Spectral relaxation for k - means clustering

Eleonora Detić, Tea Maričić, Lara Milić i Josipa Radnić

Kolegij: Uvod u složeno pretraživanje podataka

Profesor: prof.dr.sc. Zlatko Drmač

Zagreb, 19. prosinca 2021.

Sadržaj

1 Uvod

Osnovna ideja k-means algoritma je određivanje predstavnika k skupina i pridruživanje svakog podatka skupini s najbližim predstavnikom, tako da zbroj kvadrata udaljenosti podataka od predstavnika skupina kojima pripadaju bude minimalan. Skupine ćemo nazivati klasteri, a predstavnike skupina centroidi.

U ovom seminarskom radu objasnit ćemo te implementirati takav algoritam, no glavni problem kojim ćemo se baviti je poboljšanje navedenog algoritma pomoću spektralne relaksacije. Koristeći teoriju linearne algebre transformirati ćemo matricu ulaznih podataka tako da ubrzamo algoritam te dobijemo bolje rezultate.

Nakon što implementiramo gore navedeni algoritam usporedit ćemo ga s klasičnim k-means algoritmom nad četiri različite vrste podataka, te ćemo navesti dobivene rezultate i zaključke.

2 Problem optimalne particije

Često u analizi podataka radimo s velikim skupom podataka. Kako bi ubrzali proces istraživanja podataka i olakšali rad s njima javlja nam se potreba za podjelom podataka u nekakve smislene grupe, odnosno podjela po sličnosti svojstava koja su nama bitna.

Neka su zadani $a_1, \ldots, a_n \in \mathbb{R}^m$, gdje su nam a_i podaci koje želimo particionirati. Tražimo optimalnu particiju $\pi = (\pi_1, \ldots, \pi_k)$ t.d. vrijedi :

$$\bigcup_{i=1}^{k} \pi_i = \{a_1, \ldots, a_n\}\,,\ \pi_i \bigcap \pi_j = \emptyset \ za\ i \neq j$$

Neka je s_i kardinalitet od $\pi_i = \left\{a_1^{(i)}, \ldots, a_{s_i}^{(i)}\right\}$. Optimalna particija minimizira funkciju cilja

$$Q(\pi) = \sum_{i=}^{k} \sum_{s=1}^{s_i} \|a_s^{(i)} - m_i\|_2^2, \text{ gdje je } \quad m_i = \tfrac{1}{s_i} \sum_{s=1}^{s_i} a_s^{(i)}.$$

Dakle, Q minimiziramo po svim k-particijama skupa od n elemenata.

Sada ćemo navesti u koracima kako provesti algoritam za k-sredina.

ALGORITAM ZA k-SREDINE:

1. Zadaj početnu particiju $\pi_j^{(0)},\ j = 1, \ldots, k$ i izračunaj početne centroide $m_j^{(0)},\ j = 1, \ldots, k$
2. Za svaki $a_i,\ i = 1, \ldots, n$ potraži najbliži centroid i pridruži se novom skupu

$$\pi_j^{(t+1)} = \left\{a \in \{a_1, \ldots, a_n\} : \left\|a - m_j^{(t)}\right\|_2 = \min_{\ell=1:k} \|a - m_\ell\|_2\right\}$$

3. Izračunaj centroide $m_j^{(t+1)}$ skupova $\pi_j^{(t+1)},\ j = 1, \ldots, k$.
4. Provjeri kriterij zaustavljanja. Ako je zadovoljen, $\pi_j^{(t+1)},\ j = 1, \ldots, k$ je tražena particija; STOP
   Inače, $t = t + 1$ i idi na Korak 2.

Ovisno o funkciji cilja moramo odabrati pogodan kriterij zaustavljanja. Sljedeća propozicija nam osigurava da se funckija cilja iteracijama neće povećati. Odnosno, funkcija cilja je monotono padajuća i omeđena odozdo s 0. Iz toga znamo da će niz iteracija konvergirati.

Propozicija 1. $Q\left(\pi^{(t)}\right) \geq Q\left(\pi^{(t+1)}\right) \geq 0.$

3 Matrična formulacija

Definiramo matricu $A = (a_1, \ldots, a_n)$ gdje nam $a_i,\ i = 1, \ldots, n$ predstavljaju stupce matrice A dimenzije m. k-particiju π ćemo matrično zapisati tako da ćemo stupce od A presložiti u blokove onih koji pripadaju istom elementu particije,

$$A\Pi = \begin{pmatrix} A_1 & \ldots & A_i & \ldots & A_k \end{pmatrix}, A_i = \begin{pmatrix} a_1^{(i)} & \ldots & a_{s_i}^{(i)} \end{pmatrix}, i = 1, \ldots, k,$$

gdje je Π matrica permutacije. Vrijedi:

$$m_i = \tfrac{1}{s_i} A_i e,$$
$$\text{gdje je } e = \begin{pmatrix} 1 & 1 & \ldots & 1 \end{pmatrix}^T.$$

Izvod:

$$\begin{aligned} \textstyle\sum_{s=1}^{s_i} \left\| a_s^{(i)} - m_i \right\|_2^2 &= \left\| A_i - m_i e^T \right\|_F^2 = \left\| A_i - \tfrac{1}{s_i} A_i e e^T \right\|_F^2 \\ &= \left\| A_i \left(I - \tfrac{1}{s_i} e e^T \right) \right\|_F^2, \\ \text{gdje je } \left(I - \tfrac{1}{s_i} e e^T \right)^2 &= \left(I - \tfrac{1}{s_i} e e^T \right) = \left(I - \tfrac{1}{s_i} e e^T \right)^T \\ \Rightarrow \left(I - \tfrac{1}{s_i} e e^T \right)^T &\text{ ortogonalni projektor} \end{aligned}$$

Nadalje,

$$\begin{aligned} &\left\| A_i \left(I - \tfrac{1}{s_i} e e^T \right) \right\|_F^2 \\ &= \operatorname{Trag} \left(A_i \left(I - \tfrac{1}{s_i} e e^T \right) \left(I - \tfrac{1}{s_i} e e^T \right) A_i^T \right) \\ &= \operatorname{Trag} \left(A_i \left(I - \tfrac{1}{s_i} e e^T \right) A_i^T \right) \\ &= \operatorname{Trag} \left(\left(I - \tfrac{1}{s_i} e e^T \right) A_i^T A_i \right) \\ \\ &= \operatorname{Trag} \left(A_i^T A_i \right) - \tfrac{1}{s_i} \operatorname{Trag} \left(e e^T A_i^T A_i \right) \\ &= \operatorname{Trag} \left(A_i^T A_i \right) - \tfrac{1}{s_i} \operatorname{Trag} \left(e^T A_i^T A_i e \right) \\ \\ &= \operatorname{Trag} \left(A_i^T A_i \right) - \left(\tfrac{e}{\sqrt{s_i}} \right)^T A_i^T A_i \left(\tfrac{e}{\sqrt{s_i}} \right) \end{aligned}$$

U ovim oznakama uz određen izvod, funkcija cilja za particiju π glasi:

$$Q(\pi) = \textstyle\sum_{i=1}^{k} \left(\operatorname{Trag} \left(A_i^T A_i \right) - \left(\tfrac{e}{\sqrt{s_i}} \right)^T A_i^T A_i \left(\tfrac{e}{\sqrt{s_i}} \right) \right)$$

Uočavamo da je (jer $A\Pi = \begin{pmatrix} A_1 & \dots & A_i & \dots & A_k \end{pmatrix}$):

$\sum_{i=1}^{k} \text{Trag}\left(A_i^T A_i\right) = \text{Trag}\left((A\Pi)^T(A\Pi)\right) = \text{Trag}\left(A^T A\right) = \|A\|_F^2$.

Stavimo $X = \begin{pmatrix} e/\sqrt{s_1} & & & \\ & e/\sqrt{s_2} & & \\ & & \ddots & \\ & & & e/\sqrt{s_k} \end{pmatrix}$

Lako provjerimo da je $\tilde{X} = \Pi X$ ortonormalna $\left(\widetilde{X}^T\widetilde{X} = I_k\right)$

$$\sum_{i=1}^{k} \left(\frac{e}{\sqrt{s_i}}\right)^T A_i^T A_i \left(\frac{e}{\sqrt{s_i}}\right) = \text{Trag}\left(X^T\Pi^T A^T A\Pi X\right) = \text{Trag}\left(\tilde{X}^T A^T A\tilde{X}\right)$$
$$\Longrightarrow Q(\pi) = \text{Trag}\left(A^T A\right) - \text{Trag}\left(\widetilde{X}^T A^T A\tilde{X}\right) \longrightarrow \min_{\tilde{X}}$$

$\widetilde{X}$ u svakom retku ima točno jedan netrivijalan element i određuje particiju. Kako $\text{Trag}\left(A^T A\right)$ ne ovisi o particiji, odnosno o $\widetilde{X}$, problem minimizacije funkcije cilja nam se svodi na maksimizaciju $\text{Trag}\left(\widetilde{X}^T A^T A\tilde{X}\right)$. S ovim je naš algoritam ostao i dalje iste težine.

Ideja nam je povećati skup po kojem maksimiziramo ako na takvom skupu imamo dobar algoritam za računanje maksimuma. Maksimum na većem skupu je sigurno veći ili jednak maksimumu koji mi tražimo.

4 Relaksacija problema

Umjesto gore opisanih matrica $\widetilde{X}$ ćemo promatrati „veći" skup u kojem su sve ortnormalne matrice, tj. zadržavamo svojstvo ortonormalnosti $\left(X^T X = I_k\right)$, ali X ne mora imati diskretnu strukturu kao $\widetilde{X}$.
Nakon toga želimo X zamijeniti najbližom matricom unutar skupa ortonormalnih, ali koja ima diskretnu strukturu. Time ćemo promijeniti točku u kojoj se postiže maksimum, ali se nadamo da neće biti daleko. Pomoću sljedećeg teorema ćemo moći gledati problem na „većem" skupu.

Teorem 2. *(Ky Fan)*
Neka je $H \in \mathbb{R}^{n\times n}$ *simetrična matrica* $\left(H = H^T\right)$ *sa svojstvenim vrijednostima* $\lambda_1 \geq \cdots \geq \lambda_n$ *i pripadnom matricom svojstvenih vektora* $U = (u_1, \ldots, u_n), Hu_i = \lambda_i u_i, i = 1, \ldots, n; \quad U^T U = I_n.$
Tada je:

$$\max_{\substack{X\in\mathbb{R}^{n\times k}\\ X^T X = I_k}} \operatorname{Trag}\left(X^T H X\right) = \lambda_1 + \ldots + \lambda_k.$$

Za $\lambda_k > \lambda_{k+1}$ *se optimalne matrice* X *mogu opisati kao elementi skupa*

$$\left\{ \begin{pmatrix} u_1 & \ldots & u_k \end{pmatrix} Q, \quad Q \text{ ortogonalna } k \times k \right\}.$$

Iz ovog teorema slijedi nam donja ograda za funkciju cilja.

$$\min_{\pi\text{-part.}} Q(\pi) \geq \operatorname{Trag}\left(A^T A\right) - \max_{\substack{X\in\mathbb{R}^{n\times k}\\ X^T X = I_k}} \operatorname{Trag}\left(X^T A^T A X\right)$$

$$= \sum_{i=1}^{\min(m,n)} \sigma_i^2 - \sum_{i=1}^{k} \sigma_i^2 = \sum_{i=k+1}^{\min(m,n)} \sigma_i^2 = \|A\|_F^2 - \sum_{i=1}^{k} \sigma_i^2,$$

gdje su $\sigma_1 \geq \cdots \geq \sigma_{\min(m,n)}$ singularne vrijednosti od A

Kako bi mogli koristiti ovaj teorem te razumjeti donju ogradu za funkciju cilja objasniti ćemo što je SVD dekompozicija matrice A.

5 SVD dekompozicija (Singular Value Decomposition)

Neka je $A \in \mathbb{R}^{m\times n}$ matrica ranga r. Tada postoje unitarne matrice $U = (u_1, \ldots, u_m)$, $V = (v_1, \ldots, v_n)$ $(U^*U = I_m, V^*V = I_n)$ i dijagonalna Σ tako da je (npr. $m \geq n$)

$$A = U\Sigma V^* = \sum_{k=1}^{r} \sigma_i u_i v_i^*$$

$$\Sigma = \operatorname{diag}(\sigma_i), \quad \sigma_1 \geq \cdots \geq \sigma_r > 0 = \sigma_{r+1} = \cdots = \sigma_{\min(m,n)}, \text{ gdje su}$$

σ_i = singularna vrijednost matrice A
u_i = lijevi singularni vektor
v_i = desni singularni vektor
$Av_i = \sigma_i u_i$
$H = A^*A = V\Sigma^*\Sigma V^*$, $M = AA^* = U\Sigma\Sigma^*U^*$
$Hv_i = \sigma_i^2 v_i$, $Mu_i = \sigma_i^2 u_i$
σ_i^2 = svojstvena vrijednost matrice A^TA i AA^T

S obzirom da ćemo raditi nad realnim podacima, adjungirana matrica će zapravo biti transponirana matrica.

Sljedeći teorem nam omogućava da ako imamo SVD dekompoziciju matrice A možemo pronaći njoj najbližu matricu manjeg ranga.

Teorem 3. *(Eckart - Young - Mirsky - Schmidt)*
Neka je $\ell < r$ i $A_\ell = \sum_{i=1}^{\ell} \sigma_i u_i v_i^*$. Tada je

$$\min_{\operatorname{rang}(X)\leq \ell} \|A - X\|_F = \|A - A_\ell\|_F = \sqrt{\sum_{i=\ell+1}^{r} \sigma_i^2}$$
$$\min_{\operatorname{rang}(X)\leq \ell} \|A - X\|_2 = \|A - A_\ell\|_2 = \sigma_{\ell+1}$$

Pomoću ovog teorema možemo zaključiti da nam je dovoljna matrica A ranga k, gdje nam je k broj klastera.
Sada se vratimo na funkciju cilja:

$$\min_{\pi\text{-part.}} Q(\pi) \geq \operatorname{Trag}\left(A^TA\right) - \max_{\substack{X\in\mathbb{R}^{n\times k} \\ X^TX = I_k}} \operatorname{Trag}\left(X^TA^TAX\right) = \|A\|_F^2 - \sum_{i=1}^{k} \sigma_i^2$$

Iz donje ograde sada razumijemo što su singularne vrijednosti te također nam daje intuiciju zašto uzeti prvih k singularnih vektora i singularnih vrijednosti.

Ideja je da riješimo kontinuirani problem ($\max_{\substack{X\in\mathbb{R}^{n\times k} \\ X^TX = I_k}} \operatorname{Trag}\left(X^TA^TAX\right)$) čije rješenje je po Ky Fanovom teoremu oblika skupa $\left\{ \begin{pmatrix} u_1 & \ldots & u_k \end{pmatrix} Q, \quad Q \text{ ortogonalna } k \times k \right\}$. Pitanje je kako izabrati matricu Q tako da naše rješenje ima što je moguće više diskretnu strukturu. U tome će nam pomoći QR faktorizacija.

6 QR faktorizacija

Neka je $\tilde{X}$ permutirana matrica od X koja odgovara diskretnoj strukturi koja određuje particiju. Za njenu strukturu vrijedi:

1. $\tilde{X} \neq 0$ akko i-ti element u j-tom skupu π_j
2. u svakom retku je točno jedan $\tilde{X} \neq 0$
3. stupci su međusobno okomiti
4. svojstva su invarijantna na permutacije

Neka je $\tilde{X}_k$ matrica dimenzije (n, k) uredno složena takva da prvih s_1 elemenata pripada prvom klasteru, sljedećih s_2 elemenata drugom klasteru itd. Ona predstavlja matrični zapis k-particije.
Pogledajmo svojstva matrice $\tilde{X}_k^T$:

1. prvih s_1 stupaca su međusobno kolinearni, sljedećih s_2 stupaca su međusobno kolinearni...
2. navedene grupe stupaca su međusobno okomite
3. svojstva se ne mijenjaju ako permutiramo retke od$\tilde{X}_k^T$

Kao rješenje Ky Fanovog teorema dobijemo $\tilde{X}_k^T$+pogreška i iz tog rješenja želimo „izvući" matricu $\tilde{X}_k^T$ koje je diskretna. Za to nam treba sljedeći teorem.

Teorem 4. *(QR faktorizacija)*
Neka je $A \in \mathbb{R}^{m\times n}$. Tada postoje ortogonalna $m \times m$ matrica Q i gornje trokutasta $\min(m, n) \times n$ *matrica R tako da je* $A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}$
(ili $A = QR$). Matricu R možemo odabrati tako da su joj dijagonalni elementi proizvoljnih predznaka. Ako je A ranga n i ako zahtijevamo da su dijagonalni elementi od R pozitivni, onda su matrica R i prvih n stupaca matrice Q jedinstiveno određeni. Pri tome prvih n stupaca matrice Q razapinju bazu za sliku od A, a ostalih $m - n$ stupaca razapinju ortonormiranu bazu za ortogonalni komplement slike od A.

Još će nam biti potreban sljedeći teoremu:

Teorem 5. *Householderov reflektor (zrcaljenje)*
Neka su $x, y \in \mathbb{R}^\ell$ vektori jednakih euklidskih duljina, $\|x\|_2 = \|y\|_2$. Tada postoji ortogonalna simetrična matrica H tako da vrijedi $Hx = y$. Matrica H je oblika $H = I - \left(2/v^T v\right) vv^T$, $v = x - y$. H zovemo Householderova transformacija (matrica) ili Householderov reflektor (zrcaljenje)

7 QR faktorizacija s pivotiranjem

Sada ćemo navesti algoritam QR faktorizacije pomoću Householderovih transformacija sa zamjenama stupaca.

ALGORITAM:

1. Ulaz: $A = A^{(0)} \in \mathbb{R}^{m\times n}$

2. za $k = 1, \ldots, p \equiv \min(m, n)$ {
   U matrici $A^{(k-1)}(k : m, k : n)$ odredi indeks ℓ_k stupca maksimalne euklidske duljine.
   U matrici $A^{(k-1)}$ zamijeni stupce s indeksima k i $(k + \ell_k - 1)$.
   Neka je $P^{(k)}$ permutacijska matrica koja realizira tu zamjenu $\left(A^{(k-1)}P^{(k)}\right)$. ima stupce u željenom poretku).;
   $x^{(k)} = \left(A^{(k-1)}P^{(k)}\right)(k : m, k),$; $\quad \xi_k = \operatorname{sign}\left(\left(x^{(k)}\right)_1\right)\left\|x^{(k)}\right\|_2$;
   $v^{(k)} = x^{(k)} - \xi_k e_1$; $\quad H^{(k)} = I_{k-1} \oplus H\left(v^{(k)}\right)$;
   $A^{(k)} = H^{(k)}A^{(k-1)}P^{(k)}$}

3. Izlaz: $Q = H^{(1)}H^{(2)}\cdots H^{(p)}; R = A^{(p)}(1 : p, 1 : n);\ P = P^{(1)}P^{(2)}\cdots P^{(p)}$

Ovim algoritmom dobijemo:

$$\underbrace{A}_{m\times n}\ \overbrace{P}^{\text{permutacija}} = Q\begin{pmatrix} R \\ 0 \end{pmatrix}$$

$$Q^*Q = I_m.$$

$$|R_{ii}| \geq \sqrt{\sum_{k=i}^{j} |R_{kj}|^2},\ \text{za sve } 1 \leq i \leq j \leq n.$$

$$|R_{11}| \geq |R_{22}| \geq \cdots \geq |R_{\rho\rho}| \gg |R_{\rho+1,\rho+1}| \geq \cdots \geq |R_{nn}|$$

Ako je X_k matrica dobivena iz relaksirane optimizacije, onda je X_kQ također rješenje relaksiranog problema s bilo kojom ortogonalnom Q. Tada očekujemo da za neku ortogonalnu matricu Q vrijedi $X_kQ \approx \widetilde{X}_k$, tj. $Q^TX_k^T \approx \widetilde{X}_k^T$, do na permutaciju redaka i stupaca.
Uzeti ćemo kao rješenje relaksiranog problema iz Ky-Fan teorema skup oblika iz skupa $\left\{\begin{pmatrix} u_1 & \ldots & u_k \end{pmatrix}Q, \quad Q \text{ ortogonalna } k\times k\right\}$ za $Q = I_k$, te provesti gore navedeni algoritam nad tom matricom.

Onda dobijemo:

$$X_k^T P = QR = Q \begin{pmatrix} R_1 & R_2 \end{pmatrix}, \quad R_1 \in \mathbb{R}^{k \times k}$$

Sada izračunajmo matricu:

$$\hat{R} = R_{11}^{-1} \left[R_{11}, R_{12} \right] P^T = \left[I_k, R_{11}^{-1} R_{12} \right] P^T.$$

Tada je pripadnost određenom klasteru određena indeksom retka koji ima najveću apsolutnu vrijednost pripadajućeg stupca matrice $\hat{R}$.

Pogledajmo sada kako smo pristupili ovom problemu, koje podatke smo koristili te koje rezultate smo dobili. Na kraju svega vidjet ćemo što možemo zaključiti iz dobivenih rezultata.

8 Implementacija u Pythonu

Na različitim skupinama podataka provesti ćemo tri vrste klasteriranja. Prva vrsta je algoritam opisan na samom početku, u odjeljku 2, koji ćemo u nastavku rada nazivati K-means. Druga vrsta je algoritam koji ćemo zvati p-Kmeans, koji uzima za matricu podataka rješenje relaksiranog problema, tj. matricu najvećih k svojstvenih vektora iz gram matrice A. Konačno, zadnji algoritam klasteriranja nazivat ćemo p-QR i on će iskoristiti QR faktorizaciju s pivotiranjem nad matricom koja sadrži najvećih k svojstvenih vektora. Njega radi jednostavnosti zovemo p-QR Kmeans.

Prvo definirajmo pomoćne funkcije za korištenje p-Kmeans i p-QR.

1. Funkcija će računati prvih k najvećih svojstvenih vektora Gramove matrice A.

```
def spectral_relaxation(A,k):
    #A is data matrix, one column is one data

    eigenvalues, eigenvectors = eigh(np.transpose(A) @ A)

    eigenvalues = eigenvalues[A.shape[1]-k:A.shape[1]][::-1]
    X_k = np.flip(eigenvectors[:, (A.shape[1]-k):A.shape[1]] ,axis=1)

    return X_k
```

2. Funkcija će raditi QR faktorizaciju s pivotiranjem nad matricom s prvih k najvećih svojstvenih vektora Gramove matrice A.

```
    def QR_for_starting_partition(A,k):
    #A is data matrix, one column is one data

    X_k = spectral_relaxation(A,k)
    X_k = np.transpose(X_k)

    Q, R, P = scipy.linalg.qr(X_k, pivoting = True)
    P_ = np.zeros([A.shape[1],A.shape[1]])
    for i in range(A.shape[1]):
        P_[i,P[i]] = 1

    R_11 = R[0:k, 0:k]
```

```
    R_12 = R[0:k, k:R.shape[1]]
    I = np.eye(k, dtype=int)
    R = np.matmul(np.linalg.inv(R_11), R_12)

    R_kapa = np.append(I,R, axis=1) @ np.transpose(P_)
    R_kapa = np.absolute(R_kapa)

    return np.argmax(R_kapa, axis=0)
```

Sada ćemo navesti kod za navedene tri vrste algoritama:

1. Kmeans algoritam

```
def K_means_2(A, starting_partition, k, epsilon):
    #A is data frame, one data is one present as one column
    #starting_partition is starting partition, usually random
    #k is number of clasters
    #epsilon is small number, stop criterion

    A.columns = starting_partition
    partition = starting_partition

    number_of_iteration = 0
    Q_of_partition = np.inf

    while(Q_of_partition > epsilon):

        number_of_iteration += 1

        s_i = np.unique(partition, axis=0, return_counts=True)[1]
        a_i = A.groupby(level=0,axis=1).sum().add_suffix('._centroid')
        m_i = a_i / s_i

        Q_of_partition_before = 0
        for i in range(A.shape[1]):
            Q_of_partition_before  += np.linalg.norm((A.iloc[:,i]-m_i.iloc
```

```
        for i in range(A.shape[1]):
            distance_final = np.inf
            for j in range(0, m_i.shape[1]):
                distance = np.linalg.norm((A.iloc[:,i] - m_i.iloc[:,j]))
                if(distance < distance_final):
                    distance_final = distance
                    tmp = list(A.columns)
                    tmp[i] = j
                    A.columns = tmp

        partition = A.columns.values

        Q_of_partition_after = 0
        for i in range(A.shape[1]):
            Q_of_partition_after  += np.linalg.norm((A.iloc[:,i] - m_i.ilo

        Q_of_partition = Q_of_partition_before - Q_of_partition_after

    return number_of_iteration, m_i, A.columns
```

2. p-Kmeans algoritam

```
def p_K_means(A,k,epsilon):

    X_k = spectral_relaxation(A, k)
    X_k = np.transpose(X_k)

    while True:
        starting_partition = np.random.randint(0, k, A.shape[1])
        s_i = np.unique(starting_partition, axis=0, return_counts=True)[1]
        if len(s_i) == k:
            break

    return K_means_2(pd.DataFrame(X_k), starting_partition, k, epsilon)
```

3. p-QR algoritam

```
def p_QR(A,k,epsilon):

    X_k = spectral_relaxation(A,k)
    X_k = np.transpose(X_k)

    starting_partition = QR_for_starting_partition(A, k)


    return K_means_2(pd.DataFrame(X_k), starting_partition, k, epsilon)
```

Nadalje, želimo prediktirati vrijednosti podataka pomoću napisanih algoritma. Ono što smo dosad napravili jest klasteriranje podataka, no sada bismo htjeli nešto više. Plan nam je pomoću train seta opisati svaki klaster(npr. prvi klaster je znamenka broj 5) te nakon toga za nove vrijednosti podataka ne samo odrediti klaster već i vrijednost koja je pridružena pojedinom klastera.

K-means predict

```
def K_means_predict(A, D, k, epsilon):
    #A is data matrix, one column is one data
    #D is array of real clasters
    #k is number of clasters
    #epsilon is stop criterion

    partition = np.array(p_QR(A,k,epsilon)[2])
    D_informations = np.vstack([D, partition])
    A_informations = np.vstack([A, partition])

    A_train, A_test, D_train, D_test = train_test_split(np.transpose(A_inf
    A_train = np.transpose(A_train)
    A_test= np.transpose(A_test)
    D_train = np.transpose(D_train)
    D_test = np.transpose(D_test)

    solution = np.zeros(D_test[0].shape[0])

    for i in range(D_test[0].shape[0]):
        claster = D_test[1][i]
        index = np.where(A_train[-1, :] == claster)[0]
        numbers = D_train[0][index]
        if(numbers.size != 0 ):
            solution[i] = mode(numbers)

    return np.sum(D_test[0] == solution) / D_test[0].shape[0]
```

9 Opis podataka i rezultati

9.1 Opis podataka

Prva skupina podataka su točke u ravnini. Formirale smo četiri klastera točaka u svakom od kvadranata u Kartezijevoj ravnini. Krenule smo od pretpostavke da znamo koja točka se nalazi u kojem klasteru i to tako da svaki kvadrant određuje svoj klaster. Na tim podacima smo provele Kmeans algoritam.

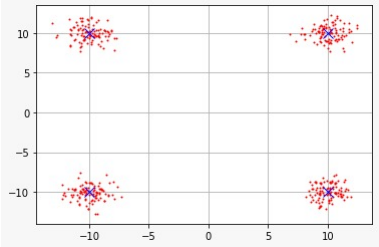
Druga skupina podataka, nad kojima smo provele sva tri navedena algoritma, su odjevni predmeti. Ručno smo izgenerirale 131 sliku odjevnog predmeta podijeljenih u tri klastera. Među slikama se nalaze slike majica, torbi i cipela.
Nakon što smo slike oblikovale da odgovovaraju rezoluciji od 32×32 piksela, svaku sliku smo rastegnule u vektor stupac te smo ih na taj način slagale u matricu naših ulaznih podataka. Kako bismo provjerile koliko „dobro" je naš algoritam klasterirao naše podatke uveli smo dodatni vektor takav da indeks svakog elementa vektora kaže u kojem klasteru se nalazi element u tom stupcu naše matrice podataka.

Treća skupina podataka su rukom pisane znamenke. Kao i gore, imale smo matricu naših podataka gdje svaki stupac odgovara jednoj znamenci. Kao kontrolni faktor naših triju algoritama također smo imale jedan vektor koji je, analogno kao i u gornjem slučaju, govori u kojem stupcu se nalazi koja znamenka.

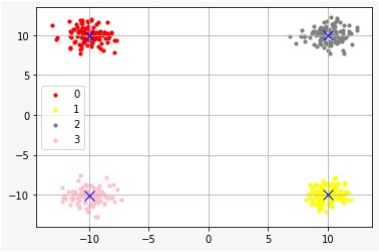
Konačno, kao posljednju skupinu podataka koristile smo stvarna mjerenja dobivena na 452 pacijenta. Svaki redak naše matrice podataka nam predstavlja informacije o jednom pacijentu za kojeg imamo 279 značajki. Svi ispitanici imaju dijagnosticiranih problema sa srcem. Za svakog od njih imamo neka medicinska mjerenja na osnovu kojih je uspostavljena dijagnoza srčane aritmije. Pomoću algoritama željeli smo predvidjeti dijagnozu. Naš skup podataka se sastojao od 12 različitih dijagnoza, stoga smo podatke smiještali u 12 različitih klastera.

9.2 Rezultati

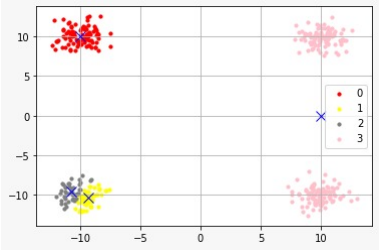
9.2.1 Rezultati - točke



Slika 1: Generirane točke



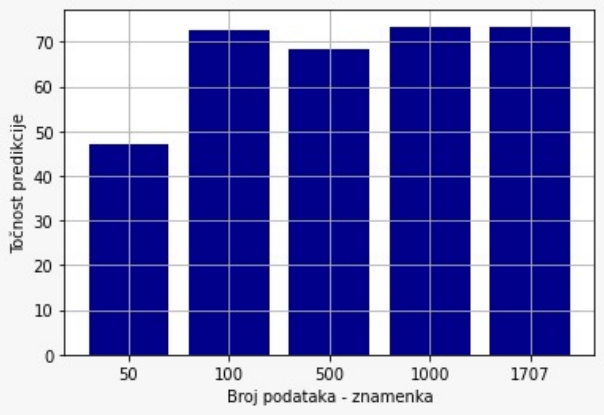
Slika 2: Primjer „lijepo” klasteriranih točaka



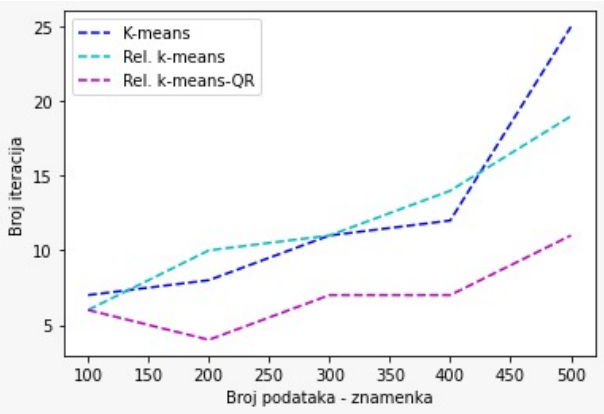
Slika 3: Primjer „loše” klasteriranih točaka

9.2.2 Rezultati - odjevni predmeti, znamenke i pacijenti

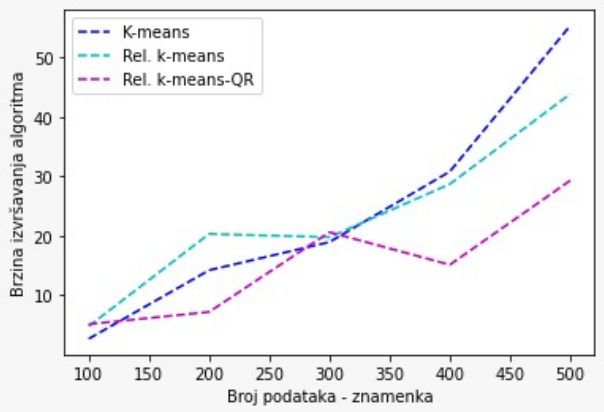
Napravit ćemo usporedbu algoritama na podacima za znamenke, te ćemo napraviti usporedbu točnosti predviđanja za sve tri skupine podataka.



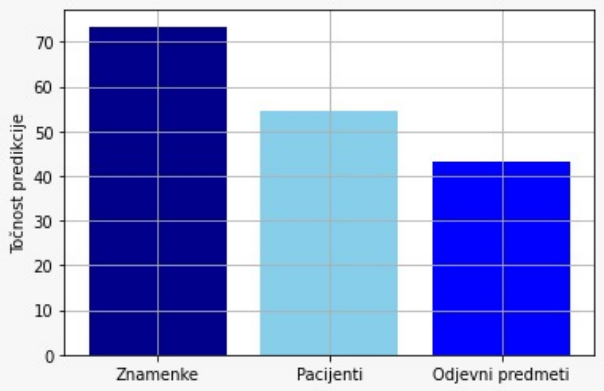
Slika 4: Točnost predikcije ovisno o broju podataka - podaci znamenke, p-QR



Slika 5: Broj iteracija ovisno o broju podataka - podaci znamenke



Slika 6: Brzina algoritma ovisno o broju podataka - podaci znamenke



Slika 7: Točnost predikcije na različitim podacima - p-QR

Iz Slike 4 možemo primjetiti da je točnost predikcije s rastom broja podataka za znamenke se povećava, osim u slučaju za 500 podataka kada broj točnosti pada s obzirom na 100 podataka.
Iz Slike 5 se vidi da broj iteracija potrebnih za izvršavanje algoritma je najmanji za algoritam p-QR bez obzira na broj podataka, što se i očekuje, dok Kmeans i p-Kmeans se isprepliću. Još možemo primijetiti da rastom broja podataka uglavnom raste broj potrebnih iteracija za sve algoritme.
Iz Slike 6 možemo vidjeti da brzina izvršavanja algoritma je uglavnom najmanji za algoritam p-QR bez obzira na broj podataka, dok ostala dva algoritma se opet isprepliću.
Iz Slike 7 možemo primijetiti da točnost predikcije pomoću p-QR algoritma je najveća

za znamenke, potom za pacijente, a najmanja za odjevne predmete.
Iz svega navedenog možemo zaključiti da smo uspjeli ubrzati standardni k-means algoritam, što je i bio cilj.

Literatura

[1] Zlatko Drmač: *USPP-2021-22-P1-P2-P3, predavanja*, Zagreb, (2021)

[2] Hongyuan Zha Xiaofeng He, Chris Ding Horst Simon, Ming Gu: *Spectral Relaxation for K-means Clustering*

[3] Helena Pelin: *Spektralno klasteriranje, diplomski rad*, Zagreb, (2016)

[4] Tim Roughgarden Gregory Valiant: *CS168: The Modern Algorithmic Toolbox Lecture 9: The Singular Value Decomposition (SVD) and Low-Rank Matrix Approximations*, Zagreb, (2021)