AGML, Sommersemester 2020 Vorlesung 21 (5 Seiten)

Hierarchisches Clustering II

Anne Driemel Letzte Aktualisierung: 10. Juli 2020

In der letzten Vorlesung haben wir uns mit dem hierarchischen Clustering befasst. Wir haben dafür eine Zielfunktion definiert und gezeigt, dass der Single-Link-Clustering Algorithmus ein optimales Ergebnis liefert. Außerdem haben wir einen wichtigen Zusammenhang zwischen minimalen Spannbäumen und dem Single-Link-Clustering gezeigt, der es uns erlaubt, das hierarchische Clustering effizient zu berechnen.

1 Hierarchisches Clustering und k-Center

Was ist aber, wenn wir ein hierarchisches Clustering mit einer anderen Zielfunktion berechnen wollen? Konkret betrachten wir den Fall, dass die dritte Clustering-Eigenschaft wie folgt abgewandelt ist.

Sei X eine Grundmenge und sei $d(\cdot,\cdot)$ eine Metrik auf X. Ein hierarchisches Clustering einer n-elementigen Menge $S \subseteq X$ ist eine geordnete Menge $\mathcal{C} = \{C_1, \ldots, C_n\}$ mit den folgenden Eigenschaften:

(i) (Jede Menge ist eine Partitionierung von S)
Für alle $1 \leq i \leq n$ hat C_i die Form $\{A_1, \ldots, A_{n-i+1}\}$ mit

(a) $\bigcup_{1 \leq j \leq n-i+1} A_j = S$ und
(b) für $j \neq k$ ist $A_j \cap A_k = \emptyset$,

(ii) (Die Mengen sind hierarchisch geschachtelt)
$C_n = \{S\}$ und für alle $1 \leq i < n$ und $A \in C_i$ existiert ein $B \in C_{i+1}$ mit $A \subseteq B$

(iii) (Die Mengen sind optimal)
Für alle $1 \leq i \leq n$ minimiert C_i die folgende Zielfunktion

$$\phi(C_i) = \max_{A \in C_i} \min_{c \in X} \max_{a \in A} d(c, a)$$

über alle möglichen Partitionierungen von S die aus genau $|C_i|$ Teilmengen von S bestehen.

Für ein einzelnes Clustering C_i ist diese Zielfunktion äquivalent zu der im k-Center-Problem, welches wir in der vorletzen Vorlesung kennengelernt haben. Für jede Menge in C_i wird der Radius einer kleinsten umschließenden Kugel gesucht, das Maximum über alle Mengen bestimmt die Zielfunktion. Wir vergleichen also jedes Clustering in der Hierarchie mit dem optimalen k-Center-Clustering. Zusätzlich wollen wir, dass die Mengen hierarchisch geschachtelt sind.

2 Ein Gegenbeispiel

Wir können zunächst feststellen, dass es Mengen S gibt für die keine Lösung existiert, die alle Eigenschaften erfüllt. Wir betrachten dafür das folgende Beispiel.

Beispiel 21.1. *Sei $X = \mathbb{R}$ mit $d(x, y) = |x - y|$, und sei $S = \{0, 4, 6, 10\}$. Ein hierarchisches Clustering von S ist zum Beispiel*

$$C_4 = \{\{0, 4, 6, 10\}\}, C_3 = \{\{0, 4\}, \{6, 10\}\}, C_2 = \{\{0\}, \{4\}, \{6, 10\}\}, C_1 = \{\{0\}, \{4\}, \{6\}, \{10\}\}.$$

Aber hier ist $\phi(C_2) = 2$, während die optimale 3-Center-Lösung den Zielwert 1 hat:

$$\phi(\{0\}, \{4, 6\}, \{10\}) = 1$$

Ein anderes hierarchisches Clustering ist

$$C_4 = \{\{0, 4, 6, 10\}\}, C_3 = \{\{0\}, \{4, 6, 10\}\}, C_2 = \{\{0\}, \{4\}, \{6, 10\}\}, C_1 = \{\{0\}, \{4\}, \{6\}, \{10\}\}.$$

Aber hier ist $\phi(C_3) = 3$, während die optimale 2-Center-Lösung den Zielwert 2 hat.

$$\phi(\{0, 4\}, \{6, 10\}) = 2$$

Wenn wir alle möglichen Mengen $C = \{C_1, \ldots, C_4\}$ welche die Clustering-Bedingungen (i) und (ii) erfüllen analysieren, dann können wir sehen, dass für jede dieser Lösungen entweder $\phi(C_2) \geq 2$ oder $\phi(C_3) \geq 3$ gilt. Also gibt es kein hierarchisches Clustering, welches Bedingung (iii) erfüllt.

3 Der Algorithmus von Dasgupta und Long

In der vorletzten Vorlesung haben wir den Algorithmus von Gonzales kennengelernt. Der Algorithmus berechnet die von ihm gewählten Zentren iterativ. In jedem Schritt ist das berechnete Clustering höchstens um einen Faktor 2 schlechter ist als das optimale Clustering. Allerdings erfüllen die Clusterings nicht unbedingt die Clustering-Bedingung (ii). Wir werden heute den Algorithmus von Dasgupta und Long kennenlernen. Dieser wandelt die Zuweisungen der Punkte zu den von Gonzales-Algorithmus berechneten Clusterzentren so ab, dass hieraus ein hierarchisches Clustering wird. Zunächst müssen wir dazu den Gonzales-Algorithmus leicht abwandeln.

```
Gonzales-Algorithmus(S = {x_1, ..., x_n})
 1. π_1 = 1
 2. d_1, ..., d_n = ∞
 3. for i in 1...n do
 4.       for j in 1...n do
 5.            //d_j speichert den Abstand von x_j zu den bisher gewählten Zentren
 6.            d_j = min(d_j, d(x_j, x_{π_i}))
 7.       r_i = max_{1≤j≤n} d_j
 8.       π_{i+1} = arg max_{1≤j≤n} d_j
 9. Return x_{π_1}, ..., x_{π_n} und r_1, ..., r_n
```

Der einzige Unterschied bis jetzt, ist dass wir den Parameter k auf n gesetzt haben und die Ausgabe erweitert haben. Der Algorithmus gibt eine Permutation der Eingabemenge $x_{\pi_1}, \ldots, x_{\pi_k}$ und die Liste der Radien der berechneten Clusterings zurück. Diese Permutation wird auch *Greedy-Permutation* genannt.

Sei $c_1, \ldots, c_n$ mit $c_i = x_{\pi_i}$ und $r_1, \ldots, r_n$ die Ausgabe des Gonzales Algorithmus für eine Menge S. Wir gruppieren die Punkte nun anhand der Radien in verschiedene *Granularitätsebenen.*

$$L_0 = \{c_1\},\ L_j = \left\{ c_{i+1} \;\middle|\; r_i \in \left(\frac{r_1}{2^j}, \frac{r_1}{2^{j-1}}\right] \right\} \text{ für } j \geq 1.$$

Außerdem definieren wir eine *Elternfunktion* auf der Menge der Punkte. Die Elternfunktion definiert einen gerichteten Graphen G auf der Punktmenge, der eine Baumstruktur hat. Die Wurzel dieses Baumes ist c_1. Sei die Elternfunktion $p : \{2, \ldots, n\} \to \{1, \ldots, n\}$ definiert als

$$p(i) = \arg\min_{1 \leq j \leq n} \left\{ d(c_i, c_j) \;\middle|\; c_j \in \bigcup_{j'=0}^{L(i)-1} L_{j'} \right\}$$

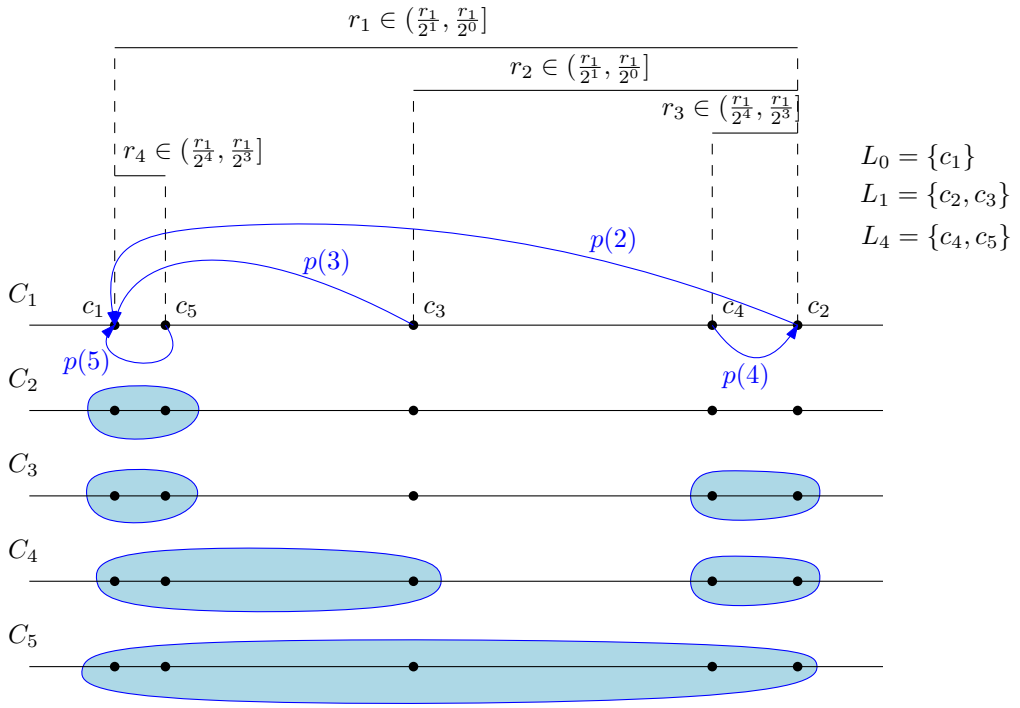

Abbildung 1: Schematische Darstellung eines hierarchischen Clusterings von Dasgupta und Long von der Menge $c_1, \ldots, c_5$ (in der Greedy-Permutation).

wobei $L(i)$ die Granularitätsebene angibt (also den Index der Menge aus $L_0, L_1, \ldots$), in der sich c_i befindet. Für einen Punkt c_i ist $p(i)$ der nächste Nachbar in der Menge der Punkte, die in einer niedrigeren Granularitätsebene liegen. Die Knoten von G sind gegeben durch die Menge $\{c_1, \ldots, c_n\}$ und jeder Knoten c_i, ausser dem Wurzelknoten, hat (genau) eine ausgehende Kante, nämlich die Kante $(c_i, c_{p(i)})$. Abbildung 1 zeigt ein Beispiel dieses Graphen.

Der Algorithmus von Dasgupta und Long berechnet ein hierarchisches Clustering welches unsere Clustering-Bedingungen (i) und (ii) erfüllt, wie folgt. Angefangen mit dem Clustering C_1 werden in jedem Schritt immer genau zwei Cluster vereinigt. Hierbei werden die Punkte in der umgekehrten Reihenfolge betrachtet, in der sie in der Greedy-Permutation auftauchen. Sei c_j ein Element dieser Reihenfolge. Dann vereinigen wir genau diese beiden Cluster, die durch eine Elternkante zwischen c_j und $c_{p(j)}$ verbunden sind. Der Pseudocode des Algorithmus ist wie folgt.

`Dasgupta-Long-Algorithmus`$(S = \{x_1, \ldots, x_n\})$
1. $c_1, \ldots, c_n, r_1, \ldots, r_n \leftarrow$ `Gonzales-Algorithmus`(S)
2. Berechne Mengen L_0 und L_j mit $L_j \neq \emptyset$
3. Berechne Werte der Elternfunktion $p(i)$ für alle $1 < i \leq n$
4. Sei $C_1 = \{\{c_1\}, \ldots, \{c_n\}\}$
5. **for** i **in** $1 \ldots n$ **do**
6. Sei $j = n - i + 1$, sei $A \in C_i$ die Menge die c_j enthält
7. Sei $j' = p(j)$, sei $B \in C_i$ die Menge die $c_{j'}$ enthält
8. Sei C_{i+1} dieselbe Menge wie C_i, nur dass A und B vereinigt sind
9. Return $C_1, \ldots, C_n$

Eine andere hilfreiche Interpretation des Graphen G in Zusammenhang mit dem hierarchi-

schen Clustering ist die folgende. Das Clustering C_i ergibt sich durch die Zusammenhangskomponenten von G, wenn die Elternkanten entfernt werden, die von den ersten i Punkten in der Greedy-Permutation ausgehen. Statt die Cluster Schritt für Schritt zu vereinigen, könnten wir uns also auch den umgekehrten Prozess vorstellen, in dem Cluster Schritt für Schritt geteilt werden, indem wir Kanten in G entfernen. Um zu überprüfen, ob der Graph G tatsächlich ein Baum ist, stellen wir die folgenden Überlegungen an. Da alle Knoten ausser der Wurzelknoten einen Elternknoten haben, können wir von einem beliebigen Knoten aus, den Elternkanten folgen, bis wir irgendwann am Wurzelknoten ankommen. Also ist der Graph zusammenhängend. Gleichzeitig wissen wir, dass der Graph genau $n-1$ Kanten hat, wobei n die Anzahl der Knoten ist.

4 Analyse der Qualität der berechneten Lösung

Wir wollen nun analysieren, wie gut die Werte der Zielfunktion $\phi(C_1), \ldots, \phi(C_n)$ auf dem berechneten Clustering sind, wobei wir jedes Clustering C_i wieder mit dem optimalen Clustering mit der gleichen Anzahl von Clustern vergleichen.

Lemma 21.2. *Für alle $c_i \in S$ gilt $d(c_i, c_{p(i)}) \leq \frac{r_1}{2^{L(i)-1}}$ wobei $L(i)$ die Granularitätsebene von c_i angibt.*

Beweis. Wir zeigen zuerst eine andere Aussage: Für alle j ist der Abstand zwischen c_i und seinem nächsten Nachbarn in der Menge $L_0 \cup L_1 \cup \cdots \cup L_j$ höchstens $\frac{r_1}{2^j}$. Sei c_k der Punkt mit höchsten Index in L_j. Dann ist

$$L_0 \cup L_1 \cup \cdots \cup L_j = \{c_1, \ldots, c_k\} =: Z$$

Aus der Analyse des Gonzales-Algorithmus folgt, dass jeder Punkt in S Abstand höchstens r_k zu seinem nächsten Nachbarn in Z hat.

Aus der Definition der Granularitätsebenen folgt, dass $c_{k+1} \in L_j$ genau dann wenn $r_k \in (\frac{r_1}{2^j}, \frac{r_1}{2^{j-1}}]$. Da $c_{k+1} \notin L_j$, folgt daraus, dass $r_k \leq \frac{r_1}{2^j}$.

Der Satz folgt nun indem wir $j = L(i) - 1$ wählen. Insbesondere haben wir hergeleitet, dass gilt

$$d(c_i, p(i)) = \min_{1 \leq j' \leq L(i)-1} d(c_i, c_{j'}) \leq \frac{r_1}{2^{L(i)-1}}$$

□

Satz 21.3. *Sei $1 \leq k \leq n$ und sei $i = n - k + 1$. Sei C^* ein optimales k-Center-Clustering einer Menge S mit k Clustern. Für das vom Dasgupta-Long-Algorithmus berechnete Clustering C_i gilt*

$$\phi(C_i) \leq 8\phi(C^*)$$

Beweis. Wir zeigen $\phi(C_k) \leq 4r_k$. Der Satz folgt dann aus Satz 19.2 (Gonzales-Algorithmus). Sei $c_i \in S$ fest. Wir folgen den Elternkanten von c_i bis wir bei einem Punkt in der Menge $Z = \{c_1, \ldots, c_k\}$ ankommen. Die Menge C_i wird vom Algorithmus berechnet. Die Clusterzentren werden aber nicht vom Algorithmus festgelegt. Wir analysieren die Kosten für Clusterzentren Z und wir weisen c_i dem Clusterzentrum $c_{p(i)}$ zu.

Sei die Sequenz der Indizes auf diesem Pfade $i_0, i_2, \ldots, i_\ell$, mit $c_{i_\ell} \in Z$. Das heißt, wir definieren $i_0 = i$, $i_1 = p(i_0)$, $i_2 = p(i_1)$, etc. Die Sequenz $i_0, i_1, \ldots, i_\ell$ ist absteigend, da die Elternkanten nur auf Element in niedrigeren Granularitätsebenen zeigen.

Wir leiten eine obere Schranke für den Abstand zwischen c_i und c_{i_ℓ} her, indem wir die Dreiecksungleichung auf die Kanten der Pfade anwenden.

$$d(c_i, c_{i_\ell}) \leq d(c_{i_0}, c_{i_1}) + d(c_{i_1}, c_{i_2}) + \cdots + d(c_{i_{\ell-1}}, c_{i_\ell})$$

Laut Lemma 21.2 können wir diese Terme beschränken und erhalten

$$d(c_i, c_{i_\ell}) \leq \frac{r_1}{2^{L(i_0)-1}} + \frac{r_1}{2^{L(i_1)-1}} + \cdots \frac{r_1}{2^{L(i_{\ell-1})-1}}$$

Da sich der Index der Granularitätsebene mit jeder Elternkante um mindestens 1 verringert, ist $L(i_0) = L(i)$ und $L(i_j) \leq L(i) - j$. Es folgt

$$d(c_i, c_{i_\ell}) \leq \frac{r_1}{2^{L(i)-1}} + \frac{r_1}{2^{L(i)-2}} + \cdots + \frac{r_1}{2^{L(i)-\ell+1}} + \cdots + \frac{r_1}{2^{L(i_{\ell-1})-1}} \leq \sum_{j=L(i_{\ell-1})-1}^{L(i)-1} \frac{r_1}{2^j}$$

Wir ersetzen $j' = j - (L(i_{\ell-1}) - 1)$ in der Laufvariable der Summe und erhalten

$$d(c_i, c_{i_\ell}) \leq \sum_{j'=0}^{\infty} \frac{r_1}{2^{j'+(L(i_{\ell-1})-1)}} = \frac{r_1}{2^{(L(i_{\ell-1})-1)}} \sum_{j'=0}^{\infty} \frac{1}{2^{j'}} \leq \frac{r_1}{2^{(L(i_{\ell-1})-2)}} \leq 4 \cdot \frac{r_1}{2^{(L(i_{\ell-1}))}}$$

In welcher Granularitätsebene ist also $c_{i_{\ell-1}}$? Wir wissen, dass $c_{i_\ell} \in Z$ und also $i_\ell \in \{1, \ldots, k\}$, da wir den Elternkanten bis zu diesem Punkt gefolgt sind. Also ist $i_{\ell-1} \geq k+1$, da wir sonst schon bei $i_{\ell-1}$ in der Menge Z terminiert wären. Daraus schliessen wir, dass $L(i_{\ell-1}) \geq L(k+1)$, und daher

$$d(c_i, c_{i_\ell}) \leq 4 \cdot \frac{r_1}{2^{(L(k+1))}}$$

Weiter ist nach der Definition der Granularitätsebenen

$$c_i \in L_j \Leftrightarrow r_{i-1} \in \left(\frac{r_1}{2^j}, \frac{r_1}{2^{j-1}}\right]$$

Für $i = k+1$ und $j = L(k+1)$ erhalten wir also $r_k > \frac{r_1}{2^{L(k+1)}}$.

Da wir die Schranke für jedes $x_i \in S$ herleiten können, folgern wir, dass

$$\phi(C_k) \leq \max_{x_i \in S} \min_{c \in Z} d(c_i, c) \leq 4 \cdot \frac{r_1}{2^{(L(k+1))}} \leq 4 r_k$$

□

Referenzen

- Sanjoy Dasgupta, Philip M. Long, Performance guarantees for hierarchical clustering. Journal of Computer and System Sciences, Volume 70, Issue 4, 2005.