Den k-Means-Algorithmus verstehen: Mit Stift & Papier und BlueJ

Andres, D.; Joachim, S.; Hennecke, M.
Universität Würzburg

DOI: 10.18420/ibis-02-01-06

Zusammenfassung

Dieser Praxisbeitrag stellt unterrichtliche Aktivitäten vor, um den *k-Means-Algorithmus* einzuführen und ihn anschließend in *BlueJ* zu implementieren. Anhand eines Beispieldatensatzes mit Fundkoordinaten von Pilzen können sich die Schülerinnen und Schüler die Funktionsweise des *k-Means-Algorithmus* selbst erschließen. Der Datensatz ist zweidimensional und klein genug, um den Algorithmus mit Stift und Papier in angemessener Zeit zu erarbeiten. Anschließend kann der Algorithmus in der den Schülerinnen und Schülern sowie Lehrkräften bekannten Umgebung *BlueJ* programmiert und das Ergebnis durch Einlesen einer csv-Datei mit den vorher verwendeten Fundkoordinaten verifiziert werden.

Einleitung

Im Moment wird täglich in der Öffentlichkeit über Künstliche Intelligenz berichtet und heftig diskutiert, ob diese eher eine Chance oder eine Bedrohung für unsere Gesellschaft darstellt. Zur Versachlichung der Diskussion besteht die Möglichkeit zu vermitteln, dass auch KI-Systeme lediglich auf Algorithmen beruhen. Einige modernere Lehrpläne haben die Thematik bereits aufgegriffen und integrieren bekannte Algorithmen aus dem Bereich der Künstlichen Intelligenz in ihre Curricula. Ein Beispiel ist der bayerische LehrplanPLUS für die Jahrgangsstufen 11 bis 13.

Bei KI-Systemen unterscheidet man zwischen dem wissensbasierten und dem datenbasierten Ansatz (auch *maschinelles Lernen*). Teilbereiche des *maschinellen Lernens* sind u.a. das *Reinforcement Learning*, das *Supervised Learning* und das *Unsupervised Learning*. Beim *Unsupervised Learning* ist das Ziel, Gruppen oder Zusammenhänge in nicht *gelabelten* (kategorisierten) Daten zu finden. Dazu müssen Daten zunächst aufbereitet und normalisiert werden, bevor sog. Clusteralgorithmen auf den Datensatz angewendet werden können. Die Ergebnisse dieser Algorithmen müssen im Anschluss interpretiert und evaluiert werden. Ein typischer Clusteringalgorithmus ist der *k-Means-Algorithmus*. Er ist beispielsweise im bayerischen LehrplanPLUS der 13. Jahrgangsstufe sowohl im grundlegenden als auch im erhöhten Anforderungsniveau als Muster für einen Clusteringalgorithmus genannt: „Die Schülerinnen und Schüler […]

beschreiben die Funktionsweise des k-Means-Algorithmus als Beispiel unüberwachten Lernens und implementieren diesen für ein Beispiel.“ (ISB o.J.)

Der *k-Means-Algorithmus* durchläuft nacheinander drei Phasen: Zuerst werden *k* Clusterzentren auf den Koordinaten von *k* zufälligen oder manuell gewählten Datenpunkten initialisiert. Danach erfolgt die Zuweisung aller Datenpunkte zu dem Clusterzentrum, das ihnen am nächsten ist. Anschließend werden Clusterzentren aktualisiert. Dafür werden für jedes Clusterzentrum jeweils komponentenweise die Mittelwerte der Koordinaten aller Datenpunkte eines Clusters berechnet. Diese sind die Koordinaten des aktualisierten Clusterzentrums. Falls sich die Position eines Clusterzentrums geändert hat, werden die Schritte Aktualisierung der Clusterzentren und Zuweisung wiederholt, bis keine Änderung mehr eintritt (vgl. Ertel 2021). Der *k-Means-Algorithmus* kann sowohl mit numerischen als auch kategoriellen Daten beliebiger Dimension ausgeführt werden. Bei Kategorien ist eine andere Metrik zur Abstandsmessung erforderlich, während bei numerischen Daten mit der euklidischen Norm gearbeitet werden kann. Aus Gründen der didaktischen Reduktion beschränkt sich das folgende Unterrichtskonzept auf den zweidimensionalen Fall mit ausschließlich numerischen Daten. Die Merkmale werden deshalb nachfolgend als x und y bezeichnet.

Es existieren verschiedene Möglichkeiten diesen Algorithmus konkret zu implementieren. Unter dem *k-Means-Algorithmus* nach Lloyd (1982) wird meist die Umsetzung des oben genannten Schemas verstanden: Erst nach vollständiger Zuweisung aller Datenpunkte zum nächsten Clustermittelpunkt (oder auch Clusterzentrum) erfolgt eine Aktualisierung der Clusterzentren. Als Abstandsmaß wird der euklidische Abstand verwendet. Im Gegensatz dazu steht z. B. der *k-Means-Algorithmus* nach MacQueen (1967), bei dem die Aktualisierung der Clusterzentren nach jeder Punktzuweisung erfolgt. Damit benötigt der Algorithmus nach MacQueen zwar wesentlich mehr Laufzeit, terminiert im Schnitt allerdings nach weniger Iterationen als der Algorithmus nach Lloyd.

Allgemein kann nicht sichergestellt werden, dass der *k-Means-Algorithmus* überhaupt terminiert oder die tatsächlich beste Clusterung findet. So bildet der *k-Means-Algorithmus* aufgrund der Mittelwertberechnung z. B. bevorzugt konvexe, gleich große Cluster (vgl. Ertel 2021 und Abbildung 2).

Unterrichtskonzept

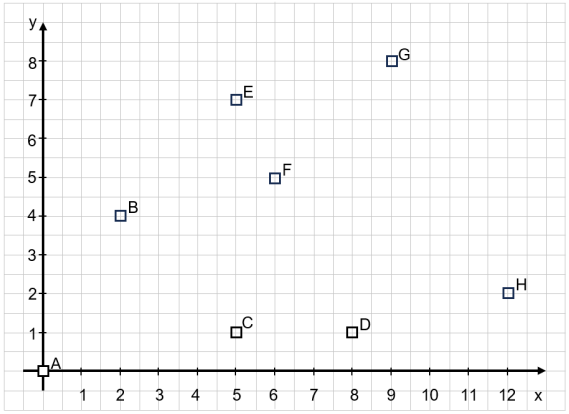

Abbildung 1: Koordinatensystem mit Pilzdatensatz

Wer in der Natur Pilze sammelt, hat vermutlich schon beobachtet, dass Pilze einer Art häufig gruppenweise zusammenstehen. Die Fundkoordinaten der acht Pilze A bis H sind im Koordinatensystem eingetragen. In diesem Beispiel wurden die Koordinaten bewusst so gewählt, dass sich rein optisch verschiedene Pilzgruppen nicht erkennen lassen (vgl. Abbildung 1).

Die unterrichtliche Fragestellung ist zunächst, wie man die Einteilung in eine vorgegebene Anzahl an Gruppen automatisieren könnte. Für dieses Problem wird der *k-Means-Algorithmus* als Lösungsvorschlag vorgestellt, den die Schülerinnen und Schüler nach einer kurzen Einführung selbst manuell auf den kleinen Pilzkoordinatendatensatz anwenden sollen. Dabei werden zwei Zentren bei der Initialisierung vorgegeben. Die Schülerinnen und Schüler erhalten so bei richtiger Durchführung des Algorithmus alle das gleiche Ergebnis und gewinnen Vertrauen in ihre Fähigkeiten.

Der bayerische LehrplanPLUS fordert zudem eine konkrete Implementierung des *k-Means-Algorithmus*. Eine Sprache für die Umsetzung ist nicht vorgegeben. Eine kompakte Implementierung des *k-Means-Algorithmus* über vorgegebene Methodenaufrufe, die beispielsweise das Berechnen des Abstands zwischen zwei Punkten oder das Finden des Minimums innerhalb einer Liste ermöglichen, ist z. B. in den Programmiersprachen R oder Python möglich. Diese beiden Sprachen haben zudem den Vorteil, dass eine einfache Visualisierung der Daten und Clusterergebnisse mit wenigen Befehlen innerhalb der Sprache für die Schülerinnen und Schüler zur Verfügung steht und somit eine visuelle Unterstützung beim Debuggen des Codes existiert. Allerdings ist die Einführung von R – zumindest in Bayern – im Curriculum der vorhergehenden Jahrgangsstufen nicht vorgesehen. Zudem arbeiten viele Lehrkräfte seit Beginn der textuellen Programmierung mit Java. Eine händische Umsetzung des *k-Means-Algorithmus* mit Java ist ebenfalls recht kompakt möglich, allerdings stellt Java keine ähnlich einfachen Befehle zur Visualisierung zur Verfügung, die Schülerinnen und Schüler schnell eigenständig erstellen und verwenden können.

Deshalb ist das zweite in diesem Beitrag vorgestellte Unterrichtsmaterial ein *BlueJ*-Projekt, in dem eine bereits implementierte Visualisierung existiert. Die Schülerinnen und Schüler können ihren Algorithmus in Lücken einer vorgegeben Klasse eintragen; dann lassen sich die Auswirkungen des von ihnen generierten Codes über wenige Klicks optisch visualisiert in einem zweidimensionalen Koordinatensystem betrachten (vgl. Abbildung 2). Neben der farblich angezeigten Clusterung werden auch die Koordinaten der Zentrumspunkte in der Visualisierung angegeben.

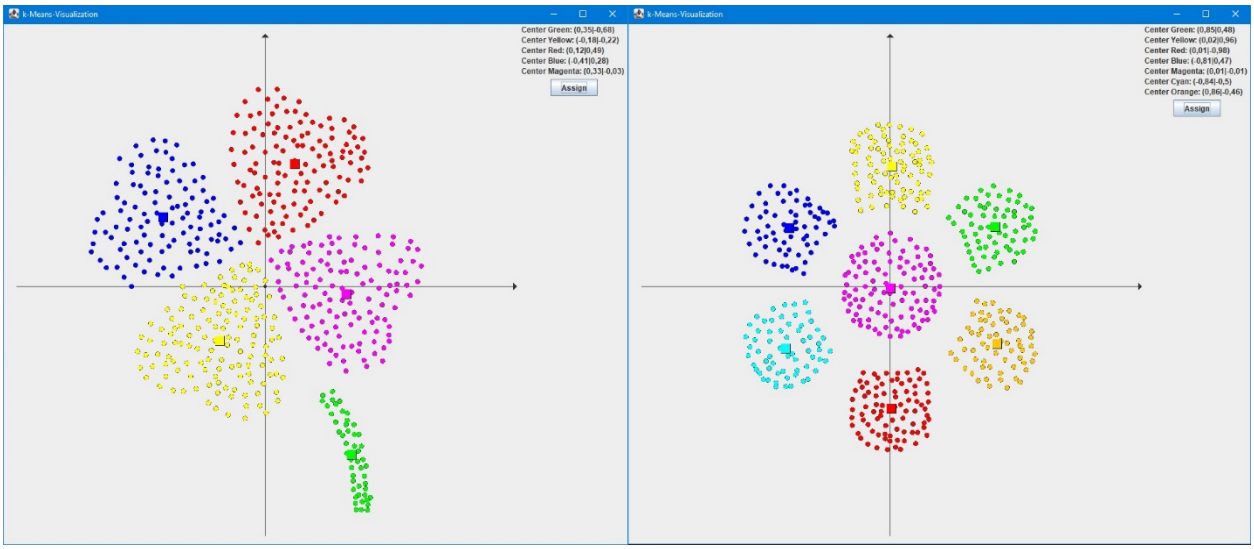

Abbildung 2: Oberfläche der Visualisierung des fertigen Clusterings der Datensätze „kleeblatt.csv“ und „blume.csv“

Material 1: *k-Means-Algorithmus* mit Stift und Papier

Nach einer kurzen Einführung in den Algorithmus führen diesen die Schülerinnen und Schüler mit Hilfe von Arbeitsblättern, verfügbar unter https://go.uniwue.de/ki, selbst durch. Beim ersten Arbeitsauftrag sind die Fundkoordinaten der Pilze bereits in ein Koordinatensystem eingetragen. Die Schülerinnen und Schüler müssen nun immer wieder Pilze dem nächstgelegenen Clusterzentrum zuweisen. Dazu sind verschiedene Möglichkeiten denkbar: So können die Abstände entweder mit dem Lineal gemessen, mit Hilfe des Satzes von Pythagoras berechnet oder dem Zirkel verglichen werden. Um einen möglichst guten Übergang zur Implementation zu erhalten, wurde im **Arbeitsauftrag 1** die **Zuweisung zu einem Clusterzentrum durch Abstandsmessung mit dem Lineal** gewählt. Dabei können die Ergebnisse der Schritte des Algorithmus in Tabellen festgehalten und ausgewertet werden. Mit der gewählten Methode gehen allerdings Messungenauigkeiten einher, welche später nach Überprüfung der Ergebnisse durch das *BluJ*-Projekt thematisiert werden müssen.

Ein Data Scientist analysiert große Mengen an Daten, um bspw. Muster zu erkennen. Eine Möglichkeit der Visualisierung von Mustern in Daten lernen die Schülerinnen und Schüler im **Arbeitsauftrag 2** kennen. Dort zeichnen sie die **Clusterhüllen für den Pilzdatensatz** ein.

Die Zentren bei der Initialisierung sind so gewählt, dass der Algorithmus die gleiche Aufteilung der Pilze in zwei Gruppen erhält, wie beim Supervised Learning mit den KI-Pilzen im Experimentiersatz „Künstliche Intelligenz“ der MEKRUPHY GMBH. Diese Aufteilung entspricht derjenigen, die durch die Label „giftig“ und „essbar“ bereits beim Supervised Learning vorgegeben war. Falls diese den Schülerinnen und Schülern bekannt ist, sollte im Unterricht die Frage thematisiert werden, ob diese Aufteilung auch bei anders gewählten Startzentren durch den *k-Means-Algorithmus* erreicht wird. Die Prognosen können später im *BlueJ*-Projekt überprüft

werden. Dies kann der Einstieg in die Problemstellung sein, dass der *k-Means-Algorithmus* bei unterschiedlichen Startzentren nicht immer zu den gleichen Clustern führt.

Material 2: *k-Means-Algorithmus* mit *BlueJ*

Um den *k-Means-Algorithmus* implementieren zu können, muss zuerst eine geeignete Umgebung bereitgestellt werden. Die Schülerinnen und Schüler benötigen zunächst Beispieldatenpunkte, auf die sie während des Algorithmus z. B. zur Mittelwertberechnung zugreifen können. Dafür müssen im Vorhinein Entscheidungen für die Speicherung der einzelnen Datenpunkte der Beispieldaten getroffen werden. Da dies im *BlueJ*-Projekt bereits vorgegeben ist, müssen die Schülerinnen und Schüler zunächst durch Einstiegsaufgaben an die Projektstruktur herangeführt werden. In **Arbeitsauftrag 3** diskutieren die Schülerinnen und Schüler deshalb verschiedene Modellierungsvorschläge für die Datentypen der Klasse POINT, die Speicherung der Clusterzuweisung sowie die Datenstruktur, die die Eingabedaten hält. Das *BlueJ*-Projekt speichert die Punkte der Eingabedaten als Objekte der Klasse POINT, welche wiederum in einem Array abgelegt werden.

Die Datenstruktur des Arrays wird somit als Vorwissen vorausgesetzt. In Bayern ist diese den Schülerinnen und Schülern je nach Zweigwahl seit der 10. oder 11. Jahrgangsstufe bekannt.

Nach dem Bearbeiten von Arbeitsauftrag 3 können die Schülerinnen und Schüler sich bereits im Projekt zurechtfinden und mit der Implementierung beginnen. Zur Unterstützung der Implementierung kann zusätzlich mit **Arbeitsauftrag 4** Pseudocode der Methoden mit Lücken zur Verfügung gestellt werden. Konkrete Hilfe bei der Implementierung etwa von Wurzelausdrücken oder der Umgang mit einem Random-Objekt, kann **Arbeitsauftrag 5** liefern. Das Projekt steht unter https://go.uniwue.de/ki in zwei Versionen zum Download zur Verfügung, einmal als fertiges, lauffähiges Projekt und einmal als Vorlage für Schülerinnen und Schüler mit Lücken in den Methoden der Klasse KMEANS.

Projektstruktur

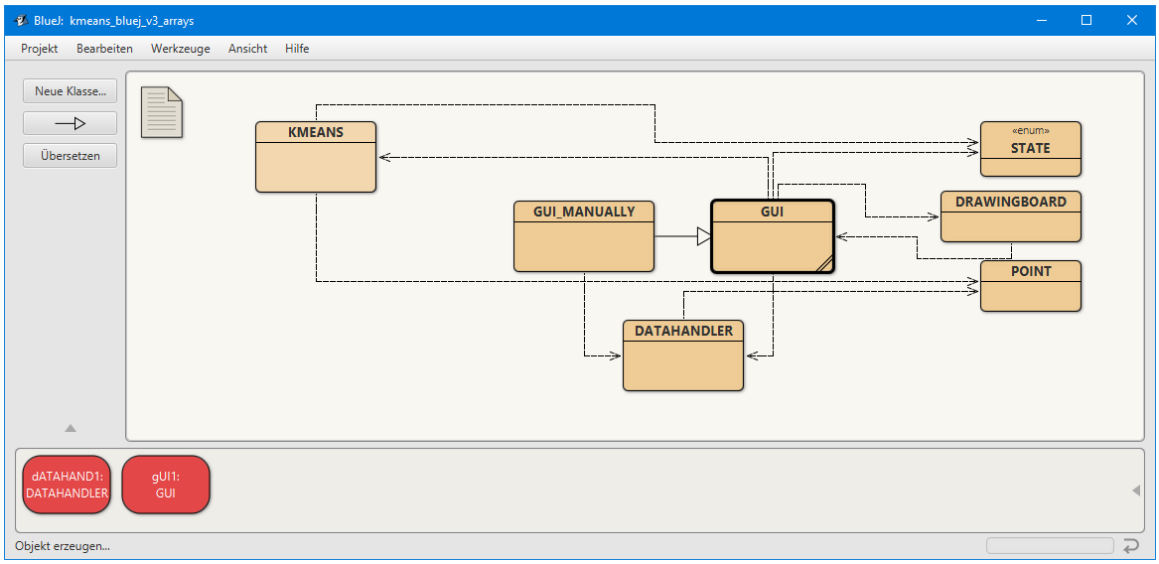

Abbildung 3 Projektstruktur auf BlueJ

Das Projekt besteht aus den sieben Klassen POINT, DATAHANDLER, STATE, GUI, GUI_MANUALLY, DRAWINGBOARD und KMEANS (vgl. Abbildung 3), von welchen lediglich KMEANS, GUI und evtl. DATAHANDLER und GUI_MANUALLY für den Arbeitsablauf der Schülerinnen und Schüler relevant sind.

Beschreibung des *BlueJ*-Projekts

Grundablauf des Programms:

Bei einer vollständig implementierten Klasse KMEANS, kann zum schnellen Testen der Methoden der Konstruktor von GUI aufgerufen werden. Dieser erzeugt einen Datensatz mit vier klar erkennbaren Clustern. Dieser wird in einem neuen Fenster angezeigt. Mit einer Schaltfläche (Button) kann der implementierte *k-Means-Algorithmus* schrittweise ausgeführt werden. Dabei werden beim Standardkonstruktor von GUI stets vier Clusterzentren initialisiert. Diese erscheinen im Koordinatensystem als farbige Quadrate. Nach der Initialisierung der Zentren benennt sich die Schaltfläche in *Assign* um und führt auf Klick die Methode `assign()` der Klasse KMEANS aus. Die Datenpunkte werden so dem ihnen nächsten Zentrum zugewiesen und im Koordinatensystem in der entsprechenden Zentrumsfarbe gefärbt. Hat sich die Zuweisung zu den Clusterzentren geändert, kann mit Klick auf die Schaltfläche *Update* die Methode `update()` ausgeführt und die neuen Koordinaten für die Clusterzentren berechnet werden. Die Zentren im Koordinatensystem verschieben sich auf diese neuen Koordinaten und die Schaltfläche wechselt zurück auf den Schritt *Assign*. So können die Methoden `assign()` und `update()` abwechselnd ausgeführt werden, bis keine Änderung der Clusterzentren mehr sichtbar ist. Ausschnitte eines Durchlaufes des *k-Means-Algorithmus* zeigt Abbildung 4.

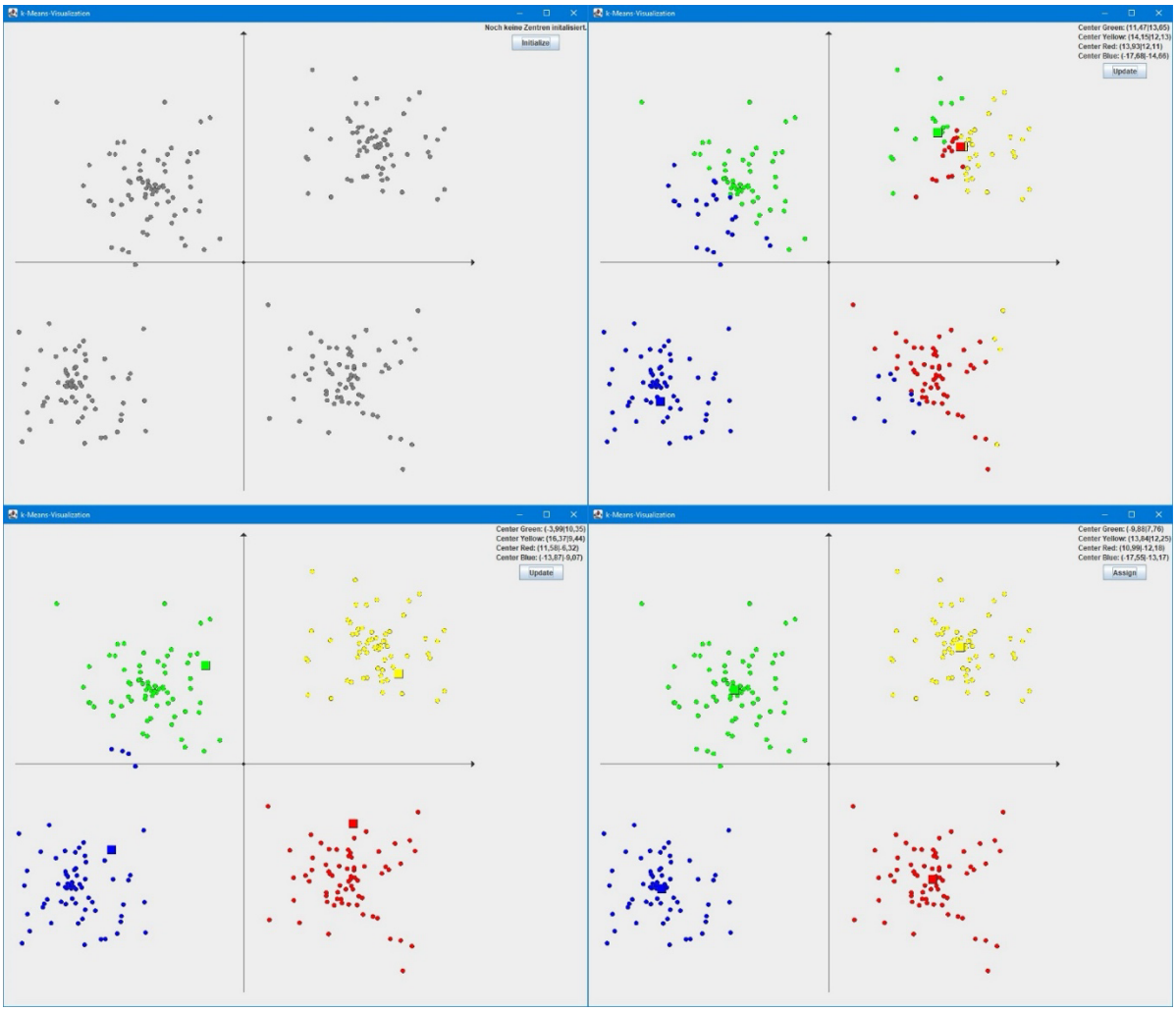

Abbildung 4 Einige Zwischenstände der Visualisierung des k-Means-Algorithmus

Möchte man den Datensatz oder die Anzahl der Cluster variieren, muss der zweite Konstruktor der Klasse GUI verwendet werden. Diesem muss ein zuvor erstelltes Objekt der Klasse DATAHANDLER sowie die gewünschte Clusterzahl *k* übergeben werden. DATAHANDLER verfügt über verschiedene Methoden, um Beispieldatensätze zu erstellen.

Schlussendlich gibt es die Möglichkeit zur Überprüfung der Implementierung die Zentren nicht über die Methode `initialize()` von KMEANS, sondern händisch per Eingabe der Koordinaten zu initialisieren. Dafür initialisiert man ein Objekt der Klasse GUI_MANUALLY, welchem man immer ein DATAHANDLER-Objekt sowie die gewünschte Clusterzahl *k* übergeben muss. Um an Material 1 anzuschließen können die Koordinaten der Pilze zunächst mittels der Methode `loadData` mit dem Parameter „pilze.csv“ von DATAHANDLER in das Programm geladen werden und anschließend mit dem Konstruktor von GUI_MANUALLY die zwei Startkoordinaten per Hand eingegeben werden. Dies widerspricht zwar der eigentlichen Implementierung des *k-Means-Algorithmus* nach Lloyd, wird aber zur Überprüfung der im Vorhinein ausgeführten manuellen Berechnung mit vorgegebenen Zentren benötigt. So ergibt sich mit dem Pilzdatensatz

und den analog zu Arbeitsauftrag 1a gewählten Startzentren $Z_{grün}$ (5|1) und Z_{gelb} (6|5) eine finale Clusterung mit Zentren $Z_{grün}$ (3,75|1,5) und Z_{gelb} (8|5,5), welche mit der in Material 1 von Hand ermittelten Clusterung im Rahmen der Messgenauigkeit übereinstimmt.

Erwartetes Schülerhandeln

Aufgabe der Schülerinnen und Schülern ist es, den Code des *k-Means-Algorithmus* in der Klasse KMEANS zu ergänzen. Zur Übersichtlichkeit sind dafür bereits drei Methoden `initialize()`, `assign()` und `update()` (in der *BlueJ* Vorlage mit leeren Methodenkörpern) vorgegeben. Nach Implementierung der Methoden, können die Schülerinnen und Schüler die Auswirkungen ihres eigenen Codes grafisch durch den Aufruf des Konstruktors der Klasse GUI überprüfen.

Beschreibung der Klassen

Im Folgenden werden nur die Konstruktoren, Attribute und Methoden aufgeführt, die für die Implementierung des *k-Means-Algorithmus* von Bedeutung sind. Die Beschreibung der im Hintergrund für die Visualisierung zuständigen Klassen ist der Dokumentation des *BlueJ*-Projektes zu entnehmen.

Die Klasse **POINT** repräsentiert einen Datenpunkt oder ein Clusterzentrum und stellt somit das Grundgerüst der Datenstruktur dar.

- Attribute
  - Hält die x- und y-Koordinate des Punktes als Gleitkommazahl.
- Methoden
  - Setter & Getter
  - `getDistance(POINT p) : double` berechnet den euklidischen Abstand zwischen diesem Punkt und einem anderen Punkt p.

DATAHANDLER erzeugt die Daten, die mit dem *k-Means-Algorithmus* geclustert werden sollen.

- Attribute
  - Die Liste `pointList`, welche alle Punkte des Datensatzes als POINT-Referenzen enthält.
- Methoden
  - `getRandomData(int countPoint, int max, int min)` bewirkt das Anlegen eines Datensatzes mit `countPoint`-vielen zufälligen Datenpunkten zwischen `max` und `min`.
  - `getRandomCluster(int countCluster, int countPoint, int max, int min)` bewirkt das Anlegen eines Datensatzes mit deutlich erkennbaren Clustern.
  - `loadData(String fileName)` lädt einen in dem Ordner „Beispieldaten“ angelegten Datensatz. Dafür muss der exakte Name des gewünschten Datensatzes z. B. „kleeblatt.csv“ in das Eingabefeld eingetragen werden.
  - `getPointlist() : POINT[]` wandelt die Liste pointList in ein Array um und gibt es zurück. So können die Schülerinnen und Schüler in KMEANS mit der ihnen vertrauten Datenstruktur des Arrays arbeiten.
- Konstruktoren

    - `DATAHANDLER()` erstellt standardmäßig eine zufällige Clusterung mit vier Clustern und 300 Datenpunkten.

In **KMEANS** wird der *k-Means-Algorithmus* nach Lloyd umgesetzt. In der *BlueJ* Vorlage sind nur die dafür benötigten Methodenköpfe gegeben, die Schülerinnen und Schüler sollen hier ihre eigene Implementation umsetzen.

- Attribute
    - Das Array `POINT[] data` speichert die einzelnen Datenpunkte. Wird KMEANS von GUI übergeben.
    - Clusterzahl `int k`. Wird KMEANS von GUI übergeben.
    - Die Liste `POINT[] centers` hält die aktuellen Zentren des *k-Means-Algorithmus*.
    - Das Array `int[] assignment` speichert die Zuordnung der Datenpunkte zu den Clusterzentren.
- Methoden
    - `initialize()` initalisiert *k* Clusterzentren an zufälligen Datenpunkten.
    - `assign()` weist die Datenpunkte ihrem nächsten Zentrum zu.
    - `update()` aktualisiert die Zentren.
    - `contains(POINT[] array, POINT point) : boolean` überprüft, ob der Punkt point bereits im Array array vorhanden ist. Wird zur Initialisierung von *k* verschiedenen Clusterzentren benötigt.

DRAWINGBOARD dient zur Umsetzung des Koordinatensystems. Die Klasse hat Methoden zum Zeichnen der Koordinatenachsen sowie der Datenpunkte und Clusterzentren.

STATE dient als Enumeration lediglich zur Verwaltung der Zustände der Visualisierung.

Die Klasse **GUI** verwaltet die Visualisierung.

- Konstruktoren
    - `GUI()` erstellt einen zufälligen, klar in vier Cluster einteilbaren Datensatz und verwendet den *k-Means-Algorithmus* mit Clusterzahl vier. Ermöglicht schnelle Überprüfung der Implementierung.
    - `GUI(DATAHANDLER datahandler, int k)` ermöglicht das Erstellen einer Visualisierung mittels selbst modifiziertem Datensatz und individuell ausgewählter Clusterzahl *k*. Das Objekt der Klasse DATAHANDLER muss hierbei in *BlueJ* im Vorhinein erstellt und dem Konstruktor übergeben werden.

GUI_MANUALLY verwaltet die gleiche Visualisierung wie GUI, ermöglicht aber die händische Initialisierung der Zentren.

- Konstruktoren
    - `GUI_MANUALLY(DATAHANDLER datahandler, int k)` ermöglicht das manuelle Eingeben der Startpunkte der Clusterzentren. Analog zu oben muss hier ebenfalls ein Objekt der Klasse DATAHANDLER vorher manuell erstellt werden.

Diskussion der Projektstruktur

Die Implementierung der Klassen POINT und KMEANS des *BlueJ*-Projektes sind angelehnt an die bayerischen illustrierenden Prüfungsaufgaben für *k-Means-Algorithmus*. Auch wenn bei

dem hier lediglich zweidimensionalen Einsatz eine Modellierung der Punkte als zweidimensionales Array denkbar wäre, wurde sich analog zu den illustrierenden Prüfungsaufgaben (gA) für das Einführen einer eigenen Klasse POINT entschieden. Dies hat zum Vorteil, dass der Algorithmus auch für anders strukturierte oder mehrdimensionale Daten mit lediglich kleinen Anpassungen des *BlueJ*-Projektes umgesetzt werden kann. Auch wird durch die Klasse POINT der Umgang mit mehrdimensionalen Strukturen vermieden.

Konkrete Implementierung des *k-Means-Algorithmus*

Einige Feinheiten der Implementierung müssen von der Lehrkraft entschieden werden. Zunächst muss im ersten Schritt der Initialisierung geklärt werden, ob *k* verschiedene Punkte als Zentren initialisieren werden sollen oder zwei Zentren auch am selben Punkt starten können. Im Verlauf des Algorithmus erfahren zwei Zentren, die am selben Punkt starten, immer dieselben Änderungen und agieren somit als wären sie ein einziges Zentrum. Somit hätte man statt zwei Zentren effektiv nur eines initialisiert. Verwendet man Eingabedaten mit mehreren hundert Datenpunkten tritt das Problem mit hoher Wahrscheinlichkeit nicht auf, bei kleineren Datensätzen wie dem Pilzdatensatz ist es aber wahrscheinlich, dass die Schülerinnen und Schüler auf dieses Problem stoßen. Möchte man bei der Implementierung sicherstellen, dass *k* verschiedene Datenpunkte als Startzentren gewählt werden, kann die Methode `contains(POINT[] array, POINT point) : boolean` der Klasse KMENAS von den Schülerinnen und Schülern genutzt oder auch selbst implementiert werden.

Eine Möglichkeit zur Differenzierung ergibt sich mit der Umsetzung der Methode `update()`. Diese kann mit einer Laufzeit von *O(nk)* analog zum beschriebenen Pseudocode in **Arbeitsauftrag 4** umgesetzt werden. Allerdings gibt es auch effizientere Implementierungen, welche besonders bei großen Datenmengen von Vorteil sein können. Eine weitere Implementierung mit Laufzeit *O(n+k)* wird im *BlueJ*-Projekt als Variante 2 mitgeliefert. Eine solche effizientere Implementierung zu finden, kann für motiviertere Schülerinnen und Schüler eine gute Differenzierungsaufgabe sein.

Obwohl das *BlueJ*-Projekt auf die Umsetzung des *k-Means-Algorithmus* nach Lloyd ausgelegt ist, ist prinzipiell auch eine Implementierung nach MacQueen möglich. In diesem Fall sind allerdings die Arbeitsblätter daran anzupassen.

Fazit

Der durch die Pilze gegebene Realitätsbezug macht unmittelbar klar, wie wichtig eine korrekte Einteilung von Daten sein kann. Dennoch werden bei dem vorgestellten Unterrichtskonzept schnell zwei Probleme des *k-Means-Algorithmus* deutlich: Durch Experimentieren mit den im *BlueJ*-Projekt mitgelieferten Beispieldatensätzen oder durch ungünstige Startzentren (vgl. Abbildung 5) wird schnell klar, dass der *k-Means Algorithmus* auch bei klar erkennbaren Clustern nicht immer zu den erwarteten Clustern führt. Insbesondere muss die gewünschte Anzahl der

Cluster dem *k-Means-Algorithmus* bereits übergeben werden, das optimale *k* wird einfach durch Betrachten der Daten in einem Koordinatensystem ermittelt. Der Nutzen des vorgestellten Verfahrens kann von den Schülerinnen und Schülern also fundiert kritisiert und hinterfragt werden. Insbesondere bietet sich eine Möglichkeit zur Diskussion über die Qualität von Daten, den Einsatz von Clusteringalgorithmen und die Vertrauenswürdigkeit von KI-Systemen, wie etwa einer Smartphone-App zur Pilzbestimmung. Dies kann Motivation sein, andere Algorithmen zur Klassifikation der Pilze zu betrachten. Weiter kann dieser Hinweis motivieren reale Anwendungsbeispiele für den *k-Means-Algorithmus* zu recherchieren. Der *k-Means-Algorithmus* ist in realen Anwendungen von großer Bedeutung. Wird er beispielsweise auf Produktionsdaten angewendet, kann er die Erkennung von Mustern und Anomalien ermöglichen, die zur Ineffizienz in der Produktion führen. Besonders, weil der *k-Means-Algorithmus* mit ungelabelten Daten arbeitet, ist er ein wichtiges Werkzeug Muster in derartigen Daten korrekt zu interpretieren.

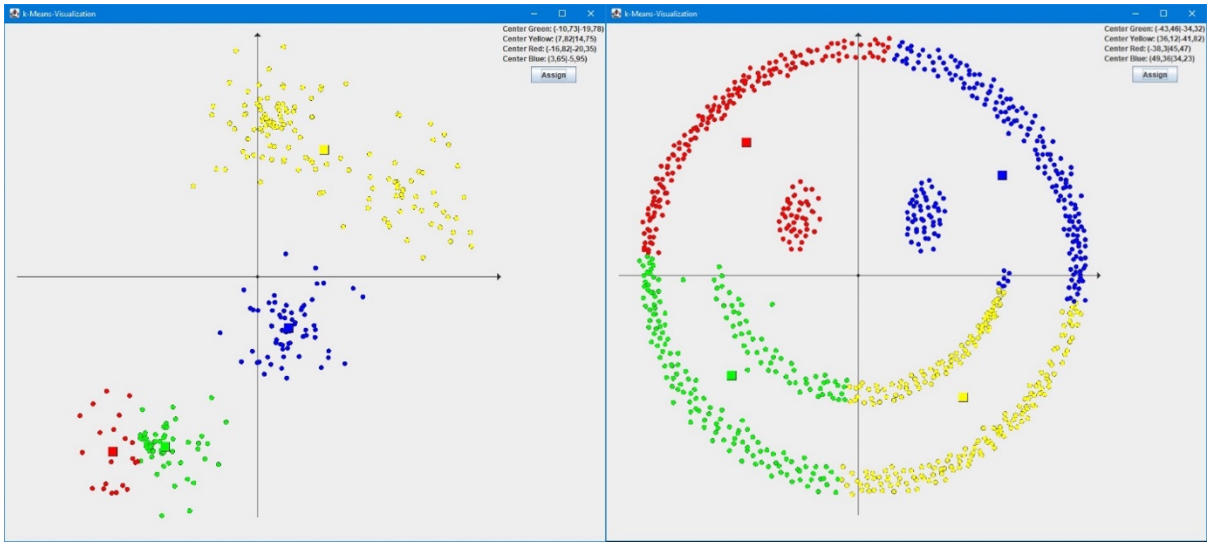

Abbildung 5 Ergebnis bei klar erkennbaren vier Clustern und ungünstiger Startzentrenverteilung sowie das Ergebnis des Clusterings des Beispieldatensatzes „smiley.csv“

Zusammengenommen können diese Probleme zu einer weiteren Unterrichtssequenz motivieren, in welcher evtl. durch einen Exkurs zu alternativen Clusteringverfahren wie dem *Hierarchischen Clustering* oder durch die Vorstellung von Validierungsverfahren für verschiedene Clusterungen die Ergebnisse des *k-Means-Algorithmus* in Abhängigkeit von *k* genauer betrachtet werden.

Quellen

Alle Webseiten/Links wurden zuletzt geprüft am 07.02.2024.

Ertel, Wolfgang (2021): Grundkurs Künstliche Intelligenz: Eine praxisorientierte Einführung, Springer, Wiesbaden. [5. Auflage]

Heesen, Bernd (2023): Künstliche Intelligenz und Machine Learning mit R: Anwendungen im Bereich Business Analytics, Springer, Wiesbaden. [1. Auflage]

Illustrierende Prüfungsaufgaben für die schriftliche Abiturprüfung in Bayern ab 2026

- Informatik grundlegendes Anforderungsniveau (gA)

https://www.isb.bayern.de/fileadmin/user_upload/Gymnasium/IlluPA/Inf/IlluPA_Informatik_Beispielaufgaben_gA.pdf

Lloyd, Stuart P. (1982): Least squares quantization in PCM, IEEE Transactions on Information Theory, 28 (2): 129–137

MacQueen, J. (1967): Some Methods for classification and Analysis of Multivariate Observations. In: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. Band 1. University of California Press, S. 281–297

Staatsministerium für Schulqualität und Bildungsforschung München: LehrplanPLUS Gymnasium Bayern, Informatik 13 und spät beginnende Informatik 13 (grundlegendes Anforderungsniveau)

https://www.lehrplanplus.bayern.de/fachlehrplan/gymnasium/13/informatik/grundlegend

Staatsministerium für Schulqualität und Bildungsforschung München: LehrplanPLUS Gymnasium Bayern, Informatik 13 (erhöhtes Anforderungsniveau)

https://www.lehrplanplus.bayern.de/fachlehrplan/gymnasium/13/informatik/erhoeht

Materialien

Die Arbeitsblätter und *BlueJ*-Projekte zum *k-Means-Algorithmus* stehen zum Download auf der Homepage der Didaktik der Informatik der Julius-Maximilians-Universität Würzburg zur Verfügung. Sie können unter dem Direktlink https://go.uniwue.de/ki kostenlos heruntergeladen werden.

Lizenz

Kontakt

Daniela Andres
Dr. Silvia Joachim
Prof. Dr. Martin Hennecke

Didaktik der Informatik, Julius-Maximilians-Universität Würzburg, https://go.uniwue.de/ddi

E-Mail: {daniela.andres,silvia.joachim,martin.hennecke}@uni-wuerzburg.de