

# 7. Grupiranje podataka primjenom algoritma

## 7.1 Cilj Vježbe

Upoznati se s problemom grupiranja podataka. Grupiranje podataka primjenom algoritma  $K$  srednjih vrijednosti. Primjena grupiranja podataka na kvantizaciju slike u boji.

## 7.2 Teorijska pozadina

Nenadzirano učenje (engl. *unsupervised learning*) je vrsta strojnog učenja gdje je cilj izvući zaključke o raspoloživim podatkovnim primjerima bez odgovarajuće izlazne veličine ili oznake. Primjenom odgovarajućih algoritama nastoji se provesti analiza podataka u skladu sa sljedećim pitanjima:

- Kako možemo grupirati slične podatkovne primjere zajedno kako bismo razumjeli njihove odnose i obrasce?
- Kako možemo smanjiti dimenziju visokodimenzionalnih podataka kako bismo ih lakše vizualizirali, analizirali ili pohranili?
- Kako možemo identificirati podatkovne primjere koje su neuobičajeni ili nisu u skladu s općim obrascima u podacima?

U ovoj vježbi studenti se upoznaju s problemom grupiranja (engl. *clustering*) podatkovnih primjera ili klaster analizom. Ovo je postupak kojim se podatkovni primjeri grupiraju u grupe na način da su slični podatkovni primjeri u istoj grupi, a različiti primjeri u različitim grupama. Cilj je pronaći skrivene grupe u podacima tj. optimalno podijeliti primjere u grupe. Podatkovni primjeri su neoznačeni - jednom kada se pronađu grupe u podacima moguće je nove podatkovne primjere dodijeliti odgovarajućoj grupi. Primjene su raznolike: segmentacija korisničkog ponašanja kako bi se npr. identificirali kupci sa sličnim interesima i navikama kupovine, detektiranje anomalija (npr. botova), grupiranje dokumenata koji se bave sličnom tematikom i sl.

Neka se podatkovni primjer sastoji  $m$  ulaznih veličina  $X = [x_1, x_2, \dots, x_m]$ . Stoga, svaki primjer je vektor vrijednosti ulaznih veličina i zapisuje se u obliku:

$$x^{(i)} = [x_1^{(i)}, x_2^{(i)}, \dots, x_m^{(i)}]^\top. \quad (7.1)$$

## 62 **Poglavlje 7. Grupiranje podataka primjenom algoritma K srednjih vrijednosti.**

Podatkovni skup koji se sastoji od takvih  $n$  raspoloživih podatkovnih primjera koji nisu označeni može se zapisati u matričnom obliku:

$$X = \begin{bmatrix} x_1^{(1)}, x_2^{(1)} \dots x_m^{(1)} \\ x_1^{(2)}, x_2^{(2)} \dots x_m^{(2)} \\ \vdots \\ x_1^{(n)}, x_2^{(n)} \dots x_m^{(n)} \end{bmatrix} \quad (7.2)$$

U ovoj vježbi koriste se dva popularna algoritma grupiranja: algoritam  $K$  srednjih vrijednosti i hijerarhijsko grupiranje kako bi se pronašle grupe u podatkovnom skupu.

### 7.2.1 **Grupiranje primjenom algoritma $K$ srednjih vrijednosti**

Algoritam  $K$  srednjih vrijednosti (engl. *K-means algorithm*) je jednostavan i često korišten algoritam grupiranja koji kao rezultat daje particioniranje podatkovnog skupa u  $K$  grupa. Svaka grupa predstavlja se  $m$  dimenzionalnim vektorom  $c^{(k)}$ ,  $k = 1, \dots, K$  koji se nazivaju centri. Nadalje, svaki podatkovni primjer pripada samo jednoj grupi. Određivanje optimalnih vrijednosti centara  $c^{(k)}$  temelji se na minimizaciji kriterijske funkcije:

$$J(c^{(k)}, k = 1, \dots, K; X) = \sum_{i=1}^n \sum_{k=1}^K b_k^{(i)} \|x^{(i)} - c^{(k)}\|^2 \quad (7.3)$$

pri čemu je  $b_k^{(i)}$  jednak 1 ili 0 ovisno pripada li podatak  $x^{(i)}$  centru  $c^{(k)}$ . Kriterijska funkcija 7.3 za dane podatkovne primjere i centre predstavlja sumu kvadrata udaljenosti svakog podatkovnog primjera do njemu najbližeg centra.

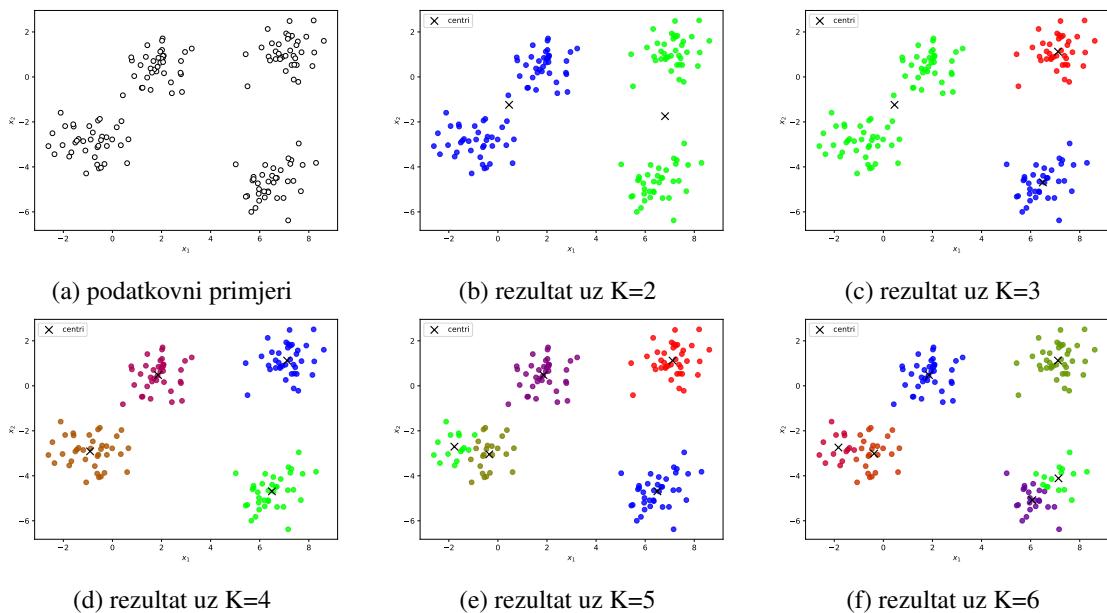
Minimizacija kriterijske funkcije provodi se iterativnim numeričkim postupkom. Pseudokod algoritma  $K$  srednjih vrijednosti je dan u tablici ?? Korisnik mora specificirati željeni broj centara odnosno grupa  $K$ . Nakon inicijalizacije centara naizmjenično se izvode dva koraka. Najprije se pridružuje svaki podatkovnog primjer najbližem centru, a zatim se preračunavaju vrijednosti centara – nova vrijednost svakog centra je centroid njemu dodijeljenih podatkovnih primjera. Postupak se ponavlja sve dok nije dostignut kriterij zaustavljanja. Kriterij zaustavljanja može biti unaprijed definiran broj iteracija ili tolerancija na normu razlike vrijednosti centara iz dvije susjedne iteracije. Algoritam uvijek konvergira. Međutim, nema jamstva da će algoritam uvijek pronaći globalno optimalno rješenje, tj. može pronaći lokalno optimalno rješenje. Iz tog razloga se uvijek preporuča nekoliko puta pokrenuti algoritam  $K$  srednjih vrijednosti uz različite vrijednosti početnih centara.

Postoje razni načini kako odabrati početne vrijednosti centara. Iako se centri mogu inicijalizirati potpuno nasumično, jedan od logičnih načina je nasumičan odabir  $K$  podatkovnih primjera kao početnih centara. Drugi pristup inicijalizacije centara je algoritam k-means++ koji povećava šanse da algoritam  $K$  srednjih vrijednosti pronađe optimalno rješenje. Osnovna ideja k-means++ je nasumično odabrati prvi centar, a zatim svaki sljedeći centar odabrati tako da je što dalje od postojećih centara. Na slici 7.1a prikazan je dvodimenzionalni podatkovni skup od 150 podatkovnih primjera. Na slici 7.1b - 7.1f prikazani su rezultati primjene algoritma  $K$  srednjih vrijednosti uz različit broj  $K$ . Primjeri koje je algoritam svrstao u iste grupe označeni su istom bojom.

Rezultat grupiranja uvelike ovisi o parametru  $K$  koji definira broj grupa i kojeg je potrebno unaprijed odrediti. Taj broj  $K$  treba odgovarati skrivenom broju grupa u podatkovnom skupu za

**Metoda 2** Pseudokod algoritma K-srednjih vrijednosti

- 1: odredi broj centara  $K$ . Odredi početne vrijednosti centara klastera  $c^{(k)}, k = 1, \dots, K$ .
- 2: Sve dok nije zadovoljen kriterij zaustavljanja ponavlja:
- 3: Za svaki podatak odredi kojem centru (grupi) pripada:
 
$$b_k^{(i)} = \begin{cases} 1 & \text{ako je } \|x^{(i)} - c^{(k)}\| \text{ najmanja od svih udaljenosti za } k = 1, \dots, K \\ 0 & \text{u suprotnom} \end{cases}$$
- 4: Osvježi vrijednosti svih centara :  $k = 1, \dots, K$ :  $c^{(k)} \leftarrow \frac{\sum_{i=1}^n b_k^{(i)} x^{(i)}}{\sum_{i=1}^n b_k^{(i)}}$



Slika 7.1: Primjena algoritma K srednjih vrijednosti na podatkovni skup s dvije ulazne veličine.

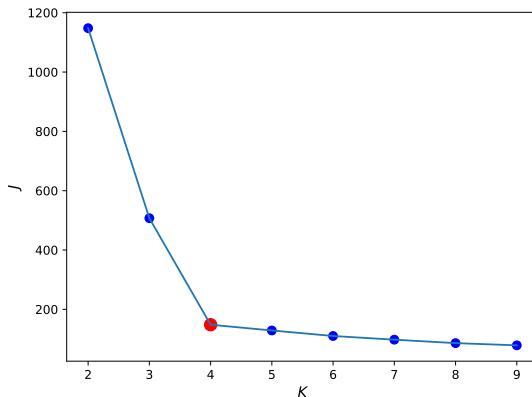
optimalnu particiju podataka, ali nam taj broj najčešće nije poznat. Postoje različiti načini kako procijeniti optimalnu vrijednost parametra  $K$ . Jedan od vizualnih načina određivanja optimalnog broja  $K$  je korištenje lakat metode (engl. *elbow method*). Grafički je potrebno prikazati vrijednost kriterijske funkcije 7.3 s obzirom na promjenu vrijednosti parametra  $K$ . Kako se povećava vrijednost  $K$ ,  $J$  će padati. Na tom grafu potrebno je pronaći vrijednost  $K$  za koju vrijednost kriterijske naglo opadne, a zatim se postepeno smanjuje (lakat). Ova vrijednost upućuje da smo identificirali optimalni broj grupa, a da daljnje povećanje broja  $K$  uzrokuje dijeljenje skrivenih grupa u podacima u još manje. Na slici 7.2 je prikazana je vrijednost  $J$  za različit broj  $K$  za primjer sa slike 7.1a. Uočava se nagli pad vrijednosti  $J$  za  $K = 4$  što i je optimalni broj grupa. Međutim, treba imati na umu da u praktičnim problemima često nije lako detektirati ovu vrijednost ili postoji više takvih vrijednosti.

### 7.3 Algoritam K srednjih vrijednosti u scikit-learn biblioteci

Algoritam  $K$  srednjih vrijednosti je u scikit-learn biblioteci implementiran je u obliku klase:

```
class sklearn.cluster.KMeans(n_clusters=8, *, init='k-means++',
```

## 64 Poglavlje 7. Grupiranje podataka primjenom algoritma K srednjih vrijednosti.



Slika 7.2: Lakat metoda – vrijednost J u ovisnosti o broju centara K.

```
n_init='warn', max_iter=300, tol=0.0001, verbose=0,
random_state=None, copy_x=True, algorithm='lloyd')
```

Najvažniji parametri su:

- n\_clusters – broj grupa  
cjelobrojna vrijednost, default=8
- init – načini inicijalizacije centara  
'k-means++', 'random', default='k-means++'
- n\_init – koliko puta će se algoritam izvršiti s različitim početnim centrima, a konačan rezultat su centri koji su postigli najmanju vrijednost kriterijske funkcije  
'auto' ili cjelobrojna vrijednost, default=10
- max\_iter – maksimalni broj iteracija algoritma  
cjelobrojna vrijednost, default=300

Najvažnije metode ove klase su:

- .fit(X) izvršava algoritam K srednjih vrijednosti
- .predict(X) izračunaj najbliži centar za svaki podataka u X

Primjer 7.1 prikazuje kako primijeniti algoritam srednjih vrijednosti na podatkovni skup  $X$ .

### ■ Primjer 7.1

```
from sklearn.cluster import KMeans
import numpy as np

# podatkovni primjeri
X = np.array([[9,1], [3,2], [3,9], [4,8], [8,2],
[7,4], [9,7], [1,4], [8,7], [1,1]])

# inicijalizacija algoritma K srednjih vrijednosti
km = KMeans(n_clusters=3, init='random',
n_init=5, random_state=0)

# pokretanje grupiranja primjera
km.fit(X)

# dodjeljivanje grupe svakom primjeru
labels = km.predict(X)
```

## 7.4 Priprema za vježbu

1. Proučite poglavlje 7.2.
2. Po potrebi dodatno proučite dokumentaciju:
  - a. Grupiranje podataka
  - b. Algoritam K srednjih vrijednosti

## 7.5 Rad na vježbi

1. Isprobajte Python primjere iz poglavlja 7.3 u Visual Studio Code IDE.
2. Riješite dane zadatke.

**Zadatak 7.5.1** Skripta `zadatak_1.py` sadrži funkciju `generate_data` koja služi za generiranje umjetnih podatkovnih primjera kako bi se demonstriralo grupiranje. Funkcija prima cijeli broj koji definira željeni broj uzoraka u skupu i cijeli broj od 1 do 5 koji definira na koji način će se generirati podaci, a vraća generirani skup podataka u obliku numpy polja pri čemu su prvi i drugi stupac vrijednosti prve odnosno druge ulazne veličine za svaki podatak. Skripta generira 500 podatkovnih primjera i prikazuje ih u obliku dijagrama raspršenja.

1. Pokrenite skriptu. Prepoznajete li koliko ima grupe u generiranim podacima? Mijenjajte način generiranja podataka.
2. Primijenite metodu  $K$  srednjih vrijednosti te ponovo prikažite primjere, ali svaki primjer obojite ovisno o njegovoj pripadnosti pojedinoj grupi. Nekoliko puta pokrenite programski kod. Mijenjate broj  $K$ . Što primjećujete?
3. Mijenjajte način definiranja umjetnih primjera te promatrajte rezultate grupiranja podataka (koristite optimalni broj grupa). Kako komentirate dobivene rezultate?

**Zadatak 7.5.2** Kvantizacija boje je proces smanjivanja broja različitih boja u digitalnoj slici, ali uzimajući u obzir da resultantna slika vizualno bude što sličnija originalnoj slici. Jednostavan način kvantizacije boje može se postići primjenom algoritma  $K$  srednjih vrijednosti na RGB vrijednosti elemenata originalne slike. Kvantizacija se tada postiže zamjenom vrijednosti svakog elementa originalne slike s njemu najbližim centrom. Na slici 7.3a dan je primjer originalne slike koja sadrži ukupno 106,276 boja, dok je na slici 7.3b prikazana resultantna slika nakon kvantizacije i koja sadrži samo 5 boja koje su određene algoritmom  $K$  srednjih vrijednosti.

1. Otvorite skriptu `zadatak_2.py`. Ova skripta učitava originalnu RGB sliku `test_1.jpg` te ju transformira u podatkovni skup koji dimenzijama odgovara izrazu 7.2 pri čemu je  $n$  broj elemenata slike, a  $m$  je jednak 3. Koliko je različitih boja prisutno u ovoj slici?
2. Primijenite algoritam  $K$  srednjih vrijednosti koji će pronaći grupe u RGB vrijednostima elemenata originalne slike.
3. Vrijednost svakog elementa slike originalne slike zamjeni s njemu pripadajućim centrom.
4. Usporedite dobivenu sliku s originalnom. Mijenjate broj grupa  $K$ . Komentirajte dobivene rezultate.
5. Primijenite postupak i na ostale dostupne slike.

## 66 Poglavlje 7. Grupiranje podataka primjenom algoritma K srednjih vrijednosti.

6. Grafički prikažite ovisnost  $J$  o broju grupa  $K$ . Koristite atribut `inertia` objekta klase `KMeans`. Možete li uočiti lakat koji upućuje na optimalni broj grupa?
7. Elemente slike koji pripadaju jednoj grupi prikažite kao zasebnu binarnu sliku. Što primjećujete?



(a) originalna slika [1]



(b) rezultantna slika

Slika 7.3: Kvantizacija boje pomoću algoritma  $K$  srednjih vrijednosti.

### 7.6 Izvještaj s vježbe

Kao izvještaj s vježbe prihvata se web link na repozitorij pod nazivom OSU\_LV.

### Literatura

[1] <https://www.kaggle.com/datasets/tolgadincer/us-license-plates>