KMESA algoritam klasterovanja

(K-Means with Simulated Annealing)

1. Uvod

Algoritam klasterovanja K-sredina (K-Means) je jedan od najpopularnijih algoritama klasterovanja zbog svoje jednostavnosti, interpretabilnosti i efikasnosti nad određenim skupovima podataka. Jedan je od najvažnijih predstavnika modela sa prototipom, tj. modela zasnovanog na reprezentativnim predstavnicima koji se u ovom slučaju nazivaju centroidi. Performanse algoritma u velikoj meri zavise od početnog koraka, tj. od inicijalizacije početnih centroida. U slučaju nepogodne inicijalizacije algoritam može da se „zaglavi“ u lokalnim optimumima i završi sa suboptimalnim klasterovanjem, što je neželjena pojava koju je potrebno izbeći. U prevazilaženju ovog problema razvijene su mnogobrojne modifikacije algoritma, a jedna od njih biće detaljno razrađena u daljem tekstu.

* 1. Detalji

Za rad algoritma neophodno je navesti tačan broj klastera k i inicijalizovati početne centroide, koji predstavljaju odgovarajuće klastere u skupu podataka. Zatim se algoritam izvršava iterativno: u svakoj iteraciji centroidi se ažuriraju na pozicije srednjih tačaka tekućih klastera koje predstavljaju, a zatim se vrši dodela tačaka iz skupa podataka najbiližim centroidima. Dodela tačaka klasterima (tj. reprezentativnim centroidima) zasniva se na funkciji različitosti (eng. disimilarity function) koja računa rastojanje tačaka do svih centroida – tačka se dodeljuje onom klasteru čijem centroidu je ona najbliža. Ova funkcija rastojanja je najčešće euklidska, ali može biti i neka druga. Iterativni postupak se ponavlja sve dok nije dostignut maksimalan broj iteracija ili dok nije ispunjen kriterijum konvergencije.

Iako algoritam nenadgledanog učenja, algoritam K-sredina ima svoj optimizacioni objektiv koji pokušava da minimizuje. Neka je ,, ..., dimenzionalni skup tačaka koje je potrebno klasterovati a ,, ..., oznake klastera sa odgovarajućim centroidima ,, ..., . Optimizacioni objektiv zasniva se na sumi srednjekvadratnih grešaka (eng. Sum of Squared Error, SSE) definisane sa

|  |  |  |
| --- | --- | --- |
|  | , | (1) |

koja predstavlja sumu kvadrata euklidskoj rastojanja tačaka iz posmatranog klastera od svog reprezentativnog predstavnika, tj. centroida . Ovaj izraz definiše grešku klastera . Na osnovu ovog izraza optimizacioni izraz postaje minimizacija ove greške po svim klasterima

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

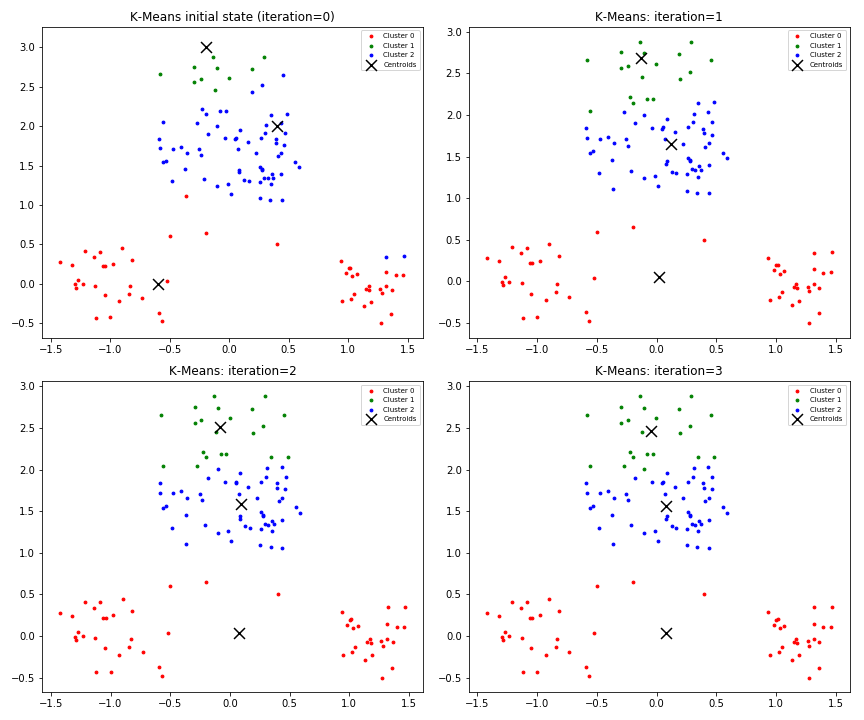
Sada je moguće jednostavno predstaviti pseudokod algoritma:

|  |
| --- |
| Algoritam: *K-sredina* |
| Ulaz: Skup tačaka ,, ..., za klasterovanje |
| Izlaz: Skup centroida ,, ..., kao reprezentativnih predstavnika  pronađenih klastera |
|  |
| Inicijalizuj početni skup centroida ,, ..., ; |
| *Dok* nije ispunjen kriterijum zaustavljanja *ponavljaj*: |
| Izvršti dodelu tačaka ,, ..., odgovarajućim klasterima na osnovu  predefinisane funkcije rastojanja; |
| Ažuriraj pozicije tekućih centroida ,, ..., u cilju minimizacije  optimizacionog izraza; |
| Vrati ,, ..., kao rešenje; |

Jasno je da korak ažuriranja centroida u cilju minimizacije (2) povlači postavljanje  
vrednosti tekućeg centroida na središnju tačku klastera :

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

gde je broj tačaka u klasteru , a predstavlja atribut tačke iz skupa podataka po kome se računa srednja vrednost. Po ovom principu ažuriranja centroida algoritam je i dobio ime K-sredina. Međutim, na osnovu iterativnog pristupa ovaj postupak ne garantuje pronalazak globalno najamnje vrednosti (2), u zavisnosti od početne inicijalizacije centroida, zbog čega je moguće da algoritam ovu vrednost minimizuje samo lokalno (tačnije, minimizacijom pojedinačne greške SSE u okviru tekućih klastera) čime se dobija suboptimalno klasterovanje, što ilustruje sledeći jednostavan primer:



Primer 1: Problem suboptimalnog klasterovanja kod algoritma K-sredina

Već u trećoj iteraciji algoritam je dostigao kriterijum konvergencije (u primeru, uslov konvergencije je da razlika u distanci odgovarajućih centroida bude manja od unapred zadatog praga tolerancije, tj. ). Očigledno je da izvršeno klasterovanje nije zadovoljavajuće i da nije dostignut globalni minimum izraza (2) već lokalni, odnosno došlo je do suboptimalnog klasterovanja.

* 1. Modifikacija algoritma

Za rešenje problema prikazanog u primeru 1 razvijene su mnogobrojne modifikacije algoritma K-sredina, a neki od njih su K-sredina sa primenom Lojdovih metoda, K-sredina sa filterom, Dvoslojni algoritam K-sredina, Brzi modifikovani globalni algoritam K-sredina sa inkrementalnom konstrukcijom klastera, Modifikovani pojačani algoritam K-sredina, Primena simuliranog kaljenja nad algoritmom K-sredina u kombinovanom modelovanju, itd. Ovaj rad koristi ideju simuliranog kaljenja za modifikaciju algoritma, spomenutu u poslednjem navedenom radu. Detalji ove modifikacije biće izložene u narednoj sekciji.

1. Simulirano kaljenje i K-sredina
   1. Uvod

Simulirano kaljenje (eng. Simulated Annealing) je probabilistička tehnika za aproksimaciju

globalnog optimuma date funkcije cilja. Specijalno, ono predstavlja metaheuristiku za aproksimaciju globalne optimizacije u velikom prostoru pretrage nekog optimizacionog problema. Naziv potiče od kaljenja u metalurgiji, tehnike koja podrazumeva zagrevanje i kontrolisano hlađenje metala zarad povećanja veličine njegovih kristala i umanjivanje unutrašnjih defekata.

U algoritmu simuliranog kaljenja ideja sporog hlađenja metala interpretira se kao postepeno smanjivanje verovatnoće prihvatanja lošijih rešenja u prostoru pretrage, čime se dopušta iscrpnija pretraga za globalnim optimumom. Uopšteno, algoritam vrši lokalnu pretragu za boljim rešenjem i ažurira svoje tekuće rešenje ukoliko je bolje rešenje pronađeno, ili ako je temperatura (tj. verovatnoća) prihvatanja lošijeg rešenja veća od nekog nasumičnog broja, generisanog iz uniformne (0,1) raspodele. Temperatura vremenom opada, tj. verovatnoća prihvatanja lošijih rešenja se vremenom smanjuje i algoritam se bliži konvergenciji.

* 1. Osnovni koncept

Za implementaciju simuliranog kaljenja nad algoritmom K-sredina, osnovna ideja ovog rada je da se simulirano kaljenje primeni nad centroidima nakon njihovog standardnog ažuriranja prema izrazu (3). Na ovaj način se zapravo prihvataju centroidi koji su lošiji u smislu minimizacije optimizacionog cilja (2), što je i osnovni koncept simuliranog kaljenja u smislu pretrage prostora rešenja za globalnim optimumom. Na ovaj način kaljeni centroidi „lutaju“ u potrazi za globalnim minimumom optimizacionog izraza. Izbor novog centroida, odnosno dodatna modifikacija centroida, zavisi od broja koji predstavlja verovatnoću prihvatanja kaljenog centroida u odnosu na standardno ažurirani centroid preko središnje tačke u klasteru. Ovaj broj se računa preko predefinisane opadajuće funkcije koja zavisi od tekuće iteracije algoritma. Pri velikom broju iteracija ova verovatnoća postaje sve manja pa se i centroidi sve manje „kale“, odnosno algoritam postaje sve bliskiji standardnom algoritmu K-sredina, što dovodi do skore konvergencije.

Dakle, modifikacija algoritma K-sredina u tekućoj iteraciji sastoji se u sledećim koracima (pri čemu se smatra da su centroidi ,, ..., već ažurirani prema izrazu (3)):

1. Izračunaj verovatnoću kaljenja svakog centroida na osnovu predefinisane funkcije u zavisnosti od tekuće iteracije algoritma;
2. Za svaki centroid generiši broj iz uniformne raspodele;
3. Ukoliko za centroid važi , generiši dimenzionalni vektor kaljenja i ažuriraj tekući centroid prema formuli:

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

gde je težina kaljenja koja može biti unapred fiksirana, ili biti izražena preko predefinisane funkcije koja zavisi od tekuće iteracije, definisane na sličan način kao i funkcija . Na ovaj način kaljeni centroidi mogu dovesti do pogoršanja sume srednjekvadratne greške u tekućoj iteraciji, ali se postiže fleksibilnost u izbegavanju lokalnih minimuma i povećava verovatnoća pronalaska globalnog minimuma.

Postavlja se pitanje: u slučaju primene simuliranog kaljenja, kako ažurirati tekući centroid, odnosno kako izračunati vektor kaljenja ? Ovaj postupak je metaheuristika koja predstavlja samu suštinu modifikovanog algoritma i biće detaljnije diskutovan u daljem tekstu.

2.3 Vektor kaljenja

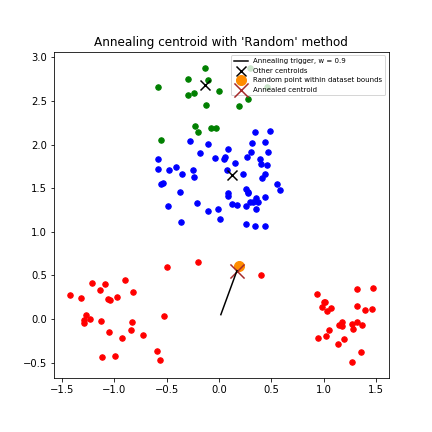
Vektor iz izraza (4) nazvan je ovde vektor kaljenja jer se računa samo u slučaju kada se simulirano kaljenje primenjuje na tekući centroid i jer upravo on utiče na promenu položaja centroida u cilju potrage za optimalnim klasterovanjem. Ovaj vektor se izračunava preko direkcione tačke (vektora) koja se generiše u dimenzionalnom prostoru, koja zapravo određuje pravac u kome će se centroid kaliti, odnosno kretati. Tako za centroid vektor kaljenja se jednostavno računa kao razlika direkcione tačke i tekuće vrednosti centroida:

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

U dimenzionalnom prostoru pretrage postoji beskonačno mnogo ovakvih vektora (tačaka ) i u slučaju njegovog proizvoljnog odabira algoritam bi više ličio na slučajnu pretragu. Stoga je vrlo važno da ovaj vektor bude pogodno izabran i da u obzir uzme relevantne informacije algoritma, kao što su npr. tekuće vrednosti centroida, vrednosti tačaka iz skupa podataka, i sl. Ovde će biti diskutovane neke ideje za generisanje vektora kaljenja koje su implementirane u radu. Metode će biti prikazane u sledećem slučaju. Posmatrajmo primer 1 i pretpostavimo da se nalazimo u prvoj iteraciji. Posmatrajmo donji centroid na pozicji koji je već ažuriran prema izrazu (3). Iz primera znamo da će se algoritam zaglaviti već u naredne dve iteracije (za zadati prag tolerancije ). Primenićemo različite metode kaljenja ovog centroida radi prikaza same ideje, dok će kompletan rad algoritma sa simuliranim kaljenjem nad istim skupom podataka biti detaljnije prikazan u sekciji 3.

2.3.1 Nasumično kaljenje

Najjednostavnije rešenje za generisanje vektora kaljenja je nasumično kaljenje (eng. Random annealing) – generiše se nasumična direkciona tačka iz prostora pretrage u okviru granica datog skupa podataka, izračuna vektor kaljenja prema (5) a zatim centroid pomeri u pravcu direkcione tačke, u zavisnosti od težine kaljenja , kao što je ilustrovano na sledećoj slici:



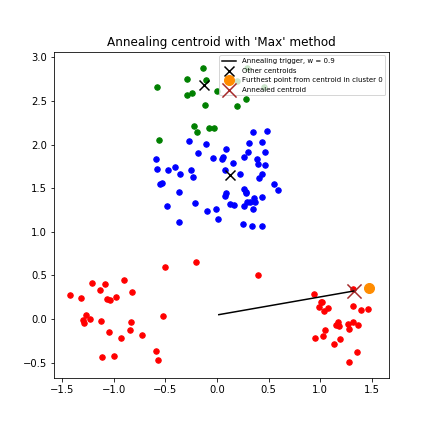
Primer 2.1: Nasumično kaljenje centroida

Narandžasta tačka na slici je nasumično generisana tačka prema kojoj se centroid “kreće” sa unapred fiksiranom težinom . Da li nasumično kaljenje stvarno pomaže videćemo u sekciji 3.

Napomena. Nakon ažuriranja centroida pomoću simuliranog kaljenja oznake klastera bi trebalo da se promene, što nije prikazano na slici. U prikazu ostalih metoda kaljenja ovo će takođe biti izostavljeno. Promena oznaka klastera (ponovna dodela tačaka klasterima) nakon simuliranog kaljenja će biti detaljnije obrađena u sekciji 3.

2.3.2 Maksimalno kaljenje

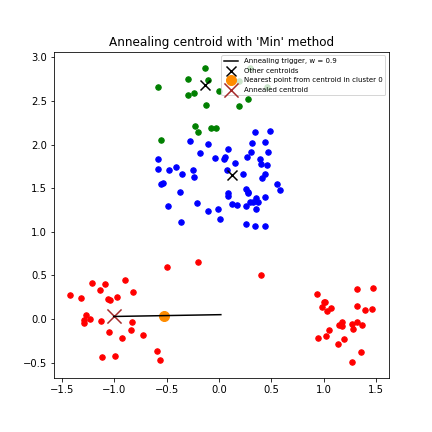
U maksimalnom kaljenju (eng. Max annealing) direkciona tačka je najudaljenija tačka iz klastera od centroida koji predstavlja taj klaster.



Primer 2.2: Maksimalno kaljenje centroida

2.3.3 Minimalno kaljenje

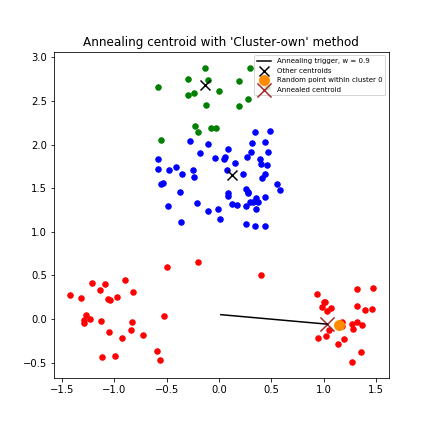
U minimalnom kaljenju (eng. Min annealing) direkciona tačka je najbliža tačka iz klastera u odnosu na centroid koji predstavlja taj klaster. S obzirom na to da se radi o najbližoj tački, u ovom slučaju minimalno kaljenje se razlikuje od ostalih metoda po tome što se centroid ne kreće prema direkcionoj tački za vektor , već za vektor čime centroid “preskače” direkcionu tačku, kao što je prikazano na slici:



Primer 2.3: Minimalno kaljenje centroida

2.3.4 Kaljenje sopstvenog klastera

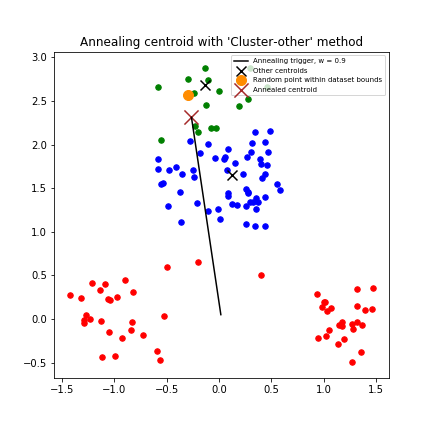
Kod kaljenja sopstvenog klastera (eng. Cluster-own annealing) direkciona tačka je nasumična tačka iz klastera čiji se centroid kali. Dakle, radi se o tački iz stvarnog skupa podataka koja pripada o tom klasteru, a ne o bilo kojoj tački iz tog opsega vrednosti za odgovarajuće atribute.



Primer 2.4: Metod kaljenja sopstvenog klastera

2.3.5 Kaljenje drugog klastera

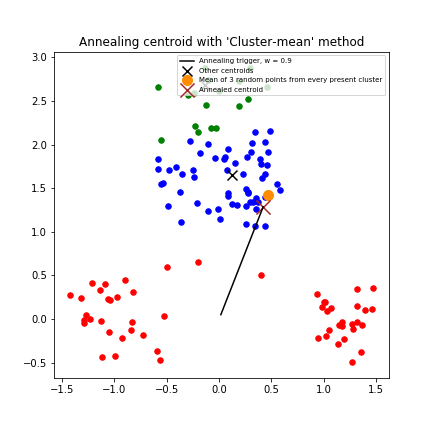
Nasuprot metodu kaljenja sopstvenog klastera, u metodu kaljenja drugog klastera (eng. Cluster-other annealing) direkciona tačka je nasumično odabrana tačka iz nekog drugog klastera, tj. iz klastera koji ne predstavlja centroid koji se kali. U sledećem primeru tačke kandidati za direkcionu tačku su plave i zelene tačke, ali ne i crvene.



Primer 2.5: Metod kaljenja drugog klastera

2.3.6 Kaljenje sredine klastera

Metod kaljenja sredine klastera (eng. Cluster-mean annealing) podrazumeva odabir nasumične tačke iz svakog klastera. Direkciona tačka se računa kao sredina ovih nasumično odabranih tačaka.

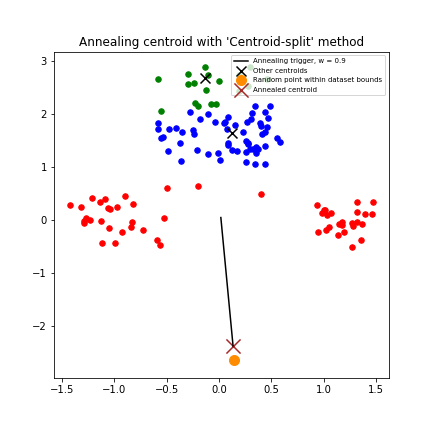


Primer 2.6: Metod kaljenja sredine klastera

Ovaj metod ima tendeciju da približuje centroide sredini skupa podataka.

2.3.7 Kaljenje razdvajanja centroida

U pristupu kaljenja koje razdvaja centroide (eng. Centroid-split annealing) računaju se distance od centroida koji se kali do svih ostalih centroida. Bira se centroid koji je najbliži tekućem centroidu koji se kali, a direkciona tačka je na suprotnoj strani od centroida :

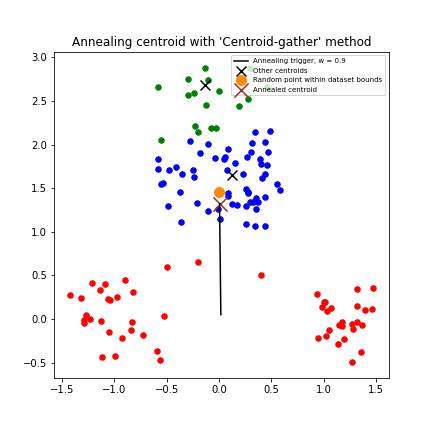


Primer 2.7: Kaljenje razdvajanja centroida

Kod ovog metoda postoji rizik da se neki centroidi znatno udalje od skupa tačaka, pa im u narednoj iteraciji neće biti dodeljena nijedna tačka iz skupa podataka, što dovodi do stvaranja praznog klastera. O ovom problemu biće više reči kasnije.

2.3.8 Kaljenje sakupljanja centroida

Kod kaljenja sakupljanja centroida (eng. Centroid-gather annealing) direkciona tačka je središnja tačka tekućih centroida.



Primer 2.8: Kaljenje sakupljanja centroida

Ovaj metod ima tendenciju, kao što je i očekivano, da centroide približuje jedan drugom. Za vrednost ovi centroidi se stapaju u jednu tačku, što nije poželjna situacija.

2.3.9 Ostali metodi

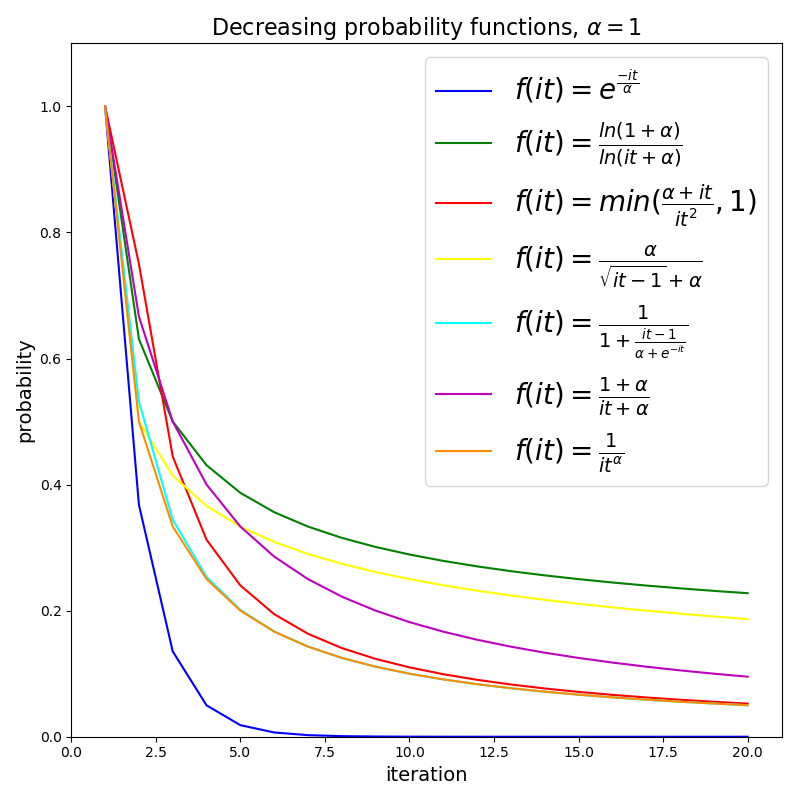
Pored navedenih, u ovom radu implementirana su još dva metoda: Maxmin kaljenje (eng. Maxmin annealing) koje kod neparnih iteracija primenjuje maksimalno kaljenje, a kod parnih minimalno kaljenje, i Kaljenje vrteške (eng. Carousel annealing) koje u svakoj iteraciji nasumično bira jedan od navedenih metoda a zatim ga primenjuje nad centroidima.

Pored navedenih, sigurno je moguće izvesti i neke kompleksnije metode kaljenja centroida koje bi u obzir uzimale i prethodne iteracije, promenu centroida u distancama, interklasterno rastojanje, promene u varijansi klastera itd. Ipak, ograničićemo se na navedene metode koje su dobre za testiranje same ideje simuliranog kaljenja u njenoj primeni nad algoritmom K-sredina. Ukoliko jednostavnije metode pokažu pozitivne indikacije, onda se isplati uložiti dodatni napor u implementaciju kompleksnijih metoda i u unapređenje samog algoritma.

U narednoj sekciji, sve ove metode biće najpre primenjene nad primerom 1, a zatim ćemo testirati algoritam nad kompleksnijim skupovima podataka. Ali pre toga ostaje još da prodiskutujemo o dva važna tehnička detalja.

2.4 Verovatnoća kaljenja

Kao što je ranije spomenuto, verovatnoća kaljenja izražena je preko opadajuće funkcije koja u zavisnosti od tekuće iteracije algoritma izračunava verovatnoću kaljenja centroida. Na narednoj slici prikazane su implementirane funkcije sa dodatnim hiperparametrom α koji funkciji daje određenu fleksibilnost i reguliše njen nagib:



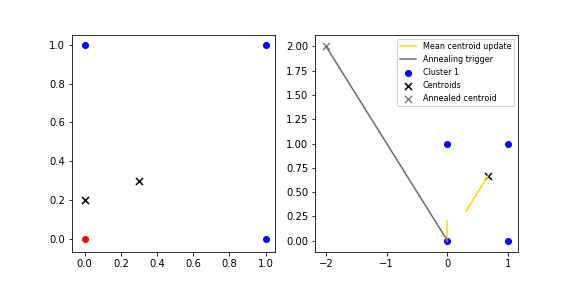
Za veće vrednosti hiperparametra funkcije sporije opadaju:

|  |  |
| --- | --- |
|  |  |

Od izbora ove funkcije i odgovarajućeg hiperparametra α zavisiće učestalost kaljenja centroida a samim tim i rad čitavog algoritma.

2.5 Problem praznog klastera

Podsetimo se da se korak dodele tačaka klasterima u algoritmu K-sredina zasniva na funkciji rastojanja – tačka se dodeljuje onom klasteru čijem je centroidu najbliža. Centroidi se potom ažuriraju na novu vrednost tako što se računaju kao središnja tačka klastera koga predstavljaju, a zatim se postupak ponavlja. Međutim, šta ako se dogodi da je neki centroid suviše daleko te mu nijedna tačka ne bude dodeljena, odnosno sve tačke budu dodeljene nekim drugim, bližim centroidima? Ovaj slučaj dovodi do praznog klastera pa se centroid bez dodeljenih tačaka ne može ažurirati, što potpuno obustavlja normalno funkcionisanje algoritma. Do ovog problema najčešće dolazi prilikom početne inicijalizacije centroida. Ukoliko su početni centroidi dobro inicijalizovani, kod standardnog algoritma K-sredina neće doći do ovakvog slučaja. Međutim, u modifikovanog algoritmu neki centroidi zbog kaljenja mogu da zauzmu nepogodnu poziciju u prostoru pretrage pri čemu im u sledećoj iteraciji neće biti dodeljena nijedna tačka. To znači da u modifikovanom algoritmu verovatnoća da dođe do problema praznog klastera je znatno veća. Ilustrujmo ovaj slučaj na jednostavnom primeru.



U inicijalnom stanju prvi centroid (centroid na poziciji ) predstavlja crveni klaster, a drugi plavi klaster. Nakon koraka ažuriranja prema središnjoj tački klastera (na drugom grafiku prikazano žutom linijom) oznake klastera se ne bi promenile. Međutim, u slučaju da se prvi centroid kali, na primer, nasumičnim kaljenjem i završi u tački , u datom koraku neće mu biti dodeljena nijedna tačka jer je svakoj tački bliži drugi centroid, pa su sve pridružene plavom klasteru. Na ovaj način se gubi crveni klaster i došlo je do problema praznog klastera.

Zbog navedenog neophodno je da se ovaj problem adresira i u projektu implementiran je pod nazivom razrešavanje praznih klastera (eng. Empty Clusters Resolution, ECR). Konkretno, implementirano je najjednostavnije rešenje: ažuriraj “prazni” centroid na neku nasumičnu tačku iz prostora ptretrage sve dok mu ne bude dodeljena bar jedna tačka iz skupa. Naravno, postoje i mnogo povoljnije metode, kao što su zamena centroida tačkom koja najviše učestvuje u SSE, nasumičnom tačkom iz klastera sa najvećim SSE, najudaljenijom tačkom, itd.

Napomena. Ovi metodi su u radu implementirani ali se algoritam iz nekog razloga zaglavljuje u beskonačnoj petlji. Zbog toga je primenjen najjednostavniji pristup.

Prikaz razrešavanja praznih klastera biće prikazan usput, u celokupnom radu algoritma.

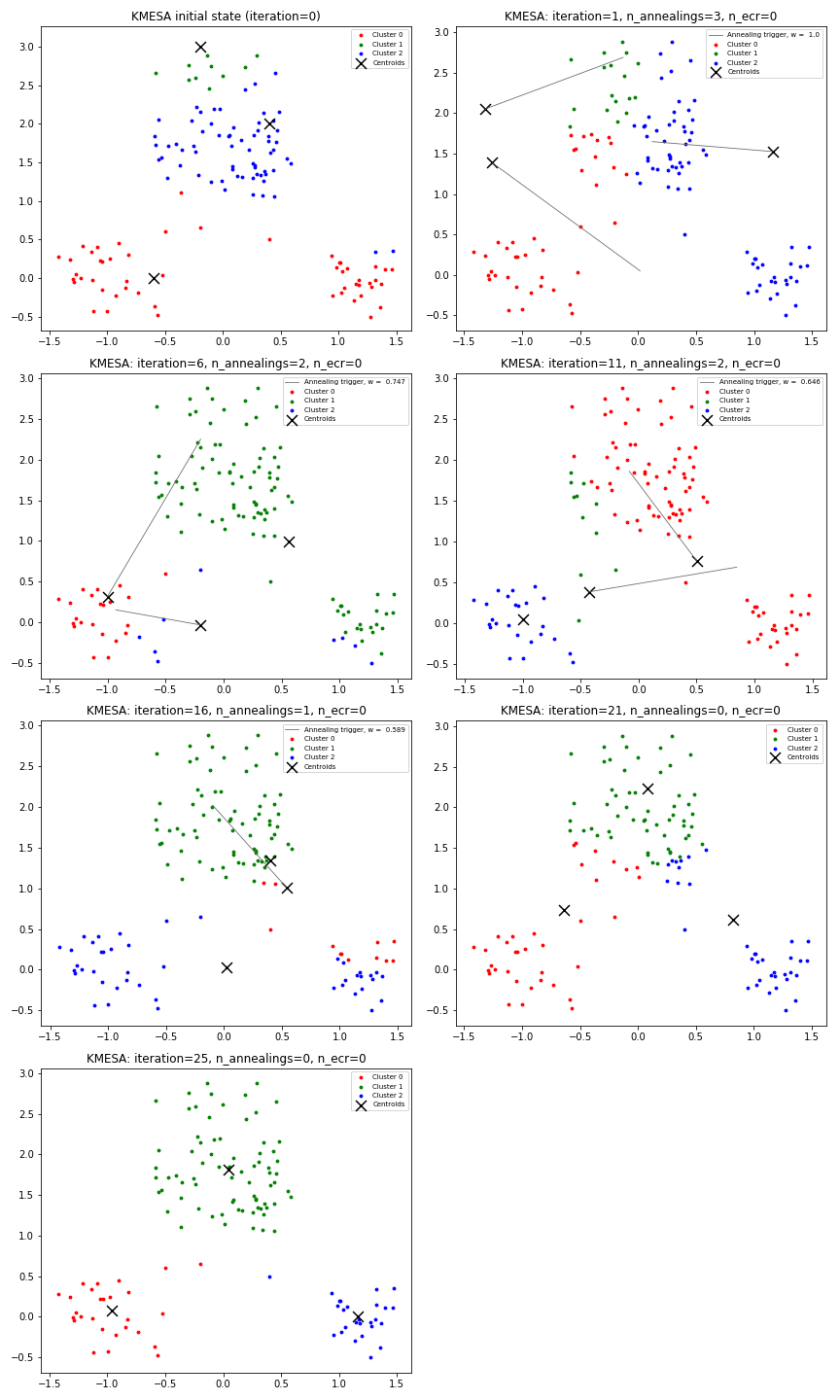
3. Testiranje algoritma

Sada pređimo na testiranje algoritma na različitim primerima. Svi detalji o parametrima, hiperparametrima, ukupnom broju izvršenih kaljenja centroida, razrešavanja praznih klastera, proteklom vremenu izvršavanja algoritma i sl. mogu se pogledati u odgovarajućim Jupyter datotekama na Github repozitorijumu.

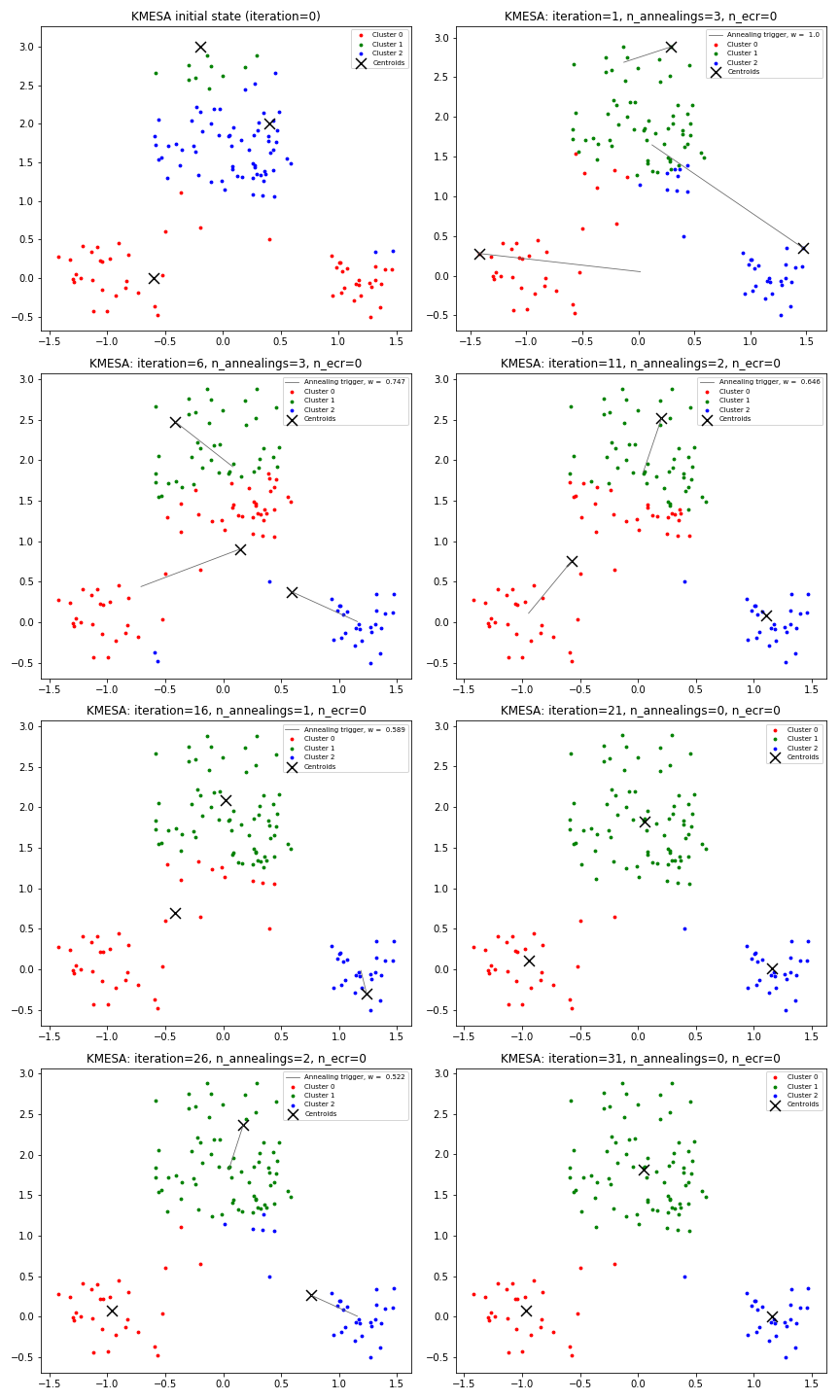
3.1 Jednostavan skup

Primenićemo modifikovani algoritam najpre na jednostavnom skupu podataka iz primera 1. Primenićemo sve metode kaljenja centroida kako bismo posmatrali njihove rezultate i pogodno odabrali najbolje za primenu nad kompleksnijim podacima. Izabraćemo početne centroide baš kao u primeru 1 da bismo najpre na najjednostavnijem primeru utvrdili da li simulirano kaljenje uopšte pomaže. Radi ilustracije, ovde će biti prikazani samo metodi nasumičnog kaljenja, maksimalnog kaljenja i kaljenja sa sakupljanjem centroida. Rad algoritma sa ostalim metodima može se pogledati na zvaničnog repozitorijumu projekta na putanji img/testing/simple\_3\_clusters.

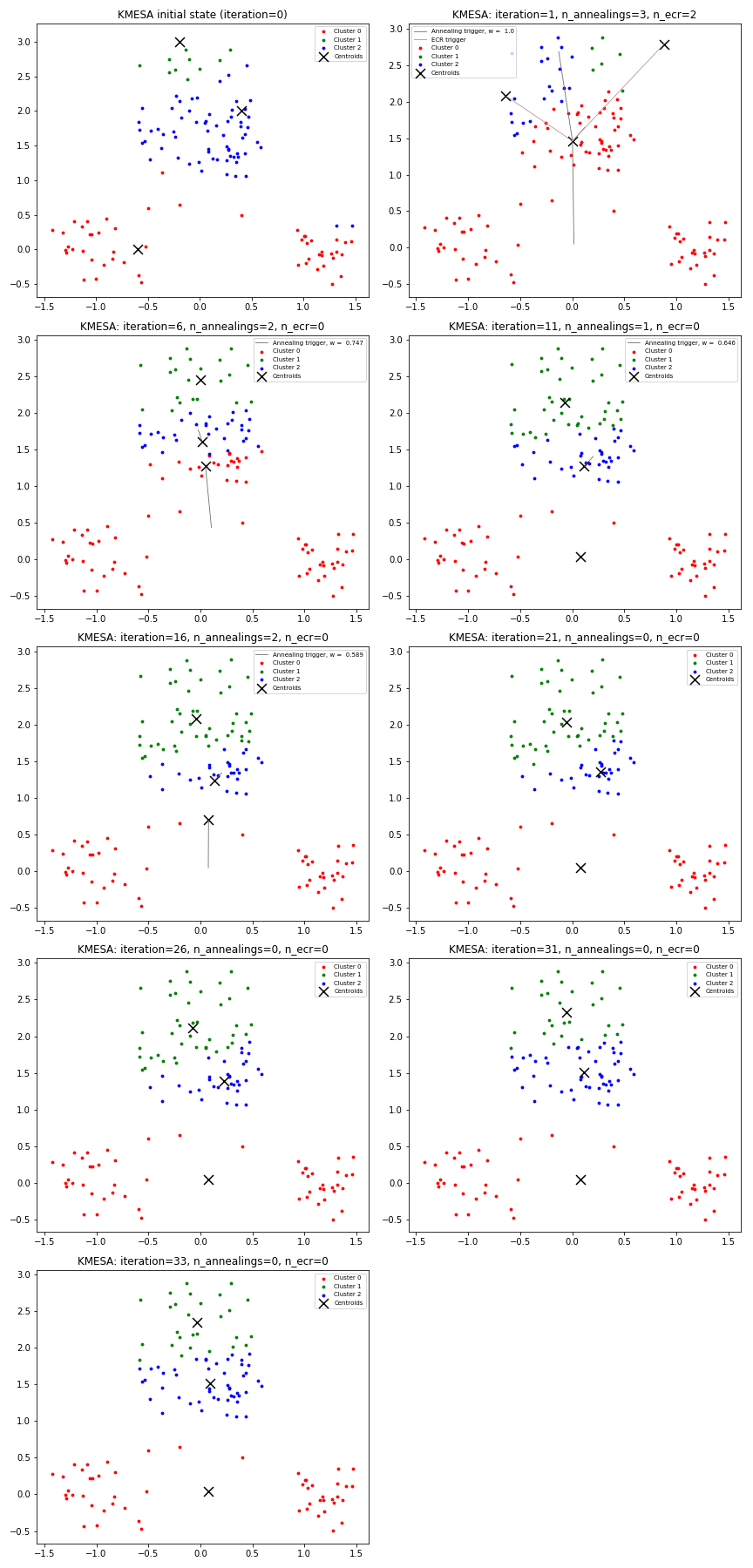
3.1.1 Klasterovanje sa nasumičnim kaljenjem



3.1.2 Klasterovanje sa maksimalnim kaljenjem



3.1.3 Klasterovanje sa kaljenjem sakupljajućih centroida

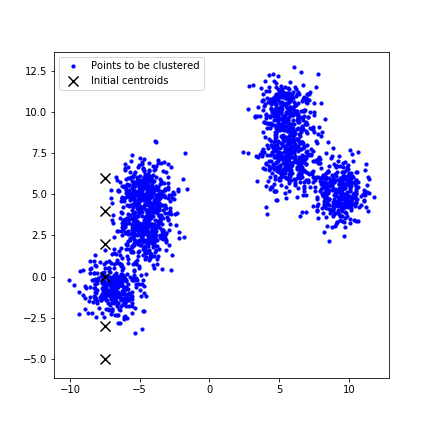


3.1.4 Zaključak

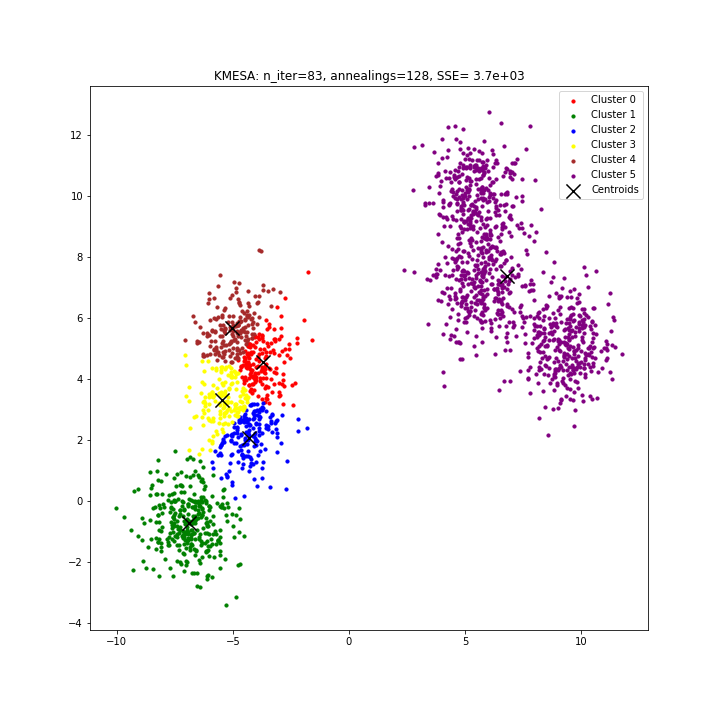
Ispostavilo se da su metodi nasumičnog i maksimalnog kaljenja doveli algoritam do globalnog minimuma, ali kaljenje sa sakupljajućim centroidima nije. Na pomenutoj putanji se mogu pogledati i ostali metodi – svi su uspeli da pronađu globalni minimum u vrlo malom broju iteracija!

3.2 Blisko povezani klasteri

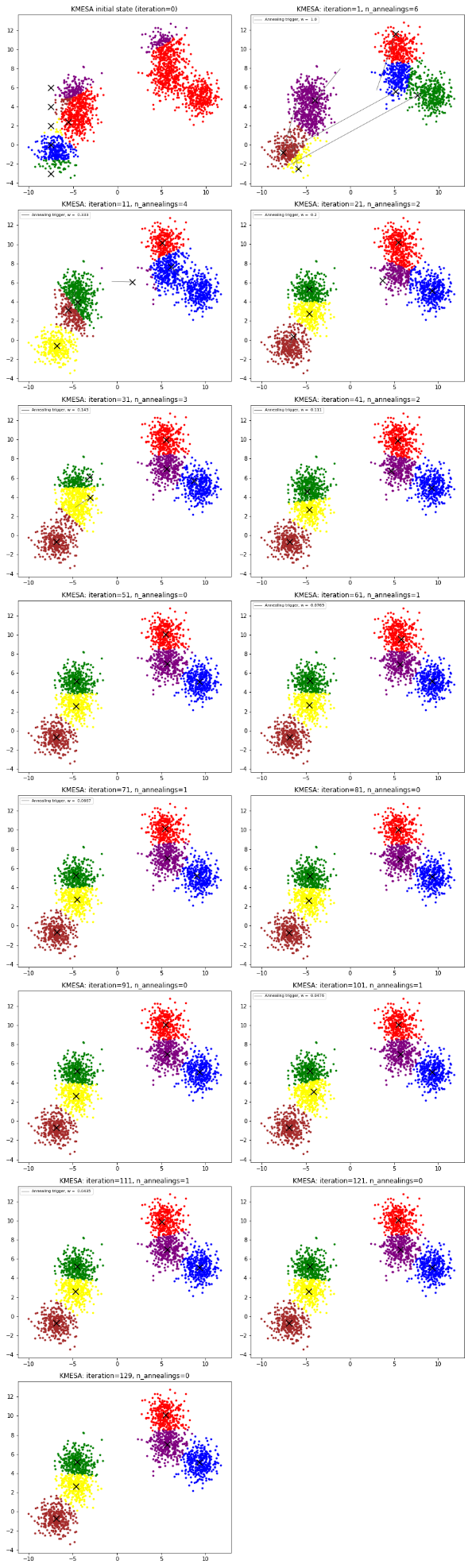
Neka je dat skup od 2000 tačaka sa fiksno postavljenim centroidima kao na slici:



Ispostavlja se da algoritam K-sredina ne uspeva da pronađe optimalno klasterovanje:



Primenimo sada modifikovani algoritam sa metodom kaljenja drugog klastera:



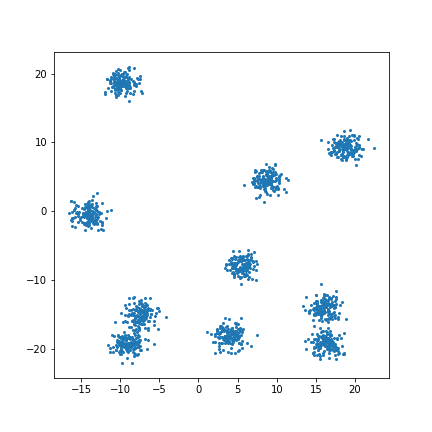
Simulirano kaljenje je uspelo da „iščupa“ algoritam iz lokalnih minimuma i na kraju ga dovode do optimalnog klasterovanja. Ipak, metoda Maxmin kaljenja to ne uspeva i završava sa istim klasterovanjem kao i običan algoritam K-sredina.

Na putanji img/testing/6\_blobs nalazi se još skup podataka sa 6 klastera. Na njemu se običan algoritam K-sredina takođe zaglavljuje, a modifikovani algoritam sa kaljenjem drugog klastera uspeva da pronađe globalni minimum optimizacionog izraza.

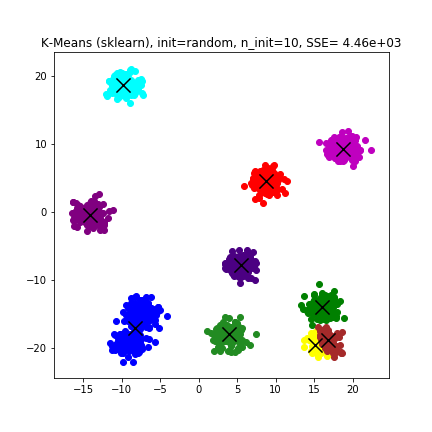
3.3 Dobro razdvojeni klasteri

Sledeći skup podataka sastoji se od 1500 tačaka koje su nešto rasprostranjenije i koje čine dobro razdvojenih 10 klastera. Radi provere efektivnosti simuliranog kaljenja metoda za incijalizaciju centroida je nasumična. Razlog za to je sledeći: ako je ideja simuliranog kaljenja zaista efektivna, onda bismo želeli da ona pomogne algoritmu da pronađe globalni minimum bez obzira na početne centroide. Iz tog razloga nije korišćen napredniji pristup u inicijalizaciji početnih centroida (u biblioteci sklearn pod nazivom k-means++).

Pogledajmo najpre kako izgleda skup podataka:



Da bismo proverili prednost modifikovanog algoritma u odnosu na standardni algoritam K-sredina, nad datim skupom pokrenut je standardni algoritam sa nasumičnom inicijalizacijom centroida, i to 10 puta. Od svih 10 puta najbolje pronađeno rešenje (sa najmanjim SSE) je sledeće:



Očigledno je da standardni algoritam nije uspeo da pronađe optimalno klasterovanje.

Zatim je pokrenut modifikovani algoritam za sve vrste kaljenja centroida, sa po dve reinicijalizacije. Ovo su dobijeni rezultati:

|  |  |
| --- | --- |
| Figure : Random annealing | Figure : Max annealing |
| Figure : Min annealing | Figure : Cluster-own annealing |
| Figure : Cluster-other annealing | Figure : Cluster-mean annealing |
| Figure : Centroid-split annealing | Figure : Centroid-gather annealing |
| Figure : Maxmin annealing | Figure : Carousel annealing |

Dakle, samo maksimalno kaljenje i kaljenje vrteške uspele su da pronađu optimalno klasterovanje. Možda je bilo moguće doći do najboljeg rešenja i nekim drugim metodama pogodnim odabirom parametara kao što su verovatnoća kaljenja i težina kaljenja, ali je za to neophodno duže i pažljivije testiranje, koje će ovde biti preskočeno.

3.4 Veliki skup tačaka

Primenićemo ovde klasterovanje nad velikim skupom podataka od 100.000 tačaka razdvojenih u 20 klastera.