^{(q)}| \phi^{(NMI)}(\hat{\lambda}_{\mathcal{T}^{(q)}}, \hat{\lambda}_{\mathcal{T}^{(q)}}^{(q)}). \quad (6)$$

Wobei $\mathcal{T}^{(i)}$ ist eine Menge von Indizes der Objekten mit bekannten Clusters für das i -te Cluster.

objektive Funktion
= Zielfunktion!



Quelle?

6 Effiziente consensus Funktion

In diesem Abschnitt werden einige Verfahren zur Lösung des Ensemble Clustering Problems vorgestellt. Es gibt zwei Arten von *consensus* Funktionen, eine basiert auf die Median Partitionierung Approximation und die zweite basiert auf das gemeinsame Auftreten von Objekten, die sogenannte *objects co-occurrence approach*.

Eine Einordnung von der wichtigsten *consensus* Funktionen werden in *Abbildung 1* vorgestellt. Aber wir werden hier uns nur auf die *Graph and hypergraph based methods* fokussieren.

6.1 Graph and hypergraph based methods

Diese Art von Clustering-Ensemble-Methoden transformieren das Kombinationsproblem zu einem Graph bzw. Hypergraph-Partitionierungsproblem. Der Unterschied zwischen den Methoden liegt auf die Art und Weise, wie der (Hyper-)Graph aus der Menge der Clusterings aufgebaut wird, und wie Kantenschnitt von der Graph definiert ist, um die *consensus* Partitionierung zu erhalten. Es gibt vier Methoden für diese Art von Clustering und zwar:

und da, die Drogen?

- 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. Quelle?
- 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 die Kantenelimination **Guten Ausdruck hier finden von Vega 2011**. Für die Implementierung dieser Methode wird die Hypergraphs-Partitionierungspaket HMETIS verwendet. Quelle?
- Meta-CLustering Algorithm (MCLA): Vor allem ist hier die Ähnlichkeit zwischen zwei Clusters C_i und C_j wird anhand des Jaccar-Indexes bezüglich der Menge der Objekte definiert **Bessere Formulierung Finden**. Somit ist die Ähnlichkeitsmatrix zwischen Clusters gebildet, welche die Adjazenzmatrix des Graphen darstellt. Aber unter der Berücksichtigung, dass die Clusters als Knoten sind und der Kante zwischen zwei Knoten gewichtet ist, wobei das Gewicht die Ähnlichkeit zwischen Clusters entspricht. Danach wird dieser Graph mit dem METIS partitioniert und das erhaltene Cluster wird als Meta-Cluster bezeichnet. Schließlich, um die eindeutige Partitionierung zu finden, wird die Anzahl jedes Objektes im Meta-Cluster berechnet und jeder Objekt wird zu einem Meta-Cluster geordnet, in dem er öfter aufgetreten ist. Dieses Algorithmus hat einen Aufwand von $O(n^2)$. Quelle?
- 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. warum ist Komplexität hier hier angegeben?
Quelle?

6.2 Darstellung von Gruppen von Clusterings als Hypergraph

Wie wir schon im vorherigen Abschnitt gesehen haben, die Clusterings sollten zu einem Hypergraph umgewandelt werden, damit eine der vier Methoden verwendet werden kann. Und das ist genau unser Ziel in diesem Abschnitt. Ein **Hypergraph** besteht aus Knoten und Hyperkanten. Eine Kante in einem

| | $\lambda^{(1)}$ | $\lambda^{(2)}$ | $\lambda^{(3)}$ | $\lambda^{(4)}$ | | $H^{(1)}$ | | | $H^{(2)}$ | | | $H^{(3)}$ | | | $H^{(4)}$ | |
|-------|-----------------|-----------------|-----------------|-----------------|-----------------------|-----------|-------|-------|-----------|-------|-------|-----------|-------|-------|-----------|----------|
| | | | | | | h_1 | h_2 | h_3 | h_4 | h_5 | h_6 | h_7 | h_8 | h_9 | h_{10} | h_{11} |
| x_1 | 1 | 2 | 1 | 1 | v_1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| x_2 | 1 | 2 | 1 | 2 | v_2 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| x_3 | 1 | 2 | 2 | ? | $\Leftrightarrow v_3$ | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| x_4 | 2 | 3 | 2 | 1 | v_4 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| x_5 | 2 | 3 | 3 | 2 | v_5 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| x_6 | 3 | 1 | 3 | ? | v_6 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| x_7 | 3 | 1 | 3 | ? | v_7 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

regulären Graph verbindet *genau* zwei Knoten. Eine **Hyperkante** ist eine Verallgemeinerung einer Kante, da sie jede Menge von Knoten verbinden kann.

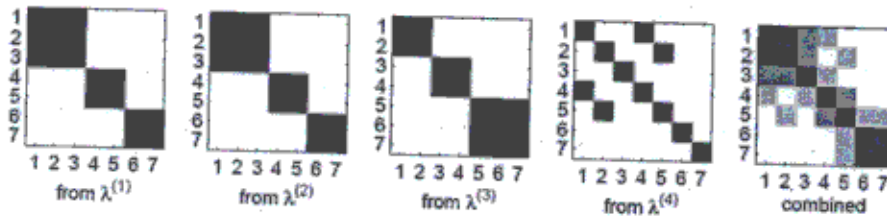
Für jeden Label Vektor (Clustering) $\lambda^{(q)} \in \mathbb{N}^n$ konstruieren wir die binäre Zugehörigkeitsindikatormatrix $H^{(q)} \in \mathbb{N}^{n \times k^{(q)}}$, wobei jeder Cluster als Hyperkante (Spalte) dargestellt wird (Siehe Tabelle 1). Alle Einträge einer Zeile in der binären Zugehörigkeitsindikatormatrix $H^{(q)}$ 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 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.

7 Cluster-based Similarity Partitioning Algorithm

In diesem Abschnitt wird das *Cluster-based Similarity Partitioning Algorithm* im Details geklärt. Wie bereit gesprochen, in diesem Algorithmus sollte zuerst eine Ähnlichkeitsmatrix werden. Basieren auf einer grob aufgelösten Suchtweise haben zwei Objekte eine Ähnlichkeit von 1, falls sie im gleichen Cluster sind und eine Ähnlichkeit von 0 ansonsten. Somit kann man eine $n \times n$ binäre Ähnlichkeitsmatrix für jedes Clustering einfach erstellen.

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 $S = \frac{1}{r} H H^T$. Abbildung 2 veranschaulicht die Erstellung der clusterbasierten Ähnlichkeitsmatrix für das in Tabelle 1 angegebene Beispiel. Nun können wir die Ähnlichkeits-



matrix verwenden, um die Objekte mit einem beliebigen sinnvollen Clustering-Algorithmus auf der Basis von Ähnlichkeit. In unserem Fall entscheiden wir uns für eine Partitionierung des induzierten Ähnlichkeitsgraphen (Knoten = Objekt, Kantengewicht = Ähnlichkeit) mit METIS wegen seiner robusten und skalierbaren Eigenschaften.

CSPA ist die einfachste und naheliegendste Heuristik, aber ihr Rechen- und Speicheraufwand Komplexität sind beide quadratisch in $O(n^2)$.

7.1 Mehrstufige Graphenbisektion

Der Graph G kann mit einem mehrstufigen Algorithmus halbiert werden. Die Grundstruktur eines mehrstufigen Algorithmus ist sehr einfach. Der Graphen G wird zunächst auf einige hundert Knoten runter vergrößert, dann wird eine Bisektion dieses viel kleineren Graphen berechnet und diese Partition dann auf den ursprünglichen Graphen zurückprojiziert Graph (feinerer Graph). Bei jedem Schritt der Entgröberung des Graphen wird die Partition weiter verfeinert. Solche Verfeinerungen verringern in der Regel den Kantenschnitt (*edge - cut*).

Also 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:

- Vergrößerungsphase *Coarsening*: Der Graph G_0 wird in eine Folge von kleineren Graphen G_1, G_2, \dots, G_m umgewandelt, so dass $|V_0| > |V_1| > \dots > |V_m|$.
- 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.
- Entgrößerungsphase *Uncoarsening*: Die Partition P_m von G_m wird auf G_0 zurückprojiziert, indem man durch die Zwischenpartitionen $P_{m-1}, P_{m-2}, \dots, P_1, P_0$ geht.

7.1.1 Vergrößerungsphase (*Coarsening*) :

In diesem Unterabschnitt beschreiben wir die grundlegende Ideen hinter der Vergrößerung mit Hilfe von *Matchings*.

In den meisten Vergrößerungsschemata wird eine Menge von Knoten von G_i zu einem einzigen Knoten des größeren Graphen der nächsten Ebene G_{i+1} kombiniert. Sei V_i^v die Menge der Knoten von G_i , die zu einem Knoten v von G_{i+1}

? was fließt?

Quelle?

als wer beschäftigt du dich sehr mit dem Details wie man die

Partition im Graphen bekommt
Das kannst du auch weniger Details und dich stattdessen mit anderen Aspekten von Clustering beschäftigen

Warum brauchen wir das?
Überleitung?

zusammengefasst wird. Wir werden v als *Multi-Knote* bezeichnen. Damit eine Bisektion eines größeren Graphen in Bezug auf den ursprünglichen Graphen gut ist, wird das Gewicht des Knotens v gleich der Summe der Gewichte der Knoten in V_i^v . Außerdem, um die Konnektivitätsinformationen im größeren Graphen zu behalten, sind die Kanten von v die Vereinigung der Kanten der Knoten in V_i^v . Für den Fall, dass mehr als ein Knoten von V_i^v Kanten von demselben Knoten u enthält, ist dann das Gewicht der Kante von v gleich der Summe der Gewichte dieser Kanten. Dies ist nützlich, wenn wir die Qualität einer Partition in einem größeren Graphen bewerten wollen. Der Kantenschnitt der Partition in einem vergrößerten Graphen ist gleich dem Kantenschnitt der gleichen Partition in einem feineren Graphen.

Bei einem Graphen $G_i = (V_i, E_i)$ kann man einen größeren Graphen durch Kollabieren benachbarter Knoten erhalten. So wird die Kante zwischen zwei Knoten kollabiert und ein *Multi-Knote* wird aus diesen beiden Knoten erstellt. Diese Idee von Kollabieren von Kanten lässt sich in Form von *Matching* beschreiben werden.

Ein *Matching* eines Graphen ist eine Menge von Kanten, von denen keine zwei die auf denselben Knoten treffen. Der nächste größere Graph G_{i+1} wird also aus G_i konstruiert, indem ein *Matching* von G_i gefunden wird und die Knoten von diesem *Matching* zu einem *Multi-Knote* kollabieren. Alle Kanten, die nicht im *Matching* waren, werden einfach im G_{i+1} kopiert. Da das Ziel des Kollabieren von Knoten mit Hilfe von *Matching* darin besteht, die Größe des Graphen G_i zu verringern, sollte das *Matching* eine große Anzahl von Kanten enthalten. Aus diesem Grund werden *maximale Matchings* verwendet, um die aufeinanderfolgenden groben Graphen zu erhalten. Ein *Matching* ist maximal wenn jede Kante im Graphen, die nicht im *Matching* enthalten ist, mindestens einen ihrer Endknoten übereinstimmt.

? Beachten Sie, dass die Anzahl der Kanten, die zum maximalen *Matching* gehören, je nach Berechnungsmethode unterschiedlich sein kann. Das maximale *Matching*, das die maximale Anzahl von Kanten hat, wird als *maximum Matching* genannt. Da jedoch die Komplexität der Berechnung eines *maximum Matching* im Allgemeinen höher ist als die Berechnung eines maximalen *Matchings* ist, werden letztere bevorzugt. Im verbleibenden Unterabschnitt beschreiben wir zwei Methoden, die wir zur Auswahl maximaler *Matchings* für die Vergrößerung.

- Random matching (RM): Ein maximales *Matching* kann mit einem 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 Knoten v gibt, dann bleibt u wie es ist in der RM. Die Komplexität von diesem Algorithmus liegt in $O(|E|)$.
- Heavy edge matching (HEM): RM ist ein einfaches und effizientes Verfahren zur Berechnung einer maximalen *Matching* und minimiert die Anzahl der

Vergrößerungsstufen in einer gierigen Weise. Unser übergeordnetes Ziel ist es jedoch, eine Partition zu finden, die den Kantenschnitt *edge-cut* minimiert. Betrachten wir einen Graphen $G_i = (V_i, E_i)$, ein Matching M_i , das zur Vergrößerung von G_i verwendet wird, und seinen größeren Graphen $G_{i+1} = (V_{i+1}, E_{i+1})$, induziert durch M_i . Wenn A eine Menge von Kanten ist, so ist $W(A)$ die Summe der Gewichte der Kanten in A . Es kann gezeigt werden, dass

$$W(E_{i+1}) = W(E_i) - W(M_i). \quad (7)$$

Somit wird das Gesamtkantengewicht des größeren Graphen um das Gewicht des *Matching* verringert. Durch die Auswahl eines maximalen Matchings M_i , dessen Kanten ein großes Gewicht haben, können wir das Kantengewicht des größeren Graphen um einen größeren Betrag verringern. Da er ein geringeres Kantengewicht hat, auch einen kleineren Kantenschnitt. Die Suche nach einem maximalen *Matching*, das Kanten mit großem Gewicht enthält, ist die Idee hinter dem *HEM*. Ein *HEM* wird mit einem randomisierten Algorithmus berechnet, der dem zuvor beschriebenen Algorithmus zur Berechnung eines *RM* ähnelt. Die Knoten werden wieder in zufälliger Reihenfolge besucht.

Anstatt jedoch einen Knoten u zufällig mit einem seiner benachbarten nicht gematchten Knoten zu matchen, wird u mit dem Knoten v so gematcht, dass das Gewicht der Kante (u, v) das Maximum über alle gültigen inzidenten Kanten ist (schwerere Kante).

Beachten Sie, dass dieser Algorithmus nicht garantiert, dass das erhaltene Matching ein maximales Gewicht (über allen möglichen Matchings), aber im Allgemeinen funktioniert er sehr gut. Die Komplexität der Berechnung eines *HEM* ist $O(|E|)$, was asymptotisch ähnlich ist zur Berechnung des *RM*.

7.1.2 Partitionierungsphase :

Die zweite Phase eines Multilevel-Algorithmus berechnet eine qualitativ hochwertige Bisektion (d. h. einen kleinen Kantenschnitt) P_m des groben Graphen $G_m = (V_m, E_m)$, so dass jeder Teil ungefähr die Hälfte des Knotengewichts des ursprünglichen Graphen enthält. Da bei der Vergrößerung die Gewichte der Knoten und Kanten des vergrößerten Graphen so eingestellt ist, dass sie die Gewichte der Knoten und Kanten des feineren Graphen widerspiegeln, enthält G_m Informationen, um auf intelligente Weise die balancierte Partitionierung und die kleinen Kantenschnitt-Anforderungen durchzusetzen.

Eine Partition von G_m kann mit verschiedenen Algorithmen erhalten werden, wie z.B:

- spektrale Bisektion *SB*
- geometrische Bisektion (wenn Koordinaten verfügbar sind)
- kombinatorische Methoden

was bedeutet das?

Quellen?

Da die Größe des gröberen Graphen G_m klein ist (d.h. $|V_m| < 100$), nimmt dieser Schritt nur wenig Zeit in Anspruch. Es wird nur ein Algorithmus in diesem Abschnitt vorgestellt und zwar **Graph growing partitioning algorithm (GGP)**. Eine einfache Methode zur Bisektion des Graphen und sie funktioniert wie 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. Diese Partition wird dann weiter verfeinert, indem sie als Eingabe für die *KL*.

7.1.3 Entgröberungsphase (*Uncoarsening*) :

Während der Vergröberungsphase wird die Partition P_m des vergrößerten Graphen G_m auf den ursprünglichen Graphen zurückprojiziert, indem man durch die Graphen $G_{m-1}, G_{m-2}, \dots, G_1$ geht. Da jeder Knoten von G_{i+1} eine bestimmte Teilmenge von Knoten von G_i enthält, erhält man P_i aus P_{i+1} durch einfaches Zuordnen der Menge der Knoten V_i^v , die mit $v \in G_{i+1}$ kollabiert sind, zu der Partition $P_{i+1}[v]$ (d.h. $P_i[u] = P_{i+1}[v] \quad \forall u \in V_i^v$).