B. Mitschang et al. (Hrsg.): Datenbanksysteme für Business, Technologie und Web (BTW 2017),
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 227

EFM-DBSCAN: Ein baumbasierter Clusteringalgorithmus unter Ausnutzung erweiterter Leader-Umgebungen

Philipp Egert[1]

Abstract:

DBSCAN ist ein dichte-basierter Clusteringalgorithmus, der beliebig geformte Cluster erkennt und sie von Rauschen trennt. Aufgrund der Laufzeit von $O(n^2)$ ist seine Anwendung jedoch auf kleine Datenkollektionen beschränkt. Um diesen Aufwand zu reduzieren, wurde der auf dem Konzept der Leader-Umgebung basierende Algorithmus FM-DBSCAN vorgestellt, der für beliebige Metriken dasselbe Clustering wie DBSCAN liefert. In dieser Arbeit wird nun basierend auf FM-DBSCAN das Verfahren EFM-DBSCAN entwickelt. EFM-DBSCAN nutzt die folgenden zwei Konzepte zur Effizienzsteigerung: (a) eine baumbasierte Partitionierung und (b) die Erweiterung der Objekte einer Leader-Umgebung um die Distanzen zu ihrem Leader. Erste Experimente zeigen, dass EFM-DBSCAN bis zu einem Faktor 17 weniger Distanzberechnungen und bis zu einem Faktor 13 weniger Rechenzeit als FM-DBSCAN benötigt. Gegenüber DBSCAN wurde ein Faktor von bis zu 10^4 eingespart.

Keywords: Density-based Clustering, DBSCAN, Leaders, Leader Neighbourhood

1 Einleitung

Dichte-basierte Clusteringverfahren sind häufig genutzte Algorithmen zum Clustern von Objekten. Sie sind in der Lage, beliebig geformte Cluster aufzufinden und diese von Rauschen zu trennen. DBSCAN [Es96] ist ein Vorreiter dieser Verfahren, der aufgrund seines eleganten Algorithmus und der nicht benötigten Vorgabe der Clusteranzahl für viele Einsatzgebiete attraktiv ist. Seine Laufzeit von $O(n^2)$ ist jedoch oft nicht praktikabel.

Zur Aufwandsverringerung wurden mehrere Ansätze entwickelt, unter anderem FM-DBSCAN [Eg16]. FM-DBSCAN ist ein exaktes Verfahren, dass für eine beliebige Metrik dasselbe Clustering wie DBSCAN liefert. Hierzu werden die Objekte in Leader-Umgebungen partitioniert und anschließend geclustert, indem die Erreichbarkeitsbegriffe von DBSCAN auf Leader-Umgebungen übertragen werden. Die Struktur der Leader-Umgebungen und die Ausnutzung der Dreiecksungleichung ermöglichen eine Beschleunigung des Clusterns. Zwar ist die Laufzeit immer noch $O(n^2)$, jedoch kann die Rechenzeit, aufgrund der Reduzierung der Anzahl der Distanzberechnungen, oft drastisch gesenkt werden.

In dieser Arbeit wird, basierend auf FM-DBSCAN, EFM-DBSCAN mit den folgenden neuen Konzepten entwickelt, um die Anzahl der Distanzberechnungen noch weiter zu verringern: (a) Eine baumbasierte Partitionierung zur Beschleunigung der Berechnung der

[1] Brandenburgische Technische Universität Cottbus–Senftenberg, Institut für Informatik, Informations- und Medientechnik, Konrad-Wachsmann-Allee 5, 03046 Cottbus, egertphi@b-tu.de

Leader-Partition und (b) die Erweiterung der Objekte einer Leader-Umgebung um die Distanzen zu ihrem Leader für schärfere Ausschlussbedingungen. Die Ausschlussbedingungen ermöglichen gleichzeitig eine effektivere Ausschlussstrategie anhand schrumpfender Leader-Umgebungen, um den Test der direkten Dichte-Erreichbarkeit zweier Leader-Umgebungen und den Test auf die Kernobjekteigenschaft zu beschleunigen.

2 Stand der Technik

In diesem Abschnitt wird der Stand der Technik vorgestellt. Hierzu werden einleitend die Algorithmen DBSCAN und FM-DBSCAN kurz vorgestellt. Abschließend werden noch weitere, aus der Literatur bekannte, Ansätze zur Beschleunigung von DBSCAN dargelegt.

2.1 DBSCAN

DBSCAN ist ein dichte-basiertes Clusteringverfahren von Ester et al. [Es96]. Ausgangspunkt ist eine endliche Menge von Objekten $O \subseteq \mathbb{U}$, wobei $\mathbb{U}$ das Universum aller möglichen Objekte ist. Über $\mathbb{U}$ sei eine Metrik d zum Vergleich der Objekte definiert. Die wesentlichen Definitionen zur Beschreibung eines Clusters werden anschließend eingeführt. Alle Definitionen sind dabei abhängig von den DBSCAN-Parametern ϵ und *minPts*.

Definition 1 (Kernobjekt)**.** Ein Objekt $o \in O$ heißt *Kernobjekt*, wenn $|N_\epsilon(o)| \geq minPts$ gilt, mit $N_\epsilon(o) = \{\hat{o} \in O \mid d(o, \hat{o}) \leq \epsilon\}$. Gilt dies nicht, heißt *o Nicht-Kernobjekt*.

Definition 2 (Direkte Dichte-Erreichbarkeit)**.** Ein Objekt $p \in O$ ist *direkt dichte-erreichbar* von $q \in O$, wenn q ein Kernobjekt und $p \in N_\epsilon(q)$ ist.

Definition 3 (Dichte-Erreichbarkeit)**.** Ein Objekt $p \in O$ ist *dichte-erreichbar* von einem Objekt $q \in O$, wenn es eine Folge von Objekten $p_1, \ldots, p_m \in O$ gibt, sodas $p_1 = q, p_m = p$ ist und es gilt: p_{i+1} ist direkt dichte-erreichbar von p_i für $1 \leq i < m$.

In DBSCAN bestimmt man einen Cluster C nun dadurch, dass man für ein Kernobjekt $o \in O$ alle dichte-erreichbaren Objekte iterativ einsammelt. Hierfür werden von o alle direkt dichte-erreichbaren Objekte bestimmt, also $N_\epsilon(o)$, und C hinzugefügt. Anschließend werden von allen Kernobjekten aus $N_\epsilon(o)$ deren direkt dichte-erreichbaren Objekte bestimmt und C beigefügt. Diese Schritte werden fortgesetzt, bis C nicht mehr erweitert werden kann. Alle Objekte die zu keinem Cluster gehören, werden als *Rauschen* bezeichnet.

2.2 FM-DBSCAN

FM-DBSCAN ist ein Clusteringverfahren, welches für eine beliebige Metrik dasselbe Clustering wie DBSCAN berechnet, jedoch effizienter ist [Eg16]. Nachfolgend werden die essenziellen Begriffe von FM-DBSCAN eingeführt und der Algorithmus erläutert.

Definition 4 (Leader-Umgebung)**.** Ein Tupel $(l, N_\epsilon^L(l))$ ist eine *Leader-Umgebung* in O, wenn $N_\epsilon^L(l) \subseteq N_\epsilon(l)$ mit $l \in N_\epsilon^L(l)$ ist. l ist der Repräsentant von $N_\epsilon^L(l)$ und heißt *Leader*.

Definition 5 (Leader-Partition)**.** Eine *Leader-Partition* von O ist eine Menge von Leader-Umgebungen $L = \{(l_1, N_\epsilon^L(l_1)), \ldots, (l_m, N_\epsilon^L(l_m))\}$ mit $d(l_i, l_j) > \epsilon$, $N_\epsilon^L(l_i) \cap N_\epsilon^L(l_j) = \emptyset$ für $i \neq j$ und $\bigcup_{i=1}^{m} N_\epsilon^L(l_i) = O$.

Definition 6 (Direkte Dichte-Erreichbarkeit)**.** Eine Leader-Umgebung $(l_p, N_\epsilon^L(l_p))$ ist *direkt dichte-erreichbar* von einer Leader-Umgebung $(l_q, N_\epsilon^L(l_q))$ in O, wenn es eine Folge von Objekten $p_1, \ldots, p_m \in (N_\epsilon^L(l_p) \cup N_\epsilon^L(l_q))$ gibt, mit $p_1 = l_q, p_m = l_p$ und es gilt: p_{i+1} ist direkt dichte-erreichbar von p_i für $1 \leq i < m$, und l_p ist ein Kernobjekt in O oder $|N_\epsilon^L(l_p)| = 1$.

Aus der Definition 6 folgt direkt: Ist $(l_p, N_\epsilon^L(l_p))$ direkt dichte-erreichbar von $(l_q, N_\epsilon^L(l_q))$, so sind alle Objekte $o \in (N_\epsilon^L(l_q) \cup N_\epsilon^L(l_p))$ dichte-erreichbar von l_q und gehören somit zum selben Cluster. Die Dichte-Erreichbarkeit wird analog zur Dichte-Erreichbarkeit von Objekten definiert. Im Anschluss werden noch drei Sätze vorgestellt, die es FM-DBSCAN ermöglichen, erheblichen Berechnungsaufwand im Vergleich zu DBSCAN einzusparen.[2]

Satz 1. Seien $(l_p, N_\epsilon^L(l_p))$ und $(l_q, N_\epsilon^L(l_q))$ zwei Leader-Umgebungen. $(l_p, N_\epsilon^L(l_p))$ ist direkt dichte-erreichbar von $(l_q, N_\epsilon^L(l_q))$ in O, genau dann, wenn l_q ein Kernobjekt ist und mindestens einer der folgenden beiden Fälle eintritt:
1. Es gibt zwei Objekte $p \in N_\epsilon^L(l_p), q \in N_\epsilon^L(l_q)$ mit $d(p, q) \leq \epsilon$ und p, q, l_p sind Kernobjekte.
2. $|N_\epsilon^L(l_p)| = 1$ und es gibt ein Kernobjekt $q \in N_\epsilon^L(l_q)$ in O mit $d(l_p, q) \leq \epsilon$.

Satz 2. Seien $p, q \in O$. Ist $d(q, p) > 2\epsilon$, so gilt: $\forall o \in N_\epsilon(p) : d(q, o) > \epsilon$.

Satz 3. Seien $p, q \in O$. Ist $d(p, q) > 3\epsilon$, so gilt: $\forall o_1 \in N_\epsilon(p), o_2 \in N_\epsilon(q) : d(o_1, o_2) > \epsilon$.

Der Algorithmus von FM-DBSCAN unterteilt sich in zwei Phasen, die Berechnung der Leader-Partition und das anschließende Clustern mithilfe der Leader-Partition.

Berechnung der Leader-Partition: Die Leader-Partition wird iterativ aufgebaut, indem jedes Objekt $o \in O$ der ersten Leader-Umgebung $(l, N_\epsilon^L(l))$ zugewiesen wird, für die $d(o, l) \leq \epsilon$ gilt. Gibt es eine solche Leader-Umgebung nicht, so wird die neue Leader-Umgebung $(o, \{o\})$ erzeugt. Die Überprüfungsreihenfolge der Leader-Umgebungen ist durch deren Erzeugungsreihenfolge festgelegt, weswegen wir diese Partitionierung auch First-come, first-served (FCFS)-Partitionierung nennen.

Clustern anhand der Leader-Partition: Die Bestimmung eines Clusters C ist analog zu DBSCAN. Ausgehend von einer Leader-Umgebung $(l, N_\epsilon^L(l))$, dessen Leader l ein Kernobjekt ist, werden alle dichte erreichbaren Leader-Umgebungen iterativ eingesammelt. Der wesentlicher Unterschied ist die Ermittlung der direkt dichte-erreichbaren Leader-Umgebungen. Hierfür bestimmt man mit Satz 3 zuerst die potentiell möglichen direkt dichte-erreichbaren Leader-Umgebungen (Kandidaten) und testet anschließend für jeden Kandidaten $(l_c, N_\epsilon^L(l_c))$, ob er wirklich direkt dichte-erreichbar von $(l, N_\epsilon^L(l))$ ist. Die maßgeblich Kosten entstehen durch die Tests der Kernobjekteigenschaft und der direkten Dichte-Erreichbarkeit. Nachfolgend werden beide Methoden beschrieben, da in EFM-DBSCAN dort wesentliche Änderungen erfolgen.

[2] Die Beweise sind aus Platzgründen in http://dbis.informatik.tu-cottbus.de/down/pdf/Leader_Umgebungen.pdf zu finden.

isCoreObject überprüft, ob das Objekt o ein Kernobjekt ist oder nicht, indem sie über die Objekte o_2 aller Leader-Umgebungen iteriert und zählt (Variable *counter*), wie oft $d(o, o_2) \leq \epsilon$ erfüllt ist. Zum Ausschluss von Leader-Umgebungen wird Satz 2 herangezogen. Handelt es sich bei o um einen Leader, so wird *counter* mit $|N_\epsilon^L(o)|$ initialisiert und $(o, N_\epsilon^L(o))$ von der weiteren Suche ausgeschlossen. Ein frühzeitiger Abbruch erfolgt, wenn $counter \geq minPts$ ist. Anhand eines im Objekt o gespeicherten Attributes kann das mehrmalige Berechnen der Kernobjekteigenschaft verhindert werden.

isDirectDensityReachable überprüft für $(l_1, N_\epsilon^L(l_1))$ und $(l_2, N_\epsilon^L(l_2))$, ob $(l_2, N_\epsilon^L(l_2))$ direkt dichte-erreichbar von $(l_1, N_\epsilon^L(l_1))$ ist. Dabei wird davon ausgegangen, dass l_1 schon als Kernobjekt erkannt wurde. Zuerst wird getestet, ob $(l_2, N_\epsilon^L(l_2))$ direkt dichte-erreichbar von $(l_1, N_\epsilon^L(l_1))$ sein kann (l_2 ist ein Kernobjekt oder $|N_\epsilon^L(l_1)| = 1$). Die eigentliche Überprüfung wird durch zwei ineinander verschachtelte Schleifen realisiert, sodass im schlimmsten Fall alle Objekte aus $N_\epsilon^L(l_1)$ mit denen aus $N_\epsilon^L(l_2)$ verglichen werden. Um dies zu verhindern, wird der Satz 2 angewendet. Haben zwei Objekte $o_1 \in N_\epsilon^L(l_1)$ und $o_2 \in N_\epsilon^L(l_2)$ eine Distanz $d(o_1, o_2) \leq \epsilon$, so wird der Satz 1 überprüft und ggf. erfolgreich abgebrochen.

2.3 Andere Arbeiten

Es existieren verschiedene Verfahren zur Beschleunigung von DBSCAN, wovon wir hier nur die exakten Verfahren betrachten, da FM-DBSCAN ebenfalls exakt ist. GriDBSCAN [MM08] basiert auf einer Gitterzerlegung des $\mathbb{R}^d$. Auf den einzelnen Gitterzellen wird DBSCAN ausgeführt und die Ergebnisse anschließend gemischt. Durch die Zerlegung in kleinere Teilprobleme lässt sich eine entsprechende Beschleunigung erzielen. Die DBSCAN-Variante von Gan und Tao [GT15] ist ein gitterbasiertes Verfahren, bei dem zuerst die Objekte einer Zelle als Kern- oder Nicht-Kernobjekte markiert und anschließend die Zellen mittels einer dem Satz 1 ähnlichen Bedingung zu Clustern vermischt werden. Die Laufzeit beträgt $O((n \log n)^{4/3})$ für $d = 3$ und $O(n^{2-2/(\lceil d/2 \rceil + 1)+\delta})$ für $d \geq 4$, mit $\delta > 0$. G-DBSCAN [KR16] nutzt ebenfalls Leader-Umgebungen, nennt sie jedoch Gruppen. Die Erzeugung der Gruppen ist dem Verfahren von FM-DBSCAN sehr ähnlich. Die Gruppen werden verwendet, um die Berechnung der ϵ-Umgebung eines Objektes zu beschleunigen. Dabei bedienen sie sich einem Satz ähnlich dem Satz 3. Die Laufzeit beträgt $O(n^2)$. FM-DBSCAN hat diverse Vorteile gegenüber den hier aufgeführten Verfahren. Es werden keine zusätzlichen Parameter benötigt, anders als bei der Arbeit [MM08]. Zusätzlich ist FM-DBSCAN flexibel für beliebige Metriken einsetzbar. Die Arbeiten [GT15; MM08] sind auf den $\mathbb{R}^d$ beschränkt. Nur die Arbeit [KR16] ist ebenfalls metrisch. Zwar wird die Berechnung der ϵ-Umgebung beschleunigt, jedoch wird sie für jedes Objekt ausgeführt, was zu einem erhöhten Aufwand führt. FM-DBSCAN hingegen nutzt zielgerichteter die Erreichbarkeitsbegriffe von Objekten aus, um Beschleunigungen zu erzielen.

3 EFM-DBSCAN

In diesem Abschnitt wird Extended Fast Metric DBSCAN (EFM-DBSCAN) präsentiert, der basierend auf FM-DBSCAN, die folgenden neuen Konzepte einführt:

(a) Eine baumbasierte Partitionierung unter Nutzung der Excluded Middle Partitioning.

(b) Erweiterung der Objekte der Leader-Umgebungen um die Distanzen zu ihrem Leader.

Die bisherige FCFS-Partitionierung sucht für ein Objekt o linear über die aktuellen Leader-Umgebungen, um eine Leader-Umgebung $(l, N_\epsilon^L(l))$ zu finden, für die $d(o, l) \leq \epsilon$ gilt. Dabei ist sie von der Einfügereihenfolge abhängig. Ist diese ungünstig, führt das zu einer erhöhten Anzahl an Distanzberechnungen. Durch die Nutzung eine Baumes (a) können, anhand der räumlichen Lage, gezielt Teilbäume und somit Leader-Umgebungen ausgeschlossen werden, sodass man Distanzberechnungen und somit Rechenzeit einspart. Die in (b) durchgeführte Erweiterung ermöglicht schärfere Ausschlussbedingungen für die Sätze 2 und 3. Diese werden für eine effektivere Ausschlussstrategie anhand schrumpfender Leader-Umgebungen genutzt, um isCoreObject und isDirectDensityReachable zu beschleunigen.

3.1 Baumbasierte Partitionierung

In diesem Abschnitt wird die neue, baumbasierte Partitionierung zur Berechnung einer Leader-Partition vorgestellt. Zuerst wird das Prinzip der Excluded Middle Partitioning (EMP) vorgestellt, worauf die Partitionierung beruht. Anschließend erfolgt die Vorstellung des Excluded Middle Partitioning Tree (EMPT), der für die Partitionierung genutzt wird.

3.1.1 Excluded Middle Partitioning

Bei der EMP [Yi99] wird die Menge der Objekte O anhand eines gewählten Pivotobjektes $o_p \in O$ in die Mengen $S_1 = \{o \in O \mid d(o, o_p) \leq d_m - \rho\}$, $S_2 = \{o \in O \mid d_m - \rho < d(o, o_p) \leq d_m + \rho\}$ und $S_3 = \{o \in O \mid d(o, o_p) > d_m + \rho\}$ zerlegt. Bei d_m handelt es sich um den Median der Distanzen der Objekte $o \in O$ zu o_p. ρ ist ein vom Nutzer vorgegebener Parameter, der die Trenndistanz zwischen S_1 und S_3 festlegt. Für alle Objekte $o_1 \in S_1$, $o_2 \in S_3$ gilt $d(o_1, o_2) > 2\rho$. Eine solche beispielhafte Zerlegung ist in Abb. 1a zu sehen. Wesentlicher Vorteil der Unterteilung ist, dass für eine Bereichssuche des Objekts o_q in O bzgl. $\epsilon \leq \rho$ nur maximal in zwei der drei Mengen S_1, S_2 und S_3 gesucht werden muss. Um nun einen Baum mittels der EMP zu erhalten, müsste man S_1, S_2 und S_3 nur nach dem gleichen Prinzip erneut rekursiv weiter partitionieren. Die Pivotobjekte stellen die Knoten des Baumes dar.

(a) Excluded Middle Partitioning (b) Excluded Middle Partitioning Tree

Abb. 1: Beispiel für die Mengen S_1, S_2 und S_3.

3.1.2 Excluded Middle Partitioning Tree (EMPT)

Mit der EMP soll nun ein Baum konstruiert werden, der die Berechnung einer Leader-Partition beschleunigt. Hierzu muss der Baum folgende Anforderungen erfüllen. Es müssen zum einen die Leader-Umgebungen in den Baum abgebildet werden und zum anderen muss gewährleistet werden, dass zwei Leader eine Distanz größer als ϵ zueinander aufweisen.

Die Abbildung der Leader-Umgebungen erfolgt dadurch, dass anstatt der Objekte nun die Leader-Umgebungen die Knoten repräsentieren. Des Weiteren werden die Parameter der Excluded Middle Partitioning angepasst. Wir setzen $d_m = 3\epsilon$ und $\rho = \epsilon$ fest und beschränken die Objekte o von S_1 nach unten mit $\rho < d(o, l)$, sodass sich folgende drei Mengen ergeben: $S_1 = \{o \in O \mid \epsilon < d(o, l) \leq 2\epsilon\}$, $S_2 = \{o \in O \mid 2\epsilon < d(o, l) \leq 4\epsilon\}$ und $S_3 = \{o \in O \mid d(o, l) > 4\epsilon\}$. Alle Objekte o, die nicht in S_1, S_2 oder S_3 liegen, gehören zu der Leader-Umgebung des Knotens, mit l als Leader. Ein Beispiel für die neue Zerlegung mit einer Leader-Umgebung (innerer grauer Kreis) ist in Abb. 1b enthalten.

Um nun sicherzustellen, dass die Leader zweier Leader-Umgebungen eine Distanz größer ϵ haben, wird der Baum iterativ aufgebaut. Initialisiert wird der Baum mit der Leader-Umgebung $(o_s, \{o_s\})$ eines zufällig gewählten Objekts $o_s \in O$. Für jedes weitere Objekt o wird nun versucht, es in eine Leader-Umgebung $(l, N_\epsilon^L(l))$ im aktuellen Baum einzufügen, für die $d(o, l) \leq \epsilon$ gilt. Hierzu wird der Baum mittels einer Tiefensuche traversiert. Dazu wird die Distanz des Leaders l vom aktuellen Knoten zum Objekt o berechnet. Gilt $d(o, l) \leq \epsilon$, so wird o in $N_\epsilon^L(l)$ eingefügt und die Suche abgebrochen. Ansonsten wird entsprechend der Distanz $d(o, l)$ entweder im linken ($d(l, o) \leq 2\epsilon$), mittleren ($2\epsilon < d(l, o) \leq 4\epsilon$) oder rechten ($d(l, o) > 4\epsilon$) Teilbaum weitergesucht. Der linke, mittlere und rechte Teilbaum repräsentieren jeweils die Mengen S_1, S_2 und S_3 bezüglich des Leaders l. Sollte in diesen Teilbäumen ebenfalls keine Leader-Umgebung gefunden werden, so muss anschließend für den linken im mittleren, für den mittleren entweder im linken ($d(l, o) \leq 3\epsilon$) oder im rechten ($d(l, o) > 3\epsilon$) und für den rechten ggf. im mittleren ($d(l, o) \leq 5\epsilon$) Teilbaum weitergesucht werden. Die Suche in weiteren Teilbäumen muss durchgeführt werden, da die ϵ-Umgebung des Objektes o mehrere Teilbäume schneiden kann. Konnte keine Leader-Umgebung gefunden werden, so wird $(o, \{o\})$ als Blattknoten eingefügt. Hierfür wandert man den Baum von der Wurzel nach unten und steigt jeweils im aktuellen Knoten entweder in den linken, mittleren oder rechten Teilbaum ab. Wurden alle Objekte eingefügt, so bilden die Knoten unsere Leader-Partition. Den eben beschriebenen Algorithmus nennen wir EMPT-Partitionierung. Von Vorteil ist, dass keine zusätzlich Parameter benötigt werden. Ein Nachteil ist, dass der Baum nicht zwangsläufig balanciert ist, sodass er zu einer Liste entarten kann. Somit wird er nicht wesentlich schlechter als die FCFS-Partitionierung.

3.2 Erweiterung der Objekte der Leader-Umgebungen

Die Erweiterung der Objekte o einer Leader-Umgebung $(l, N_\epsilon^L(l))$ um die Distanzen $d(o, l)$ erfolgt durch die Abbildung $d^l : N_\epsilon^L(l) \to \mathbb{R}$, mit $d^l(o) = d(o, l)$. Die Distanzen werden nur einmal und nicht bei jedem Aufruf von d^l berechnet. Die erstmalige Berechnung erfolgt bei der Bestimmung der Leader-Partition, da dort die Distanzen zum Leader berechnet werden

müssen. Eine Aktualisierung von d^l erfolgt ggf. noch in der Neuverteilung der Objekte von FM-DBSCAN [Eg16]. Mit der Erweiterung werden nun schärfere Ausschlussbedingungen für die Sätze 2 und 3 formuliert. Diese werden dann dazu genutzt, um eine effektivere Ausschlussstrategie anhand schrumpfender Leader-Umgebungen zur Beschleunigung der Methoden isCoreObject und isDirectDensityReachable zu entwickeln.

3.2.1 Schärfere Ausschlussbedingungen

Sei d^l_{max} die maximale Distanz eines Objektes einer Leader-Umgebung zu seinem Leader, dann lassen sich die folgenden Sätze formulieren[3]:

Satz 4. Sei $q \in O$ ein Objekt und $(l, N^L_\epsilon(l))$ eine Leader-Umgebung in O. Ist $d(q,l) > \epsilon + d^l_{max}$, so gilt: $\forall o \in N^L_\epsilon(l) : d(o,q) > \epsilon$.

Satz 5. Seien $(l_1, N^L_\epsilon(l_1))$ und $(l_2, N^L_\epsilon(l_2))$ zwei Leader-Umgebungen in O. Ist $d(l_1,l_2) > \epsilon + d^{l_1}_{max} + d^{l_2}_{max}$, so gilt: $\forall o_1 \in N^L_\epsilon(l_1), o_2 \in N^L_\epsilon(l_2) : d(o_1,o_2) > \epsilon$.

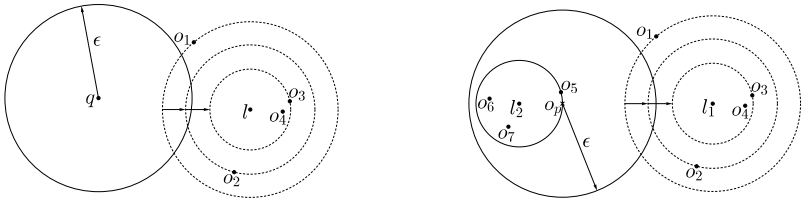
Die Sätze 4 und 5 sind eine Verbesserung der Sätze 2 und 3, da nicht mehr ϵ, sondern d^l_{max} als maximale Distanz für eine Leader-Umgebung verwendet wird. Da $d^l_{max} \leq \epsilon$ ist, kann mit den Sätzen 4 und 5 die Suche eher abgebrochen werden, als mit den Sätzen 2 und 3.

3.2.2 Ausschlussstrategie anhand schrumpfender Leader-Umgebungen

Eine verbesserte Ausschlussstrategie für isCoreObject und isDirectDensityReachable ergibt sich, wenn man den folgenden Zusammenhang betrachtet. Für eine Leader-Umgebung $(l, N^L_\epsilon(l))$ stellt das Tupel $(l_2, N^L_\epsilon(l_2))$, mit $l_2 = l$, $l_2 \in N^L_\epsilon(l_2)$ und $N^L_\epsilon(l_2) \subseteq N^L_\epsilon(l)$, ebenfalls eine Leader-Umgebung dar. Da $N^L_\epsilon(l_2)$ eine Teilmenge von $N^L_\epsilon(l)$ ist, gilt zusätzlich $d^l_{max} \geq d^{l_2}_{max}$. Diese Eigenschaft kann man sich zunutze machen, wenn man die Objekte o_i einer Leader-Umgebung $(l, N^L_\epsilon(l))$ immer absteigend sortiert nach $d^l(o_i)$ abarbeitet. Sei $o_1, o_2, \ldots, o_n$ eine solche Sortierung für $N^L_\epsilon(l)$, dann stellen die Tupel $(l_i, N^L_\epsilon(l_i))$, mit $l_i = l$, $N^L_\epsilon(l_i) = \{o_i, \ldots o_n\}$, $d^{l_i}_{max} = d(o_i, l)$ und $1 \leq i \leq n$, eine Folge von schrumpfenden Leader-Umgebungen dar, auf die man den Satz 4 für ein Objekt q anwenden kann. Analog kann man das für zwei schrumpfende Leader-Umgebungen und Satz 5 tun.

Beispiele für den Abbruch anhand von Satz 4 oder 5 sind in Abb. 2 zu sehen. Die Folge der schrumpfenden Leader-Umgebungen ist durch die gestrichelten Kreise und die schwarzen Pfeile dargestellt. In Abb. 2a kann bei der von o_3 begrenzten Leader-Umgebung anhand von Satz 4 abgebrochen werden, da es keine Überlappung mit der ϵ-Umgebung von q gibt. In Abb. 2b erfolgt ein Abbruch mit Satz 5 auch erst mit Objekt o_3. Die eingezeichnete ϵ-Umgebung gehört zu dem am nächsten liegenden Objekt o_p zu l_1, was noch in $(l_2, N^L_\epsilon(l_2))$ enthalten sein könnte. Solange die schrumpfenden Leader-Umgebungen von $(l_1, N^L_\epsilon(l_1))$ diese ϵ-Umgebung schneiden, ist ein Abbruch mit Satz 5 nicht möglich.

[3] Die Beweise sind aus Platzgründen in http://dbis.informatik.tu-cottbus.de/down/pdf/Leader_Umgebungen.pdf zu finden.



(a) Abbruch anhand von Satz 4 (b) Abbruch anhand von Satz 5

Abb. 2: Beispiele für schrumpfende Leader-Umgebungen und der Abbruch anhand von Satz 4 und 5

Für die Integration der Ausschlussstrategie werden beide Methoden angepasst. In isCoreObject muss in der Schleife, in der über die Objekte o_2 der aktuellen Leader-Umgebung $(l, N_\epsilon^L(l))$ iteriert wird, der Test von Satz 4 vor der Berechnung von $d(o, o_2)$ eingefügt werden, um die Suche für das zu Objekt o in $(l, N_\epsilon^L(l))$ frühzeitig abzubrechen.

Alg. 1 isDirectDensityReachable$((l_1, N_\epsilon^L(l_1)), (l_2, N_\epsilon^L(l_2)), L, \epsilon, minPts)$

```
1: if isCoreObject(l2, L, ϵ, minPts) ∨ |N_ϵ^L(l2)| = 1 then
2:     N2 := N_ϵ^L(l2); d_max^l2 = max_{o∈N2} d^l2(o);
3:     for all o1 ∈ N_ϵ^L(l1) do
4:         if d(l1, l2) > d^l1(o1) + d_max^l2 + ϵ then return false;
5:         if d(o1, l2) ≤ ϵ ∧ isCoreObject(o1, L, ϵ, minPts) = true then return true;
6:         for all o2 ∈ N2 do
7:             if d(o1, l2) > d^l2(o2) + ϵ then break;
8:             if d(o2, l1) ≤ ϵ ∧ (|N_ϵ^L(l2)| = 1∨ isCoreObject(o2, L, ϵ, minPts) = true) then return true;
9:             if d(o2, l1) > d^l1(o1) + ϵ then
10:                N2 := N2 \ {o2}; d_max^l2 = max_{o∈N2} d^l2(o);
11:                if N2 = ∅ then return false;
12:            if d(o1, o2) ≤ ϵ∧ isCoreObject(o1, L, ϵ, minPts) = true then
13:                if |N_ϵ^L(l2)| = 1∨ isCoreObject(o2, L, ϵ, minPts) = true then return true;
14: return false;
```

Die Anpassung von isDirectDensityReachable ist aufwendiger, weswegen der Code in Alg. 1 dargestellt ist. L ist die Menge der Leader-Umgebungen. Satz 4 wird für die Objekte aus $N_\epsilon^L(l_1)$ in der Z. 7 analog zu der Anpassung von isCoreObject überprüft. Um Satz 4 auch für die Objekte aus $N_\epsilon^L(l_2)$ anzuwenden, wird eine temporäre Kopie N_2 von $N_\epsilon^L(l_2)$ verwendet. Aus N_2 wird, falls Satz 4 anwendbar ist (Z. 9), das entsprechende Objekt entfernt und die neue maximale Distanz zum Leader l_2 ermittelt (Z. 10). Ist N_2 leer, kann die Methode als fehlgeschlagen abgebrochen werden (Z. 11). Der Test von Satz 5 befindet sich in der Z. 4. Der Test der direkten Dichte-Erreichbarkeit mittels Satz 1 ist in den Z. 1, 8 und 12–13 enthalten. Die sortierte Abarbeitung hat zusätzlich den Vorteil, dass gerade Objekte, die am Rand von Leader-Umgebungen liegen, auf die Distanz kleiner gleich ϵ getestet werden. Diese sind eher als die inneren Objekte dazu geeignet die direkte Dichte-Erreichbarkeit zu gewährleisten, sodass zeitiger abgebrochen werden kann. Nachteilig ist die Sortierung der Leader-Umgebungen. Dieser zusätzliche Aufwand fällt jedoch durch die eingesparten Distanzberechnungen nicht erheblich ins Gewicht (siehe Abschnitt 4).

4 Evaluation

EFM-DBSCAN wurde mit DBSCAN, G- und FM-DBSCAN verglichen. Alle Experimente liefen auf einem Intel® Core™ i7-2600K Prozessor. Die Daten wurden im Hauptspeicher gehalten. Als Effizienzmaß wurde die Anzahl der Distanzberechnungen (#Distanzen) und die Rechenzeit genutzt. Jeder Lauf wurde fünfmal mit einer zufällig generierten Permutation der Objekte ausgeführt und die Ergebnisse gemittelt. Als synthetische Datenkollektion kam *Gaussian Mixture Clusters* (12 Dimensionen)[4] zum Einsatz. Als reale Datenkollektion wurde *Caltech256* [GHP07] verwendet, die 30.607 Bilder enthält, für die das Feature *Dominant Color* (MPEG-7) extrahiert wurde. Für *Caltech256* wurde die Earth Mover's Distanz (EMD) und für *Gaussian Mixture Clusters* die euklidische Distanz genutzt.

Abb. 3 stellt die Skalierbarkeit der Verfahren mit wachsender Kollektionsgröße dar. EFM-DBSCAN benötigt deutlich weniger Distanzberechnungen und Rechenzeit als die anderen Verfahren. Gegenüber DBSCAN und G-DBSCAN konnte ein Faktor in der Größenordnung von 10 bis 10^2 eingespart werden. Im Vergleich zu FM-DBSCAN wird ein Faktor von 7 bis 17 weniger Distanzberechnungen und ein Faktor 3 bis 14 weniger Rechenzeit benötigt. Es zeigt sich also, dass die eingeführten Verbesserungen eine Beschleunigung erzielen und der zusätzliche Sortieraufwand vernachlässigbar ist. Obwohl G-DBSCAN gegenüber DBSCAN weniger Distanzberechnungen ausführt, ist die Rechenzeit höher. Dies ist auf den Mehraufwand zur Reduzierung der Distanzberechnungen und der kompletten Berechnung der ϵ-Umgebung zurückzuführen.

In Abb. 4 ist der Einfluss von ϵ auf die Performanz der Verfahren dargestellt. FM- und EFM-DBSCAN erzielen auf beiden Kollektionen die besten Ergebnisse. Wie erwartet, nähern sich die beiden Verfahren mit wachsendem ϵ immer weiter an. Das liegt an der sinkenden Anzahl Leader-Umgebungen, was das dazu führt, dass der Baum der EMPT-Partitionierung nicht ausreichend befüllt ist und die Häufigkeit der Tests der direkten Dichte-Erreichbarkeit sinkt. Somit werden die eingeführten Verbesserungen seltener angewendet. Dennoch erzielt EFM-DBSCAN gerade für aufwendig zu berechnende Metriken (EMD) und kleinere Werte für ϵ eine entsprechende Verbesserung gegenüber FM-DBSCAN. Für *Gaussian Mixture Clusters* und *Caltech256* wird bis zu einem Faktor 17 und 5 an Distanzberechnungen und bis zu einem Faktor 13 und 4 an Rechenzeit eingespart.

5 Zusammenfassung und Ausblick

In dieser Arbeit wurde das Verfahren EFM-DBSCAN vorgestellt, welches auf FM-DBSCAN basiert. Dabei wurden zwei neue Konzepte zu Reduzierung der Anzahl der Distanzberechnungen eingeführt: (a) eine baumbasierte Partitionierung und (b) die Erweiterung der Objekte einer Leader-Umgebung um die Distanzen zu ihrem Leader. Die Experimente zeigen, dass EFM- gegenüber FM-DBSCAN bis zu einem Faktor 17 weniger Distanzberechnungen und bis zu einem Faktor 13 weniger Rechenzeit benötigt. Im Vergleich zu DBSCAN wurde bis zu einem Faktor 10^4 eingespart. In zukünftigen Arbeiten soll die Wahl von ϵ und *minPts* bzgl. einer guten Qualität und hohen Effizienz untersucht werden.

[4] Generiert mittels RapidMiner (https://rapidminer.com/) in den Größen n = 10.000, 20.000, . . . , 100.000.

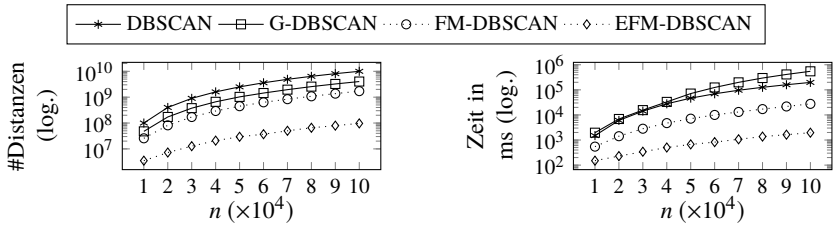
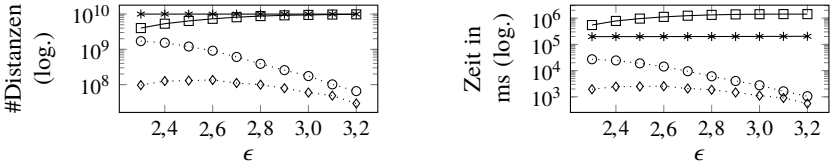
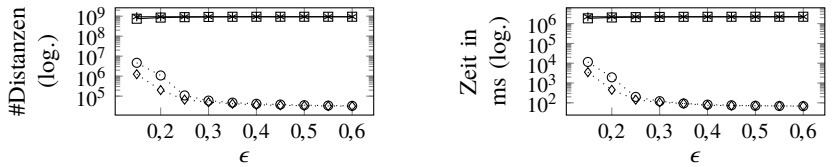


Abb. 3: Skalierbarkeit mit der Kollektionsgröße n auf *Gaussian Mixture Clusters* ($minPts = 5$, $\epsilon = 2{,}3$)



(a) *Gaussian Mixture Clusters* ($minPts = 5$, $n = 100.000$)



(b) *Caltech256* ($minPts = 2$, $n = 30.607$)

Abb. 4: Performanz für variierendes ϵ

Literatur

[Eg16] Egert, P.: FM-DBSCAN: Ein effizienter, dichte-basierter Clustering-Algorithmus. In: Proc. of the 28th GI-Workshop GvDB. S. 44–49, 2016.

[Es96] Ester, M.; Kriegel, H.; Sander, J.; Xu, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: Proc. of the 2nd Int. Conf. on Knowledge Discovery and Data Mining. S. 226–231, 1996.

[GHP07] Griffin, G.; Holub, A.; Perona, P.: Caltech-256 Object Category Dataset, Techn. Ber. 7694, California Institute of Technology, 2007.

[GT15] Gan, J.; Tao, Y.: DBSCAN Revisited: Mis-Claim, Un-Fixability, and Approximation. In: Proc. of the 2015 ACM SIGMOD. S. 519–530, 2015.

[KR16] Kumar, K. M.; Reddy, A. R. M.: A fast DBSCAN clustering algorithm by accelerating neighbor searching using Groups method. Pattern Recognition 58/, S. 39–48, 2016.

[MM08] Mahran, S.; Mahar, K.: Using grid for accelerating density-based clustering. In: Proc. of 8th IEEE Int. Conf. on Computer and Info. Techn. S. 35–40, 2008.

[Yi99] Yianilos, P. N.: Excluded middle vantage point forests for nearest neighbor search. In: In DIMACS Implementation Challenge, ALENEX'99. 1999.