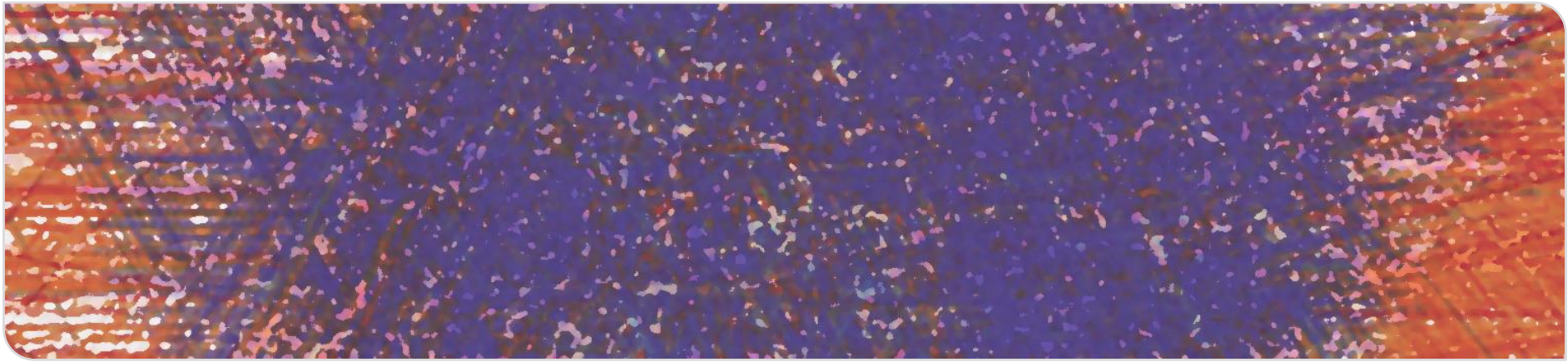
Karlsruher Institut für Technologie

AGD – WS 2020/21 – Übungssitzung 4

Jakob Bach (jakob.bach@kit.edu)



Zeitplan

Woche	Datum*	Übung
1	03.11.2020	
2	10.11.2020	
3	17.11.2020	Abgabe Blatt 1
4	24.11.2020	
5	01.12.2020	Abgabe Blatt 2
6	08.12.2020	Sitzung 1
7	15.12.2020	Abgabe Blatt 3

Woche	Datum*	Übung
8	22.12.2020	
9	12.01.2021	Abgabe Blatt 4
10	19.01.2021	Sitzung 2
11	26.01.2021	Keine Abgabe
12	02.02.2021	Sitzung 3
13	09.02.2021	Abgabe Blatt 5
14	16.02.2021	Sitzung 4

* regulärer Vorlesungstermin am Dienstag; Abgabetermin der Übungsblätter kann ein anderer Tag sein

Agenda

Inhalt: Übungsblatt 5, Vorlesungskapitel 9-10

- Umfrage: Vorbereitung auf Sitzung
- Eure Fragen: Code-Aufgaben, Online-Tests, Sonstiges
- Theorie-Aufgaben
- Kurze Feedback-“Runde“

Ankündigungen:

- Umfrage zur Übung im ILIAS vom 16.02. bis zum 21.02.
- Abgabedatum der Qualifikationsaufgabe für Praktikum: 14.03.2021

Fragen (1)

- Ist das Praktikum zu empfehlen?
  - Ja ;-)
  - Noch mal umfassenderer Blick auf Data Science, große Aufgaben (nicht nur isolierte Betrachtung der Algorithmen wie in Vorlesung/Übung)
- Was ist der Inhalt des Praktikums?
  - 2 Phasen mit jeweils 6-7 Wochen
  - Aufgabe 1: Teilnahme an Data Mining Cup 2021 – in Vergangenheit meist Regressions-/ Klassifikationsaufgabe mit Daten aus dem Einzelhandel
  - Aufgabe 2: jetzt noch nicht bekannt, Aufgabe aus einem aktuellen Forschungsgebiet unseres Lehrstuhls
  - Arbeit in Teams (vermutlich Dreierteams)

Fragen (2)

- Wie werden die Praktikums-ECTS verbucht, da das Praktikum unbenotet ist?
  - Wie normale Lehrveranstaltung im Notenauszug, nur ohne Note (ähnlich Schlüsselqualifikation)
- Warum haben die Testdaten in der Qualifikationsaufgabe keine Labels – das Testen der Performance ist dadurch doch nicht möglich?
  - Performance testet Jakob anhand eurer Vorhersage ;-)
  - In der Praxis wären die Testdaten neue Daten, die nach Training des Vorhersagemodells hereinkommen
  - Testdaten sollen (hier und auch generell) nicht zum Training (oder Auswahl von Vorhersagemodellen) genutzt werden
  - Für das Vergleichen von Vorhersagemodellen in eurer Lösung könnt ihr selbst die Trainingsdaten aufteilen (Trainingsdaten - Validierungsdaten)

Fragen (3)

- Was ist das Praktikum „Analysis of Complex Data Sets“?
  - Findet nicht statt, auch wenn es im Modulhandbuch steht
  - Einzige „aktive“ Praktika des Lehrstuhls sind Datenbank-Praktikum im WS und Data-Science-Praktikum (früher: AGD-Praktikum) im SS
  - Außerdem jedes Semester Seminar, Titel ändert sich laufend (Themen werden von Doktoranden gestellt): http://dbis.ipd.kit.edu/3051.php
- Ist die letzte Vorlesung prüfungsrelevant?
  - Im Zweifelsfall: ja
  - Feste Aussage diesbezüglich kann aber nur Prof. Böhm treffen
- Woher kommt der Unterschied zwischen AGD und AGD2 bzgl. ECTS?
  - AGD2 hat keine Übung
  - Stoffumfang der Vorlesung tatsächlich vergleichbar

Theorieaufgaben – Hierarchisches Clustering

- Was sind die Vorteile/Nachteile von divisivem Clustering gegenüber agglomerativem Clustering?
  - Generell: agglomerativ beginnt lokal (bottom-up), divisiv beginnt global (top-down)
  - Generell: Ergebnis von agglomerativem Clustering hängt stark von Linkage Criterion (Distanzfunktion zwischen zwei Clustern) ab
  - Vorteil divisiv: startet mit Blick auf Verteilung der Gesamtdaten (bei agglomerativ dagegen erst später größere Cluster, Informationsverlust durch Aggregation)
  - Vorteil agglomerativ: Komplexität von $O(n^3)$, da n Iterationen mit je $O(n^2)$ Distanzberechnungen; bei divisivem Clustering allein im ersten Schritt $O(2^n)$ Möglichkeiten, die Daten aufzuteilen (daher nicht alle Möglichkeiten betrachtet)
  - Zum Vergleich: k-means hat $O(n * k * \#Iterationen)$, OPTICS/DBSCAN haben $O(n * \log n)$ (mit räumlicher Indexstruktur und „passender“ Verteilung der Daten)

Theorieaufgaben – OPTICS allgemein

- Was sind Vor- und Nachteile von OPTICS gegenüber DBSCAN?
  - Parameter ε bei DBSCAN normalerweise schwer zu wählen
    - Bei OPTICS kann man einfach großes $\underline{\varepsilon}$ wählen
    - Wenn $\underline{\varepsilon}$ zu groß, geht Laufzeit allerdings gegen $O(n^2)$ (räumliche Indexstruktur nutzlos)
  - OPTICS liefert kein festes Clustering, sondern eine Datenstruktur (Reachability Plot)
    - Geeignetes ε lässt sich ablesen
    - DBSCAN-Ergebnisse für verschiedene ε lassen sich schnell daraus bestimmen $\rightarrow O(n)$
    - In Datenstruktur lassen sich womöglich Cluster unterschiedlicher Dichte erkennen (bei Umwandlung in DBSCAN-Ergebnis wird aber eine einheitliche Dichtegrenze festgelegt)
  - Reale Laufzeit bei OPTICS kann schlechter als bei DBSCAN sein
    - Theoretische Komplexität wie bei DBSCAN $O(n * \log n)$ (mit räumlicher Indexstruktur und „passender“ Verteilung der Daten)
    - Durch $\underline{\varepsilon}$ statt (kleinerem) ε verringert sich aber Nutzen der räumlichen Indexstruktur
    - Zeitlicher Zusatzaufwand für Verwaltung/Update der Priority Queue (erhöht aber theoretische Komplexität nicht), außerdem zusätzlicher Speicheraufwand dafür

Theorieaufgaben – OPTICS-Algorithmus

- Wie viele Datenobjekte müssen bei der Ausgabe eines Datenobjekts von OPTICS betrachtet werden?
  - Objekte in ε-Nachbarschaft müssen betrachtet werden (siehe `ExpandClusterOrder()` in Vorlesungsfolien)
  - Für Nachbar-Objekte jeweils (siehe `ControlList::update()`) Reachability Distance berechnet und je nach Situation
    - Objekt in Priority Queue eingefügt (wenn Objekt noch nicht darin vorhanden)
    - Wert (Priorität) des Objektes in der Priority Queue angepasst (wenn neue Reachability Distance kleiner als bisherige Reachability Distance)
    - Nichts getan (Objekt bereits in Priority Queue und potentielle neue Reachability Distance größer als bisherige)

Theorieaufgaben – Clustering mit kategorischen Attributen / Link-basiertes Clustering

- Warum ist Clustering mit kategorischen Attributen besonders (in Bezug auf Link-basiertes Clustering)?
  - Generell bei kategorischen Attributen:
    - Abstand zwischen Werten meist nur als 0 oder 1 (gleich oder nicht) beschreibbar
    - Wenn viele Attribute (z.B. mögliche Itemsets bei Einkauf), dann viele davon oft 0
    - Abhilfe: Jaccard-Koeffizient
  - Link-basiertes Clustering
    - Ähnlichkeit zwischen Datenobjekten zunächst z.B. mit Jaccard-Koeffizient beschrieben, dann aber binärisiert (ist Ähnlichkeit über Schwellwert oder nicht?)
    - Eigentliches Ähnlichkeitsmaß: wie viele Nachbarn (Objekte, deren Ähnlichkeit zu gegebenem Objekt über Schwellwert liegt) haben zwei Datenobjekte gemeinsam?
    - Alternatives Ähnlichkeitsmaß: Jaccard-Koeffizient der Mengen der Nachbarn

Theorieaufgaben – CLARANS

- Wie findet man bei CLARANS die Nachbarn?
  - Hinweis: nur Konzept in Vorlesung behandelt, nicht genauer Algorithmus
  - Konzeptionell lässt sich (partitionierendes) Clustering als das Problem beschreiben, die Menge der Clusterzentren zu finden
    - In Vorlesung mit Graph beschrieben, ist aber nicht notwendig, Graph existiert nur implizit
  - „Nachbar“ einer Lösung: eines der bisherigen Clusterzentren durch einen anderen (Nicht-Zentrums-)Punkt austauschen
  - Es gibt viele solche Nachbarlösungen
  - CLARANS testet iterativ jeweils eine zufällige Nachbarlösung
    - Falls besserer Zielfunktionswert, neue Lösung übernommen
    - Falls keine Verbesserung, Zähler erhöht; nach vorgegebener Anzahl erfolgloser Versuche abgebrochen
    - Ganzer Prozess eine vorgegebene Anzahl von Versuchen wiederholt (mit jeweils anderer Initialisierung), beste Lösung ausgewählt

Theorieaufgaben – Silhouetten-Koeffizient (1)

- Wie ist der Silhouetten-Koeffizient definiert und wie interpretiert man ihn?
  - a(o)
    - Durchschnittlicher Abstand von Objekt o zu anderen Objekten desselben Clusters
    - Sollte bei einem guten Clustering klein sein
    - Misst Kompaktheit des Clusters
  - b(o)
    - Minimaler durchschnittlicher Abstand von Objekt o zu Objekten eines anderen Clusters
    - Sollte bei einem guten Clustering groß sein
    - Misst Separierung der Cluster
  - S(o) - Silhouette von Objekt o, kombiniert a(o) und b(o)
  - S(C) - Silhouette eines Clusters (Durchschnitt über Objekte in Cluster)
  - Silhouetten-Koeffizient: Durchschnitt der Silhouette über alle Objekte
  - Exakte Formeln siehe Vorlesung

Theorieaufgaben – Silhouetten-Koeffizient (2)

- Wie ist der Silhouetten-Koeffizient definiert und wie interpretiert man ihn?
  - Silhouetten-Koeffizient bewertet Qualität eines Clustering-Ergebnisses
  - Kann zur Auswahl eines Clustering-Algorithmus oder zur Wahl der Parameter eines Clustering-Algorithmus genutzt werden
  - Wertebereich [-1,1]
    - 1: Distanz von 0 zu Objekten im selben Cluster (z.B. Cluster mit nur einem Objekt)
    - 0: gleicher durchschnittlicher Abstand zu Objekten im selben Cluster wie zu Objekten im nächstgelegenen anderen Cluster
    - -1: Distanz von 0 zu Objekten in anderem Cluster (sehr theoretischer Fall)

Theorieaufgaben – Silhouetten-Koeffizient (3)

- Wie ist der Silhouetten-Koeffizient definiert und wie interpretiert man ihn?
  - Für Verfahren wie DBSCAN ist Silhouetten-Koeffizient nicht geeignet
    - Silhouette geht von kompakten, wohl-separierten Clustern aus, nicht Clustern beliebiger Form (z.B. ein Kreis umgeben von einem Ring)
    - Wie Ausreißer-Punkte des DBSCAN-Ergebnisses für Silhouette behandeln?
      1) Jeweils in neue Ein-Objekt-Cluster stecken: Silhouetten-Koeffizient wird unangemessen gut (diese Cluster haben Silhouette von 1)
      2) Alle in ein Cluster stecken: Silhouetten-Koeffizient wird unangemessen schlecht (Ausreißer-Cluster wird höchstwahrscheinlich nicht zusammenhängend sein)
      3) Ignorieren: unschön, weil ein Teil des Clustering-Ergebnisses vom Koeffizienten nicht bewertet wird
      4) Dem jeweils nächsten Cluster zuordnen: Ausreißer-Punkte haben aber vermutlich hohe Distanz zu restlichen Punkten in Cluster, daher schlechte Silhouette

Theorieaufgaben – CF-Tree / BIRCH

- Berücksichtigt der CF-Tree die vorgegebene Anzahl an Clustern?
  - CF-Tree selbst hat nicht Anzahl der Cluster als Parameter, sondern B (Kapazität innerer Knoten), B‘ (Kapazität von Blattknoten), T (Grenzwert für Größe von Blättern) → Anzahl der Knoten ergibt sich nur implizit daraus
  - Blattknoten sind Elementarcluster
  - CF-Tree gibt uns eine mögliche Clustering-Hierarchie
  - Verschiedene Möglichkeiten, aus CF-Tree ein Clustering mit einer festen Anzahl von Clustern zu machen (Folie 52 der Clustering-Vorlesung)
  - Weiterhin aber auch andere Clustering-Verfahren auf CF-Tree anwendbar, die keine feste Anzahl von Clustern als Eingabeparameter haben
  - Ursprünglicher BIRCH-Algorithmus nutzt agglomeratives hierarchisches Clustering auf den Blattknoten, mit optionalen Schritten zur Bereinigung und Verfeinerung des Clustering-Ergebnisses

Theorieaufgaben – Arten von Outlier Detection

- Welche Arten von Ansätzen gibt es bei Outlier Detection?
  - Ausreißer als Nebenprodukt von Clustering-Algorithmen wie DBSCAN
  - Statistische Verfahren: betrachte z.B. die Punkte als Ausreißer, die 2/3/4 Standardabweichungen vom Mittelwert entfernt sind
  - Distanzbasierte Verfahren
    - z.B. DB(p,D)-Outlier, ermittelt mit Index-basiertem, Nested-Loop oder Zellen-basiertem Algorithmus
    - z.B. Ausreißer-Ranking/-Scoring basierend auf kNN-Distanz
  - Dichtebasierte Verfahren, z.B. LOF
    - Wichtig: auch lokale Verteilung der Distanzen der Nachbarschaft betrachtet (also nicht allein der Dichte-Aspekt ausschlaggebend für die Kategorie, denn Distanz-basierte Verfahren kann man mathematisch zum Teil auch über Dichten formulieren)

Theorieaufgaben – Zellenbasierte Outlier Detection

- Ist es eher gut oder eher schlecht bzw. welche Bedeutung hat es, wenn in der „Hauptzelle“ des zellenbasierten Outlier-Detection-Ansatzes viele Datenpunkte liegen?
  - Generell gut für Zellen-basiert:
    - viele Punkte in L1-Nachbarschaft → keine Ausreißer in Zelle
    - wenige Punkte in L2-Nachbarschaft → nur Ausreißer in Zelle
  - In anderen Fällen müssen wir Tupel der Zelle einzeln inspizieren
  - Wenn also schon viele Punkte in Zelle selbst (mehr, als die DB(p,D)-Definition für Ausreißer erlaubt), ist das aus algorithmischer Sicht gut, weil diese Punkte alle keine Ausreißer sind

Theorieaufgaben – Aggregationsfunktionen (1)

- Wie ist der Zusammenhang zwischen self-maintainable und den restlichen Eigenschaften (distributiv, algebraisch, holistisch)?
  - Self-maintainable beim Löschen: aus altem Aggregatwert und Wert des gelöschten Punktes lässt sich neuer Aggregatwert bestimmen
  - Distributiv → Self-maintainable beim Löschen?
    - Nein, Gegenbeispiel Minimum: Wenn ich einen Punkt entferne, der genau den minimalen Wert hat, kenne ich das neue Minimum nicht (Scan über Daten nötig)
  - Algebraisch → Self-maintainable beim Löschen?
    - Nein, Gegenbeispiel Mittelwert: Wenn ich einen Punkt entferne, kenne ich den neuen Mittelwert nicht (zusätzlich noch Wissen über Anzahl der Datenpunkte nötig)
  - Holistisch → Self-maintainable beim Löschen?
    - Nein, Gegenbeispiel Median: Wenn ich einen Punkt entferne, kenne ich den neuen Median nicht (Scan über Daten nötig)

Theorieaufgaben – Aggregationsfunktionen (2)

- Wie ist der Zusammenhang zwischen self-maintainable und den restlichen Eigenschaften (distributiv, algebraisch, holistisch)?
  - Self-maintainable beim Einfügen: aus altem Aggregatwert und Wert des eingefügten Punktes lässt sich neuer Aggregatwert bestimmen
  - Distributiv → Self-maintainable beim Einfügen?
    - Ja: Neuen Punkt als ein-elementige Menge betrachten, Aggregatwert darauf berechnen und mit dem Aggregatwert für den restlichen Datenbestand kombinieren, um neuen Aggregatwert zu erhalten (funktioniert wegen Distributivität)
  - Algebraisch → Self-maintainable beim Einfügen?
    - Nein, Gegenbeispiel Mittelwert: Wenn ich einen Punkt einfüge, kenne ich den neuen Mittelwert nicht (zusätzlich noch Wissen über Anzahl der Datenpunkte nötig)
  - Holistisch → Self-maintainable beim Einfügen?
    - Nein, Gegenbeispiel Median: Wenn ich einen Punkt einfüge, kenne ich den neuen Median nicht (Scan über Daten nötig)

Theorieaufgaben – Aggregationsfunktionen (3)

- Wie ist der Zusammenhang zwischen self-maintainable und den restlichen Eigenschaften (distributiv, algebraisch, holistisch)?
  - Umkehrung: self-maintainable → distributiv / algebraisch / holistisch?
  - Self-maintainable bei Einfügen
    - Ist zumindest für folgende Beispiele zugleich distributiv: Minimum, Maximum, Summe, Count
    - Formale Definition von Distributivität sollte generell erfüllt sein
    - Wer ein Gegenbeispiel findet: gerne E-Mail schreiben
  - Self-maintainable bei Löschen
    - Klassische Beispiele (Summe, Count) sind zugleich distributiv
    - Unklar (zumindest dem Übungsleiter), ob man Implikation „self-maintainable bei Löschen → distributiv“ beweisen kann, denn Definition von Distributitivität bezieht sich auf Aggregation, während Löschen eine Disaggregation ist (Invertierung)