K-means algoritam u bioinformatici

Ančić, Doris

PRIRODOSLOVNO–MATEMATIČKI FAKULTET
SVEUČILIŠTA U SPLITU

DORIS ANČIĆ

K-MEANS ALGORITAM U BIOINFORMATICI

DIPLOMSKI RAD

Split, travanj 2024.

PRIRODOSLOVNO–MATEMATIČKI FAKULTET
SVEUČILIŠTA U SPLITU

ODJEL ZA MATEMATIKU

K-MEANS ALGORITAM U BIOINFORMATICI

DIPLOMSKI RAD

Student(ica):
Doris Ančić

Mentor(ica):
doc. dr. sc. Ivo Ugrina

Split, travanj 2024.

TEMELJNA DOKUMENTACIJSKA KARTICA

PRIRODOSLOVNO–MATEMATIČKI FAKULTET
SVEUČILIŠTA U SPLITU
ODJEL ZA MATEMATIKU

DIPLOMSKI RAD

K-MEANS ALGORITAM U BIOINFORMATICI

Doris Ančić

Sažetak:

U diplomskom radu predstavljen je k-means algoritam te, na primjeru analize ekspresije gena, njegova primjena u bioinformatici. U tu svrhu prvo smo predstavili potrebne pojmove iz biologije. Na kraju smo predstavili Lloydov algoritam kao jedan od najpoznatijih k-means algoritama. Također, u radu je dana i jedna implementacija Lloydovog algoritma za koju je korišten programski jezik Python.

Ključne riječi:

grupiranje, k-means algoritam, ekspresija gena, Načelo dobrog grupiranja, Lloydov algoritam

Podatci o radu:

47 stranice, 18 slika, 7 literaturnih navoda, jezik izvornika: hrvatski

Mentor(ica): *doc. dr. sc. Ivo Ugrina*

Članovi povjerenstva:

prof. dr. sc. Milica Klaričić Bakula

Stipe Marić

Povjerenstvo za diplomski rad je prihvatilo ovaj rad *28. ožujka 2024.*

BASIC DOCUMENTATION CARD

FACULTY OF SCIENCE, UNIVERSITY OF SPLIT
DEPARTMENT OF MATHEMATICS

MASTER'S THESIS

K-MEANS ALGORITHM IN BIOINFORMATICS

Doris Ančić

Abstract:

In this master's thesis k-means algorithm is introduced, along its application in bioinformatics with an example of gene expression analysis. To achieve this, the necessary terms and concepts from biology are introduced first. At the end, Lloyd's algorithm is presented as one of the most popular k-means algorithm. Additionally, one of the possible implementations of Lloyd's algorithm is given in this thesis, for which programming language Python was used.

Key words:

clustering, k-means algorithm, gene expression, The Good Clustering Principle, Lloyd's algorithm

Specifications:

47 pages, 18 figures, 7 references, original in: Croatian

Mentor: *doc. dr. sc. Ivo Ugrina*

Committee:

prof. dr. sc. Milica Klaričić Bakula

Stipe Marić

This thesis was approved by a Thesis commettee on *March 28, 2024.*

Uvod

K-means algoritam je, zbog svoje jednostavnosti i brzine, često i široko upotrebljavan algoritam za grupiranje podataka. Neka od područja u kojima se algoritam koristi su marketing, otkrivanje prijevara, bioinformatika, računalni vid, društvene znanosti...

Pojam "*k-means*" prvi puta spominje James MacQueen 1967. godine, iako sama ideja seže do 1956. godine i Huga Steinhausa. Standardni algoritam je prvi predložio Stuard Lloyd 1957. godine kao tehniku za pulsno-kodnu modulaciju, unatoč tome što nije objavljen kao članak sve do 1982. godine. Edward W. Forgy je 1965. godine objavio suštinski istu metodu, te se zbog toga u literaturi Lloydov algoritam može naći i pod imenom Lloyd-Forgyjev algoritam.

U bioinformatici *k-means* algoritam koristi se kako bi olakšao znanstvenicima analize podataka, prepoznavanje uzoraka, identificiranje teško uočljivih veza među podatcima, otkrivanje bioloških procesa, ... Najpoznatiji primjeri primjene *k-means* algoritma u tom području su analize ekspresije gena, analize sekvenci proteina te analize metaboličkih puteva.

U ovom radu naglasak će biti na primjeni *k-means* algoritma u bioinformatici, a poseban fokus bit će na analizi ekspresije gena te primjeni algoritma u navedenoj analizi.

Sadržaj

BASIC DOCUMENTATION CARD

Poglavlje 1

Motivacija: Evolucijska povijest proizvodnje vina

Jedan od prvih organizama koje su ljudi "pripitomili" bio je kvasac. Godine 2011. znanstvenici su, dok su iskopavali staro groblje u armenijskoj špilji, otkrili 6000 godina staru vinariju zajedno s vinskom prešom, posudama za fermentaciju, pa čak i čašama za piće. Ova vinarija bila je veliki tehnološki izum koji je zahtijevao razumijevanje kako kontrolirati rod kvasca *Saccharomyces* - rod kvasca koji se koristi za proizvodnju kruha i alkohola, a kojem pripadaju i kvašćeve gljivice.

Međutim, naše zanimanje za alkohol možda traje i duže od 6000 godina. Godine 2008. znanstvenici su otkrili da su perorepe rovke "alkoholičari". Piće koje konzumiraju je tzv. palmino vino proizvedeno iz cvjetova Bertram palme te je prirodno fermentirano pomoću kvasca *Saccharomyces* koji živi na njenim cvjetovima. Ovo otkriće sugerira kako naša sklonost prema alkoholu ima genetičku osnovu, te da je starija od armenijske vinarije milijunima godina.

Količina palminog vina koju rovke unose bila bi smrtonosna za većinu sisavaca. Srećom, rovke su razvile učinkovite načine probavljanja alkohola te

na taj način izbjegavaju opijanje, a samim time smanjuju rizik da ih predator ubije. Zbog tolerancija na alkohol koje rovke posjeduju, znanstvenici vjeruju da im alkohol pruža neke evolucijske prednosti, kao što su na primjer zaštita od srčanog udara. Također, znanstvenici smatraju da je moguće da su naši noviji, odnosno evolucijski bliži, pretci primati bili "alkoholičari" te da smo od njih naslijedili povezivanje alkohola s povećanjem unosa kalorija.

1.1 Dijauksični pomak

Vrsta kvasca kojeg ćemo proučavati kako bismo došli do problema grupiranja (eng. *clustering*) jest *Saccharomyces cerevisiae*, tj. pivski kvasac, pomoću kojeg možemo dobiti vino jer glukozu koja se nalazi u voću pretvara u etanol. Stoga možemo započeti s jednostavnim pitanjem: ako *S. cerevisiae* često živi na vinovoj lozi, zašto se zgnječeno grožđe mora čuvati u dobro zatvorenim bačvama kako bi se dobilo vino?

Jednom kada ostane bez zaliha glukoze, *S. cerevisiae* mora pronaći način kako preživjeti. Zbog toga ona mijenja svoj metabolizam, a etanol koji je upravo proizvela postaje njena nova zaliha hrane. Ovakva metabolička inverzija, nazvana dijauksični pomak (eng. *diauxic shift*), može se dogoditi samo u prisutnosti kisika. Bez kisika, *S. cerevisiae* hibernira sve dok kisik ili glukoza ne postanu dostupni. Drugim riječima, ako vinari ne zatvore dobro svoje bačve, kvasac u njima će probaviti etanol koji je proizveo, te na taj način uništiti vino.

Dijauksični pomak je složeni proces koji utječe na ekspresiju mnogih gena, odnosno na proces u kojem se gen uključuje u stvaranje RNA i proteina. Sukladno tome, dijauksički pomak morao je proizaći iz velikog evolucijskog događaja koji je pretku *Saccharomycesa* omogućio nevjerojatnu prednost u

odnosu na njegove "suparnike". Naime, *Saccharomyces* može ubiti svoje "suparnike" proizvodnjom etanola koji je toksičan za većinu bakterija i drugih kvasaca te uz to može koristiti prikupljeni etanol kao izvor energije. Međutim, kako i kada je *Saccharomyces* razvio dijauksični pomak, te koji su geni uključeni u taj proces?

Poglavlje 2

Identificiranje gena koji su odgovorni za dijauksični pomak

2.1 Dvije evolucijske hipoteze s različitim sudbinama

Japanski genetičar Susumu Ohno postavio je hipotezu da postoje rijetki evolucijski događaji zvani duplikacija cijelog genoma (eng. *Whole Genome Duplications*, ili *WGDs*). Svoj WGD model predstavio je 1970. godine unatoč nedostatku dokaza koji bi potvrdili njegovu hipotezu. Ipak, Ohno je vjerovao kako će WGD biti potreban u vrijeme kritične evolucijske inovacije kako bi vrste mogle implementirati neku novu, revolucionarnu, funkciju kao što je to, primjerice, dijauksični pomak.

Zamislimo, na primjer, vrijeme prije više milijuna godina kada su se razvile prve biljke koje su davale plodove, ali nijedan organizam nije mogao probaviti glukozu koju su ti plodovi proizvodili te iskoristiti proizvedeni etanol. Prve vrste koje bi to mogle napraviti, imale bi veliku evolucijsku prednost

unatoč tome što probavljanje glukoze nije jednostavan zadatak. Umjesto kreiranja novih gena, dijauksični pomak zahtijevao bi stvaranje novih metaboličkih puteva s mnogo gena koji rade zajedno. Ohno je tvrdio da bi WBD osigurao platformu za takvo revolucionarno otkriće, jer bi tada svaki duplicirani gen imao dvije kopije. Jedna kopija mogla bi se razvijati bez ugrožavanja postojeće funkcije gena koju bi preostala kopija obavljala.

Tri godine kasnije, 1973., Ohno je predstavio tzv. *Random Breakage Model* kao novi model evolucije kromosoma. Hipoteza ovog modela tvrdi da su točke loma preraspodijela genoma (eng. *breakage points of rearrangements*) nasumične, što dalje implicira da žarišne točke preraspodijela genoma (eng. *rearrangement hotspots*) u genomu sisavca ne postoje. U Ohnovom modelu žarišne točke bila su područja u genomu koja su pokazivala povišene stope preraspodijele genoma, dok je sama preraspodijela bila zapravo preokret segmenta kromosoma. Kao za WGD model, dokazi nisu postojali ni za *Random Breakage Model*.

Random Breakage model i WGD model imali su značajno drugačije sudbine. Naime, *Random Breakage Model* biolozi su prihvatili još u vrijeme kada je tek bio predložen i bio je dogma sve do 2003. godine kada je pobijen. Nasuprot tome, WGD model u početku je dočekan sa skepticizmom i 25 godina nije privlačio pažnju znanstvenika, dijelom zbog toga što se samo 13% gena *S. cerevisiae* duplicira.

Godine 1997. Wolfe i Shields dali su prve računske argumente u korist WGD modela za *S. cerevisiae*. Tvrdili su da činjenica što se samo 13% gena *S. cerevisiae* duplicira nije iznenađujuća jer čak i da stotine gena pridonosi evolucijskoj inovaciji nakon WGD-a, većina tih gena ne bi bila potrebna za tu inovaciju. Stoga bi duplicirani geni bili bombardirani mutacijama sve dok se ne bi pretvorili u pseudogene, te na taj način, nakon milijuna godina, nestali

iz genoma.

2.2 Koji geni kvasca upravljaju dijauksičnim pomakom?

Jedan od mnogih koraka koje kvasac obavlja tijekom fermentacije jest pretvorba acetaldehida u etanol. Ako kisik polako postaje dostupan, onda se nakupljeni etanol pretvara natrag u acetaldehid. Obje reakcije katalizira enzim alkohol dehidrogenaza (Adh). Kod *S. cerevisiae* aktivnost enzima Adh kodiraju dva gena, Adh_1 i Adh_2, koji su nastali dupliciranjem zajedničkog gena pretka. Enzim kojeg kodira Adh_1 ima povećanu sposobnost proizvodnje etanola, dok enzim kojeg kodira Adh_2 ima povećanu sposobnost konzumiranja etanola.

Godine 2005. Michael Thomson koristio je višestruko usklađivanje Adh gena različitih vrsta kvasaca kako bi rekonstruirao davni gen *Saccharomycesa*. Taj gen pokazao je sklonost pretvaranju acetaldehida u etanol koje je bilo slično ponašanju gena Adh_1. Thomson je stoga zaključio da je prije WGD-a za *Saccharomyces*, alkohol hidrogenaza bila većinski uključena u proces stvaranja, a ne u proces konzumacije, etanola. Nakon WGD-a Adh_1 je nastavio izvršavati svoju osnovnu funkciju, dok je Adh_2 mogao pomoći u pokretanju dijauksičnog pomaka.

Zamislimo da smo mogli pratiti svih n gena kvasca u m vremenskih točaka s obje strane dijauksičnog pomaka. Kao rezultat dobili bismo $n \times m$ matricu ekspresije gena E, gdje broj $E_{i,j}$ označava razinu ekspresije gena i u vremenu j. i-ti redak matrice E nazivamo **vektor ekspresije gena** i. Gledajući samo vektore ekspresija gena kvasca moguće je uočiti različite obrasce ponašanja gena s obzirom na dijauksični pomak. Također, pomoću matrice E mogli

bismo uočiti i gene čija se ekspresija jedva mijenja, gene čija se ekspresija brzo povećava prije, a potom naglo smanjuje nakon dijauksičnog pomaka, ...

Iako ćemo se u ovom radu fokusirati na ekspresiju gena s obzirom na dijauksični pomak, važno je naglasiti kako su matrice ekspresije gena česte u biološkoj analizi. Primjerice, ako je vektor ekspresije novosekvencioniranog gena sličan vektoru ekspresije gena čija je funkcija poznata, biolozi mogu posumnjati da ti geni obavljaju slične funkcije. Također, geni sa sličnim vektorima ekspresije mogu implicirati da su koregulirani, odnosno da njihovu ekspresiju kontrolira isti transkripcijski faktor. To, nadalje, sugerira na tzv. "krivnja po povezanosti" strategiju koja se koristi za zaključivanje funkcija gena. Ova strategija zasniva se na poznavanju funkcija nekih gena, te potencijalno širenje tih funkcija na gene sa sličnim vektorima ekspresije.

Konačno, analiza ekspresije gena važna je u biomedicinskim istraživanjima kao što su to, na primjer, analiza tkiva prije i nakon primjene lijeka ili kontrastiranje kancerogenih ili nekancerogenih stanica. Tako je, primjerice, ekspresijska analiza dovela do tzv. MammaPrinta - dijagnostičkog teksta koji, na temelju ekspresijske analize 70 ljudskih gena povezanih s aktivacijom i supresijom tumora, utvrđuje vjerojatnost pojave raka dojke. Ipak, nakon navedenih primjena analize ekspresije gena postavlja se pitanje: "Koje metode se mogu koristiti kako bismo analizirali podatke o ekspresiji gena?"

Poglavlje 3

Uvod u grupiranje (eng. *clustering*)

3.1 Mjerenje ekspresije gena

Postoji više načina kojima možemo mjeriti ekspresiju gena, a neki od njih su masena spektometrija, sekvencioniranje RNA i mjerenje ekspresije gena pomoću DNA nizova.

U eksperimentu masene spektometrije biolozi generiraju skup spektara i uspoređuju ih s proteomom - skupom svih bjelančevina i proteomskih oblika koje neki organizam proizvodi tijekom života. Broj spektara koji odgovaraju peptidima iz određenog proteina nudi zamjenu za razinu ekspresije promatranog proteina. Kako bi procijenili ekspresiju proteina iz ove zamjene, bioinformatičari moraju uzeti u obzir različite duljine proteina, slabu fragmentaciju nekih peptida (što dovodi do poteškoća u njihovoj identifikaciji) i druge praktične probleme.

U eksperimentu sekvencioniranja RNA generiramo niz čitanja iz transkriptoma (eng. *transcriptome*) ili skupa svih RNA molekula u stanici nas-

tale transkripcijom, pri čemu je transkriptom skup svih kodirajućih i nekodirajućih RNA molekula nastalih transkripcijom. Procjenjivanjem količine svakog RNA transkripta koji kodira protein u uzorku, dobivamo zamjenu za ekspresiju rezultirajućeg proteina. Brojni procesi utječu na proizvodnju proteina u stanici uz transkripciju, kao što su translacija, posttranslacijske modifikacije i razgradnja proteina. Valja napomenuti da ovi dodatni čimbenici mogu zbuniti korelaciju između količine transkripta i ekspresije odgovarajućeg proteina.

Kada mjerimo ekspresiju gena pomoću DNA nizova možemo koristiti DNK nizove koji nose tzv. sonde (eng. *probes*) koje su usmjerene na svaki gen u promatranoj vrsti. Svaku probu karakterizira intenzitet, koji nudi zamjenu za broj transkripata danog gena prisutnih u uzorku. Nedostatak DNK nizova je taj što ciljaju samo na identifikaciju poznatih transkripata i često ne uspijevaju procijeniti nepoznate transkripte. Na primjer, mnoge vrste raka uzrokovane su rijetkim mutacijama i ostale bi neotkrivene pri korištenju DNK nizova.

3.2 Mikronizovi

Postoji nekoliko različitih tehnologija mikronizova, ali sve su zasnovane na istim osnovnim principima. Niz visoke gustoće dizajniran je i konstruiran tako da sadrži mrežu od mnogo točaka, od kojih je svaka sastavljena od različite poznate DNK, RNK ili fragmenta proteina. Niz se zatim izlaže otopini koja sadrži nepoznatu mješavinu komplementarnih tvari koje su označene bojom ili izotopom. Ako se tvari u otopini hibridiziraju na određenim točkama niza, tada se njihova prisutnost i približna zastupljenost u otopini mogu mjeriti kvantificiranjem količine oznake koja je prisutna na određenim mjestima u

nizu.

Iako mikronizovi imaju mnoge namjene, uobičajena je upotreba za mjerenje ekspresije gena. Nizovi ekspresije gena sadrže tisuće različitih gena, od kojih je svaki predstavljen pomoću jednog ili više genskih fragmenata koji su pohranjeni na nizu.

Niz se zatim hibridizira s komplementarnom DNA (cDNA) pripremljenom iz RNA koja je pak izolirana iz bioloških tkiva. Promatrajući intenzitet hibridizacije na svakoj točki u nizu, može se procijeniti jesu li, i u kojem stupnju, različiti geni izraženi u tkivima iz kojih su ekstrakti napravljeni.

3.3 Mikronizovi i analiza ekspresije gena

3.3.1 Mikronizovi

Godine 1997. Joseph DeRisi proveo je prvi veliki eksperiment ekspresije gena uzorkovanjem kulture *S. cerevisiae* sedam puta u satima -6, -4, -2, 0, +2, +4 te +6, gdje nulti sat predstavlja sat dijauksičnog pomaka. Budući da *S. cerevisiae* ima otprilike 6400 gena, ovaj eksperiment rezultirao je matricom ekspresije gena reda 6400×7.

Kako nijedna od tehnika navedena u poglavlju 3.1 nije bila razvijena do 1997. godine, DeRisi je koristio mikronizove. Način na koji je DeRisi koristio mikronizove je sljedeći:

Nakon ekstrahiranja mnogih RNA transkripata izraženih u stanicama kvasca, DeRisi je pretvorio svaki RNA transkript u komplementarnu DNA (cDNA) pomoću enzima reverzna transkriptaza, i obilježio te cDNA na stakalcu. Zatim je hibridizirao cDNA sa fluorescentno obilježenom RNA iz uzorka kako bi izmjerio razine ekspresije različitih gena kvasca. Količina

cDNA otisnuta na svakoj točki mikroniza može uvelike varirati, što je komplikacija koju je DeRisi trebao riješiti kako bi se osiguralo da se intenziteti fluorescencije mogu usporediti između točaka i nizova. Zbog toga je DeRisi hibridizirao dva uzorka koji odgovaraju dvjema različitim vremenskim oznakama za svaki niz, a zatim je uzorke označio različitim fluorescentnim bojama tako da su se uzorci mogli razlikovati *softwareom* za obradu slike.

Vrijednosti ekspresije dobivene iz mikroniza predstavljene su kao omjer fluorescentnih intenziteta dvaju uzoraka. Stoga se ekspresija mjeri kao relativne promjene u ekspresiji pojedinačnih gena između uzoraka i vremenskih točaka.

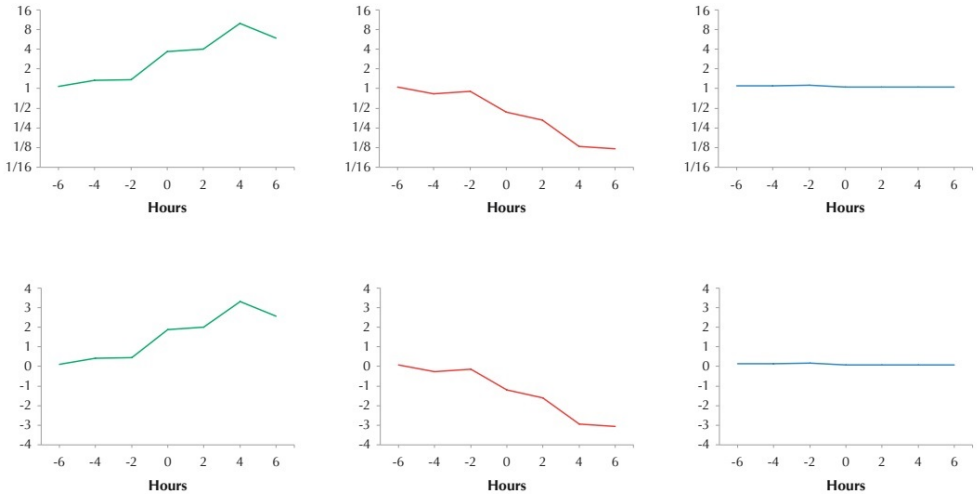
Mikronizovi danas se koriste rijetko, ali pristup koji je DeRisi koristio za analizu mikronizova radi podjednako dobro za moderne tehnologije ekspresije gena.

3.3.2 Analiza ekspresije gena

Promotrimo sliku 3.1 i uočimo da je uzorak vektora ekspresije gena YPR055W ravan tijekom dijauksičnog pomaka. Iz toga možemo zaključiti kako ovaj gen vjerojatno nije uključen u dijauksični pomak. S druge strane, ekspresija gena YLR258W (slika 3.1) znatno se mijenja tijekom dijauksičnog pomaka, pa možemo pretpostaviti da je ovaj gen uključen u dijauksični pomak. Doista, ako provjerimo bazu podataka genoma *Saccharomyces*, otkrit ćemo da je gen YLR258W glikogen sintaza. Ovaj enzim kontrolira proizvodnju glikogena - polisaharida saharoze koji je glavni spremnik glukoze u stanicama kvasca.

Biolozi u praksi često koriste logaritam vrijednosti ekspresije gena (slika 3.1). Nakon ove transformacije pozitivne vrijednosti vektora ekspresije gena

3.3. Mikronizovi i analiza ekspresije gena



Slika 3.1: Vektori ekspresije $(1.07, 1.35, 1.37, 3.70, 4.00, 10.00, 5.88)$, $(1.06, 0.83, 0.90, 0.44, 0.33, 0.13, 0.12)$, i $(1.11, 1.11, 1.12, 1.06, 1.05, 1.06, 1.05)$ tri gena kvasca (YLR258W, YPL012W, i YPR055W, redom) prikazani na grafu. Svaki vektor ekspresije $(e_1, ..., e_m)$ prikazan je kao unija segmenata koji povezuju točke (j, e_j) i $(j + 1, e_{j+1})$ za svaki $j \in [1, m - 1]$. U DeRisijevom eksperimentu, razina ekspresije u početnoj točki odgovara osnovnoj razini ekspresije; primijetimo da je razina ekspresije približno 1 za sva tri gena. Vrijednosti veće od 1 ukazuju na povećanu ekspresiju, dok vrijednosti manje od 1 ukazuju na smanjenu ekspresiju. Vektori ekspresije ista tri gena čije su razine ekspresije izražene kao logaritmi s bazom 2: $(0.11, 0.43, 0.45, 1.89, 2.00, 3.32, 2.56)$, $(0.09, -0.28, -0.15, -1.18, -1.59, -2.96, -3.08)$, i $(0.15, 0.15, 0.17, 0.09, 0.07, 0.09, 0.07)$.

odgovaraju povećanoj ekspresiji gena, dok negativne vrijednosti odgovaraju smanjenoj ekspresiji.

Slika 3.2 pokazuje matricu ekspresije deset gena kvasca nakon logaritmi-

ranja.

Gene	Expression Vector						
YLR361C	0.14	0.03	-0.06	0.07	-0.01	-0.06	-0.01
YMR290C	0.12	-0.23	-0.24	-1.16	-1.40	-2.67	-3.00
YNR065C	-0.10	-0.14	-0.03	-0.06	-0.07	-0.14	-0.04
YGR043C	-0.43	-0.73	-0.06	-0.11	-0.16	3.47	2.64
YLR258W	0.11	0.43	0.45	1.89	2.00	3.32	2.56
YPL012W	0.09	-0.28	-0.15	-1.18	-1.59	-2.96	-3.08
YNL141W	-0.16	-0.04	-0.07	-1.26	-1.20	-2.82	-3.13
YJL028W	-0.28	-0.23	-0.19	-0.19	-0.32	-0.18	-0.18
YKL026C	-0.19	-0.15	0.03	0.27	0.54	3.64	2.74
YPR055W	0.15	0.15	0.17	0.09	0.07	0.09	0.07

Slika 3.2: Podmatrica veličine 10 $\times$ 7 DeRisijeve matrice ekspresije gena veličine 6400 $\times$ 7. Podmatrica prikazuje vrijednosti ekspresije deset gena kvasca nakon logaritmiranja s bazom 2. Geni sa slike 3.1 prikazani su pripadnom bojom.

3.4 Grupiranje genoma kvasca

Cilj ovog rada jest podijeliti skup svih gena kvasca u k disjunktnih grupa (eng. *cluster*) tako da geni koji se nalaze u istoj grupi imaju sličan vektor ekspresije. U praksi broj grupa nije poznat unaprijed, stoga biolozi često primjenjuju algoritam grupiranja na podatke o ekspresiji gena s različitim vrijednostima k, te odabiru onu vrijednost k koja ima najviše smisla. Radi jednostavnosti, pretpostavit ćemo da je vrijednost k fiksna. Slika 3.3 pokazuje podjelu gena sa slike 3.2 u tri grupe koje ukazuju povećanu, smanjenu i ravnu ekspresiju gena tijekom dijauksičnog pomaka.

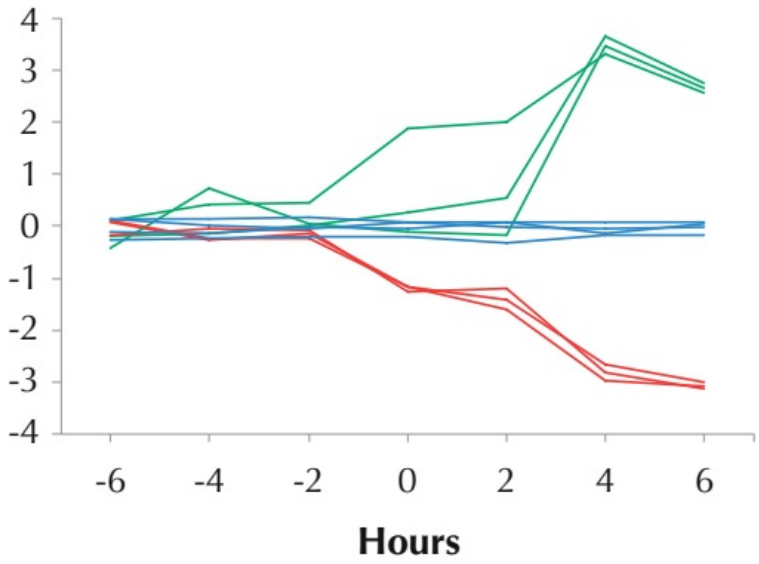
3.4. Grupiranje genoma kvasca

Iako je dijauksični pomak važan događaj u životu *S. cerevisiae*, on nema mnogo utjecaja na većinu funkcija kvasca. Zbog toga možemo pretpostaviti da većina gena *S. cerevisiae* ima ravnu ekspresiju tijekom dijauksičnog pomaka. Takve gene željeli bismo izuzeti iz razmatranja, te na taj način smanjiti matricu ekspresije.

Razine ekspresije većine gena kvasca gotovo se ne mijenjaju prije i poslije dijauksičnog pomaka (plava krivulja na slici 3.3). Ti geni, također, imaju svojstvo da su sve vrijednosti njihovog vektora ekspresije blizu nuli. U našoj analizi isključit ćemo one gene s vektorima ekspresije čije se vrijednosti nalaze u intervalu [-2.3, 2.3]. Ovim smo smanjili originalni skup od 6400 gena kvasca na skup podataka od 230 gena čije se ekspresije znatno mijenjaju oko dijauksičnog pomaka.

Na slici 3.3 možemo vidjeti da gen YLR258W ima drugačiji obrazac promjene u usporedbi s genom YGR043C. To ukazuje da bi podjela 230 gena kvasca u samo dvije grupe, tj. one s rastućim i padajućim razinama ekspresije, bilo previše jednostavno. Naš cilj je, zbog toga, grupirati te gene na temelju sličnih obrazaca ponašanja.

Gene	Expression Vector						
YLR361C	**0.14**	0.03	-0.06	0.07	-0.01	-0.06	-0.01
YMR290C	0.12	-0.23	-0.24	-1.16	-1.40	-2.67	**-3.00**
YNR065C	-0.10	**-0.14**	-0.03	-0.06	-0.07	-0.14	-0.04
YGR043C	-0.43	-0.73	-0.06	-0.11	-0.16	**3.47**	2.64
YLR258W	0.11	0.43	0.45	1.89	2.00	**3.32**	2.56
YPL012W	0.09	-0.28	-0.15	-1.18	-1.59	-2.96	**-3.08**
YNL141W	-0.16	-0.04	-0.07	-1.26	-1.20	-2.82	**-3.13**
YJL028W	-0.28	-0.23	-0.19	-0.19	**-0.32**	-0.18	-0.18
YKL026C	-0.19	-0.15	0.03	0.27	0.54	**3.64**	2.74
YPR055W	0.15	0.15	**0.17**	0.09	0.07	0.09	0.07



Slika 3.3: Retci matrice ekspesije gena sa slike 3.2 podijeljeni u tri grupe. Zeleni geni prikazuju povećanu ekspresiju, crveni smanjenu, dok plavi geni prikazuju ravnomjerno ponašanje i malo je vjerojatno da će biti povezani s dijauksičnim pomakom. Elementi s najvećom apsolutnom vrijednosti svakog vektora ekspresije istaknuti su masnim slovima, dok je na grafu prikazano svih deset vektora ekspresije.

Poglavlje 4

Načelo dobrog grupiranja

Kako bismo identificirali grupe gena sa sličnim obrascem ekspresije, razmišljat ćemo o vektoru ekspresije duljine m kao o točki u m-dimenzionalnom prostoru. Na ovaj način će geni sa sličnim vektorima ekspresije biti obližnje točke. Idealno, grupe bi trebale zadovoljavati sljedeće načelo:

Načelo dobrog grupiranja: Svaki par točaka iz iste grupe trebao bi imati manju udaljenost nego bilo koji par točaka koje pripadaju različitim grupama.

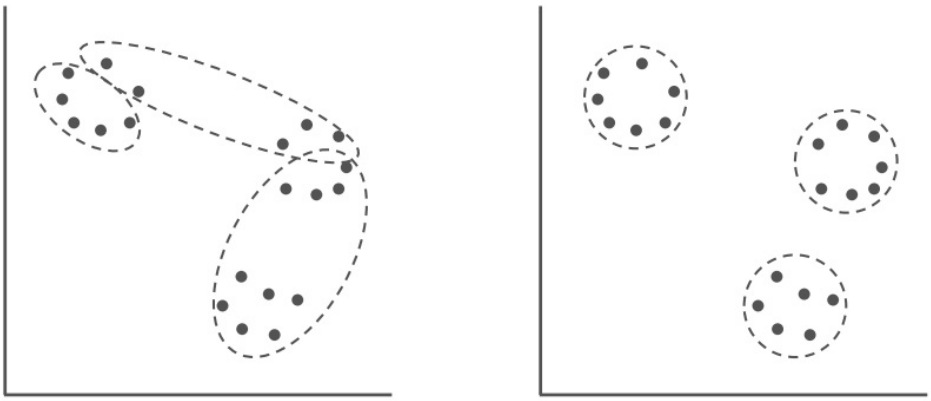
Problem dobrog grupiranja:

Podijeliti skup točaka u grupe.

Ulaz: n-člani skup u m-dimenzionalnom prostoru i prirodni broj k.

Izlaz: Particija danog n-članog skupa u k grupa koje zadovoljavaju Načelo dobrog grupiranja.

Ovim načelom analizu ekspresije gena pretvorili smo u problem particioniranja n-članog skupa točaka m-dimenzionalnog prostora u k grupa.



Slika 4.1: (Lijevo) Podjela 20 točaka koja ne zadovoljava Načelo dobrog grupiranja. (Desno) Podjela koja zadovoljava Načelo dobrog grupiranja.

Promatrajući sliku 4.1, prirodno bismo podijelili točke sa slike u dvije grupe. Međutim, te dvije grupe ne bi zadovoljavale Načelo dobrog grupiranja. Štoviše, takva podjela uopće ne postoji. Iz tog razloga morat ćemo problemu pristupiti drugačije kako bismo mogli osmisliti dobro definiran računski problem grupiranja.

Poglavlje 5

Grupiranje kao optimizacijski problem

Umjesto da o grupiranju razmišljamo kao o podjeli točaka nekog skupa u k grupa, pokušat ćemo izabrati k-člani skup točaka koje će predstavljati središta grupa. Radi lakšeg snalaženja, s $Data$ označimo skup točaka koje želimo podijeliti u grupe, a s $Centers$ skup središta tih grupa. Elemente skupa $Centers$ želimo izabrati tako da oni minimiziraju neku funkciju udaljenosti skupova $Data$ i $Centers$ u odnosu na sve moguće izbore središta grupa. Kako bismo mogli dobro definirati funkciju udaljenosti skupova $Data$ i $Centers$, prvo definiramo euklidsku udaljenost dviju točaka.

Definicija 5.1 *Neka su* $v = (v_1, \ldots, v_m)$ *i* $w = (w_1, \ldots, w_m)$ *dvije točke u m-dimenzionalnom prostoru.* ***Euklidska udaljenost točaka*** v *i* w*, u oznaci* $d(v, w)$*, dana je formulom*

$$d(v, w) = \sqrt{\sum_{i=1}^{m} (v_i - w_i)^2}.$$

Nadalje, za danu točku $DataPoint$ u m-dimenzionalnom prostoru i k-člani skup $Centers$, definiramo udaljenost točke $DataPoint$ od skupa $Centers$.

Definicija 5.2 *Euklidska udaljenost točke $DataPoint$ i središta grupe koji je od nje najmanje udaljen nazivamo* ***udaljenost točke*** *$DataPoint$* ***od skupa*** *$Centers$, i pišemo s $d(DataPoint, Centers)$.*

Dakle, vrijedi

$$d(DataPoint, Centers) = \min_{x \in Centers} d(DataPoint, x).$$

Sada možemo definirati udaljenost svih točaka skupa $Data$ i točaka skupa $Centers$.

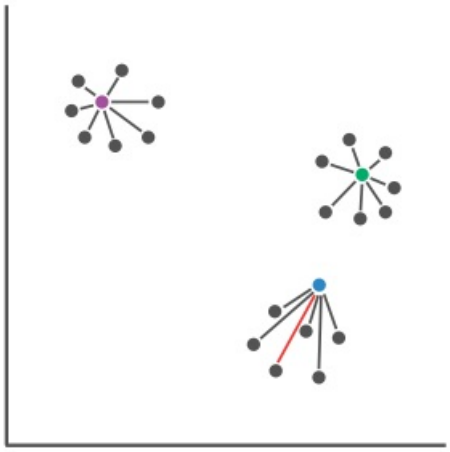
Definicija 5.3 ***Udaljenost svih točaka skupa*** *$Data$* ***i točaka skupa*** *$Centers$ definiramo kao maksimum od $d(DataPoint, Centers)$ po svim točkama $DataPoint$, te je označavamo s $MAXDISTANCE(Data, Centers)$.*

Očito je

$$MAXDISTANCE(Data, Centers) = \max_{DataPoint \in Data} d(DataPoint, Centers).$$

Napomena 5.4 *Pojmovi iz definicija 5.2 i 5.3 definirani su kako bi se olakšalo dobro definiranje problema grupiranja, te se stoga ne bi smjeli miješati sa standardnim definicijama udaljenosti točke od skupa i udaljenosti dvaju skupova.*

Nakon što smo definirali potrebne udaljenosti, možemo formulirati dobro definirani problem grupiranja.



Slika 5.1: Skup točaka $Data$ sa slike 4.1 podijeljene u tri grupe čija središta tvore skup $Centers$. Za svaku točku $DataPoint$ skupa $Data$ vrijednost $d(DataPoint, Centers)$ jednaka je duljini dužine koja je povezuje s najbližim središtem. Vrijednost $MAXDISTANCE(Data, Centers)$ jednaka je duljini najduže takve dužine, a na slici je prikazana crvenom bojom.

k-centers problem grupiranja (eng. k-Centers Clustering Problem):

Za dani skup točaka pronađite k središta koji minimiziraju maksimalnu udaljenost između točaka zadanog skupa i središta:

Ulaz: Skup točaka $Data$ i prirodni broj k.

Izlaz: k-člani skup $Centers$ središta koji minimiziraju udaljenost $MAXDISTANCE(Data, Centers)$ po svim mogućim izborima k središta.

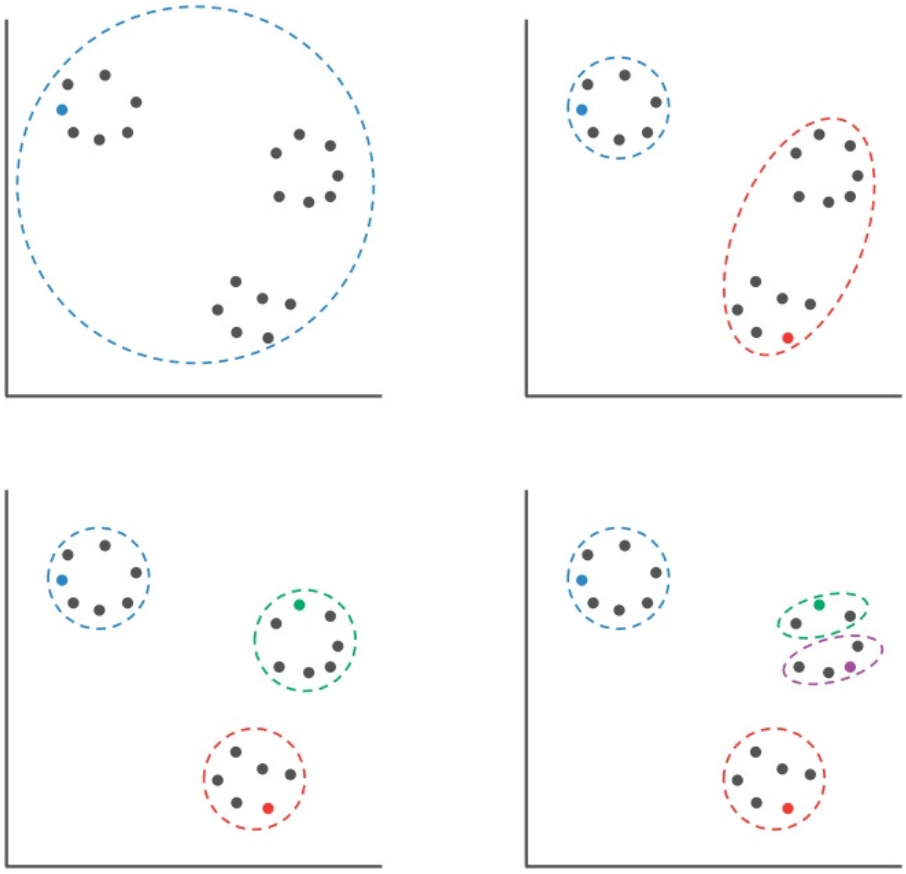
Poglavlje 6

Najdalje prvo prelaženje (eng. *Farthest First Traversal*)

Iako se *k-centers problem grupiranja* može lako izreći, to je zapravo jedan od primjera NP-problema. *Farthest First Traversal* heuristika, čiji je pseudokod prikazan kao Algoritam 6.1, odabire središta grupa iz skupa *Data*. Algoritam odabire proizvoljnu točku kao prvo središte grupe, a potom iterativno dodaje ostale, birajući one točke iz skupa *Data* koje su najudaljenije od već odabranih središta, pri čemu od ponovljenih vrijednosti nasumično bira jednu.

Algoritam 6.1 *FARTHEST FIRST TRAVERSAL*

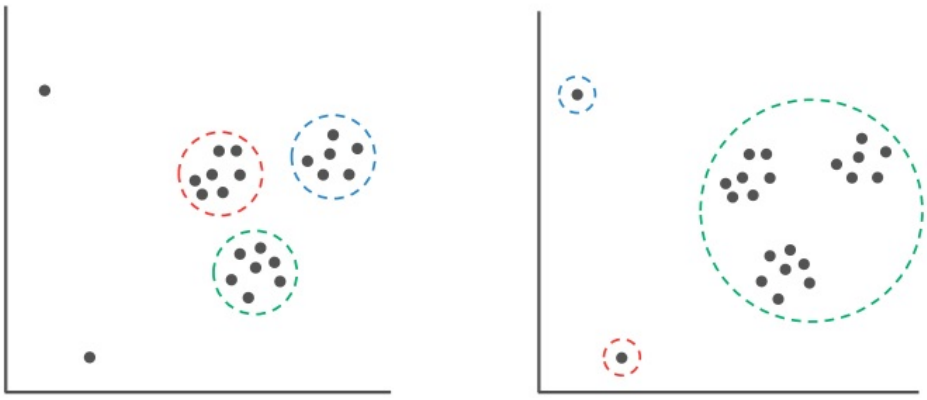
FARTHESTFIRSTTRAVERSAL(Data,k)
Centers ← *skup čiji je član nasumično odabrana točka iz skupa Data*
while $|Centers| < k$ ***do***
    DataPoint ← *točka skupa Data koja maksimizira d(DataPoint,Centers)*
    add DataPoint to Centers
end while
return *Centers*



Slika 6.1: Primjena $FARTHESTFIRSTTRAVERSAL$ heuristike na skup sa slike 4.1 i $k = 4$. (Gore lijevo) Proizvoljna, plavo obojana, točka skupa izabrana je kao prvo središte. Sve točke pripadaju jednoj grupi. (Gore desno) Budući da je najudaljenija od plavo obojane točke, crveno obojana točka izabrana je kao drugo središte. (Dolje lijevo) Nakon što smo za sve točke izračunali minimalne udaljenosti od svakog od dva postojeća središta, uočavamo da točka s najvećom udaljenosti od oba središta je ona obojana u zeleno. Ta točka postaje treće središte. (Dolje lijevo) Analognim postupkom dobijemo četvrto središte koje je obojano ljubičastom bojom.

FARTHESTFIRSTTRAVERSAL brz je algoritam, a njegovo rješenje aproksimira optimalno rješenje *k-centers problema grupiranja*. Međutim, rijetko se koristi u analizi ekspresije gena.

U *k-Center problemu grupiranja*, odabiremo skup $Centers$ tako da njegove točke minimiziraju vrijednost $MAXDISTANCE(Data, Centers)$, tj. maksimum udaljenosti bilo koje točke iz skupa $Data$ i njoj najbližeg središta grupe. Biologe, ipak, često zanima analiza tipičnih devijacija, umjesto onih maksimalnih s obzirom da oni mogu odgovarati *outlierima* koji predstavljaju eksperimentalne pogreške.



Slika 6.2: (Lijevo) Skup podataka s tri lako uočljive grupe i dva *outliera*. (Desno) S obzirom da se $FARTHESTFIRSTTRAVERSAL$ heuristika oslanja na $MAXDISTANCE$ kako bi izračunala nova središta, ako pokušamo podijeliti dane točke u tri grupe, dva outliera bit će izabrana kao dvije jednočlane grupe bez obzira koju točku odaberemo kao prvo središte. U tom slučaju, sve preostale točke bit će dodijeljene jednoj grupi.

Poglavlje 7

k-Means grupiranje (eng. k-Means Clustering)

7.1 Distorzija kvadratne pogreške (eng. *Squared error distortion*)

Kako bismo ukazali na ograničenja funkcije $MAXDISTANCE$, uvest ćemo novu funkciju bodovanja.

Definicija 7.1 *Neka je Data n-člani skup podataka i Centers k-člani skup središta. Distorzija kvadratne pogreške skupova Data i Centers, koju ćemo označavati s* $DISTORTION(Data, Centers)$*, je srednja kvadratna udaljenost točaka iz skupa Data i njihovog najbližeg središta, odnosno*

$$DISTORTION(Data, Centers) = \frac{1}{n} \sum_{DataPoint \in Data} d(DataPoint, Centers)^2.$$

Distorzija kvadratne pogreške dovodi nas do sljedeće izmjene *k-centers problema grupiranja*:

---

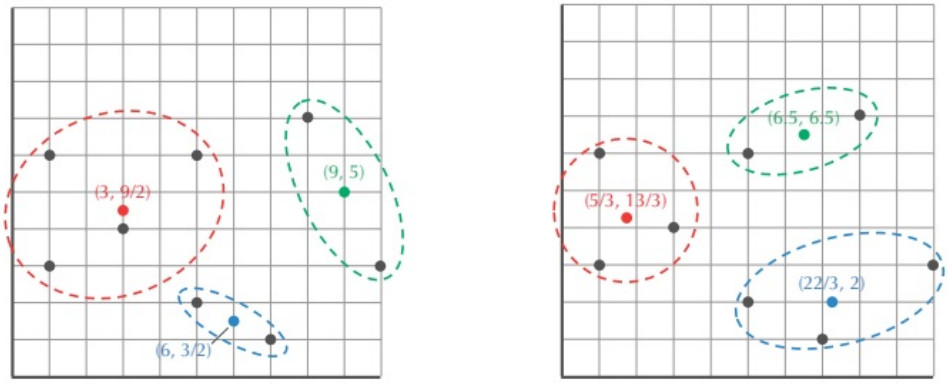
k-means problem grupiranja:

Za dani skup točaka pronađite k središta koji minimiziraju distorziju kvadratne pogreške:

Ulaz: Skup točaka *Data* i prirodni broj k.

Izlaz: k-člani skup *Centers* središta koji minimiziraju udaljenost $DISTORTION(Data, Centers)$ po svim mogućim izborima k središta.

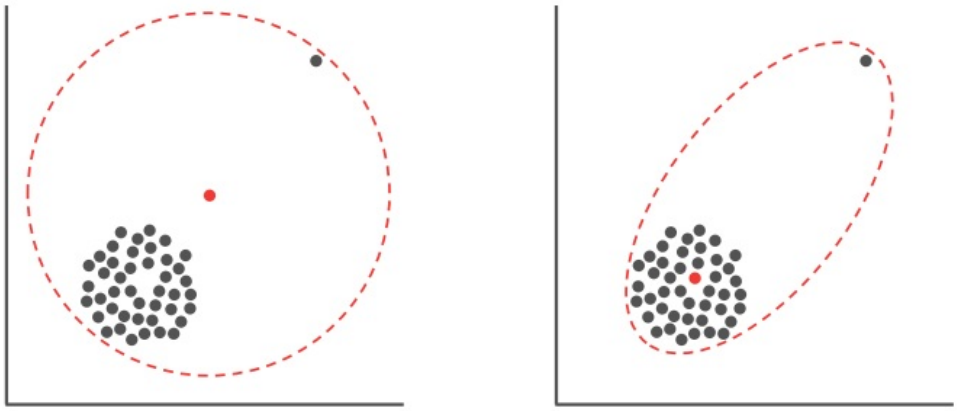
Iako *k-centers* i *k-means problemi grupiranja* izgledaju slično, ne moraju uvijek dati iste rezultate. Ključna razlika između ova dva problema je u tome što u *k-means problemu grupiranja* tzv. *outlieri* mnogo manje utječu na smještaj središta grupa.



Slika 7.1: (Lijevo) Tri obojana središta s pripadnim grupama koja rješavaju *k-Centers problem grupiranja* za zadane točke. (Desno) Tri obojana središta s pripadnim grupama koja rješavaju *k-Means problem grupiranja*.

7.2 *k-means grupiranje* i središte gravitacije

Za $k > 1$ *k-means problem grupiranja* je NP-problem. Međutim, za k=1, problem se svodi na pronalaženje jednog središta x koji minimizira distorziju kvadratne pogreške. Iako je particioniranje skupa u jednu grupu trivijalno, nejasno je kako pronaći jedno središte koje minimizira distorziju kvadratne pogreške. Ovaj jednostavniji problem želimo riješiti kako bismo lakše mogli dizajnirati heuristiku za slučaj kada je $k > 1$.



Slika 7.2: Položaj središta drugačiji je u različitim formulacijama problema grupiranja. (Lijevo) U *k-Centers problemu grupiranja*, središte grupe odabrano je tako da je najveća udaljenost između središta i bilo koje točke u grupi minimizirana. Zbog toga na položaj središta grupe mogu znatno utjecati *outlieri*. (Desno) U *k-Means problemu grupiranja*, utjecaj *outliera* na položaj središta znatno je manji. Ovakvo ponašanje poželjnije je u analizi bioloških podataka, u kojima *outlieri* često odgovaraju pogrešnim podatcima.

Definicija 7.2 ***Središte gravitacije skupa*** *Data definiramo kao točku čija je i-ta koordinata srednja vrijednost i-tih koordinata svih točaka skupa Data.*

Primjer 7.3 *Neka su dane točke (3,8), (8,0) i (7,4). Njihovo središte gravitacije je točka*

$$(\frac{3+8+7}{3}, \frac{8+0+4}{3}) = (6, 4).$$

Teorem 7.4 (Teorem o središtu gravitacije) *Središte gravitacije skupa Data je jedinstvena točka koja za k=1 rješava k-means problem grupiranja.*

Poglavlje 8

Lloydov algoritam

Lloydov algoritam jedan je od najpoznatijih heuristika za rješavanje *k-means problema grupiranja*. Algoritam prvo iz skupa *Data* bira k proizvoljnih središta (tj. k elemenata skupa $Centers$), te potom iterativno izvodi sljedeća dva koraka:

- *Centers to Clusters*: nakon što su središta odabrana, dodjeljuje svaku točku grupi koji odgovara njenom najbližem središtu; ako postoje ponovljene vrijednosti, nasumično bira jednu
- *Clusters to Centers*: nakon što su točke dodijeljene grupama, gravitacijsko središte svake grupe postaje novo središte grupe

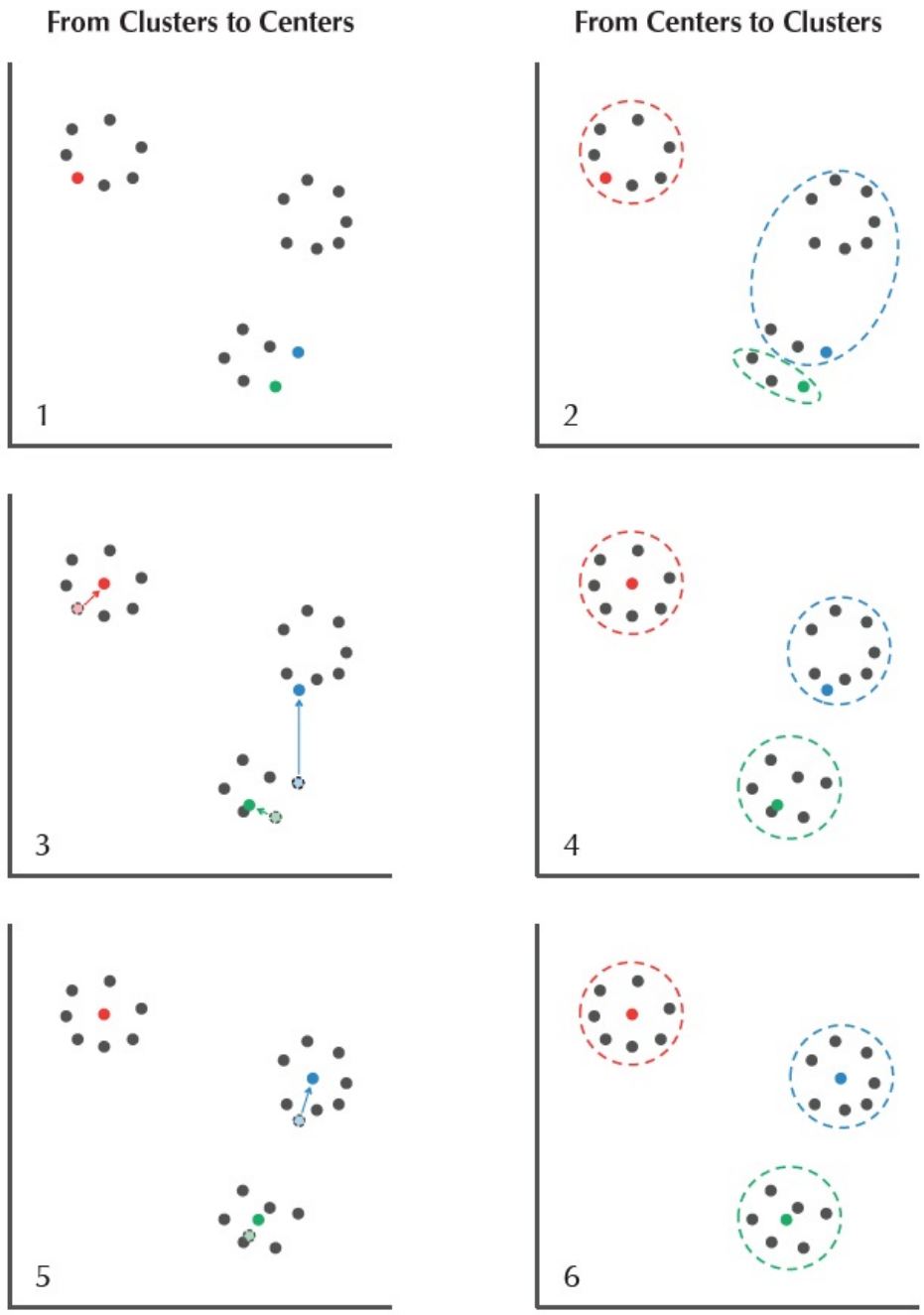
Na slici 8.1 možemo uočiti da se središta sa svakom iteracijom sve manje i manje pomiču. Kažemo da je Lloydov algoritam **konvergira** ako se središta, a samim time i grupe, prestanu mijenjati između iteracija. Ako Lloydov algoritam ne konvergira, potrebno je u bilo kojem koraku smanjiti distorziju kvadratne pogreške zbog jednog od mogućih razloga:

- ako je točka, u koraku *"Centers to Clusters"*, dodijeljena novom središtu,

onda udaljenost te točke od novog središta mora biti manja od udaljenosti te točke od starog središta.

- ako u koraku *"Clusters to Center"* središte postane gravitacijsko središte grupe, onda je po Teoremu o središtu gravitacije ono jedina točka koja minimizira distorziju kvadratne pogreške za sve točke grupe.

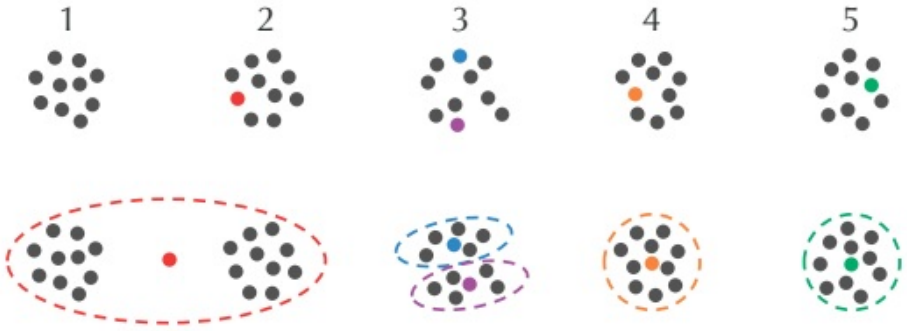
Iako se distorzija kvadratne pogreške smanjuje u svakom koraku, Lloydov algoritam ne mora zbog toga konvergirati. Na primjer, može se dogoditi da distorzija kvadratne pogreške postaje sve manja i manja, što bi dovelo do beskonačnog procesa. Jedan od takvih slučajeva je da se kvadratna distorzije pogreške smanji za 1/2, zatim za 1/4, potom za 1/8, itd. Ipak, takvo nešto se u praksi ne može dogoditi jer broj iteracija u Lloydovom algoritmu ne prelazi broj svih podjela točaka u k grupa.



Slika 8.1: Lloydov algoritam uz dani $k = 3$. Na grafu broj 1, izabiru se tri proizvoljne točke kao središta. Na sljedećim grafovima iterira se *Centers to Clusters* korak, a zatim *Clusters to Centers*. Na posljednjem grafu, grafu 6, Lloydov algoritam konvergira.

8.1 Inicijalizacija Lloydovog algoritma

Slika 8.2 ilustrira da grupiranje može poći po krivu ukoliko ne obratimo pažnju na korak inicijalizacjie Lloydovog algoritma. Kao što je prikazano, ako iz nakupine (eng. *clump*) 1 ne odaberemo ni jedno središte, ali izaberemo dva središta iz nakupine 3, te po jedno središte iz nakupina 2, 4 i 5, nakon prve iteracije Lloydovog algoritma sve točke iz nakupina 1 i 2 bit će dodijeljene crvenom središtu koji se pomakne na otprilike pola između nakupina 1 i 2. Također, nakupina 3 podijelit će točke u dvije grupe, dok će se središta nakupina 4 i 5 pomaknuti prema sredini nakupina u kojima se nalaze. Tada će Lloydov algoritam brzo konvergirati, što će dati netočnim grupiranjem.



Slika 8.2: (Gore) Pet grupa s deset točaka u dvodimenzionalnom prostoru. Lloydov algoritam je inicijaliziran tako da nakupina 1 ne sadrži nijedno središte, nakupina 3 sadrži dva središta, te preostale nakupine sadrže po jedno središte. (Dolje) Lloydov algoritam spojio je točke iz nakupina 1 i 2 u jednu grupu, a nakupinu 3 je podijelio u dvije grupe.

8.2 *k-means++* inicijalizator

Do sada nismo obraćali mnogo pažnje načinu odabira inicijalnih središta u Lloydovom algoritmu koji ih odabire nasumično. Slično heuristici *FARTHESTFIRSTTRAVERSAL*, *k-MEANS++INITIALIZER* odabire k središta jedno po jedno, ali umjesto da bira točke koje su najudaljenije od već odabranih, ono izabire svaku točku nasumično na način da će udaljenije točke imati veću vjerojatnost da budu odabrane. Posebno, vjerojatnost izabiranja središta *DataPoint* iz skupa *Data* je proporcionalna kvadratu udaljenosti točke *DataPoint* od već odabranih središta, tj. bit će proporcionalna s $d(DataPoint, Centers)^2$.

Primjer 8.1 *Pretpostavimo da imamo tri točke te da su njihovi kvadrati udaljenosti od postojećih središta iz skupa Centers jednaki 1, 4 i 5. Tada je vjerojatnost da k-MEANS++INITIALIZER odabere svaku od njih kao sljedeće središte jednako* $\frac{1}{10}$, $\frac{4}{10}$ *i* $\frac{5}{10}$ *redom.*

Algoritam 8.2 *KMEANS++ INITIALIZER*

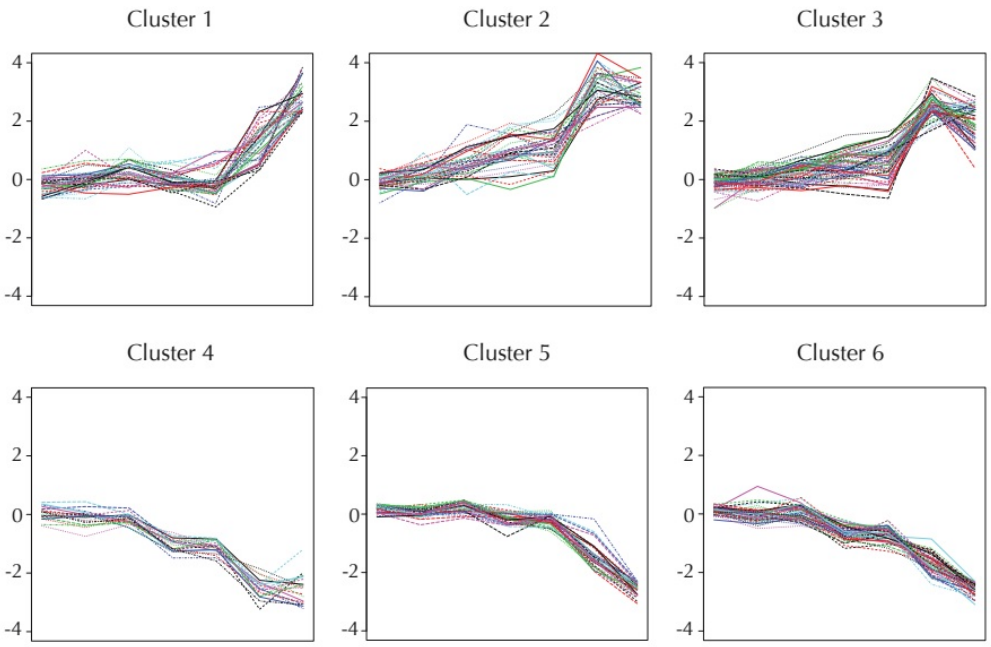
KMEANS++INITIALIZER(Data,k)
Centers $\leftarrow$ *skup čiji je član nasumično odabrana točka iz skupa Data*
while $|Centers| < k$ ***do***
DataPoint $\leftarrow$ *nasumično odabrana točka skupa Data s vjerojatnosti koja je proporcionalna* $d(DataPoint, Centers)^2$
add DataPoint to Centers
end while
return *Centers*

Poglavlje 9

Grupiranje gena uključenih u dijauksični pomak

Budući da odabir biološki bitne vrijednosti k može biti izazovan, mi ćemo (donekle proizvoljno) grupirati 230 gena kvasca u šest grupa. (slika 9.1)

Grafovi na slici 9.1 prikazuju šest obrazaca ponašanja gena uključenih u dijauksični pomak, te postavljaju pitanja za daljnja biološka razmatranja kojima se u ovom radu nećemo baviti. Neki od njih su, primjerice, koji regulacijski mehanizmi prisiljavaju gene iz prve grupe da povećaju svoju ekspresiju. Koji mehanizmi uzrokuju smanjenje ekspresije gena iz četvrte grupe? Kako navedene promjene pridonose dijauksičnom pomaku?

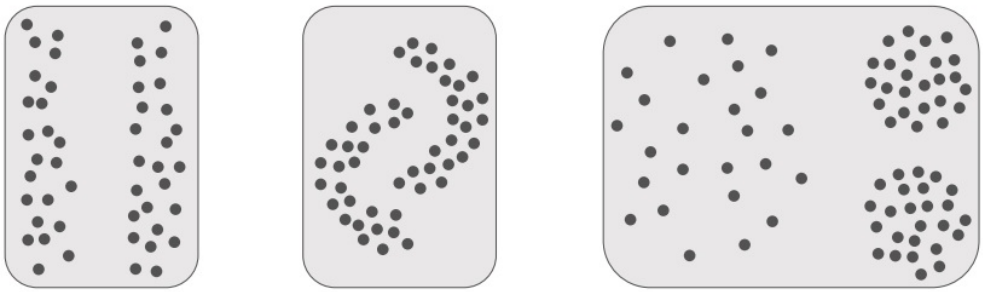


Slika 9.1: Primjenom Lloydovog algoritma (uz $k = 6$) na smanjeni skup podataka koji sadrži 230 gena kvasca dobivamo šest grupa koje otkrivaju šest različitih tipova regulatornog ponašanja. Te grupe redom sadrže 37, 36, 58, 19 i 44 gena. Vektori ekspresija svih gena u svakoj od grupa prikazani su na zasebnim grafovima.

Poglavlje 10

Ograničenja *k-means grupiranja*

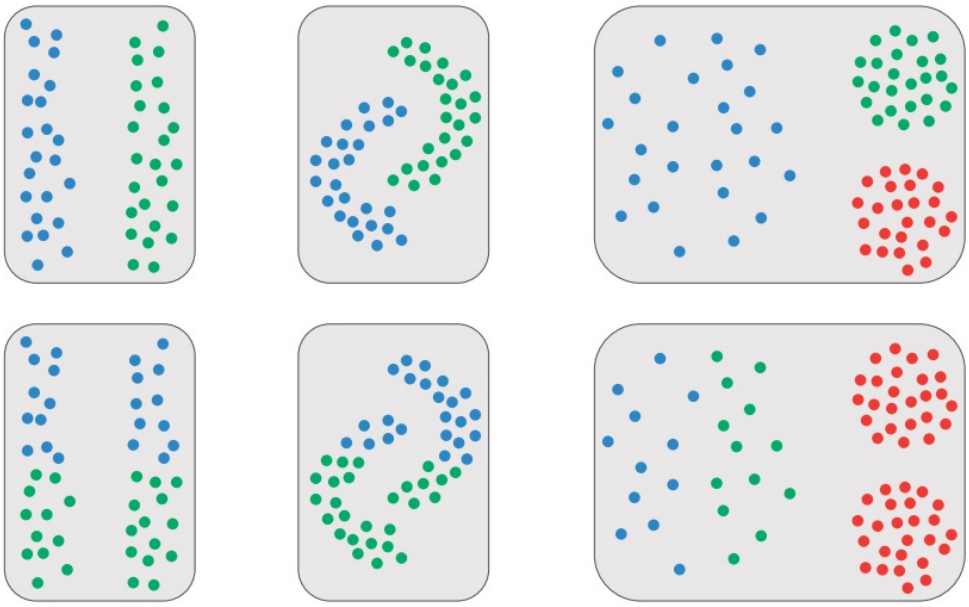
Promotrimo grupe sa slike 10.1.



Slika 10.1: Teži problem grupiranja za $k = 2$ (lijevo i sredina) i $k = 3$ (desno).

U slučaju "izazovnijeg" problema grupiranja, Lloydov algoritam ne može odrediti grupe koji nam se čine očitima. (10.2)

Slabost prethodno definiranog *k-means problema grupiranja* je to što nas ono prisiljava na "strogo" dodjeljivanja svake točke točno jednoj grupi. Ova strategija nema mnogo smisla za točke koje su otprilike jednako udaljene od dva središta.



Slika 10.2: Ljudsko oko (gore) i Lloydov algoritam (dolje) često se ne slažu u slučajevima izduženih skupova (lijevo), skupova neglobularnog oblika (sredina), te skupova s vrlo različitim gustoćama točaka (desno).

Poglavlje 11

Implementacija Lloydovog algoritma

U ovom poglavlju proučit ćemo zadatke BA8A, BA8B i BA8C s web-stranice `https://rosalind.info/` te ujedno prikazati moguća rješenja za svaki od njih. Navedeni zadatci vezani su uz implementacije pomoćnih heuristika i funkcija potrebnih za Lloydov algoritam, kao i za samu implementaciju istog. Napomenimo samo da je dio prikazanog koda prilagođen obliku ulaznih i izlaznih vrijednosti zadanih na spomenutoj web-stranici.

Primjer 11.1 (BA8A: Implementacija *FarthestFirstTraversal* heuristike) *Prisjetimo se heuristike $FarthestFirstTraversal$ iz poglavlja 6.*

$FarthestFirstTraversal$ heuristika ne odabire k središta iz cijelog m-dimenzionalnog prostora, već iz zadanog skupa $Data$, a započinje nasumičnim odabirom jedne od točaka koja postaje prvo središte. Heuristika potom iterativno skupu $Centers$ dodaje nova središta - točke iz skupa $Data$ koje su najudaljenija od prethodno odabranih središta.

Pseudokod prikazan je prethodno kao Algoritam 6.1, a njegovu implementaciju u programskom jeziku Python možemo vidjeti na slici 11.1.

***Ulaz**: Prirodni brojevi k i m, te skup točaka Data u m-dimenzionalnom prostoru*

***Izlaz**: k-člani skup Centers*

```
import math
import numpy as np

def euclidDist(point1,point2):
  r=0
  for i in range(len(point1)):
    r+=(point1[i]-point2[i])**2
  return math.sqrt(r)

def dpCenters(dp,centers):
  dist=list()
  for center in centers:
    dist.append(euclidDist(dp,center))
  mini=min(dist)
  return mini, centers[dist.index(mini)]

def maxDistance(data,centers):
  d=dict()
  for dp in data:
    minDist,center=dpCenters(dp,centers)
    d[tuple(dp)]=minDist
  maxi=max(d.values())
  for key in d.keys():
    if d[key]==maxi:
      point=key
      break
  return point

def farthestFirstTraversal(k,data):
  centers=list()
  centers.append(tuple(data[0]))
  while len(centers)<k:
    point=maxDistance(data,centers)
    if point not in centers: centers.append(point)
  return centers


if __name__=='__main__':
  with open(path,'r') as f:
    input=f.read().splitlines()

  k,m=input[0].split(' ')

  data=input[1:]
  for i in range(len(data)):
    data[i]=data[i].split(' ')
    data[i]=list(np.float_(data[i]))

  res=farthestFirstTraversal(int(k),data)
  for r in res:
    print(*r,sep=' ', file=open('output.txt','a'))
```

Slika 11.1: Implementacija heuristike $FarthestFirstTraversal$

Primjer 11.2 (BA8B: Implementacija funkcije *SquaredErrorDistortion*) *Za zadane skupove Data i Centers definiramo funkciju SquaredErrorDistortion kao prosječnu kvadratnu udaljenost svake točke iz skupa Data od njoj najbližeg središta, tj. točke iz skupa Centers. Sama funkcija Squared Error Distortion pobliže je opisana u poglavlju 7.1.*

***Ulaz**: Prirodni brojevi k i m, te skupovi točaka Centers i Data*

***Izlaz**: Distorzija kvadratne pogreške zaokružena na tri decimale*

```
import math
import numpy as np

def euclidDist(point1,point2):
    r=0
    for i in range(len(point1)):
        r+=(point1[i]-point2[i])**2
    return math.sqrt(r)

def dpCenters(dp,centers):
    dist=list()
    for center in centers:
        dist.append(euclidDist(dp,center))
    mini=min(dist)
    return mini, centers[dist.index(mini)]

def distortion(data,centers):
        sum=0
        for dp in data:
            sum+=(dpCenters(dp,centers)[0])**2
        res=(1/len(data))*sum
        return round(res,3)


if __name__=='__main__':
    with open(path,'r') as f:
        input=f.read().splitlines()

    k,m=input[0].split(' ')

    centers=input[1:1+int(k)]
    for i in range(len(centers)):
        centers[i]=centers[i].split(' ')
        centers[i]=list(np.float_(centers[i]))

    data=input[1+int(k)+1:]
    for i in range(len(data)):
        data[i]=data[i].split(' ')
        data[i]=list(np.float_(data[i]))

    res=distortion(data,centers)
    print(res)
```

Slika 11.2: Implementacija *ComputetheSquaredErrorDistortion*

Primjer 11.3 (BA8C: Implementacija Lloydovog algoritma) *Lloydov algoritam najpoznatija je heuristika za $k-MeansClustering$ problem. Heuristika prvo iz skupa Data odabire k točaka koji postaju središta grupa, odnosno elementi skupa $Centers$), a potom iterativno izvršava korake $CenterstoClusters$ i $ClusterstoCenters$, koji su kao i cijeli Lloydov algoritam pobliže opisani u poglavlju 8.*

***Ulaz**: Prirodni brojevi k i m, te skup točaka Data u m-dimenzionalnom prostoru*

***Izlaz**: k-člani skup $Centers$ dobiven primjenom Lloydovog algoritma na skup Data, pri čemu je prvih k točaka tog skupa odabrano kao prvih k središta.*

```python
import numpy as np
import math
from BA8A import euclidDist

def dpCenters(dp,centers):
    dist=list()
    for center in centers:
        dist.append(euclidDist(dp,center))
    mini=min(dist)
    return dist.index(mini)

def centersToClusters(data,centers):
    indC=list(dpCenters(dp,centers) for dp in data)
    return indC

def clusterToCenter(data,centers,i):
    assignedClusters=centersToClusters(data,centers)
    d=list()
    for dp,cluster in zip(data,assignedClusters):
        if i==cluster:
            d.append(dp)
    mean=np.round(np.mean(np.array(d), 0),3)
    return mean

def clustersToCenters(data,centers,k):
    newCenters=[clusterToCenter(data,centers,i) for i in range(k)]
    return newCenters
```

```
def lloyd(data,centers,k):
    flag=False
    while not flag:
        oldCenters=centers
        centers=clustersToCenters(data,centers,k)
        if np.array_equal(centers,oldCenters): flag=True
    return centers


if __name__=='__main__':
    with open(path,'r') as f:
        input=f.read().splitlines()

    k,m=input[0].split(' ')
    k=int(k)
    m=int(m)
    k,m=input[0].split(' ')
    k=int(k)
    m=int(m)
    data=input[1:]
    centers=input[1:1+k]

    for i in range(len(centers)):
        centers[i]=list(map(float, centers[i].split(' ')))

    for i in range(len(data)):
        data[i]=list(map(float,data[i].split(' ')))


    for el in lloyd(data,centers,k):
        print(*["%.3f" % (koord) for koord in el], sep=' ',file=open('output.txt','a'))
```

Slika 11.3: Implementacija Lloydovog algoritma

Zaključak

U diplomskom radu istražen je *k-means* algoritam te njegova primjena u bioinformatici, preciznije, u analizi ekspresije gena.

U prvom dijelu rada dana je motivacija te uvod u neke od osnovnih pojmova i procesa iz biologije, a svojevrsna matematička pozadina u drugom dijelu. U radu su, također, razmotrena i ograničenja *k-means* algoritma, te Lloydov algoritam kao jedan od najpoznatijih primjera *k-means* algoritama. Uz to, dani su primjeri implementacija heuristika i funkcija vezanih uz Lloydov algoritam, te primjer implementacije samog algoritma.

Na temelju navedenog vidjeli smo da se *k-means* algoritam pokazao kao vrijedan alat pri analizi ekspresije gena. Osim toga, algoritam je ujedno brzo i jednostavno rješenje velikog skupa različitih problema, te stoga ne čudi što ima široku primjenu u raznim područjima.

Literatura

[A] Phillip Compeau, Pavel Pevzner, Bioinformatics Algorithms: An Active Learning Approach, Vol II

[B] Phillip Compeau, Pavel Pevzner, Bioinformatics Algorithms: An Active Learning Approach, Vol I

[C] Anonymous, (2023), Proteom. [Internet], raspoloživo na:`https://en.wikipedia.org/wiki/Recombination_hotspot`, [17.9.2023.].

[D] Anonymous, (2021), Recombination hotspot. [Internet], raspoloživo na: `https://hr.wikipedia.org/wiki/Proteom`, [17.9.2023.].

[E] Anonymous, (2023), Transcriptome. [Internet], raspoloživo na: `https://en.wikipedia.org/wiki/Transcriptome`, [18.9.2023.].

[G] Dan Halligan, (2022). [Internet], raspoloživo na: `https://github.com/danhalligan/rosalind.info/blob/main/rosalind/bioinformatics_textbook_track/ba8c.py`, [20.12.2023.].

[H] Anonymous, (2024). [Internet], raspoloživo na: `https://en.wikipedia.org/wiki/K-means_clustering`, [26.3.2024.].

Sve slike, osim onih iz poglavlja 11, preuzeta su iz [A] i [B].