Clusteralgorithmen für Textdaten

Daniel Pullwitt
Institut für Informatik, Graduiertenkolleg Wissensrepräsentation

10. April 2003

Zusammenfassung

Im Umfeld des Information Retrieval, speziell im Bereich explorativer Datenanalyse, spielen Clusterverfahren eine wichtige Rolle. Der Artikel gibt einen Überblick über häufig eingesetzte Verfahren unter besonderer Gewichtung schwerpunktbasierter Verfahren mit nicht überlappender vollständiger Clustereinteilung. Zusätzlich wird ein eigenentwickelter Algorithmus, der informationstheoretische *k-means*, vorgestellt.

1 Einleitung

Im Folgenden wird ein Überblick über die begrifflichen Grundlagen der Clusterbildung gegeben (Abschnitt 3), gefolgt von Beschreibungen der Methoden, beschränkt auf für Textdaten untersuchte (Abschnitte 4-7. Anschließend folgt eine allgemeine vergleichende Abschätzung der Eigenschaften verschiedener Methoden (Abschnitt 8).

2 Formale Konventionen

$\mathbb{E}$ Der Raum der zu clusternden Elemente.

E Menge der zu clusternden Elemente.

m Anzahl der zu clusternden Elemente (Kardinalität von E, $m = \#E$).

e_i Ein einzelnes Element der Menge E.

C Menge der zu ermittelnden Cluster.

k Anzahl der zu Cluster Elemente (Kardinalität von C, $k = \#C$).

c_j Ein einzelner Cluster der Menge C.

$\bar{c}_j$ Die Modellrepräsentation des Clusters c_j (wenn vorhanden).

$d_E(e_r, e_s)$ Abstand zwischen zwei Elementen $e_r, e_s \in E$.

$d_C(c_u, c_v)$ Abstand zwischen zwei Clustern $c_u, c_v \in C$.

t Diskreter Zeitschritt für iterative Verfahren ($t = 0, \ldots, t_{\max}$).

$x^{(t)}$ Wert der Variablen x zum Iterationszeitpunkt t.

3 Grundlagen des Clusterns

Eine *Clustereinteilung* beschreibt eine Gruppierung der Elemente einer Menge, die im Sinne eines Begriffes der Zusammengehörigkeit natürlich ist, oder formal:

Def. 1 *Eine* Clustereinteilung C *einer Menge von Elementen* $E = (e_1, \ldots, e_m)$ *ist eine Zerlegung von* E *in* k *Teilmengen*

$$C = \{c_j \subseteq E | j = 1, \ldots, k\} \tag{1}$$

derart, dass die Elemente eines Clusters $e \in c_j$ *in einem noch zu definierenden Sinne zusammengehörig sind, während die Elemente verschiedener Cluster* $e_r \in c_u, e_s \in c_v, u \neq v$ *eine geringere Zusammengehörigkeit (im gleichen Sinne) aufweisen.*

Die *Zusammengehörigkeit* von Elementen wird im Allgemeinen durch ein Maß für die *Ähnlichkeit* $s_E(.,.)$ bzw. den *Abstand* $d_E(.,.)$ im Elementraum $\mathbb{E}$ definiert. Ein Ähnlichkeitsmaß $s_E(.,.)$ lässt sich,

sofern es von begrenztem Wertebereich ist, in ein Abstandsmaß überführen durch

$$d_E(e_r, e_s) = 1 - \frac{s_E(e_r, e_s)}{\max_{e_r, e_s \in E} s_E(e_r, e_s)} \quad (2)$$

Maximale Zusammengehörigkeit in einem Cluster wird also durch die Maximierung der Ähnlichkeit bzw. die Minimierung des Abstandes innerhalb des Clusters erreicht. In den folgenden Betrachtungen wird der Einfachheit halber von der Verwendung eines Abstandsmaßes ausgegangen.

Der Begriff der *Clustereinteilung* nach Definition 1 ist sehr allgemein gehalten, da Clustereinteilungen auf verschiedene Arten vorgenommen werden können. Es ist daher notwendig, ihn durch weitere Eigenschaften näher zu charakterisieren:

Def. 2 *Eine Clustereinteilung $C = \{c_1, \ldots, c_k\}$ einer Elementmenge E heißt* vollständig, *wenn jedes Element $e \in E$ einem Cluster zugeordnet ist, d. h. es gilt $\bigcup_{c_j=1}^{k} c_j = E$. Anderenfalls heißt die Einteilung* partiell.

Def. 3 *Eine Clustereinteilung $C = \{c_1, \ldots, c_k\}$ heißt* nicht überlappend, *wenn kein Element mehr als einem Cluster zugeordnet ist, d. h. es gilt $\forall c_u, c_v \in C : c_u \cap c_v = \emptyset$. Anderenfalls heißt die Einteilung* überlappend.

Def. 4 *Ein* Clusterverfahren *ist eine Methode, die ausgehend von der Elementmenge $E = (e_1, \ldots, e_m)$ unter Verwendung eines geeigneten Maßes für die Ähnlichkeit bzw. den Abstand von Elementen eine* Clustereinteilung *C bestimmt.*

Wird ein metrischer Raum $\mathbb{E}$ mit der Metrik $d_E(.,.)$ verwendet, so können Cluster c durch prototypische Elemente $\bar{c}$ in $\mathbb{E}$ repräsentiert werden. Eine prototypische Repräsentation ist durch *Schwerpunkte* möglich.

Def. 5 *Ein* Schwerpunkt *$\bar{c}$ bzw.* Zentroid *(*centroid*) eines Clusters c ist gegeben als* prototypisches *Element des Elementraumes $\mathbb{E}$, das sich als Mittelung über alle Elemente des Clusters ergibt*

$$\bar{c} = \frac{1}{\#c} \sum_{e \in c} e \quad (3)$$

Der Zentroid *bildet damit ein* Modell *des Clusters.*

Die Schwerpunkte beschreiben eine Partitionierung als vollständige Zerlegung des Elementraumes $\mathbb{E}$ in Klassen:

Def. 6 *Eine* Partitionierung *P eines Elementraumes $\mathbb{E}$ ist dessen disjunkte Zerlegung in Klassen $P = \{p_1, \ldots, p_l\}$, d. h. es gelten $p_u \cap p_v = \emptyset \Leftrightarrow u \neq v$ und $\bigcup_{j=1}^{l} p_j = \mathbb{E}$.*

Der Begriff der *Partitionierung* ist damit eine auf den Elementraum übertragene Analogie zur nicht überlappenden *Clustereinteilung.*

4 Hierarchisch agglomeratives Clustern

Die Methode des *hierarchisch agglomerativen Clusterns* (*HAC*) [13, 14] wird in der Literatur häufig als grundlegender Algorithmus eingeführt. Ausgehend von der Menge aller einelementigen Cluster werden diese entsprechend des verwendeten Kriteriums der Zusammengehörigkeit schrittweise paarweise vereinigt, bis die angestrebte Anzahl von Clustern erreicht ist.

Der Algorithmus des *HAC* ist wie folgt definiert:

1. Definiere die Menge der einelementigen Startcluster

   $$C^{(0)} = \{c_1^{(0)} = \{e_1\}, \ldots, c_m^{(0)} = \{e_m\}\}$$

   und setze den Iterationszähler t auf 0.

2. Im Iterationsschritt t finde die beiden Cluster $c_i^{(t)}, c_j^{(t)}$ mit minimalem Abstand

   $$d_C(c_i^{(t)}, c_j^{(t)}) = \min_{c_r^{(t)}, c_s^{(t)} \in C^{(t)}, c_r^{(t)} \neq c_s^{(t)}} d_C(c_r^{(t)}, c_s^{(t)}) \quad (4)$$

3. Fasse die beiden Cluster $c_i^{(t)}, c_j^{(t)}$ zu einem neuen zusammen und bilde die um ein Element kleinere Clustermenge

   $$C^{(t+1)} = (C^{(t)} \setminus \{c_i^{(t)}, c_j^{(t)}\}) \cup \{c_i^{(t)} \cup c_j^{(t)}\} \quad (5)$$

   Erhöhe den Iterationszähler t um 1.

4. Solange die angestrebte Anzahl Zielcluster noch nicht erreicht wurde, d. h. $\#C^{(t)} > k$, gehe zu Schritt 2.

Der Algorithmus erzeugt in jedem Schritt als Zwischenergebnis die Ergebnisse für $m, m-1, \ldots, 1$ Cluster. Liegt die Reihenfolge der Vereinigungen vor, ist es nach komplettem Durchlauf, d. h. bis nur noch ein Cluster $C^{(m-1)} = \{E\}$ mit der gesamten Elementmenge bleibt, mit vergleichsweise geringem Aufwand möglich eine Clusterung beliebiger Auflösung zu ermitteln. Es ist für k Cluster lediglich notwendig, $m - k$ Vereinigungsoperationen (Schritt 3) auszuführen, die kostenaufwendige Suche nach der optimalen Vereinigung (Schritt 2) entfällt.

Es handelt sich bei *HAC* um einen *greedy* Algorithmus, d. h. es wird in jedem Schritt die *lokal optimale* Vereinigung ausgeführt unter Vernachlässigung der Auswirkung auf weitere Schritte.

4.1 Varianten der Clusterbildung

Ausgehend von dem Abstandsmaß $d_E(.,.)$ ergeben sich verschiedene Methoden die Vereinigung von Clustern in Schritt 3 sowie das Maß des *Clusterabstands* $d_C(.,.)$ zu definieren.

4.1.1 Single Link Clustering

Ein Cluster c_j ergibt sich aus der Vereinigung aller Elemente $e_{i(j)} \in E$ die diesem zugeordnet sind. Der Abstand $d_C(c_u, c_v)$ zwischen zwei Clustern c_u, c_v wird definiert als der minimale Abstand über allen Elementpaaren aus den Clustern

$$d_C(c_u, c_v) = \min_{e_r \in c_u, e_s \in c_v} d_E(e_r, e_s) \tag{6}$$

Die Verwendung des minimalen Abstandes führt zu inhomogenen Clustern. So kann über ein einzelnes Element eine Kette von weiteren Elementen zugeordnet werden, die zu den restlichen Clusterelementen einen hohen Abstand aufweisen. Die aus dem *Single Link Clustering* resultierenden Cluster weisen damit eine starke Streuung auf und langgezogene Strukturen auf.

4.1.2 Complete Link Clustering

Die Clusterdefinition als Vereinigung aller zugeordneten Elemente entspricht der des *Single Link Clustering*. Der Abstand $d_C(c_u, c_v)$ zwischen zwei Clustern c_u, c_v wird definiert als der maximale Abstand über allen Elementpaaren aus den Clustern

$$d_C(c_u, c_v) = \max_{e_r \in c_u, e_s \in c_v} d_E(e_r, e_s) \tag{7}$$

Durch die Verwendung des maximalen Elementabstandes ergeben sich kleine, stark abgegrenzte sphärische Cluster, d. h. alle Elemente weisen einen geringen paarweisen Abstand auf.

4.1.3 Group Averaging Clustering

Ein Cluster wird analog zum *Single Link Clustering* definiert als Vereinigung aller zugeordneten Elemente. Der Abstand $d_C(c_u, c_v)$ zwischen zwei Clustern c_u, c_v ergibt sich als der durchschnittliche Abstand über alle Elementpaare der Cluster

$$d_C(c_u, c_v) = \frac{1}{\#c_u \#c_v} \sum_{e_r \in c_u, e_s \in c_v} d_E(e_r, e_s) \tag{8}$$

Die Clustergebnisse des *Group Averaging Clustering* liegen in ihrer Struktur zwischen den langgezogenen Ketten des *Single Link Clustering* und den stark abgegrenzten Clustern des *Complete Link Clustering*.

4.1.4 Schwerpunktbasierte Cluster

Wird die Elementmenge E auf einer *quantitativen Skala* repräsentiert und ist das verwendete Abstandsmaß $d_E(.,.)$ *metrisch*, so können die Cluster c_j durch *Schwerpunkte* (Gleichung 3) über die Clusterelemente repräsentiert werden. Die Schwerpunkte stellen *Modelle* ihrer Cluster im Elementraum $\mathbb{E}$ dar.

Der Abstand zwischen den Clustern c_u, c_v kann damit als Abstand der zugehörigen Modelle im Elementraum ausgedrückt werden

$$d_C(c_u, c_v) = d_E(\bar{c}_u, \bar{c}_v) \tag{9}$$

Der Vorteil schwerpunktbasierter Verfahren liegt in der geringeren Komplexität der Abstandsberechnung ($\mathcal{O}(1)$ gegen $\mathcal{O}(\#c_u \#c_v)$ bei *Group Averaging Clustering*). Im Allgemeinen verringert sich auch die Komplexität bezüglich des Speicherplatzes, da ein einzelnes Clustermodell $\bar{c}_j$ die Menge der Clusterelemente $\{e \in c_j\}$ ersetzt.

4.2 Die Information Bottleneck Methode

Eine spezielle Variante des *HAC* Algorithmus für Wahrscheinlichkeitsverteilungen bildet der von der *Information Bottleneck* Methode [11] abgeleitete

Algorithmus [10]. Die Idee des Ansatzes besteht darin, Elemente und deren Merkmale $\omega \in \Omega$ als 2 Zufallsvariablen aufzufassen, deren gemeinsame empirische Verteilung durch die Merkmalsausprägungen in der vorliegenden Elementmenge E beschrieben wird.

Eine der Variablen, die Elementmenge E mit den Ausprägungen $e \in E$, wird komprimiert, d. h. zur Clustervariablen C mit einer geringeren Menge an Ausprägungen $c \in C$ zusammengefasst, unter maximaler Erhaltung der *gemeinsamen Information*

$$I(E;\Omega) = \sum_{\omega\in\Omega} I(\omega) = \sum_{e\in E,\omega\in\Omega} p(e,\omega)\log\frac{p(e,\omega)}{p(e)p(\omega)} \tag{10}$$

zwischen Elementen und Merkmalen in der Clusterstruktur. Der Grad der zu maximierenden Informationserhaltung lässt sich ausdrücken durch den Quotienten $\frac{I(C;\Omega)}{I(E;\Omega)}$.

Für das hierarchische Clustern werden daher ein spezielles Abstandsmaß $d_{\mathrm{IBN}}(.,.)$ und eine angepasste Vereinigungsoperation der Cluster (Gleichung 5) konstruiert. Für die mit Wahrscheinlichkeitsvektoren repräsentierten Ausgangscluster $c^{(0)} \in C^{(0)}$ wird die Auftrittswahrscheinlichkeit $p(c^{(0)}) = \frac{1}{\#C^{(0)}}$ definiert. Die Vereinigung zweier Cluster $c_i^{(t)}, c_j^{(t)}$ ergibt sich als mit den Wahrscheinlichkeiten gewichtete Mittelung der Clusterverteilungen

$$\tilde{c} = \frac{p(c_i^{(t)})}{p(\tilde{c})}c_i^{(t)} + \frac{p(c_j^{(t)})}{p(\tilde{c})}c_j^{(t)} \tag{11}$$

$$p(\tilde{c}) = p(c_i^{(t)}) + p(c_i^{(t)}) \tag{12}$$

Im Fall nichtüberlappender Cluster (Definition 3) ergibt sich als optimale Lösung für das Abstandsmaß

$$d_{\mathrm{IBN}}(c_i^{(t)}, c_j^{(t)}) = \left(p(c_i^{(t)}) + p(c_j^{(t)})\right)\cdot \left(\frac{p(c_i^{(t)})}{p(\tilde{c})}D_{\mathrm{KL}}(c_i^{(t)},\tilde{c}) + \frac{p(c_j^{(t)})}{p(\tilde{c})}D_{\mathrm{KL}}(c_j^{(t)},\tilde{c})\right) \tag{13}$$

wobei sich $\tilde{c}$ nach Gleichung (11) ergibt und

$$D_{\mathrm{KL}}(x_i,x_j) = \sum_{\omega\in\Omega} p(\omega|x_i)\log\frac{p(\omega|x_i)}{p(\omega|x_j)} \tag{14}$$

die *relativen Entropie* bzw. *Kullback-Leibler* Divergenz bezeichnen. Gleichung (14) gibt an, wie viel Information verloren geht, drückt man x_i durch x_j aus. Der Abstand (13) beschreibt damit den Verlust an Information durch Vereinigung der beiden Cluster $c_i^{(t)}, c_j^{(t)}$, gewichtet mit den jeweiligen Clusterwahrscheinlichkeiten. Es ist anzumerken, dass $d_{\mathrm{IBN}}(.,.)$ keine *Metrik* ist, da die Dreiecksungleichung nicht erfüllt wird.

4.3 Komplexitätsbetrachtung

Die Komplexität des Algorithmus wird im Wesentlichen von der verwendeten Suchmethode in Schritt 2 bestimmt. Die folgenden Betrachtungen (vergleiche [16]) vernachlässigen die zusätzlichen Kosten für die Ermittlung der paarweisen Abstände. Im Falle komplexer Daten, beispielsweise der in der betrachteten Anwendung verwendeten extrem hochdimensionalen Vektoren, sind diese Kosten nicht trivial.

- *Verwendung der Datensätze* (*stored data*) Die Abstände zwischen Datensätzen werden bei Bedarf neu ermittelt. Daraus ergibt sich ein Platzbedarf von $\mathcal{O}(m)$ (Datensätze) sowie eine Laufzeit von $\mathcal{O}(m^3)$, wobei die Komplexität der Abstandberechnung stark ins Gewicht fällt.

- *Abstandsmatrix* (*stored matrix*) Sämtliche paarweise Abstände werden zu Beginn ermittelt und gespeichert. Der Platzbedarf ergibt sich damit als $\mathcal{O}(m^2)$ (Matrix), der Zeitaufwand beträgt $\mathcal{O}(m^2) + \mathcal{O}(m^3)$ (Matrixerstellung + lineare Suche). Im Falle des *Single Link Clustering* wird für jedes Element nur der jeweils nächste Nachbar benötigt [15], d. h. der Platzbedarf beträgt nur $\mathcal{O}(m)$ und die Zeitkomplexität $\mathcal{O}(m^2)$.

- *Sortierte Abstandsmatrix* (*sorted matrix*) Analog zur einfachen Abstandsmatrix, allerdings werden die Einträge sortiert, s. d. die Suche der nächsten zu vereinigenden Cluster die Komplexität $\mathcal{O}(1)$ hat. Der Platzbedarf beträgt wie bei der unsortierten Abstandsmatrix $\mathcal{O}(m^2)$, die Zeitkomplexität $\mathcal{O}(m^2)$ (Erstellen der Matrix), $\mathcal{O}(m^2 \log m^2)$ (Sortieren der Matrix), $\mathcal{O}(m^2)$ (Erstellen der Clusterhierarchie).

Diese Methode kann nicht mit *Group Averaging Clustering* oder *Schwerpunkten* verwendet werden, da jede Vereinigung neue Clusterabstände erzeugt, die in die sortierte Liste eingefügt werden müssen und eine Neusortierung erforderlich machen.

Die Verwendung einer Matrix ist vorteilhaft bei komplexen Elementen, d. h. bei relativ hohen Kosten für die Abstandsbestimmung. Bei Platzbeschränkungen auf realen Systemen bietet sich die sortierte Matrix an, die aufgrund des sequentiellen Zugriffes eine externe Speicherung großer Teile der Daten erlaubt. Zusätzlich können hybride Verfahren verwendet werden, die nur Teile der Abstandsmatrix speichern und fehlende Daten bei Bedarf neu berechnen.

5 Single Pass Clustering

Der Algorithmus des *Single Pass Clustering* (*SPC*) [9] betrachtet jedes Element nur ein einziges Mal um seine Zuordnung zu bestimmen. Es wird entschieden, ob das Element zu einem existierenden Cluster passt oder ob ein neuer Cluster definiert werden muß.

Der *SPC* Algorithmus ist wie folgt definiert:

1. Definiere den Ausgangscluster $c_1^{(1)} = \{e_1\}$ als den, der das erste Element enthält.

2. Für das Element e_t bestimme die Ähnlichkeit bzw. den Abstand zu allen existierenden Clustern $c_j^{(t)} \in C^{(t)}$.

3. Liegt die ermittelte Ähnlichkeit (der Abstand) zum besten Cluster $c_j^{(t)}$ über (unter) einer Schwelle α, so wird e_t dem Cluster $c_j^{(t)}$ zugefügt
$$c_j^{(t+1)} = c_j^{(t)} \cup e_t \tag{15}$$
Anderenfalls wird ein neuer Cluster definiert $C^{(t+1)} = C^{(t)} \cup \{e_t\}$

4. Solange noch nicht betrachtete Elemente der Elementmenge E existieren, erhöhe t und gehe zu Schritt 2.

Das Verfahren unterstützt praktisch alle Varianten der Clusterbildung die für das *hierarchisch agglomerative Clustern* eingeführt wurden (Abschnitt 4.1) und weist darüber hinaus die sehr günstige Zeitkomplexität von $\mathcal{O}(m \log m)$ auf.

Nachteilig ist die Empfindlichkeit der Clusterstruktur von der Reihenfolge der Präsentation der Elemente und von Ausreißern, welche die Bildung unnötig vieler kleiner Cluster bewirken. Ein weiterer Effekt ist, dass Cluster, die zu Beginn definiert werden, relativ viele Elemente erhalten, während den spät definierten anfangs (falsch) zugeordnete fehlen. Es besteht keine Möglichkeit, die Anzahl der zu erstellenden Cluster a priori vorzugeben.

6 Iterative Verbesserung von Clustern

Die Klasse der Verfahren zur iterativen Verbesserung von Clustern basiert auf dem Ansatz, ausgehend von einer initialen Einteilung der Elemente in Cluster diese schrittweise zu verbessern. Der zugehörige Algorithmus hat im wesentlichen die Gestalt:

1. Bestimme die initale Clustereinteilung $C^{(0)}$ und setzt den Iterationszeitpunkt auf $t = 0$.

2. Ordne jedes Element $e \in E$ einem der Cluster aus $C^{(t)}$ zu.

3. Optimiere die Cluster anhand der ihnen zugeordneten Elemente und einer *Kostenfunktion* in geeigneter Weise und bilde die Menge neuer Cluster $C^{(t+1)}$. Erhöhe den Iterationszähler $t := t + 1$.

4. Überprüfe ein vorgegebenes Zielkriterium $\tau(C^{(t)})$. Ist dieses nicht erfüllt, so gehe zu Schritt 2, anderenfalls beende den Prozess.

Verfahren iterativer Verbesserung können auch unter dem Aspekt des *Aufteilen und Mischens* betrachtet werden. Inhomogene Cluster hinsichtlich einer geeigneten Bewertungsfunktion werden aufgeteilt, beispielsweise mittels eines (von der Komplexität her) einfachen Clusterverfahrens. Ähnliche Cluster, deren Vereinigung den geringsten Qualitätsverlust hinsichtlich der Bewertungsfunktion ergibt, werden zu einem einzigen zusammengefasst. Wird eine feste Anzahl Cluster angestrebt, sind beide Grundoperationen kombiniert anzuwenden.

Der Prozess wird im Wesentlichen bestimmt von der Methode zur Clusteradaption in Schritt 3 und der Menge der initialen Cluster $C^{(0)}$. Ersterer Punkt ist spezifisch für das jeweilige Verfahren während zur Bildung der initialen Cluster verschiedene Optionen bestehen:

- *Zufällige Initialisierung*: Die Cluster werden in einer nichtdeterministischen Weise gebildet. In Abhängigkeit von der Clustermodellierung existieren verschiedene Untertypen:
  - *Zufällige Auswahl*: Aus den Eingabeobjekten werden zufällig Elemente gezogen und zu Clustern gruppiert.
  - *Zufällige Bildung*: Werden *Schwerpunkte* über *quantitativen Daten* zur Repräsentation der Cluster benutzt, so können die initialen Cluster als zufällig bestimmte Elemente auf dieser Skala festgelegt (ausgewürfelt) werden.
  - *Zufällige Bildung um globalen Schwerpunkt*: Bei *quantitativen Daten* und Repräsentation mit *Schwerpunkten* kann zunächst der globale Schwerpunkt der Daten ermittelt werden. Ausgehend von diesem Punkt können die einzelnen Cluster als dessen zufällige Variationen gesetzt werden.
- *Vorclustern*: Die initialen Cluster werden ermittelt durch Einsatz einer beliebigen Clustermethode. Dies ist vor allem sinnvoll mit Verfahren geringer Komplexität, wie zum Beispiel dem *Single Pass Clustering* (Abschnitt 5). Ebenfalls geeignet sind das *Clustern einer Stichprobe*, z. B. mittels des *Buckshot*-Algorithmus (Abschnitt 7.1), oder das *Clustern in Teilmengen*, das sich im *Fractionating*-Algorithmus (Abschnitt 7.2) findet. Untersuchungen von [4] zeigen eine durchgehende Verbesserung von mittels *HAC* (Abschnitt 4) generierten Clustern durch iterative Algorithmen.

Für alle nichtdeterministischen Initialisierungen gilt, dass diese zwar einfach realisierbar sind, aber praktisch keines eine generell günstige Startsituation ermittelt, da die exakte (unbekannte) Verteilung der Daten vernachlässigt wird. Bei zufälliger Auswahl hängt die Qualität von der Repräsentativität der ausgewählten Stichprobe ab. Die zufällige Bildung um den globalen Schwerpunkt sichert zwar eine ungefähre Lage inmitten der Eingabedaten, ist aber nicht sensitiv gegen extreme Unterschiede in der lokalen Dichte.

Es besteht auch immer die Möglichkeit verschiedenen Initialisierungen zu verwenden und das beste Ergebnis zu selektieren. Die Praktikabilität hängt aber wesentlich von der Komplexität der Verfahren ab.

6.1 *k-means* Clustering

Ein Verfahren zur iterativen Verbesserung von schwerpunktbasierten Clustern durch Optimierung einer Kostenfunktion ist der k-means Algorithmus [8]. Für eine feste Anzahl k von Clustern $\{c_j | j = 1, \ldots, k\}$ wird eine Kostenfunktion $f(.)$ minimiert. Die zu clusternde Menge E muss in einem metrischen Raum $\mathbb{E}$ liegen und die Elementrelation als Abstandsmaß definiert sein, die Elemente $e \in E$ werden als Vektoren $\vec{e}$ beschrieben.

Die Cluster c_j werden mittels Schwerpunktvektoren $\vec{c}_j$ repräsentiert, die zu minimierende Kostenfunktion definiert als

$$f(c_j) = \sum_{e_i \in c_j} d_E(\vec{c}_j, \vec{e}_i)^2 \tag{16}$$

Der Algorithmus ist wie folgt definiert:

1. Initialisiere die Clusterschwerpunkte $\vec{c}_j^{(0)}$ und setze den Iterationszeitpunkt $t = 0$.
2. Ermittle die Clusterzuordnung für alle Elemente $e \in E$
$$\gamma^{(t)}(e, C^{(t)}) = \arg\min_j \{d_E(\vec{c}_j^{(t)}, \vec{e}) | c_j \in C^{(t)}\} \tag{17}$$
Existieren mehrere Cluster, für die der Abstand den Minimalwert erreicht, ordne das Element willkürlich einem der Cluster zu.
3. Bestimme den neuen Clusterschwerpunkt aus den zugeordneten Elementen
$$\vec{c}_j^{(t+1)} = \frac{1}{\#c_j^{(t)}} \sum_{e \in c_j^{(t)}} \vec{e} \tag{18}$$

4. Erhöhe den Iterationszähler $t := t + 1$. Ist die Abbruchbedingung erfüllt, beende den Algorithmus, sonst gehe zu Schritt 2.

Die Zeitkomplexität des Algorithmus liegt bei $\mathcal{O}(mt\,(k+1))$ (Clusterzuordnung und Aktualisierung), der Platzbedarf bei $\mathcal{O}(m+2k)$ (Elemente, Schwerpunkte und neue Schwerpunkte akkumuliert während jeder Iteration).

Der k-means Algorithmus realisiert einen Gradientenabstieg entlang des verwendeten Abstandsmaßes, der in das nächstgelegene (lokale) Minimum strebt. Das Ergebnis ist damit von der Startkonfiguration $\{\vec{c}_1^{\,(0)}, \ldots, \vec{c}_k^{\,(0)}\}$ abhängig, für deren Wahl die Erwägungen zu Beginn dieses Abschnittes gelten.

Die Abbruchbedingung in Schritt 4 ist nicht näher spezifiziert, möglich sind aber eine feste Anzahl Iterationsschritte (beschränkte Zeitkomplexität) oder eine Funktion die von der Qualität der Cluster abhängt.

6.1.1 Sphärischer *k-means*

Wird eine Normierung der Merkmalsvektoren der Elemente auf Einheitslänge verwendet, d. h. werden die Elemente auf die Oberfläche einer n-dimensionalen Einheitskugel projiziert, so ist eine Modifikation des Algorithmus notwendig, um sicherzustellen, dass auch die Clustermodelle auf der Kugeloberfläche liegen.

Die entsprechende Abwandlung, der sphärische *sphärische* k-means [3], basiert auf der Verwendung des *Kosinusabstands*

$$d(d_r, d_s) = 1 - \cos(\measuredangle(\vec{d}_r, \vec{d}_s)) = 1 - \frac{< \vec{d}_r, \vec{d}_s >}{\|\vec{d}_r\| \|\vec{d}_s\|} \tag{19}$$

in Schritt 2 und Kombination mit Renormierung in Schritt 3. Dazu werden zunächst alle zugeordneten Elementvektoren aufsummiert

$$\vec{s}_j^{\,(t)} = \sum_{e \in c_j^{(t)}} \vec{e} \tag{20}$$

und anschließend auf die Oberfläche der Einheitskugel projiziert

$$\vec{c}_j^{\,(t+1)} = \frac{\vec{s}_j^{\,(t)}}{\|\vec{s}_j^{\,(t)}\|} \tag{21}$$

Eine für Komplexitätsbetrachtungen günstige Eigenschaft des *sphärischen* k-means liegt in der Möglichkeit, einen der Teil der Abstandsberechnungen zwischen Element- und Clustervektoren durch Abschätzungen zu ersetzen. Dazu wird der beobachtete Effekt ausgenutzt, dass die Clustermodelle nach den ersten Iterationen, die der Groborientierung dienen, nur noch relativ kleine Veränderungen erfahren. Die maximale Änderung des Abstandes des Elementes e zum Cluster c in einem Iterationsschritt ist begrenzt durch

$$\begin{aligned} | < \vec{e}, \vec{c}^{\,(t)} > - < \vec{e}, \vec{c}^{\,(t+1)} > | \leq \\ \|\vec{e}\| \|\vec{c}^{\,(t)} - \vec{c}^{\,(t+1)}\| = \|\vec{c}^{\,(t)} - \vec{c}^{\,(t+1)}\| \end{aligned} \tag{22}$$

Der Algorithmus erfordert zusätzlich die Führung einer $m \times k$-Matrix der geschätzten minimalen Abstände aller Element-Cluster Paare. In Schritt 2 des k-means Algorithmus wird zunächst für jedes Paar der geschätzte Abstand der Matrix nach Gleichung (22) aktualisiert. Für den Cluster, dem das Element bisher zugeordnet war, wird der Abstand exakt berechnet, unterschreitet einer der geschätzten minimalen Abstände diesen exakten Wert, so dass eine potentielle Neuzuordnung notwendig ist, so wird auch er neu berechnet.

Experimentelle Daten nach [3], zeigen mit dieser Methode eine signifikante Einsparung an Abstandsberechnungen, insbesondere bei vielen Clustern. So zeigt sich in einzelnen Iterationsschritten eine Ersparnis von bis zu 2/3 der Abstandsberechnungen, die den Mehraufwand für die Führung der zusätzlichen Matrix mehr als kompensiert.

6.1.2 Informationstheoretischer *k-means*

Aus dem guten experimentellen Abschneiden der *Information Bottleneck* Clustermethode (Abschnitt 4.2) entstanden eigene Überlegungen, die Effektivität des Ansatzes mit dem günstigen Komplexitätsverhalten des k-means Algorithmus zu kombinieren. Die entwickelte Idee ist, den Verlust an Information zu minimieren, der mit der Repräsentation eines Elementes $e \in E$ durch den Cluster $c \in C$ verbunden ist.

Für die informationstheoretische Modellierung wird davon ausgegangen, dass sowohl Elemente als auch Clustermodelle durch n-dimensionale Vektoren von Wahrscheinlichkeiten der Merkmale $\omega \in \Omega$ gegeben sind, d. h. jeder Wert in $[0, 1]$ liegt und die Summe über jeden Vektor 1 ergibt.

In Schritt 2 des k-means Algorithmus, der Clusterzuordnung, wird für jedes Clustermodell $\vec{c}$ ein hypothetisches Modell $\vec{c}'$ ermittelt, das sich aus der Vereinigung mit dem aktuell präsentierten Element ergibt

$$\vec{c}' = \frac{p(c)\vec{c} + p(e)\vec{e}}{p(c) + p(e)}, \tag{23}$$

wobei die Elemente im Allgemeinen gleich gewichtet sind, d. h. es gilt $p(e) = \frac{1}{\#E}$ und die Clusterwahrscheinlichkeit als Summe der Wahrscheinlichkeiten der zugeordneten Elemente sich demzufolge definiert als

$$p(c^{(t)}) = \sum_{e \in c^{(t)}} p(e) = \frac{\#c^{(t)}}{\#E} \tag{24}$$

Der Abstand eines Elementes zu einem Cluster wird definiert als der mit der Clusterzuordnung verbundene Informationsverlust für das zugeordnete Element in Form der *relativen Entropie* (Gleichung 14) zwischen Element und hypothetischem Modell (*weighted kullback leibler*)

$$d_{\mathrm{WKL}}(\vec{e}, \vec{c}) = D_{\mathrm{KL}}(\vec{e}, \vec{c}') \tag{25}$$

Die Aktualisierung eines Clusters in Schritt 3 des Algorithmus erfolgt durch bestimmen der mittleren Wahrscheinlichkeit für jedes Merkmal der zugeordneten Elemente, d. h. in gewohnter Weise durch Schwerpunktbildung über den Elementen. Die Clusterwahrscheinlichkeit nach der Vereinigung ergibt sich als Summe der Elementwahrscheinlichkeiten entsprechend Gleichung (24).

Die Gewichtung mittels Wahrscheinlichkeiten in Gleichung (23) gegenüber der direkten Verwendung des Modellvektors, wie z. B. in [12], hat 2 Effekte: Zum Einen vermeidet sie in der relativen Entropie undefinierte (unendlich große) Werte, da Terme der Form $x \log \frac{x}{0}, x \neq 0$ verhindert werden, ohne die Notwendigkeit einer Glättung (siehe dazu z. B. [1]). Zum Anderen bewirkt sie eine Trägheit der Cluster proportional zur Anzahl ihnen zugeordneter Elemente. Damit neigen kleine Cluster aufgrund generell geringerer Abstände leichter zur Anpassung im Vergleich zu großen Clustern, die bereits eine relevante Datenkonzentration beschreiben. Kleine Cluster sind damit in der Lage, durch große Sprünge im Elementraum $\mathbb{E}$, diesen schnell nach dichtbesetzten Regionen abzusuchen, während große Cluster praktisch nur lokale Feinadaption vornehmen.

Der Abstand $d_{\mathrm{WKL}}(.,.)$ ist keine Metrik, so ist bereits die Symmetrieeigenschaft nicht gegeben. Zusätzlich ist der Algorithmus nicht formal exakt, da der Aktualisierungsschritt weiterhin nach der Euklidischen Metrik erfolgt, während der Abstand nicht-euklidisch und nicht-metrisch ist. Die mittels des Verfahrens experimentell gewonnenen Ergebnisse sind jedoch, auch bei wenigen Iterationsschritten, qualitativ besser als die mit anderen Methoden erzielten. Der *informationstheoretische* k-means erfüllt damit die angestrebten Eigenschaften, d. h. qualitativ hochwertige Ergebnisse bei günstigem Komplexitätsverhalten.

6.2 Selbst organisierende Karten

Die *selbst organisierende Karte* (*Self-Organizing Map*) (*SOM*) [7] ist ein Sonderfall unter den hier vorgestellten Clusteralgorithmen. Die Karte beschreibt eine nachbarschaftserhaltende nichtlineare Abbildung hochdimensionaler Daten in einen niedrigdimensionalen *diskreten Raum* A und ist damit eine Methode der *nichtlinearen Hauptkomponentenanalyse*. Der Algorithmus eignet sich speziell zur *visuellen Datenanalyse* [5], d. h. dem Entdecken und Identifizieren von Strukturen in unbekannten Daten, da Cluster im Elementraum $\mathbb{E}$ sich im Ausgaberaum A wiederspiegeln und dort, bei geeigneter Visualisierung, leicht identifiziert werden können.

Die Grundvariante des Algorithmus [7] ist definiert für den *Euklidischen Abstand*

$$d(d_r, d_s) = \|\vec{d}_r - \vec{d}_s\| = \sqrt{\sum_{j=1}^{n} \left(\omega_j^{\mathrm{p}}(d_r) - \omega_j^{\mathrm{p}}(d_s)\right)^2} \tag{26}$$

bzw. den *Kosinusabstand* (Gleichung 19) über einem reellwertigen Vektorraum $\mathbb{E} \subseteq \mathbb{R}^n$, d. h. die Elemente $e \in E$ sind definiert als Vektoren $\vec{e} \in \mathbb{E}$. Im Fall auf Einheitslänge normierter Elementvektoren sind die Ergebnisse mit beiden Metriken äquivalent [7]. Der Algorithmus bestimmt nicht direkt die Cluster, sondern lediglich eine *Partitionierung* des Raumes $\mathbb{E}$. Der diskrete Ausgaberaum A, d. h. die Karte, hat üblicherweise die Form eines regelmäßigen Gitters, in dem jedem Gitterpunkt a eine Partition p, repräsentiert von einem Schwerpunktvektor $\vec{p} \in \mathbb{E}$, zugeordnet wird. Die Anordnung der Partitionen im Raum A wird durch eine Abstandsfunk-

tion $d_A(.,.)$ beschrieben, s. d.

$$d_A(p_u, p_v) = d_A(a_u, a_v) \tag{27}$$

Für Clusteranalysen mit Visualisierung wählt man üblicherweise den Ausgaberaum A als zweidimensionales Gitter mit rechteckiger oder hexagonaler Anordnung der Gitterpunkte.

Der iterative Lernalgorithmus der *selbst organisierende Karte* ist so definiert, dass zum Einen die Partitionen die dichten Regionen des Elementraumes $\mathbb{E}$ beschreiben, zum Anderen in $\mathbb{E}$ benachbarte Partitionen (d. h. aneinander grenzende Partitionen) auch auf der Karte A möglichst benachbart sind. Die Karte beschreibt damit idealerweise eine nachbarschaftserhaltende Abbildung $\mathbb{E} \to A$, soweit dies die *intrinsische Dimension* der Daten in $\mathbb{E}$, d. h. die minimal benötigte Dimension des reellwertigen Vektorraumes, in den die Elementmenge $E \subset \mathbb{E}$ Abstand erhaltend eingebettet werden kann, und die Dimension des Ausgaberaumes A zulassen.

Der Algorithmus ist wie folgt definiert:

1. Initialisiere die Schwerpunktvektoren der Partitionen $\vec{p}_j^{(0)}, \vec{p} = 1, \ldots, l$ und setze den Iterationszeitpunkt $t = 0$.

2. Bestimme zum Zeitpunkt t für das präsentierte Element $e^{(t)}$ die enthaltende Partition (den Gewinner)

$$w^{(t)} = \arg \min_{p_j^{(t)} \in P^{(t)}} d_E(\vec{e}^{(t)}, \vec{p}_j^{(t)}) \tag{28}$$

3. Aktualisiere alle Partitionen mit dem präsentierten Element

$$\vec{p}_j^{(t+1)} = \vec{p}_j^{(t)} + \epsilon^{(t)} h(d_A(w^{(t)}, p_j), \sigma^{(t)}) \cdot \left(\vec{e}^{(t)} - \vec{p}_j^{(t)}\right), \tag{29}$$

bzw. für den *Kosinusabstand*

$$\vec{p}_j^{(t+1)} = \vec{p}_j^{(t)} + \epsilon^{(t)} h(d_A(w^{(t)}, p_j), \sigma^{(t)}) \frac{\|\vec{p}_j^{(t)}\|}{\|\vec{e}^{(t)}\|} \vec{e}^{(t)} \tag{30}$$

d. h. bewege den Schwerpunktvektor in $\mathbb{E}$ auf das Element $\vec{e}$ zu, entsprechend der Parameter:

   - Die *Lernrate* $\epsilon^{(t)}$ beschreibt allgemein die Größenordnung von Änderungsoperationen in Abhängigkeit vom Zeitschritt t. Dieser Parameter wird auch als *Temperatur* bezeichnet, da in Anlehnung an physikalische Prozesse die Aktivität bzw. Beweglichkeit der Schwerpunkte proportional zur Größe dieses Parameters ist.
   - Die *Nachbarschaftsreichweite* $\sigma^{(t)}$ beschreibt als Parameter der *Nachbarschaftsfunktion* $h(.,.)$ die *Plastizität* der Karte durch Regulierung der *Größe*des von einer Aktualisierung betroffenen Kartenbereiches bestimmt. Generell gilt, dass, mit zunehmenden Abstand einer Partition vom Gewinner in A, die Stärke der Änderung reduziert wird.

4. Erhöhe den Iterationszähler $t := t + 1$. Ist die Abbruchbedingung erfüllt, beende den Algorithmus, sonst gehe zu Schritt 2.

Der Lernprozess der Karte wird im wesentlichen durch die Parameter ϵ und σ kontrolliert. Zu Beginn werden beide Parameter relativ groß gewählt, so dass jedes präsentierte Element große Bereiche der Karte beeinflusst. In dieser Phase, der *initiale Ordnungsphase* werden die Schwerpunkte der Partitionen zunächst grob um den Schwerpunkt der Datenverteilung im Elementraum $\mathbb{E}$ platziert und die Karte *entfaltet*, d. h. entsprechend der Nachbarschaftsstruktur ausgerichtet. Im weiteren Verlauf des Lernens werden beide Parameter allmählich verkleinert, s. d. kleinere Kartenbereiche in kleinerem Maße verändert werden. In dieser *Konvergenzphase* prägt die Detailstruktur der Daten in $\mathbb{E}$ die Karte. Wählt man σ so klein, dass in Schritt 3 nur der Gewinner aktualisiert wird, bricht die Nachbarschaftsstruktur zusammen und der Algorithmus verhält sich wie k-means (Abschnitt 6.1).

Ein Cluster c auf der Karte lässt sich definieren als die Menge von Partitionen, die in A einen zusammenhängenden Bereich bilden und in $\mathbb{E}$ hinreichend nahe beeinander liegen, d. h. der Abstand im Elementraum zwischen den Partitionsschwerpunkten $\vec{p}$ ist innerhalb eines Clusters vergleichsweise geringer als zu Partitionen anderer Cluster. Da ein Cluster im Allgemeinen von mehreren Partitionen beschrieben wird, ergeben sich im Unterschied zur Repräsentation mit einem Schwerpunkt (z. B. im Fall von k-means) Grenzen zwischen zwei Clustern, die nicht durch eine einzelne Hyperebene beschrieben werden können. Mit hinreichend vielen Partitionen sind damit praktisch Cluster beliebiger Form

möglich. Um eine bestimmte Anzahl k Cluster zu ermitteln, ist es daher erforderlich die Anzahl l der Partitionen als ein Vielfaches der gewünschten Anzahl Cluster festzulegen und durch Variation der Trennschärfe die Clusteridentifikation zu steuern.

Im Fall eines zweidimensionalen Ausgaberaumes A erlaubt die von der Karte beschriebene Abbildung $\mathbb{E} \rightarrow A$ eine direkte Visualisierung, die für andere Clustermethoden nur unter zusätzlichem Aufwand zu erhalten ist.

Der Platzbedarf des Algorithmus liegt bei $\mathcal{O}(m+l)$ (Elemente und Schwerpunkte) und verhält sich damit ähnlich zu dem des k-means ($\mathcal{O}(m+2k)$), allerdings ist im Allgemeinen die Zahl der Partitionen l größer als die der Cluster k. Die Zeitkomplexität liegt bei $\mathcal{O}(mt\,(l+\nu))$, wobei $\nu < l$ die durchschnittliche Anzahl der von einer Aktualisierung betroffenen Partitionen bezeichnet und vom Verlauf der Nachbarschaftsreichweite $\sigma^{(t)}$ abhängt. Die Komplexität liegt damit deutlich über der von k-means ($\mathcal{O}(mt\,(k+1))$), aber noch unterhalb der hierarchischer Algorithmen ($\mathcal{O}(m^2)$).

6.2.1 *Batch Map*

Eine alternative Variante ist der *Batch Map* Algorithmus [6], in dem die Partitionsvektoren erst nach der Zuordnung aller Elemente aktualisiert werden:

1. Initialisiere die Schwerpunktvektoren der Partitionen setze den Iterationszeitpunkt $t = 0$.
2. Bestimme zum Zeitpunkt t für jede Partition $p_j^{(t)}$ die zugeordneten Elemente
$$p_j^{(t)} = \{e \in E | j = \arg\min_i d_E(\vec{e}, \vec{p}_i^{(t)})\} \quad (31)$$
3. Aktualisiere die Partitionen
$$\vec{p}_j^{(t+1)} = \frac{\sum_i h(d_A(p_j, p_i), \sigma^{(t)}) \sum_{e \in \vec{p}_i^{(t)}} \vec{e}}{\sum_i h(d_A(p_j, p_i), \sigma^{(t)}) \# p_i^{(t)}} \quad (32)$$
4. Erhöhe den Iterationszähler $t := t + 1$. Ist die Abbruchbedingung erfüllt, beende den Algorithmus, sonst gehe zu Schritt 2.

Der Vorteil des *Batch Map* Algorithmus liegt in der Zeitkomplexität von $\mathcal{O}(mt\,(l+1) + tl^2)$, gegenüber k-means fällt nur der Mehraufwand durch die komplexere Aktualisierung nach Gleichung (32) an.

7 Clustern über Teildaten

Für bestimmte Aufgabenstellungen ist es notwendig die Clustereinteilung sehr schnell zu bestimmen. Um die Zeitkomplexität signifikant reduzieren zu können sind aggressive Optimierungen notwendig, die Näherungslösungen bestimmen. Die im Folgenden vorgestellten Verfahren wurden für online Anwendungen im Bereich des Clusterns von Textdokumenten entwickelt, d. h. für das Erzeugen der Cluster auf Anfrage, sind aber keineswegs auf diesen Aufgabenbereich beschränkt. Sie sind als Erweiterung existierender Verfahren zu betrachten und beschreiben Strategien zur Clusterbildung über Teildaten unter Verwendung existierender Methoden.

7.1 *Buckshot*-Algorithmus

Der *Buckshot*-Algorithmus [2] ermittelt Cluster anhand einer Stichprobe der Elementmenge E. Es wird dabei angenommen, dass die Cluster mittels *Schwerpunkte* repräsentiert werden. Das Ziel beim Entwurf des Verfahrens lag in hoher Geschwindigkeit ($\mathcal{O}(km)$) für die on-line Berechnung von Clusterschwerpunkten.

Aus der Gesamtmenge E der Objekte wird eine Stichprobe E' so gewählt, dass das Clusterverfahren mit der angestrebten Zeitkomplexität läuft, d. h. bei quadratischer Komplexität beträgt der Umfang der Stichprobe $\#E' = \sqrt{mk}$. über diese Stichprobe wird eine Clustereinteilung ermittelt, der dann alle Objekte aus E zugeordnet werden.

Der Algorithmus ist nichtdeterministisch in der Wahl der Stichprobe, die erzeugten Cluster variieren demzufolge über unterschiedliche Durchläufe. Die Wahl der Stichprobe ist ebenfalls kritisch wenn kleine Cluster angestrebt werden, da entsprechende Elemente möglicherweise nicht gezogen werden.

Zeit- und Platzkomplexität werden wesentlich vom verwendeten Clusterverfahren bestimmt. Erstere lässt sich über die Größe der Stichprobe begrenzen, dies beeinflusst jedoch die Qualität der Cluster.

7.2 *Fractionation*-Algorithmus

Die Idee hinter dem *Fractionation*-Algorithmus [2] ist die schnelle Clusterberechnung durch Partitionierung des Suchraumes. Die Menge der zu clus-

ternden Objekte E wird in eine Reihe von disjunkten Teilmengen zerlegt die separat in Cluster eingeteilt werden, die anschließend zu einer einzigen Clustermenge vereinigt werden. Dieser Prozess wird iterativ solange wiederholt, bis die angestrebte Clusteranzahl erreicht ist.

Im Detail lässt sich das Verfahren wie folgt formulieren:

1. Bilde die Startmenge von Clustern $C^{(0)}$ indem jedes Element $e \in E$ einen eigenen Cluster bildet.
$$C^{(0)} = \{\{e\} | e \in E\}$$
Setze den Iterationszähler $t = 0$.

2. Unterteile die Clustermenge $C^{(t)}$ in eine Menge von $x^{(t)} = \#C^{(t)}/n$ Partitionen
$$P^{(t)} = \{P_1^{(t)}, \ldots, P_{x^{(t)}}^{(t)}\}, \quad \bigcup_{i=1}^{x^{(t)}} P_i^{(t)} = C^{(t)},$$
$$\forall i \neq j : P_i^{(t)} \cap P_j^{(t)} = \emptyset$$
der Größe $n > k$.[1]

3. Berechne die Clustereinteilung $C(P_i^{(t)})$

4. Bilde die neue Clustermenge als Vereinigung der Einzelergebnisse $C^{(t+1)} = \bigcup_i C(P_i^{(t)})$ und erhöhe den Iterationszähler $t := t + 1$.

5. Ist die gewünschte Clustereinteilung noch nicht erreicht, d. h. gilt $\#C^{(t)} > k$, setze den Vorgang bei Schritt 2 fort.

Nimmt man an, dass jeder einzelne Clusterprozess (Schritt 3) die Größe der lokalen Partition auf den Anteil ρ reduziert, so ist es möglich den Prozess als das Erzeugen von $1/\rho$-nären Bäumen anzusehen, der terminiert wenn nur noch k Wurzeln existieren.

Entscheidend für die Qualität der erreichten Cluster ist die initiale Aufteilung der Elemente. Idealerweise sollte jede der Partitionen $P_i^{(0)}$ die einander jeweils ähnlichsten Elemente enthalten. Allerdings ist dies auch das Merkmal, das von dem Clusterprozess ermittelt werden soll. Im Falle multipler quantitativer Attribute mit ähnlichem Wertebereich lässt sich eine recht gute Annäherung erzielen, indem die Elemente nach dem jeweils dominierenden Attribut gruppiert werden.

Die Zeitkomplexität lässt sich abschätzen als $\mathcal{O}(\frac{m}{n} y(n)(1 + \rho + \rho^2 + \ldots)) = \mathcal{O}(\frac{m}{n} y(n))$, wobei $y(n)$ die Komplexität des verwendeten Clusterverfahrens bezeichnet.

8 Vergleich

Ohne im Detail auf experimentelle Daten und Bewertungsmöglichkeiten einzugehen[2], sind verschiedene generelle Aussagen über die Verfahren möglich: Die durchgängig besten Ergebnisse erzielen die informationstheoretisch motivierten Verfahren (Abschnitte 4.2 und 6.1.2), gefolgt von den Methoden zur iterativen Verbesserung (Abschnitt 6), wobei die Effektivität des *SOM* Algorithmus nur in relativ geringem Maß unter der von *k-means*(Abschnitt 6.1) liegt. Der *SPC* Algorithmus (Abschnitt 5) weist die geringste Effektivität bei günstigster Zeitkomplexität auf, ebenso sind die Methoden über Teildaten (Abschnitt 7) primär aus Effizienzgründen interessant, ihre Effektivität ist geringer als bei den entsprechenden Verfahren über kompletten Daten.

Literatur

[1] K. W. Church and W. A. Gale. A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of english bigrams. *Computer Speech and Language*, 5:19–54, 1991.

[2] D. R. Cutting, J. O. Pedersen, D. R. Karger, and J. W. Tukey. Scatter/Gather: A cluster-based approach to browsing large document collections. In *ACM SIGIR*, pages 318–329. ACM Press, 1992.

[3] I. S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections.

[1] Genau genommen besitzen $\#C^{(t)} \bmod n$ Partitionen eine Größe von $n + 1$ Elementen, da die Clustermenge im Allgmeinen eine Kardinalität ungleich $\alpha n, \alpha \in \mathbb{N}$ aufweist.

[2] Evaluierung von Clusterstrukturen und ein experimenteller Vergleich verschiedener Verfahren ist Teil meiner noch nicht erschienenen Dissertation.

In R. L. Grossman, C. Kamath, P. Kegelmeyer, V. Kumar, and R. R. Namburu, editors, *Data Mining for Scientific and Engineering Applications*, pages 357–382. Kluwer Academic Publishers, 2001.

[4] M. Goldszmidt and M. Sahami. A probabilistic approach to full-text document clustering. Technical Report ITAD-433-MS-98-044, SRI International, 1998.

[5] S. Kaski. *Data exploration using self-organizing maps*. PhD thesis, Helsinki University of Technology, Espoo, Finland, 1997.

[6] T. Kohonen. New develpoments of learning vector quantization and the self-organizing map. In *Symposion on Neural Networks; Alliances and Perspectives*, 1992.

[7] T. Kohonen. *Self-Organizing Maps*. Springer, 1995.

[8] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. L. Cam and J. Neyman, editors, *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley and Los Angeles, CA, 1967. University of California Press.

[9] E. Rasmussen. Cluster algorithms. In W. B. Frakes and R. Baeza-Yates, editors, *Information Retrieval*, pages 419–442. Prentice Hall, Eaglewood Cliffs, New Jersey, 1992.

[10] N. Slonim and N. Tishby. Agglomerative information bottleneck. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Proceedings of Neural Information Processing Systems (NIPS-99)*, pages 617–623. MIT Press, 1999.

[11] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *Proceedings of the 37th Allerton Conference on Communication and Computation*, pages 368–377, 1999.

[12] S. Vaithyanathan and B. Dom. Model selection in unsupervised learning with applications to document clustering. In *Proceedings of the 6th International Conference on Machine Learning*, pages 433–443, 1999.

[13] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.

[14] E. M. Voorhees. Implementing agglomerative hierarchical clustering algorithms for use in document retrieval. *Information Processing and Management*, 22(6):465–476, 1986.

[15] P. Willett. A note on the use of nearest neighbors for implementing single linkage document classifications. *Journal of the American Society for Information Science*, 35:149–152, 1984.

[16] P. Willett. Recent trends in hierarchic document clustering: A critical review. *Information Processing and Management*, 24(5):577–597, 1988.