UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Nejc Ilc

PRIMERJAVA METOD ZA RAZVRŠČANJE VZORCEV V GRUČE

Diplomska naloga
na univerzitetnem študiju

Mentor: prof. dr. Andrej Dobnikar

Ljubljana, 2009

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
fax: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01531/2008
Datum: 15.12.2008

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **NEJC ILC**

Naslov: **PRIMERJAVA METOD ZA RAZVRŠČANJE VZORCEV V GRUČE**
COMPARISON OF THE METHODS FOR PATTERN CLUSTERING

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Namen naloge je prikazati nekaj najbolj znanih in uspešnih metod za razvrščanje vzorcev v gruče, kot so 'k-means', ECMC, EM GMM in CSC, ter analizirati njihove uspešnosti z notranjimi in zunanjimi kriteriji na različnih problemskih domenah. Na osnovi takšne analize je mogoče podati vrednotenje omenjenih metod za posamezne tipične probleme, kar lahko bistveno vpliva na odločitev za določeno metodo razvrščanja za vsak primer posebej.

Mentor:

prof. dr. Andrej Dobnikar

Dekan:

prof. dr. Franc Solina

Namesto te strani **vstavite** original izdane teme diplomskega dela s podpisom mentorja in dekana ter žigom fakultete, ki ga diplomant dvigne v študentskem referatu, preden odda izdelek v vezavo!

Zahvala

Za usmerjanje in vse koristne nasvete pri izdelavi diplomske naloge se zahvaljujem: mentorju, prof. dr. Andreju Dobnikarju, doc. dr. Branku Šteru, doc. dr. Urošu Lotriču in Jerneju Zupancu. Posebna zahvala velja Robertu Jenssenu, ki je dal na razpolago svojo kodo algoritma CSC.

Velika hvala tudi družini in vsem bližnjim, ki so me potrpežljivo prenašali v času pisanja.

Vsem vzorcem, ki so se pustili razvrstiti.

Kazalo

Seznam uporabljenih kratic in simbolov

x_i	i-ti vzorec, podatek
$x_i^{(d)}$	d-ta razsežnost (atribut) i-tega vzorca
P	množica vseh vzorcev
N	število vzorcev v množici P, $\|P\|$
D	razsežnost vzorca $x \in P$
C	množica vseh gruč; razvrstitev
K	število vseh gruč v razvrstitvi C
N_{C_i}	število vzorcev v gruči C_i
$\mathrm{sim}(x, y)$	mera podobnosti med vzorcema x in y
$\mathrm{dis}(x, y)$	razdalja med vzorcema x in y
U	matrika pripadnosti vzorcev gručam

Povzetek

Na področju strojnega učenja velja razvrščanje vzorcev v gruče za osnovno disciplino, ki pa je zaradi obilice med seboj različnih problemskih domen za reševanje zelo težak problem. Raziskovalci so se ga lotevali na mnogo načinov, kar ima za posledico veliko število različnih metod in postopkov. Poleg tega je razvrščanje vzorcev v gruče izredno slabo definiran pojem, predvsem kar se tiče ocenjevanja pravilnosti rezultata. Ni namreč univerzalnega ali enotnega merila, ki bi najbolje določal podobnost med vzorci v isti gruči in ločeval dobre razvrstitve od slabih.

V diplomski nalogi obravnavamo štiri izbrane metode za razvrščanje vzorcev, od katerih ima vsaka svoje zanimive posebnosti. To so: KMC, ECMC, EM GMM in CSC. Poleg tega se na področju razvrščanja pojavljajo vedno novi kriteriji ocenjevanja uspešnosti, ki so sami po sebi tudi predmet medsebojne primerjave. Namen je opraviti vsestransko analizo načina delovanja izbranih metod in objektivno ovrednotiti rezultate razvrščanja posameznih tipičnih problemskih domen. Uporabljenih je šest zunanjih in štirje notranji kriteriji ocenjevanja, na podlagi katerih je podana končna ocena uspešnosti posameznih metod in ovrednotena njihova uporabnost v konkretnih primerih. Množice vzorcev, nad katerimi je bilo opravljeno razvrščanje, so bile izbrane tako, da odražajo tipične probleme na tem področju.

Končni izsledki primerjave kažejo, da uporaba znanja iz teorije informacije, ki ga izkorišča novejša metoda CSC, pripomore k boljšemu rezultatu glede na izbrane kriterije in množice vzorcev. Prav tako se odpira še precej možnosti za njeno izboljšavo in je hkrati tudi motivacija za uporabo drugih pristopov k reševanju problema razvrščanja.

Ključne besede:

razvrščanje vzorcev, primerjava metod, notranji kriteriji, zunanji kriteriji

Abstract

Clustering or cluster analysis is a fundamental machine learning task, which is, unfortunatelly, an ill-posed problem, caused by large diversity of problem domains. Many different approaches have been used to solve it, which consequently reflects as a long list of clustering methods. Moreover, it is hard to determine, which clustering of particular data is better than another, because there does not exist an universal similarity metric, which would be the most appropriate for all different problems.

In the thesis, four chosen methods for clustering are being examined, each of which has its interesting features. These are: KMC, ECMC, EM GMM in CSC. In addition, new criteria for the evaluation of clustering correctness appear, which are inherently subject to a peer comparison. My intention was to carry out a comprehensive analysis of the chosen methods and objectively evaluate the results of the clustering of individual typical problem domain. To achieve this, four internal and six external evaluation criteria or indices were used. On their basis final evaluation of the effectiveness of various methods is given. Several synthetic and real data sets on which the clustering has been performed out have been selected to reflect the typical problems in this field.

The final results of the comparison shows that the application of knowledge of information theory, which exploits novel CSC method, contribute to a better outcome depending on the selected criteria and the data sets. It also opens up considerable potential to continue its improvement and is also the motivation for using alternative approaches to solve the clustering problem.

Key words:

clustering, method comparison, internal indices, external validation

Poglavje 1

Uvod

Zlobna mačeha je zažugala Pepelki in ji z glasom, ki je kipel od zaničevanja in prezira siknila: „Medtem ko bom s tvojima lepima polsestrama neznansko uživala na princevem plesu, boš ti, ničvrednica, delala tole!“ Vzela je posodo z raznobarvnimi zrni leče in jo stresla v ognjišče, kjer je bilo polno pepela. „Ko se vrnemo, mora biti vsa leča prebrana in to po barvah! Gorje ti, če bo vmes tudi kaj črvive!“ se je še glasilo navodilo mačehe. In pa: „Uživaj!“ Pepelka je s strtim srcem zrla v zmešnjavo pred seboj.

V vsakdanjem življenju se pogosto srečujemo s podobnim problemom kot Pepelka – z razvrščanjem vzorcev v gruče. Ljudje smo nagnjeni k temu, da podobne stvari združujemo skupaj, ali kot radi pravimo, mečemo v isti koš. Primerov, ko naši izjemni možgani opravljajo razvrščanje objektov v skupine, je ogromno, največkrat se tega procesa niti ne zavedamo. Znan primer je denimo razvrščanje kemijskih elementov v periodni sistem. Razvrščanje poteka tudi, ko skupino ljudi pred seboj nezavedno razdelimo na prijazne, brezbrižne in morebitno nevarne osebe. Vendar obstajajo številni problemi, ko je objektov ali vzorcev za razvrščanje enostavno preveč ali pa imajo preveliko število lastnosti, da bi jih človek uspešno razvrstil še za časa svojega življenja. Predstavljajte si samo, da bi radi vse gradivo v mestni knjižnici, ali raje na svetovnem spletu, razvrstili po tematiki, ali denimo vso kdajkoli posneto glasbo po zvrsteh. Za človeka je to precej dolgotrajen in mučen postopek, zato je veliko truda vloženega v razvoj avtomatskih postopkov, ki razvrstijo dane objekte oziroma vzorce v skupine oziroma gruče. Ali si bo tudi Pepelka iz uvodne zgodbe omislila stroj, ki bo namesto nje opravil zamudno nalogo razvrščanja, pa je stvar njene pripravljenosti, da se zgodba srečno konča. Če bo namreč

celo noč čepela pred ognjiščem, si bo princ na plesu izbral drugo dekle, mogoče celo njeno polsestro.

V svetu stroke je razvrščanje vzorcev v gruče (angl. clustering) opredeljeno kot osnovno področje strojnega učenja, kjer vzorce ali podatke razvrstimo v gruče ali skupine na podlagi medsebojne podobnosti [2]. Področje razvrščanja vzorcev se ukvarja z ugotavljanjem povezav med vzorci, preiskovanjem njihove strukture in raziskovanjem skritega znanja v podatkih. Zato spada pod okrilje širokega pojma razpoznavanja vzorcev (angl. pattern recognition) kot tudi v zadnjih letih modernega izraza podatkovnega rudarjenja (angl. data mining). Statistiki imajo področje razvrščanja opredeljeno kot posebno disciplino znotraj multivariatne analize [5].

Razvrščanje vzorcev v gruče je nenadzorovan (angl. unsupervised) proces, kjer vnaprejšnje znanje o vzorcih ni zbrano, kar pomeni, da ne poznamo primerov že uvrščenih ali označenih vzorcev in tudi ni vnaprej definiranih gruč. To je pomembna razlika, ki ločuje razvrščanje od uvrščanja oziroma klasifikacije vzorcev (angl. classification). Slednje deluje kot nadzorovano učenje, kjer je število skupin vnaprej znano in prav tako razvrstitev določenega dela vzorcev, ki sestavljajo učno množico.

Uporaba metod za razvrščanje vzorcev je zelo široka. Uporabimo jih lahko ob prvem srečanju z neznanimi podatki, ko želimo ugotoviti njihovo medsebojno povezanost in najti značilne oblike, poteze, morebitna odstopanja posameznih vzorcev. Primerna je tudi za zgoščevanje podatkov, ko izmed množice vzorcev izluščimo zgolj najbolj tipične skupine in na ta način poenostavimo problem. Opravljanje taksonomske analize je prav tako problem pisan na kožo hierarhičnemu razvrščanju v gruče (primer: razvrščanje živali v debla, razrede, redove, družine, rodove in vrste). Znanje s področja razvrščanja vzorcev v gruče se s pridom uporablja v medicini pri analizi genskega materiala, kot pomoč pri diagnozi bolezni glede na značilne simptome, v oglaševanju pri iskanju skupin potrošnikov s podobnim vedenjskim vzorcem, v knjižnicah in na svetovnem spletu pri razvrščanju besedil po tematiki, pri preučevanju potresov za ugotavljanje nevarnih območij glede na koncentracijo epicentrov, pri segmentaciji slik in drugje.

Kmalu, ko začnemo razmišljati o razvrščanju vzorcev, se pojavi ključno vprašanje: kako najbolje opraviti to nalogo? In takoj zatem: kaj sploh pomeni najbolje opraviti nalogo razvrščanja? Nedvoumno odgovoriti na ti dve vprašanji je žal nemogoče, ker ni enotnega merila, ki bi povezovalo podobne vzorce med seboj in ločilo različne. V primeru Pepelke iz naše zgodbe bi lahko denimo bledo rumeno in rumeno-zeleno lečo razvrstili v gručo z isto barvo ali pa v dve gruči, ker bi mislili, da sta ti dve barvi različni. Prav tako bi bilo

lahko navodilo mačehe, naj razvrsti lečo na podlagi velikosti. Rezultat bi bil najbrž popolnoma drugačen. Kaj od tega je potemtakem bolje?

Namen mojega dela je primerjati izbrane algoritme za razvrščanje med seboj z izbranimi kriteriji za ocenjevanje uspešnosti in ugotoviti morebitne prednosti določenih algoritmov pred ostalimi.

Cilji dela so naslednji: predstaviti delovanje izbranih algoritmov za razvrščanje vzorcev v gruče in njihova implementacija, opis in uporaba notranjih in zunanjih kriterijev ocenjevanja razvrstitev vzorcev med sabo in izbira reprezentativnih testnih množic vzorcev, nad katerimi bo izvedeno razvrščanje.

V naslednjem poglavju podajam formalizacijo problema razvrščanja in definicije osnovnih pojmov na tem področju. Nato sledi opis izbranih metod za razvrščanje, notranjih in zunanjih kriterijev za ocenjevanje uspešnosti razvrščanja in končno predstavitev množic vzorcev, ki bodo uporabljene za primerjavo metod med seboj. V poglavju 4 je predstavljena izvedba primerjave in njeni rezultati, ki so sprotno komentirani. Na koncu sledijo še sklepne ugotovitve, ki povzemajo prednosti in slabosti izbranih algoritmov in nakazujejo možnosti za nadaljnje delo na tem področju.

Poglavje 2

Problem razvrščanja

Postavimo sedaj natančnejšo definicijo problema razvrščanja vzorcev v gruče [4]:

> Denimo, da imamo množico vzorcev P, ki vsebuje posamezne vzorce x_i, tako da velja: $P = \{x_1, \ldots, x_N\}$, kjer je N število vseh vzorcev. Vzemimo celo število K, ki pomeni število gruč, v katere bomo razvrščali vzorce x_i. Problem razvrščanja je torej najti preslikavo
>
> $$f : P \to \{1, \ldots, K\} \quad , \tag{2.1}$$
>
> kjer je vsak vzorec x_i razvrščen v natanko eno gručo C_j, $1 \leq j \leq K$. Torej je
>
> $$C_j = \{x_i | f(x_i) = C_j, 1 \leq i \leq N \text{ in } x_i \in P\} \quad . \tag{2.2}$$
>
> Rezultat razvrščanja v gruče je množica gruč, ki ji pravimo tudi razvrstitev $C = \{C_1, C_2, \ldots, C_K\}$.

Naj omenim, da ta definicija ne zajema posebnih primerov, ko se lahko gruče medsebojno prekrivajo, kar pomeni, da lahko posamezni vzorec hkrati pripada več gručam. Takih primerov v tem delu ne bom obravnaval in za vse vključene algoritme postavljena definicija velja.

Vzorce razvrščamo v gruče po nekem ključu ali kriteriju – temu bomo v nadaljevanju rekli mera podobnosti (angl. similarity measure). Mero podobnosti med dvema vzorcema x_i in x_j bomo označevali kot $\text{sim}(x_i, x_j)$. Podobnosti obratna količina pa je razdalja (angl. distance), označena kot $\text{dis}(x_i, x_j)$ in se v literaturi pogosteje uporablja. Posamezni vzorec mora biti tako bolj podoben vzorcu iz iste gruče kot nekemu iz druge gruče. To lastnost gruče (recimo

ji C_j) lahko matematično formalno zapišemo kot: $\forall x_l, x_m \in C_j$ in $x_i \notin C_j$ velja

$$\mathrm{sim}(x_l, x_m) > \mathrm{sim}(x_l, x_i) \quad . \tag{2.3}$$

Pripadnost posameznega vzorca določeni gruči izraža matrika pripadnosti U velikosti $N \times K$ z elementi u_{ij}, za katere velja

$$u_{ij} = \begin{cases} 1 \ ; & x_i \in C_j, \\ 0 \ ; & x_i \notin C_j \end{cases} \quad . \tag{2.4}$$

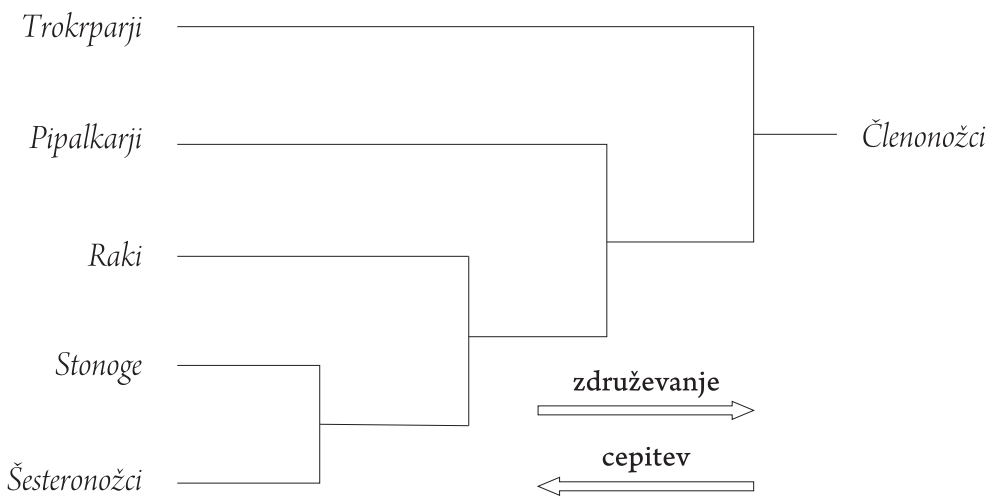
Izziv in cilj raziskovanja je torej najti postopek oziroma algoritem, ki bo vzorce razvrstil v gruče na način, da bodo znotraj gruč vzorci, ki so si na nek določen način podobni, podobnost med gručami pa naj bo čim manjša. Ker imamo ljudje različne poglede na to, kaj si je podobno, se je tekom let razvilo veliko število algoritmov za razvrščanje vzorcev in skupaj z njimi tudi raznih mer podobnosti med vzorci. Kako lahko algoritme razdelimo glede na način, ki ga ubirajo pri razvrščanju, si bomo ogledali v naslednjem podpoglavju, nato bomo osvetlili tudi pojem mere podobnosti in razdalje.

2.1 Delitev algoritmov za razvrščanje

V grobem lahko algoritme za razvrščanje vzorcev v gruče razdelimo na hierarhične (angl. hierarchical) in delitvene (angl. partitional).

Hierarhični postopki zgradijo celo strukturo množic gruč; na najvišjem nivoju so vsi vzorci v eni skupni gruči, na najnižjem pa je vsak vzorec svoja gruča. Znotraj enega nivoja pripada po definiciji vsak vzorec natanko eni gruči. Hierarhični postopki se naprej delijo na združevalne (angl. agglomerative) in cepitvene (angl. divisive). Združevalni gradijo strukturo množic od spodaj navzgor – na vsakem nivoju združujejo določene gruče skupaj, dokler ne nastane ena sama velika gruča, ki vsebuje vse vzorce. Nasprotno začnejo cepitveni algoritmi na vrhu z eno gručo, v kateri so vsi vzorci, in na vsakem nivoju določeno gručo razcepijo na dve. Postopek se konča, ko je vsak vzorec v svoji gruči. Postopek hierarhičnega razvrščanja lahko nazorno prikažemo z drevesom, ki mu pravimo dendrogram. Prikazan je na sliki 2.1.

Delitveni algoritmi ne gradijo drevesa množic, temveč eno samo množico gruč. Najpogosteje iterativno razdelijo množico vzorcev na K gruč, pri čemer K imenujemo želeno število gruč in je vhodni podatek v algoritem. Cilj teh algoritmov je tipično optimizacija funkcije, ki odraža kvaliteto razvrščanja. Ta funkcija je navadno vsota kvadratov razdalj vzorcev do težišča pripadajoče



Slika 2.1: Primer dendrograma za filogenetsko razvrščanje debla členonožcev. Puščici prikazujeta smer poteka cepitvenih oziroma združevalnih algoritmov.

gruče, ni pa nujno. Poleg tega je pojem razdalja lahko definiran na mnogo načinov, denimo kot najbolj znana evklidska razdalja.

2.2 Mera podobnosti in razdalja

Cilj razvrščanja vzorcev v skupine je združevanje vzorcev, ki so si med seboj podobni. Podobnost je zelo ohlapen pojem, katerega lastnosti je treba natančneje definirati. Podobnost količinsko opišemo s preslikavo – mero podobnosti sim, ki vsakemu paru vzorcev (x, y), kjer $x, y \in P$, priredi neko realno število

$$\text{sim} : (x, y) \mapsto R \quad . \tag{2.5}$$

Mera podobnosti mora izpolnjevati naslednji zahtevi [5]:

- $\text{sim}(x, y) = \text{sim}(y, x)$ (simetričnost) in
- $\text{sim}(x, x) \leq \text{sim}(x, y)$ (prema mera) ali
  $\text{sim}(x, x) \geq \text{sim}(x, y)$ (obratna mera).

Če prema mera izpolnjuje pogoj: $\text{sim}(x, x) = \text{sim}^*$, lahko dobimo iz nje mero različnosti, ki je potemtakem

$$\text{dis}(x, y) = \text{sim}(x, y) - \text{sim}^* \quad . \tag{2.6}$$

Mera različnosti mora imeti naslednje lastnosti:

- $\text{dis}(x, y) \geq 0$ (nenegativnost),
- $\text{dis}(x, x) = 0$,
- $\text{dis}(x, y) = \text{dis}(y, x)$ (simetričnost).

Mero različnosti lahko poimenujemo tudi razdalja, če izpolnjuje še dva pogoja:

- $\text{dis}(x, y) = 0 \Rightarrow x = y$ (razločljivost) in
- za $\forall z \in P:\ \text{dis}(x, y) + \text{dis}(y, z) \geq \text{dis}(x, z)$ (trikotniška neenakost).

Z merami lahko izmerimo razdaljo med dvema točkama oziroma vzorcema in tako ocenimo njuno podobnost. Sklepamo, da manjša razdalja pomeni večjo podobnost in obratno, denimo $\text{sim} = 1 - \text{dis}$, mogoče pa so tudi razne druge transformacije.

Poglejmo si sedaj dva primera razdalj v D-razsežnem prostoru med vzorcema $x = (x^{(1)}, x^{(2)}, \ldots, x^{(D)})$ in $y = (y^{(1)}, y^{(2)}, \ldots, y^{(D)})$. Razdalja $\text{dis}(x, y)$ med njima je, na primer, lahko definirana kot:

- razdalja Minkowskega:

$$\text{dis}(x, y) = \left(\sum_{i=1}^{D} |x^{(i)} - y^{(i)}|^p \right)^{\frac{1}{p}} \quad \text{ali} \tag{2.7}$$

- razdalja Mahalanobisova

$$\text{dis}(x, y) = \sqrt{(x - y)' \Sigma^{-1} (x - y)} \quad , \tag{2.8}$$

  kjer je Σ kovariančna matrika znotraj gruč.

Za vzorce, ki zavzamejo številske vrednosti, so največ v uporabi naslednji posebni primeri razdalj Minkowskega (enačba 2.7):

- razdalja Manhattan ali „city-block“ ($p = 1$):

$$\text{dis}(x, y) = \sum_{i=1}^{D} |x^{(i)} - y^{(i)}| \quad , \tag{2.9}$$

- evklidska razdalja ($p = 2$):

$$\mathrm{dis}(x, y) = \sqrt{\sum_{i=1}^{D} (x^{(i)} - y^{(i)})^2} \quad , \tag{2.10}$$

- razdalja Čebiševa ali trdnjavska razdalja ($p = \infty$):

$$\mathrm{dis}(x, y) = \max_i |x^{(i)} - y^{(i)}| \quad . \tag{2.11}$$

Omeniti je potrebno, da obstajajo še druge razdalje oziroma mere podobnosti, ki so primerne za binarne oziroma nominalne vzorce. Pri teh merimo ujemanje med parom vzorcev s pomočjo frekvenc v kontingenčni tabeli. Več o tem v poglavju 3.2.2, ko obravnavamo kriterije podobnosti dveh razvrščanj, kar lahko prevedemo tudi na problem računanja podobnosti med dvema vzorcema. Pomembno je, da pred samim razvrščanjem danih vzorcev v gruče izberemo najbolj ustrezno mero, saj od tega v veliki meri zavisi končni rezultat.

2.3 Postopek razvrščanja vzorcev v gruče

Vsako razvrščanje vzorcev v gruče je zgodba zase, saj ne moremo uporabiti istih mer podobnosti oziroma algoritmov razvrščanja na vseh množicah vzorcev, ker preprosto za vsak primer obstajajo drugačne značilnosti, ki pogojujejo izbor določenih tehnik. Večkrat se je potrebno intuitivno odločati, vsekakor pa je znanje strokovnjakov, iz področja katerih je vzeta množica vzorcev, zelo dobrodošlo. Razvrščanje v gruče je postopek, ki v splošnem zajema pet sklopov in vsak od njih prinaša široko paleto možnih izvedb. Na kratko si jih oglejmo:

1. **Izbira objektov:** na začetku izberemo množico objektov, katerih značilnosti opazujemo ali merimo. Merjenim objektom lahko rečemo tudi vzorci.

2. **Določitev množice spremenljivk:** izmed vseh značilnosti merjenih objektov moramo izbrati tiste, ki imajo določen pomen oziroma pojasnevalno moč. Značilnosti opišemo s spremenljivkami (atributi). Pred nadaljevanjem je potrebno poskrbeti, da so spremenljivke ustrezno standardizirane, da se njihov prispevek v razvrščanju izenači. Število opazovanih spremenljivk določa razsežnost prostora, v katerem se nahajajo vzorci.

3. **Računanje podobnosti med vzorci:** mero podobnosti izberemo glede na naravo spremenljivk (številski ali nominalni tip) in pričakovano strukturo podatkov.

4. **Uporaba algoritma za razvrščanje v gruče:** poznamo ogromno postopkov za razvrščanje in vsak ima svoje posebnosti delovanja. Ni postopka, ki bi bil v vseh primerih in pogledih najboljši. Izberemo torej tistega, ki je prilagojen naši problemski domeni, lahko pa jih izberemo več in kot rezultat upoštevamo najboljšega.

5. **Ocena dobljene rešitve:** ta korak se pogosto namenoma ali pomotoma izpušča, pa vendar je zelo pomemben, saj šele z njim dobimo pravi vpogled v rezultate razvrščanja. Poznamo zunanje in notranje kriterije za oceno kvalitete razvrščanja. Tudi na tem koraku je ponudba možnih izbir zelo široka.

Če vse skupaj kot povzetek orišem s Pepelkino prigodo iz začetka poglavja: izbrani objekti so zrna leče v pepelu. Pepel je prav tako objekt, lahko pa ga smatramo tudi kot šum, ki otežuje problem razvrščanja. V množici spremenljivk opazovanih objektov nastopajo: barva zrn, njihova velikost, oblika in zdravje (črviva/zdrava zrna). Mačeha je izrecno zahtevala, da naj se upoštevata le barva in zdravje, zato ti dve spremenljivki tvorita izbrano množico atributov. Mera podobnosti med vzorci mora biti povezana z njihovo barvo, torej lahko računamo razlike barvnih komponent (denimo v barvnem modelu RGB) posameznih zrn in na ta način definiramo podobnost. Tako bi tudi lahko zaznali črviva zrna, ki bi najbrž imela temnejše barvne odtenke. Kateri algoritem razvrščanja bo v naslednjem koraku izbrala Pepelka oziroma njeni možgani, naj ostane pravljična skrivnost, gotovo pa ni skrivnost ocenjevanje rezultata razvrščanja – v vsakem primeru bo s strani mačehe ugotovljen kup nepravilnosti pri razvrščanju in Pepelki tudi tokrat ne bo prizaneseno.

Poglavje 3

Metodologija dela

V tem poglavju bom opisal delovanje izbranih algoritmov za razvrščanje[1]: KMC (angl. K-Means Clustering), ECMC (angl. Evolving Clustering Method with Constrained minimization), EM GMM (angl. Expectation Maximization Gaussian Mixture Model) in CSC (angl. Cauchy-Schwarz divergence Clustering), kriterije za notranje in zunanje ocenjevanje uspešnosti algoritmov in množice vzorcev.

Razlogi, zakaj so bili v primerjavo izbrani ravno prej našteti algoritmi, so naslednji: algoritem KMC (metoda K-tih voditeljev) je eden temeljnih in najpreprostejših na tem področju. Algoritem ECMC je njegova izpeljanka, ki pa ima nekaj zanimivih potez. Zanimalo me je, ali prispevajo k boljšim rezultatom. Nadaljnje je zanimiva uporaba verjetnostnih porazdelitev v namen razvrščanja, ki vsakemu vzorcu pripiše določeno verjetnost, da pripada neki gruči. Ta pristop izkorišča algoritem EM GMM, ki je v literaturi pogosto na seznamu primerjanih. Bistveni član zasedbe pa je nazadnje omenjen algoritem CSC, ki uporablja popolnoma drugačen pristop kot ostali in je med vsemi tudi najmlajši, kar se letnice odkritja tiče. Med preučevanjem literature še nisem zasledil primerjave teh algoritmov med seboj, zato domnevam, da znajo biti rezultati zanimivi in koristni.

Opisi algoritmov so podrobno obravnavani v podpoglavju 3.1. Rezultate razvrščanja v gruče je potrebno ustrezno ovrednotiti in oceniti. V ta namen služijo notranji in zunanji kriteriji za ocenjevanje, ki so podrobneje opisani v poglavju 3.2. V postopku primerjave algoritmov kot tudi samih kriterijev za ocenjevanje, je posebna skrb namenjena izboru ustreznih množic vzorcev. Opis izbranih množic in utemeljitev izbire je podana v poglavju 3.3.

[1]Imena algoritmov bom uporabljal v obliki kratic zavoljo kompaktnosti in enostavnosti zapisa.

3.1 Opis algoritmov za razvrščanje v gruče

3.1.1 KMC

Algoritem KMC, ali prevedeno metoda K-tih voditeljev, je iterativna, lokalno optimizacijska metoda, ki jo uvrstimo med delitvene algoritme. Postopek premešča vzorce med gručami, dokler ni dosežen lokalni optimum oziroma je minimizirana kriterijska funkcija. Algoritem se začne tako, da izberemo K točk v prostoru vzorcev, ki postanejo voditelji oziroma težišča K-tih gruč. Parameter K torej pomeni število gruč, med katere bo algoritem razvrstil vzorce in mora biti podan kot vhod. Voditelje označimo kot m_k, $k = 1, \ldots, K$. Nato se izračuna razdalja (denimo evklidska) med vsakim vzorcem in voditelji. Posamezni vzorec se nato uvrsti v gručo tistega voditelja, ki mu je najbližji. To storimo za vse vzorce, tako da na koncu vsak vzorec pripada natanko eni gruči. Za vsako gručo nato izračunamo novo težišče (srednjo vrednost), ki predstavlja novo pozicijo voditelja.

Težišče m_k je definirano kot

$$m_k = \frac{1}{N_{C_k}} \sum_{i=1}^{N_{C_k}} x_{ki} \quad , \tag{3.1}$$

kjer je N_{C_k} število vzorcev v gruči C_k, x_{ki} pa i-ti vzorec v gruči $C_k = \left\{x_{k1}, x_{k2}, \ldots, x_{kN_{C_k}}\right\}$.

Postopek ponavljamo, dokler se pozicija voditeljev ne spreminja več, oziroma dokler ne dosežemo vnaprej določenega števila ponovitev.

Zapišimo algoritem KMC po korakih:

Korak 1: Določimo začetna težišča (voditelje) m_k, $k = 1, \ldots, K$.

Korak 2: Izračunamo razdalje med vzorci in voditelji. Vzorec x_i uvrstimo v gručo C_k, če je $\mathrm{dis}(x_i, m_k) < \mathrm{dis}(x_j, m_l)$ za $l = 1, \ldots, K$, $l \neq k$. To naredimo za vse vzorce $x \in P$, kjer je P množica vseh vzorcev.

Korak 3: Za vsako gručo C_k izračunamo novo težišče m_k po enačbi 3.1.

Korak 4: Preverimo ali so težišča m_k spremenila svoj položaj glede na prejšnjo iteracijo. Če so, gremo na korak 2, sicer končamo. Končamo tudi v primeru, ko smo prekoračili določeno število iteracij.

Hitro se nam lahko zastavi vprašanje, kako v koraku 1 določimo začetna težišča. Najpogosteje se to naredi naključno (enakomerno po celem področju

vzorcev), lahko tudi uporabimo določeno hevristiko, kot recimo to, da morajo biti začetna težišča kar najbolj razpršena med vzorci. Najbolje pa je, če se za začetne pozicije odločimo na podlagi predhodne analize podatkov in domnev o strukturi preučevanih pojavov. Kakorkoli, končni rezultat, ki je razvrstitev $C = \{C_1, \ldots, C_K\}$, je močno odvisen od začetne izbire težišč. Zato je potrebno cel algoritem ponoviti večkrat in kot končni rezultat vzeti najboljšo razvrstitev.

Spet smo naleteli na besedo najboljše, ki zahteva pojasnilo. Še prej pa moramo povedati, katero kriterijsko funkcijo hočemo optimizirati pri tej metodi. Načeloma je lahko poljubna funkcija razdalje, ki naj skozi iteracije monotono pada; vsota kvadratov razdalje vzorcev do težišča gruče po vseh gručah je denimo moja izbira, ni pa edina. Formalno jo zapišemo kot

$$J = \sum_{k=1}^{K} \sum_{i=1}^{N_{C_k}} \mathrm{dis}(x_{ki}, m_k)^2 \quad . \tag{3.2}$$

Algoritem KMC išče minimum kriterijske funkcije J podane v enačbi 3.2. Najboljša razvrstitev vzorcev je torej tista, ki da najmanjšo vrednost funkcije J.

Omeniti velja, da so mogoče razne prilagoditve opisanega algoritma. Tako lahko postavimo omejitev, da morajo biti voditelji nujno izbrani izmed vzorcev (angl. K-medoids algoritem). Prav tako obstaja algoritem, ki ne deluje trdo, ampak mehko (angl. fuzzy), kjer vsak vzorec pripada v gručo z določeno stopnjo pripadnosti. Se pravi, da lahko en vzorec pripada dvema ali več različnim gručam hkrati. Tak algoritem se imenuje Fuzzy C-means, vendar njegov opis presega okvire te naloge.

Implementacija

Algoritem KMC spada v standardni nabor metod za analizo podatkov, zato so njegove implementacije že vključene v razna programska okolja. Za izvedbo tega dela sem uporabljal programski paket MATLAB in znotraj njega že vgrajeno funkcijo `kmeans`.

3.1.2 ECMC

Leta 2001 sta Kasabov in Song [18] predlagala novo metodo razvrščanja v gruče, imenovano ECM, kar po angleško pomeni Evolving Clustering Method. Postopek je bil razvit v prid novega sistema mehkega sklepanja za adaptivno učenje v povezavi z dinamičnim napovedovanjem časovnih vrst [8] (angl. DENFIS – Dynamic Evolving Neural-Fuzzy Inference System). Naloga algoritma ECM je razdelitev vhodnega prostora spremenljivk z namenom ustvarjanja

mehkih pravil sklepanja. V prvotni obliki algoritem ECM deluje sprotno (angl. online) na podlagi maksimizacije medsebojne razdalje med gručami. Podana pa je tudi razširjena različica algoritma, ki opravi še optimizacijo določene kriterijske funkcije. Ta različica se imenuje ECMC (angl. Evolving Clustering Method with Constrained minimization) in sem jo tudi uporabil v primerjavi. Algoritem ECMC, tako kot KMC, spada med delitvene algoritme razvrščanja.

Princip delovanja je dokaj preprost. Vhod v algoritem ni več število gruč K, ampak največja razdalja, ki jo še dopustimo med vzorcem in središčem njegove gruče (največji polmer). Označimo jo z D_{thr} in pomeni prag razdalje (angl. distance threshold). Seveda D_{thr} vpliva na K, vendar zveza ni enostavno določljiva.

V prvem delu algoritem jemlje vzorce iz vhodne množice enega po enega in po potrebi ustvarja nove gruče ali posodablja že obstoječe. V drugem delu, ko so vsi vzorci že razvrščeni, pa izvedemo minimizacijo kriterijske funkcije v postopku optimizacije z omejitvijo.

Za razdaljo med dvema D-razsežnima vzorcema $x, y \in P$, je izbrana utežena evklidska razdalja, določena kot

$$\text{dis}(x, y) = \sqrt{\frac{\sum_{i=1}^{D} |x^{(i)} - y^{(i)}|^2}{D}} \quad . \tag{3.3}$$

Središče posamezne gruče C_k bomo označili podobno kot težišča pri metodi KMC, se pravi z m_k. Polmer vsake gruče C_k označimo z R_k, vzorce pa kot x_i.

Algoritem ECMC po korakih (prvi del):

Korak 1: Vzamemo prvi vzorec x_1 iz množice P. Ustvarimo prvo gručo C_1 s polmerom $R_1 = 0$, v kateri je samo x_1, ki je hkrati tudi središče gruče – m_1.

Korak 2: Če so bili razvrščeni že vsi vzorci, algoritem končamo, sicer nadaljujemo. Izračunamo razdalje med trenutnim obravnavanim vzorcem x_i in vsemi (k) že ustvarjenimi središči m_j, $j = 1, \ldots, k$, kar zapišemo:

$$D_{ij} = \text{dis}(x_i, m_j) \quad . \tag{3.4}$$

Korak 3: Izmed vseh razdalj D_{ij}, $j = 1, 2, \ldots, k$, izberemo take, ki zadoščajo pogoju $D_{it} \leq R_t$, $t = 1, 2, \ldots, T$. Izmed njih izberemo najmanjšo:

$$D_{iM} = \min_t(D_{it}) = \min_t \text{dis}(x_i, m_t) \quad .$$

Vzorec x_i uvrstimo v gručo C_M in se vrnemo na korak 2. Če je $T = 0$ (vse razdalje D_{ij} so večje od polmerov obstoječih gruč), nadaljujemo z naslednjim korakom.

Korak 4: Za vsa središča gruč m_j, $j = 1, 2, \ldots, k$, izračunamo vrednosti $S_{ij} = D_{ij} + R_j$ in izmed vseh izberemo najmanjšo S_{ia}, da velja:

$$S_{ia} = D_{ia} + R_a = \min_j S_{ij} \quad , \; j = 1, 2, \ldots, k \quad .$$

Korak 5: Če je $S_{ia} > 2 \cdot D_{\text{thr}}$, vzorec x_i ne pripada v nobeno že obstoječo gručo. Ustvari se nova gruča po postopku, opisanem v koraku 1 in nadaljujemo s korakom 2.

Korak 6: Če je $S_{ia} < 2 \cdot D_{\text{thr}}$, spada vzorec x_i v gručo C_a, s tem da moramo popraviti pozicijo središča in velikost polmera. Novi polmer R_a^{novi} meri $\frac{S_{ia}}{2}$, novo središče m_a^{novi} pa se premakne na daljico med vzorcem x_i in prejšnjo pozicijo središča m_a, tako da je razdalja med novim središčem m_a^{novi} in vzorcem x_i enaka R_a^{novi}. Vrnemo se na korak 2.

Opisani prvi del algoritma na koncu zagotavlja, da noben vzorec x_i ni oddaljen od središča svoje gruče za več kot D_{thr}. Končno število gruč je K. Sedaj je potrebno določiti kriterijsko funkcijo, ki jo želimo v drugem delu algoritma minimizirati. Zaradi konsistentnosti z ostalimi algoritmi jo označimo z J, njena definicija pa se glasi:

$$J = \sum_{k=1}^{K} \sum_{i=1}^{N_{C_k}} \text{dis}(x_{ki}, m_k) \quad , \tag{3.5}$$

kjer je N_{C_k} število vzorcev v gruči C_k in x_{ki} i-ti vzorec v gruči C_k. Sedaj sestavimo matriko pripadnosti U z elementi u_{ik}, za katere mora veljati za vsak $l \neq k$:

$$u_{ik} = \begin{cases} 1 \; ; & \text{dis}(x_i, m_k) \leq \text{dis}(x_i, m_l), \\ 0 \; ; & \text{sicer} \end{cases} \quad . \tag{3.6}$$

Torej, če je i-temu vzorcu x_i najbližje središče m_k, potem je $u_{ik} = 1$, sicer 0.

Med minimizacijo funkcije J moramo upoštevati omejitev, da ne sme biti noben vzorec oddaljen od središča svoje gruče za več kot D_{thr}. Omejitvena neenačba se torej glasi:

$$\text{dis}(x_{ki}, m_k) \leq D_{\text{thr}} \quad , \; k = 1, 2, \ldots, K \quad , i = 1, 2, \ldots, N_{C_k} \quad . \tag{3.7}$$

V drugem delu algoritma ECMC iterativno posodabljamo matriko U in premikamo središča m_k, da dosežemo minimum kriterijske funkcije J.

Algoritem ECMC po korakih (drugi del):

Korak 7: Zgradimo matriko pripadnosti vzorcev v gruče U kot pravi enačba 3.6.

Korak 8: Z uporabo metode optimizacije z omejitvami, glede na enačbi 3.5 in 3.7, določimo nova središča gruč.

Korak 9: Izračunamo kriterijsko funkcijo J po enačbi 3.5. Če je njena vrednost pod določeno tolerančno vrednostjo, oziroma je razlika od prejšnje iteracije pod določenim pragom, oziroma smo že presegli največje dovoljeno število iteracij, algoritem končamo. Sicer nadaljujemo s korakom 7.

Metoda ECMC se razlikuje od tipičnih delitvenih metod po tem, da kot vhodni parameter eksplicitno ne zahteva števila gruč K, temveč največjo dovoljeno razdaljo med vzorci znotraj gruče do središča gruče. Posledično to pomeni, da je morebitna primerjava z ostalimi algoritmi precej težavna, če prej ne poskrbimo, da je rezultat vseh algoritmov razvrstitev v K gruč. Kako to dosežemo pri metodi ECMC, je opisano v poglavju 4.

Implementacija

Algoritem ECM je bil, tako kot vse ostalo programje v tem delu, implementiran v okolju MATLAB. Za izvedbo nadgradnje algoritma v ECMC, torej za dodajanje optimizacije z omejitvami, je bila uporabljena funkcija `fmincon` MATLAB-ovega paketa za optimizacijo Optimization Toolbox [14].

3.1.3 EM GMM

V tem poglavju bom predstavil verjetnostni pristop k reševanju problema razvrščanja v gruče. Najprej si bomo ogledali, kako lahko razporejenost vzorcev modeliramo z uporabo skupka ali mešanice Gaussovih verjetnostnih porazdelitev [2] (od tu kratica GMM – Gaussian Mixture Model). Kot bomo videli, pri tem zopet potrebujemo optimizacijski algoritem, ki ustrezno nastavi parametre modela in tako čim bolj optimalno razvrsti vzorce v gruče. Ta algoritem se v angleščini imenuje Expectation Maximization oziroma na kratko EM. Pa pojdimo lepo po vrsti.

Pri postopku modeliranja verjetnostne porazdelitve vzorcev se velikokrat uporablja *Gaussova* ali normalna porazdelitev, ki jo za D-razsežni vzorec x zapišemo tako:

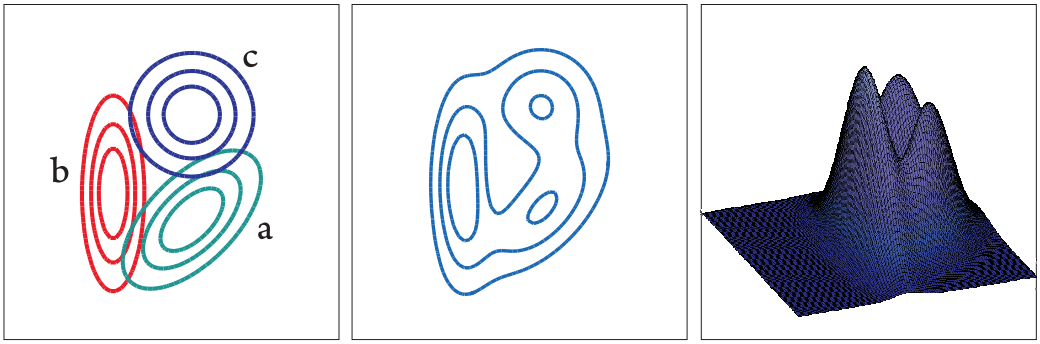
$$N(x|\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^D|\Sigma|}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} \quad , \qquad (3.8)$$

kjer je μ D-razsežni vektor srednjih vrednosti, Σ kovariančna matrika velikosti $D \times D$, $|\Sigma|$ njena determinanta in oznaka ^{T} transpozicija vektorja. Σ je lahko različnih oblik (glej sliko 3.1(a)):

a) splošna oblika: Σ vsebuje D^2 načeloma različnih vrednosti,

b) diagonalna oblika: $\Sigma = diag(\sigma_i^2)$ – D različnih vrednosti (σ_i^2 je varianca i-te dimenzije); eliptične krivulje, ki povezujejo točke z isto verjetnostjo, so poravnane s koordinatno osjo,

c) proporcionalna enotski matriki I: $\Sigma = \sigma^2 I$; krivulje, ki povezujejo točke z isto verjetnostjo, so koncentrični krogi.



(a) Gaussove porazdelitve za različne oblike Σ.

(b) Superpozicija Gaussovih porazdelitev – mešanica.

(c) Prostorska ponazoritev mešanice Gaussovih porazdelitev.

Slika 3.1: Grafična upodobitev mešanice Gaussovih porazdelitev.

Od izbire kompleksnosti Σ je odvisna tudi kompleksnost nastavljanja iskanih parametrov, po drugi strani pa s kompleksnejšo obliko lahko zajamemo določeno strukturo v podatkih, ki je drugače ne bi.

Ker z eno samo Gaussovo porazdelitveno funkcijo težko opišemo kompleksnejše podatke, uporabimo mešanico Gaussovih porazdelitev (angl. mixture of

Gaussians), ki ni nič drugega kot linearna kombinacija (superpozicija) enostavnih Gaussovih porazdelitev (slika 3.1(b)), ki se imenujejo komponente mešanice in imajo vsaka svoj vektor μ in matriko Σ. Navedimo primer za superpozicijo K Gaussovih porazdelitev:

$$p(x) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k) \quad . \tag{3.9}$$

Koeficiente mešanja (angl. mixture coefficients) označimo z π_k in velja:

$$\sum_{k=1}^{K} \pi_k = 1 \quad \text{in} \quad 0 \leq \pi_k \leq 1 \quad . \tag{3.10}$$

Koeficiente mešanja lahko razumemo kot verjetnost, da bo izbrana k-ta komponenta mešanice, torej p(k), $N(x|\mu_k, \Sigma_k)$ pa kot pogojno verjetnost $p(x|k)$ vzorca x, če smo izbrali k-to komponento mešanja. Uvedemo še diskretne spremenljivke s_i, ki določajo pripadnost posameznega vzorca x določeni gruči. Torej $s_i = k$ pomeni, da x_i pripada gruči k. Potem drži, da

$$\pi_k = p(s = k) \quad . \tag{3.11}$$

Spremenljivke s_i imenujemo skrite spremenljivke (angl. latent, hidden variables) – naš cilj je, da najdemo njihovo vrednost v postopku optimizacije kriterijske funkcije.

Oblika mešanice Gaussovih porazdelitev je torej odvisna od treh množic parametrov: $\pi = \{\pi_1, \ldots, \pi_K\}$, $\mu = \{\mu_1, \ldots, \mu_K\}$ in $\Sigma = \{\Sigma_1, \Sigma_2, \ldots, \Sigma_K\}$. Ena možnost za njihovo nastavitev je uporaba funkcije največje podobnosti (angl. maximum likelihood).

Funkcija podobnosti (angl. likelihood function) je funkcija vzorcev ($x \in P$) in parametrov modela (Θ) in je definirana s

$$p(P|\Theta) = p(\{x_1, x_2, \ldots, x_N\} |\pi, \mu, \Sigma) = \prod_{i=1}^{N} p(x_i|\pi, \mu, \Sigma) \quad . \tag{3.12}$$

Cilj funkcije največje podobnosti je poiskati parametre $\Theta = \{\pi, \mu, \Sigma\}$, tako da ti maksimizirajo logaritem funkcije podobnosti (angl. log likelihood function). Za množico vzorcev $P = \{x_1, x_2, \ldots, x_N\}$ in z upoštevanjem 3.12 je logaritem funkcije podobnosti L enak

$$L = \ln p(P|\pi, \mu, \Sigma) = \sum_{i=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k N(x_i|\pi_k, \mu_k, \Sigma_k) \right\} \quad . \tag{3.13}$$

Na žalost analitične rešitve za funkcijo največje podobnosti (zaradi logaritma zunaj notranje vsote) ni, zato je potrebno poseči po nečem drugem. Tu se izkaže za izredno močno orodje Expectation Maximization ali EM algoritem, ki ga bomo uporabili za iskanje funkcije največje podobnosti s skritimi spremenljivkami.

Najprej je potrebno L odvajati po srednji vrednosti μ_k in odvod postaviti na 0. Dobimo:

$$\mu_k = \frac{\sum_{i=1}^{N} r_{ik} x_i}{\sum_{i=1}^{N} r_{ik}} \quad , \tag{3.14}$$

kjer je r_{ik} posteriorna verjetnost pripadnosti vzorca x_i k-ti komponenti mešanice Gaussovih funkcij, kar lahko zapišemo kot $p(s_i = k|x_i, \mu_k, \Sigma_k)$ in se po Bayesovem teoremu izračuna kot:

$$r_{ik} = \frac{\pi_k N(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_i|\mu_j, \Sigma_j)} \quad . \tag{3.15}$$

Podobno kot prej odvajamo L po Σ_k^{-1} in odvod postavimo na 0. Dobimo enačbo za Σ_k, ki se glasi

$$\Sigma_k = \frac{\sum_{i=1}^{N} r_{ik}(x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^{N} r_{ik}} \quad . \tag{3.16}$$

Sedaj nas čaka samo še odvajanje logaritma funkcije podobnosti L po π_k. Tu je potrebno biti pazljiv, saj moramo upoštevati omejitev, da je $\sum_{k=1}^{K} \pi_k = 1$. To dosežemo z uporabo Lagrangovega multiplikatorja (angl. Lagrange multiplier) in maksimizacijo izraza $L' = L + \lambda(1 - \sum_{k=1}^{K} \pi_k)$. L' torej odvajamo po π_k in odvod izeenačimo z 0. Po izračunu dobimo:

$$\pi_k = \frac{\sum_{i=1}^{N} r_{ik}}{N} \quad . \tag{3.17}$$

Število $N\pi_k$ lahko interpretiramo kot efektivno število vzorcev v gruči C_k.

Sedaj imamo vse pripravljeno za razlago delovanja algoritma EM. Bistvena sta dva koraka, E in M, po katerih je algoritem tudi dobil ime. V koraku E izračunamo vrednost skritih spremenljivk, če so parametri modela Θ nespremenljivi, fiksni. V koraku M pa popravimo parametre glede na vrednost skritih spremenljivk. V primeru modela mešanice Gaussovih porazdelitev so s_i skrite spremenljivke, parametri Θ pa π_k, μ_k in Σ_k.

Algoritem EM za model mešanice Gaussovih porazdelitev po korakih:

Korak 1: Nastavimo začetne vrednosti parametrom π_k, μ_k in Σ_k za vsako komponento $k = 1, 2, \ldots, K$. Izračunamo začetno vrednost funkcije L (po enačbi 3.13).

Korak 2 (korak E): Izračunamo vrednosti r_{ik} (po enačbi 3.15) – uporabimo trenutno nastavljene parametre π_k, μ_k in Σ_k.

Korak 3 (korak M): Izračunamo nove vrednosti parametrov μ_k^{novi} (po enačbi 3.14), Σ_k^{novi} (po enačbi 3.16) in π_k^{novi} (po enačbi 3.17). Uporabimo vrednosti r_{ik} izračunane v koraku 2.

Korak 4: Izračunamo vrednost logaritma funkcije podobnosti L po enačbi 3.13 in preverimo konvergenco bodisi parametrov bodisi funkcije L. V primeru, da ni dosežena, gremo na korak 2, sicer končamo.

Dodajmo še, da je v koraku 1 mogoče nastaviti parametre μ_k z uporabo nekaj iteracij algoritma KMC, ki da boljše izhodišče kot naključna izbira. Začetni π_k nastavimo na $1/K$, Σ_k pa je navadno enotska matrika ali njen ekvivalent.

Predpostavimo sedaj mogočo situacijo, da ima j-ta komponenta Gaussove mešanice srednjo vrednost μ_j enako, kot je pozicija vzorca x_n, torej velja $\mu_j = x_n$. V limiti, ko gre $\sigma_j \to 0$, gre funkcija L proti neskončnosti. Zato maksimizacija funkcije L ni več rešljiv problem, kar imenujemo problem singularnosti. V tem primeru je potrebno uporabiti hevristične postopke, da se takim primerom izognemo. Ena izmed možnosti je, da ob zaznani singularnosti srednjo vrednost μ kritične komponente mešanja ponastavimo na neko naključno vrednost, kovariančno matriko iste komponente pa na neko veliko vrednost. Nato nadaljujemo z optimizacijo.

Implementacija

Uporabljena je bila implementacija algoritma v prostem paketu za MATLAB, ki se imenje Netlab toolbox [2].

3.1.4 CSC

Raziskave na področju razvrščanja v gruče so vodile k uporabi kompleksnejših mer podobnosti, ki so bile sposobne zajeti strukture vzorcev, ki presegajo statistike drugega reda, kot so denimo kriterijske funkcije, ki minimizirajo varianco (primer je denimo vsota kvadratov razdalje). V ta namen so se v

[2]Netlab toolbox je prosto dostopen na: http://www.ncrg.aston.ac.uk/netlab/ .

zadnjem času začele uporabljati mere, vzete iz informacijske teorije, kot so entropija [9], medsebojna informacija in divergenca. Jenssen idr. [10] je na podlagi Cauchy-Schwarzeve neenakosti[3] osnoval novo kriterijsko funkcijo in algoritem za razvrščanje vzorcev v gruče, ki to funkcijo maksimizira. Algoritma v svojem članku avtorji eksplicitno ne poimenujejo, zato sem si to pravico za potrebe kompaktnosti in preglednosti vzel sam in algoritem poimenoval CSC, kar v angleščini pomeni Cauchy-Schwarz divergence Clustering. Poslovenjeno bi to metodo lahko poimenoval razvrščanje vzorcev v gruče na osnovi Cauchy-Schwarzeve divergence.

Osnovna ideja izhaja iz Cauchy-Schwarzeve neenakosti med dvema vektorjema x in y

$$||x||^2||y^2|| \geq (x^T y)^2 \quad , \tag{3.18}$$

kjer je x^T transponiran vektor x. Iz tega sledi:

$$-\ln \frac{x^T y}{\sqrt{||x||^2||y^2||}} \geq 0 \quad . \tag{3.19}$$

Če skalarni produkt dveh vektorjev zamenjamo s skalarnim produktom dveh verjetnostnih porazdelitev $p(x)$ in $q(x)$, dobimo definicijo za Cauchy-Schwarzevo divergenco D_{CS}:

$$D_{CS}(p,q) = -\ln \frac{\int p(x)q(x)dx}{\sqrt{\int p^2(x)dx \int q^2(x)dx}} \quad . \tag{3.20}$$

Na pravkar zapisano mero lahko gledamo kot na približek Kullback-Leiblerjeve divergence med dvema porazdelitvama. Ima vse tri lastnosti [6] (pozitivna, enaka 0, če $p(x) = q(x)$ in simetrična), ki so potrebne, da je to mera različnosti (glej razdelek 2.2). Naš cilj je maksimizirati divergenco (različnost) s tem, da minimiziramo argument logaritma.

Če želimo uporabiti mero različnosti zapisano v enačbi 3.20, moramo iz podatkov najprej oceniti verjetnostni porazdelitvi $p(x)$ in $q(x)$. To storimo z uporabo metode Parzenovega okna, kjer kot okensko funkcijo uporabimo večrazsežno Gaussovo (normalno) jedrno funkcijo $N(x|\Sigma)$, kjer je $\Sigma = \sigma^2 I$ (σ^2 je varianca in I enotska matrika). Za definicijo normalne porazdelitve glej enačbo 3.8 v poglavju 3.1.3. Ocena gostote verjetnosti vzorcev $p(x)$ v gruči C_k s skupno N_{C_k} vzorci je:

[3]Schwarz je v literaturi pogosto napačno imenovan kot Schwartz.

$$\hat{p}(x) = \frac{1}{N_{C_k}} \sum_{i=1}^{N_{C_k}} N(x - x_i|\sigma^2 I) \quad . \tag{3.21}$$

Da bo vsa stvar v nadaljevanju preglednejša, uvedimo funkcijo pripadnosti M_{ij}:

$$M_{ij} = \begin{cases} 1\ ; & x_i \text{ in } x_j \text{ pripadata različnima gručama,} \\ 0\ ; & \text{sicer} \end{cases} \quad . \tag{3.22}$$

Poleg M_{ij} definirajmo še funkcijo pripadnosti $M_{C_{k_{ij}}}$ za poljubno gručo vzorcev C_k:

$$M_{C_{k_{ij}}} = \begin{cases} 1\ ; & x_i \text{ in } x_j \text{ pripadata gruči } C_k, \\ 0\ ; & \text{sicer} \end{cases} \quad . \tag{3.23}$$

Enačbo 3.20 zapišemo kot $D_{CS} = -\ln V_{CS}$. Sedaj lahko ocenimo V_{CS} kot (vmesni koraki so razloženi v [10, 11]):

$$\hat{V}_{CS} = \frac{\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} M_{ij} N(x_i - x_j|2\sigma^2 I)}{\sqrt{\prod_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N} M_{C_{k_{ij}}} N(x_i - x_j|2\sigma^2 I)}} \quad . \tag{3.24}$$

Dobili smo enačbo, ki predstavlja našo kriterijsko funkcijo za razvrščanje v gruče. Potrebujemo le še način, kako to funkcijo minimizirati. Razvit je bil poseben algoritem, ki spominja na hierarhične pristope k razvrščanju. Podobnost je namreč v tem, da začnemo z razvrščanjem vzorcev v večje število gruč, kot je želeno število K. Nato v vsakem koraku določimo najslabšo gručo, v smislu divergence, in njene vzorce razporedimo po ostalih gručah. Postopek ponavljamo, dokler ne ostane K gruč. To je glavna ideja algoritma.

Algoritem CSC po korakih:

Korak 1: Naključno izberemo vzorec in mu dodamo še $N_{\text{zač}} - 1$ najbližjih[4] vzorcev. Tako dobimo prvo gručo. Postopek ponavljamo, dokler ni ustvarjenih $K_{\text{zač}}$ gruč.

Korak 2: Izberemo še nerazvrščen vzorec x in ga dodamo taki gruči C_k, da velja:

$$k = \arg\max_i D_{CS}(C_1, \ldots, C_i + x, \ldots, C_{K_s}) \quad ,$$

[4]Razdalja med x_i in x_j je mišljena kot vrednost $N(x_i - x_j|2\sigma^2 I)$

kjer je K_s število gruč v trenutnem nivoju algoritma (K_s gre od $K_{\text{zač}}$ do K). Ciljno gručo za vzorec x torej izberemo tako, da ob dodajanju vzorca x vanjo, najbolj povečamo D_{CS} med gručami. Postopek ponovimo za vse nerazvrščene vzorce. Če je trenutno število gruč K_s enako želenemu (končnemu) številu K, algoritem ustavimo, sicer nadaljujemo na korak 3.

Korak 3: Določimo najslabšo gručo. To je tista gruča, ki ob tem, da je izvzeta iz izračuna D_{CS} za preostale gruče, največ prispeva k povečanju divergence. Izračunamo vrednost D_{CS} za vse vzorce, razen za vzorce gruče C_i. To storimo za $i = 1, 2, \ldots, K_s$. Najslabša gruča je C_w, če velja, da je bila vrednost D_{CS} največja takrat, ko te gruče nismo upoštevali. Pomeni, da so v tem primeru ostale gruče najbolj ločene med seboj. Število gruč K_s zmanjšamo za 1 in vse vzorce iz gruče C_w označimo kot nerazvrščene. Gremo na korak 2.

Vhodov v algoritem je sedaj več: $K_{\text{zač}}$, $N_{\text{zač}}$, K, σ. Eksperimenti [10] in lastna empirična dognanja so pokazali, da je priporočljiva vrednost za $K_{\text{zač}}$ okoli 20 in $N_{\text{zač}}$ okoli 10, oziroma toliko, da je z začetnimi gručami pokritih vsaj 50 % vzorcev. Izbira vrednosti parametrov je tudi odvisna od obravnavane množice vzorcev. Zlasti pa to velja za parameter σ, ki mora biti pazljivo izbran, da bo rezultat dober. V drugih virih [11] je opisano pravilo, ki ga je za oceno parametra σ postavil Silverman in se za D-razsežni prostor glasi tako:

$$\sigma_S = \hat{\sigma} \left[\frac{4}{(2D+1)N} \right]^{\frac{1}{D+4}} , \tag{3.25}$$

kjer je $\hat{\sigma} = \sqrt{D^{-1} \sum_i \Sigma_{ii}}$ in $\sum_i \Sigma_{ii}$ vsota diagonalnih elementov kovariančne matrike vzorcev.

Znano je, da je težje ocenjevati σ, ko D raste, saj pride do večje razpršenosti vzorcev po prostoru. Pojav, ko volumen eksponentno raste z razsežnostjo in s tem otežuje reševanje problemov, je Richard Bellman poimenoval prekletstvo razsežnosti (angl. curse of dimensionality) in tudi na obravnavanem področju razvrščanja povzroča preglavice.

Nismo še povedali, kako na koraku 2 algoritma CSC izberemo naslednji še neuvrščeni vzorec. Možnosti je več, naj omenimo dve:

- Naključno med vsemi nerazvrščenimi vzorci. Slabost tega je v nestabilnosti procesa, če so izbrani vzorci daleč od začetnih gruč. Pred-

nost pa je v enostavnosti izvedbe.

- Izberemo vzorec, ki je najbližji prototipu katerekoli gruče. Kot prototip lahko vzamemo težišče vzorcev znotraj gruče ali pa je prototip kar vsak vzorec posebej. V zadnjem primeru je izbira računsko precej zahtevna, saj je potrebno izračunati razdalje od vseh nerazvrščenih do vseh razvrščenih vzorcev. Prednost tega je, da je proces razvrščanja stabilnejši.

Implementacija

Algoritem CSC je implementiran v okolju MATLAB, za izvorno kodo pa se zahvaljujem avtorju algoritma, Robertu Jenssenu.

3.2 Kriteriji za ocenjevanje uspešnosti razvrščanja

Vsak algoritem za razvrščanje vzorcev v gruče ima svojo notranjo mero uspešnosti, oziroma kriterijsko funkcijo, ki mu na določenem koraku izvajanja pove, kako dobro so vzorci razvrščeni. Pogosto je vrednost te funkcije tudi pogoj za konec izvajanja. Iz opisa algoritmov v poglavju 3.1 vidimo, da so kriterijske funkcije med seboj precej raznolike in postavi se vprašanje, kako potem vrednotiti posamezno razvrstitev vzorcev. Vsaka metoda zase je po svojem kriteriju najboljša, saj ta kriterij oziroma mero podobnosti pravzaprav optimizira, ko pa želimo narediti primerjavo med metodami, potrebujemo neodvisne kriterije ocenjevanja. Glede na to, za kakšno vrsto primerjave gre, ločimo notranje in zunanje kriterije, včasih poimenovane tudi indeksi ali kazalci. Notranji kriterij meri kvaliteto rezultata algoritma znotraj ene same razvrstitve, zunanji kriterij pa primerja dve razvrstitvi različnih algoritmov med seboj.

3.2.1 Notranji kriteriji

Notranji kriterij je zelo podoben kriterijski funkciji, ki je lastna algoritmu, in na podlagi razdalj med vzorci znotraj gruč in gručami med sabo pove, kako je neka razvrstitev vzorcev dobra.

Pomembna uporaba notranjih kriterijev ali indeksov je ugotavljanje pravega števila gruč na vzorcih. To ni trivialen problem in je v resnici posebno pereč. Večina algoritmov, zlasti delitvenih, zahteva kot vhodni podatek število gruč K, v katere naj razdeli vzorce. Dejstvo pa je, da tega števila največkrat

vnaprej ne poznamo in je potrebno ugibanje. Zato je praksa taka, da posamezni algoritem za razvrščanje poženemo večkrat za različno vrednost K in vsak rezultat ovrednotimo z notranjim kriterijem. Glede na potek vrednosti kriterija lahko sklepamo na pravilno število gruč. Postopek se izkaže za uporabnega, če so gruče med seboj dobro ločene in meje jasne, težave pa se pojavijo, ko gre za razvrščanje zahtevnejših podatkov (v smislu razsežnosti, prekrivanja, zapletenih oblik).

Sam bom notranje kriterije uporabil v smislu napovedovanja pravilnega števila gruč v vzorcih in jih v nadaljevanju tudi med seboj primerjal. Za primerjavo sem izbral štiri kriterije:

- Davies-Bouldinov kriterij (*DB*),
- Calinski-Harabaszov kriterij (*CH*),
- kriterij homogenost-ločljivost (angl. Homogeneity-Separation – *Hom & Sep*),
- kriterij uteženega inter-intra razmerja (angl. Weighted inter-intra ratio – *Wtertra*).

Preden si pogledamo vsak kriterij posebej, določimo nekaj označb, ki veljajo v nadaljevanju poglavja. Vzorci $x = \{x_1, x_2, \ldots, x_N\}$ iz množice P imajo razvrstitev $C = \{C_1, C_2, \ldots, C_K\}$, kjer je K število gruč. Vrednost $\mathrm{dis}(x_i, x_j)$ je razdalja med vzorcema x_i in x_j ter je poljubno določena – v mojem primeru je to evklidska razdalja (glej 2.2).

Davies-Bouldin

Davies-Bouldinov kriterij je definiran kot [15]:

$$DB(C) = \frac{1}{K} \sum_{k=1}^{K} \max_{k \neq l} \left\{ \frac{\Delta(C_k) + \Delta(C_l)}{\delta(C_k, C_l)} \right\} \quad , \tag{3.26}$$

kjer je $\Delta(C_k)$ notranja razdalja v gruči C_k in $\delta(C_k, C_l)$ razdalja med gručama C_k in C_l. $\Delta(C_k)$ je definirana s

$$\Delta(C_k) = \frac{\sum_{x_i \in C_k} \mathrm{dis}(x_i, m_k)}{N_{C_k}} \quad , \; k = 1, 2, \ldots, K \quad , \tag{3.27}$$

kjer je m_k središče gruče C_k, izračunano kot: $m_k = \frac{\sum_{x_i \in C_k} x_i}{N_{C_k}}$.

Razdalja med gručama $\delta(C_k, C_l)$ oziroma njunima središčema je definirana:

$$\delta(C_k, C_l) = \text{dis}(m_k, m_l) \quad . \tag{3.28}$$

Davies-Bouldinov kriterij izraža kompaktnost vzorcev znotraj gruče, kjer mora biti razdalja med njimi čim manjša, in ločenost gruč ene od druge, kjer naj bo razdalja med središči gruč čim večja. Manjša vrednost indeksa DB – boljši rezultat.

Calinski-Harabasz

Calinski-Harabasz ali indeks CH se nad množico K-tih gruč C, med katere je razdeljenih N vzorcev, izračuna kot [12]

$$CH(C) = \frac{sl(B)/(K-1)}{sl(W)/(N-K)} \quad , \tag{3.29}$$

kjer $sl(B)$ pomeni sled (diagonalo) matrike B, za W velja podobno. B predstavlja matriko razpršenosti med gručami, W pa matriko razpršenosti znotraj gruče. Sledi matrik B in W sta definirani kot

$$sl(B) = \sum_{k=1}^{K} N_{C_k} \text{dis}(m_k - m)^2 \quad , \tag{3.30}$$

$$sl(W) = \sum_{k=1}^{K} \sum_{i=1}^{N_{C_k}} \text{dis}(x_i - m_k)^2 \quad , \tag{3.31}$$

kjer je N_{C_k} število vzorcev v gruči C_k, m_k središče gruče C_k in m središče vseh vzorcev skupaj.

Višja vrednost kriterija CH pomeni boljši rezultat.

Homogenost-ločljivost

Homogenost in ločljivost sta v resnici dva kriterija, ki pa naj bi nastopala hkrati. Se pravi, da želimo hkrati homogenost znotraj gruč in ločljivost med gručami. Določena sta kot [3, 17]

$$Hom(C) = \frac{1}{N} \sum_{k=1}^{K} \sum_{x_i \in C_k} \text{dis}(x_i, m_k) \quad , \tag{3.32}$$

$$Sep(C) = \frac{\sum_{k \neq l} N_{C_k} N_{C_l} \text{dis}(m_k, m_l)}{\sum_{k \neq l} N_{C_k} N_{C_l}} \quad , \tag{3.33}$$

kjer je m_k središče gruče C_k. $Hom(C)$ je torej povprečna razdalja vzorcev znotraj gruče do njenega središča. Z izboljševanjem rezultata razvrščanja ta indeks pada. $Sep(C)$ pomeni uteženo povprečno razdaljo med središči gruč. Vrednost tega indeksa naj bi z izboljševanjem rezultata razvrščanja naraščala. V obeh primerih iščemo koleno oziroma točko, kjer se vrednost skokovito spremeni. Ta naloga je vse prej kot trivialna in obstaja več načinov, kako najti koleno funkcije (poiskati največji drugi odvod funkcije, največjo spremembo v magnitudi, poiskati točko, ki je najdlje od funkciji prilegajoče se premice, itd.). V tem delu bom koleno poteka trenutno obravnavanih kriterijev določal vizualno, kar je tudi pogosta praksa.

Uteženo inter-intra razmerje

Uteženo inter-intra razmerje, na kratko *Wtertra* [19], je kriterij podoben prejšnjim, s to razliko, da ne računa razdalj do središč gruč in da uvaja idejo kaznovanja prevelikih vrednosti K. Vrednost $intra(C_k)$ in $inter(C_k, C_l)$ določimo takole:

$$intra(C_k) = \frac{1}{(N_{C_k} - 1)N_{C_k}} \sum_{x_i,\, x_j \in C_k} \mathrm{sim}(x_i, x_j),\ i \neq j \quad , \tag{3.34}$$

$$inter(C_k, C_l) = \frac{1}{N_{C_k} N_{C_l}} \sum_{x_i \in C_k,\, x_j \in C_l} \mathrm{sim}(x_i, x_j),\ k \neq l \quad . \tag{3.35}$$

Naš cilj je maksimizirati *intra* in minimizirati *inter* količino. Napišimo njuno uteženo razmerje:

$$Wr(C) = 1 - \frac{(N-K)\sum_{k=1}^{K}\sum_{l=k+1}^{K} N_{C_k} \cdot inter(C_k, C_l)}{N\sum_{k=1}^{K}(N_{C_k} - 1) \cdot intra(C_k)} \quad . \tag{3.36}$$

Želimo, da je visoka številka gruč kaznovana, ker navadno ni zaželena. Zato uvedemo faktor kaznovanja, ki se linearno povečuje, ko število gruč K raste. Zapišimo končno enačbo za kriterij *Wtertra*:

$$Wtertra(C) = \left(1 - \frac{2K}{N}\right) Wr(C) \quad . \tag{3.37}$$

Količina *Wtertra* zavzame vrednosti na intervalu $[0, 1]$. Če je 0, pomeni, da si vzorci znotraj gruče niso v povprečju nič bolj podobni kot vzorci med različnimi gručami. Obratno pri vrednosti 1. Ko ugotavljamo pravo (optimalno) število gruč s tem kriterijem, gremo s številom K navzgor (od 2 do

končnega števila) in opazujemo vrednost *Wtertra*. Število gruč K pred prvim padcem vrednosti *Wtertra* je optimalno število gruč po tem kriteriju.

3.2.2 Zunanji kriteriji

V postopku primerjanja algoritmov za razvrščanje vzorcev v gruče nam pravilna razvrstitev (množica gruč) največkrat ni znana. V takem primeru lahko uporabimo zgolj notranje kriterije ocenjevanja. Če pa temu ni tako, torej poznamo pravilno razvrstitev, lahko rezultate algoritmov primerjamo z njo na podlagi zunanjih kriterijev ocenjevanja. To je zopet zelo pisana druščina, izmed katere sem izbral 6 kriterijev oziroma indeksov:

- Randov indeks (angl. Rand Index – RI),
- popravljen Randov indeks (angl. Adjusted Rand Index – ARI),
- Jaccardov indeks (angl. Jaccard's Index – JI),
- razvrstitvena napaka (angl. Clustering Error – CE),
- variacija informacije (angl. Variation of information – VI),
- mera $ADCO$ (angl. Attribute Distribution Clustering Orthogonality).

Princip delovanja vseh metod zunanjega primerjanja (razen zadnjih dveh) je podoben – štetje soglasij in nesoglasij med pari vzorcev. Pri tem si pomagamo s kontingenčno tabelo (angl. contingency table, confusion matrix). Za razvrstitvi C in C' jo prikazuje tabela 3.1.

Tabela 3.1: Kontingenčna tabela za razvrstitvi C in C'. Znak + označuje ujemanje, − pa neujemanje parov vzorcev.

$C \backslash C'$	+	−
+	$N_{(++)}$	$N_{(+-)}$
−	$N_{(-+)}$	$N_{(--)}$

V tabeli 3.1 nam števila $N_{(++)}$, $N_{(+-)}$, $N_{(-+)}$ in $N_{(--)}$ pomenijo naslednje:

- $N_{(++)}$: število parov vzorcev, ki so v isti gruči v obeh razvrstitvah, C in C',

- $N_{(+-)}$: število parov vzorcev, ki so v isti gruči v razvrstitvi C in v različnih gručah v razvrstitvi C',
- $N_{(-+)}$: število parov vzorcev, ki so v isti gruči v razvrstitvi C' in v različnih gručah v razvrstitvi C,
- $N_{(--)}$: število parov vzorcev, ki so v različnih gručah v obeh razvrstitvah C in C'.

Veljati mora naslednje:

$$N_{(++)} + N_{(+-)} + N_{(-+)} + N_{(--)} = \frac{N(N-1)}{2} = \binom{N}{2} = N_p \quad , \tag{3.38}$$

kjer je N število vseh vzorcev in N_p število vseh možnih parov med njimi. Z ozirom na napisano, si sedaj oglejmo posamezne kriterije.

Randov indeks

W. M. Rand [16] je svoj kriterij oziroma mero podobnosti med dvema razvrstitvima, C in C', definiral kot

$$RI(C, C') = \frac{N_{(++)} + N_{(--)}}{N_p} \quad . \tag{3.39}$$

Kriterij $RI(C, C')$ lahko zavzema vrednosti od 0 do 1; 1 je takrat, kadar se razvrstitvi C in C' popolnoma ujemata, 0 pa, ko popolnoma razlikujeta. Ker bomo pozneje vseh 6 zunanjih kriterijev uporabljali v istem kontekstu, bi bilo bolj pregledno, če bi vsi delovali podobno. Se pravi, da bi pri popolnem ujemanju vsi zavzeli vrednost 0 in obratno. Zato vrednost kriterija RI odštejemo od 1.

Popravljen Randov indeks

Randov indeks ima pomanjkljivost, in sicer da njegova srednja vrednost nad dvema naključnima spremenljivkama ne zavzame konstantne vrednosti [22], zato sta Hubert in Arabie [7] predlagala izboljšavo. Pred zapisom enačbe pa je potrebno kontingenčno tabelo 3.1 razširiti tako, da zapišemo ujemanja za vsako gručo posebej, kot to prikazuje tabela 3.2.

Srednjo vrednost indeksa RI zapišemo kot
$E[RI(C, C')] = E_{RI}$:

Tabela 3.2: Razširjena kontingenčna tabela za razvrstitvi C in C'.

$C \backslash C'$	C'_1	C'_2	$\dots$	C'_K	Vsota
C_1	N_{11}	N_{12}	$\dots$	N_{1K}	$N_{1.}$
C_2	N_{21}	N_{22}	$\dots$	N_{2K}	$N_{2.}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
C_K	N_{K1}	N_{K2}	$\dots$	N_{KK}	$N_{K.}$
Vsota	$N_{.1}$	$N_{.2}$	$\dots$	$N_{.K}$	N

$$E_{RI} = E\left[\sum_{i,j}\binom{N_{ij}}{2}\right] = \left[\sum_{i}\binom{N_{i.}}{2}\sum_{j}\binom{N_{.j}}{2}\right] / \binom{N}{2} \quad . \tag{3.40}$$

Sedaj lahko popravljen Randov indeks ali ARI zapišemo kot

$$ARI(C, C') = \frac{\sum_{i,j}\binom{N_{ij}}{2} - E_{RI}}{\frac{1}{2}\left[\sum_i \binom{N_{i.}}{2} + \sum_j \binom{N_{.j}}{2}\right] - E_{RI}} \quad . \tag{3.41}$$

Zanj velja isto kot pri originalnemu RI: vrednost 1 pomeni popolno ujemanje dveh razvrstitev, 0 pa njuno popolno neujemanje. Zaradi konsistence z ostalimi kriteriji bomo to pojmovanje obrnili, podobno kot pri RI.

Jaccardov indeks

Paul Jaccard [20] je zapisal svojo mero podobnosti dveh razvrstitev takole:

$$JI(C, C') = \frac{N_{(++)}}{N_{(++)} + N_{(+-)} + N_{(-+)}} \quad . \tag{3.42}$$

Ta mera se veliko uporablja tudi kot mera podobnosti med dvema binarnima spremenljivkama. Za našo uporabo bo prav prišla vrednost $1 - JI(C, C')$, ki je 1, ko sta si C in C' popolnoma različni in 0 ob popolnem ujemanju.

Razvrstitvena napaka

Razvrstitvena napaka (angl. clustering error) [13] je odstotek napačno razvrščenih vzorcev razvrstitve C', če jo primerjamo z C. Za razliko od prej

omenjenih indeksov tu ne gre za štetje parov vzorcev, temveč za ugotavljanje ujemanja gruč (angl. set matching). Razvrstitveno napako izračunamo tako, da seštejemo nediagonalne elemente razširjene kontingenčne tabele 3.2 (predstavljajo nam napake) pri vseh možnih permutacijah stolpcev in vrstic. Nato izmed vseh permutacij izberemo tisto, ki ima najmanjšo vrednost vsote in jo poimenujemo razvrstitvena napaka ali *CE*. Ker jo želimo v odstotkih, pred seštevanjem vse elemente kontingenčne tabele normiramo s številom vzorcev N. Pri popolnem ujemanju dveh razvrstitev je *CE* enak 0, ob popolnem neujemanju pa 1.

Variacija informacije

M. Meilă [13] je predlagala, da bi se za primerjavo dveh razvrstitev vzorcev v gruče (razvrstitvi poimenujmo tako kot do sedaj, C in C'), uporabila teorija informacije in mere, ki spadajo v to področje. Glavna ideja je, da ugotovimo, koliko informacije o razvrstitvi C izgubimo in koliko o C' jo je potrebno še pridobiti, ko gremo od opazovanja C na opazovanje razvrstitve C'.

Najprej definirajmo entropijo (nedoločenost) razvrstitve C:

$$H(C) = -\sum_{k=1}^{K} p(k) \log p(k) \quad , \tag{3.43}$$

kjer je $p(k)$ verjetnost, da naključno izbran vzorec iz množice vzorcev P spada v gručo C_k razvrstitve C. Izračuna se po enačbi: $p(k) = \frac{N_{C_k}}{N}$. $H(C)$ je torej nedoločenost povezana z razvrstitvijo C, na isti način pa izračunamo tudi $H(C')$. Na entropijo $H(C)$ lahko gledamo na sledeči način: entropija $H(C)$ je nedoločenost, v katero gručo v razvrstitvi C spada naključno izbrani vzorec iz P. Entropija ni odvisna od števila vzorcev, temveč zgolj od relativnih razmerij med velikostmi gruč.

Zanima nas še, kolikšna je medsebojna informacija (angl. mutual information) med dvema razvrstitvama, se pravi, kolikšna je informacija, ki jo ena razvrstitev vsebuje o drugi. Medsebojno informacijo označimo z $I(C; C')$ in je definirana kot

$$I(C; C') = \sum_{k=1}^{K} \sum_{k'=1}^{K'} p(k, k') \log \frac{p(k, k')}{p(k)p(k')} \quad , \tag{3.44}$$

kjer je $p(k, k')$ verjetnost, da nek vzorec pripada gruči C_k v razvrstitvi C in gruči $C'_{k'}$ v razvrstitvi C'. Definirana je tako:

$$p(k,k') = \frac{|C_k \cap C'_{k'}|}{N} \quad . \tag{3.45}$$

Predlagan kriterij za primerjavo dveh razvrstitev – variacija informacije (VI) se glasi:

$$VI(C,C') = H(C) + H(C') - 2I(C;C') \quad . \tag{3.46}$$

Mera VI zavzema vrednosti med 0 (razvrstitvi sta si enaki) in $\log N$ (razvrstitvi sta si popolnoma različni). Količino mere VI lahko potemtakem normiramo z $\log N$, da zavzema vrednosti med 0 in 1 kot vsi ostali kriteriji v primerjavi.

Mera ADCO

Problem do sedaj obravnavanih zunanjih kriterijev ocenjevanja razvrstitev je ta, da v obzir ne jemljejo razporejenosti vzorcev, temveč zgolj njihovo pripadnost gručam. E. Bae, idr. [1] je zato predlagal novo metodo primerjanja, ki temelji na ugotavljanju ujemanja gostote vzorcev med seboj. Predstavljajmo si, da vsako razsežnost vzorcev (oziroma njihove atribute) razrežemo na q enakih pasov. Vsi pasovi skupaj tvorijo nekakšno hiper (večrazsežno) mrežo. Vsak vzorec se hkrati nahaja v natančno eni celici te mreže. Mera ADCO pokaže, kako podobno so po teh celicah hiper mreže porazdeljeni vzorci dveh različnih razvrstitev.

Začnimo formalnosti s tem, da definiramo d_i kot i-to razsežnost sicer D-razsežne množice vzorcev P. Če si predstavljamo vzorce iz množice P urejene v matriko dimenzij $N \times D$, potem je d_i i-ti stolpec te matrike. Ko si to dobro predstavljamo, vse stolpce razrežemo na q enakih pasov. Potem vpeljimo d_i^j kot množico vrednosti razsežnosti d_i, ki pripadajo pasu j. Sedaj lahko za vsako gručo razvrstitve C preštejemo, koliko vzorcev se uvrsti v določen pas posamezne razsežnosti. To imenujemo gostota gruče C_k v pasu j razsežnosti d_i ali na kratko

$$dens_{C_k}(d_i, j) = |x \in C_k; x(d_i) \in d_i^j| \quad . \tag{3.47}$$

Izračunamo vse gostote preko vseh K gruč in vseh q pasov vseh D razsežnosti znotraj posamezne gruče. Iz dobljenih vrednosti za vsako gručo posebej sestavimo vektor gostot dolžine $D \cdot q$, ki je za gručo C_k videti tako:

$$\begin{aligned} dens_{C_k} \quad = \quad & \{dens_{C_k}(d_1,1), dens_{C_k}(d_1,2), \ldots, dens_{C_k}(d_1,q), \\ & dens_{C_k}(d_2,1), \ldots, dens_{C_k}(d_D,q)\} \quad . \end{aligned} \tag{3.48}$$

Sedaj lahko definiramo operacijo produkta med dvema razvrstitvama, C in C', kot skalarni produkt med vektorji gostote:

$$C \cdot C' = \sum_{k=1}^{K} dens_{C_k} \cdot dens_{C'_k} \quad . \tag{3.49}$$

Visoka vrednost skalarnega produkta dveh vektorjev gostote pomeni visoko podobnost. Ker v enačbi 3.49 primerjamo podobnost razvrstitev paroma po gručah (torej C_1 z C'_1, C_k z C'_k), moramo upoštevati tudi vse možne druge permutacije razvrstitve C', da bo rezultat neodvisen od izbrane označbe gruč (pri razvrščanju oznaka gruč namreč ni pomembna). Izmed vseh možnih permutacij torej izberemo tisto z največjo vrednostjo, ki jo formalno zapišemo kot

$$PWS(C, C') = \max_{P} [C \cdot P(C')] \quad , \tag{3.50}$$

kjer gre P preko vseh permutacij gruč v razvrstitvi C'. V zadnjem koraku želimo prilagoditi obravnavano kriterijsko mero ostalim, da bo zavzemala vrednosti od 0 (ujemanje razvrstitev je popolno) do 1 (ujemanja ni). Za to je potrebna normalizacija z največjo možno mero podobnosti, ki je: $MaxS(C, C') = \max(C \cdot C, C' \cdot C')$. Končno torej dobimo izraz

$$ADCO(C, C') = 1 - \frac{PWS(C, C')}{MaxS(C, C')} \quad , \tag{3.51}$$

pri čemer vrednost 0 pomeni popolno ujemanje in 1 popolno neujemanje dveh razvrstitev.

3.3 Množice vzorcev

Težavnost razvrščanja vzorcev v gruče je pogojena s tem, kakšno množico vzorcev si za razvrščanje izberemo. Lahko so gruče v njej jasno definirane in določljive, lahko pa je meja med njimi zelo zabrisana. Pomembna je tudi oblika gruč – so kompaktne ali so razpotegnjene. Nekaterim algoritmom delajo preglavice vzorci, ki ležijo daleč stran od ostalih (samotarji ali angl. outliers) in težijo k temu, da postanejo svoja gruča. Pojavi se tudi velik problem razvrščanja vzorcev, ki imajo veliko razsežnost, sploh če je ta celo večja od števila vzorcev.

Da bi izbrane algoritme čim bolj vsestransko primerjal, sem izbral testne množice vzorcev, ki imajo tipične problematične značilnosti, o katerih sem govoril v prejšnjem odstavku. Nekatere od njih so umetno ustvarjene, druge so

vzete iz realnih meritev. Vse množice vzorcev so bile pred obdelavo standardizirane, in sicer vsaka razsežnost posebej na interval $[-1, 1]$ s srednjo vrednostjo 0. V taki obliki so v nadaljevanju tudi prikazane.

V primerjavi algoritmov med seboj bodo nastopale naslednje množice vzorcev[5]:

- *Gauss4* *,
- *Sinus2* *,
- *TriSpot* *,
- *Luna* *,
- *Iris*,
- *Wine*,
- *Lung cancer*.

3.3.1 Gauss4

Množica vzorcev *Gauss4* vsebuje 400 dvorazsežnih vzorcev, ki so oblikovani v 4 dotikajoče se gruče. Vsaka gruča vsebuje 100 vzorcev. Izris vzorcev prikazuje slika 3.2. Središča gruč so postavljena v oglišča kvadrata s stranico dolžine 4 in središčem v (0,0). Vzorci so bili ustvarjeni naključno po normalni verjetnostni porazdelitvi s srednjo vrednostjo v ogliščih kvadrata in varianco 1.

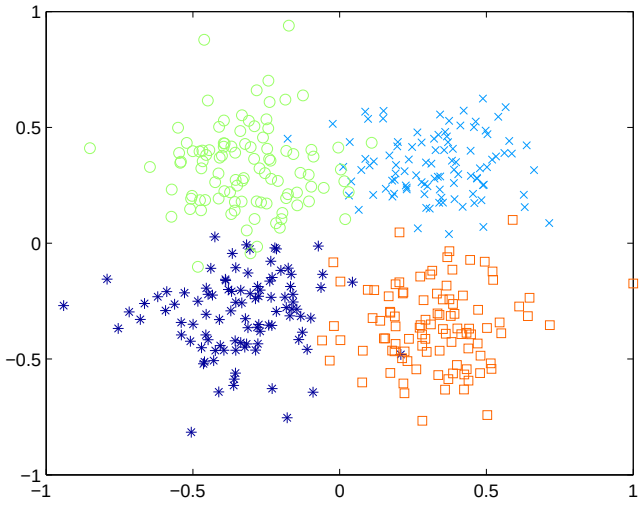
Značilnost: Prekrivanje gruč med sabo, nejasne meje med njimi.

3.3.2 Sinus2

800 dvorazsežnih vzorcev je v množici *Sinus2* razporejenih v obliki dveh sinusnih krivulj, ena nad drugo, kot prikazuje slika 3.3. Vsaka vsebuje 400 vzorcev, ki so razporejeni enakomerno po normalni porazdelitvi okrog nosilne linije $sin(-\pi, \pi)$, premaknjene za 2 navzgor oziroma navzdol.

Značilnost: Gruči sta izrazito podolgovati. Največja razdalja med vzorcem in središčem njegove gruče je večja kot razdalja med vzorci različnih gruč.

[5]Umetne množice vzorcev, označene z *, sem poimenoval sam.



Slika 3.2: Izris množice vzorcev *Gauss4*, ki vsebuje 4 gruče – prikazane so z različnimi barvnimi znaki.

3.3.3 TriSpot

550 dvorazsežnih vzorcev je v množici *TriSpot* razporejenih v obliki treh podolgovatih gruč s po 150 vzorci, sredi katerih je še ena okrogla s 100 vzorci. Množica vzorcev je vzeta iz [9], prikazuje pa jo slika 3.4.

Značilnost: Gruče so različnih oblik in imajo različno število vzorcev.

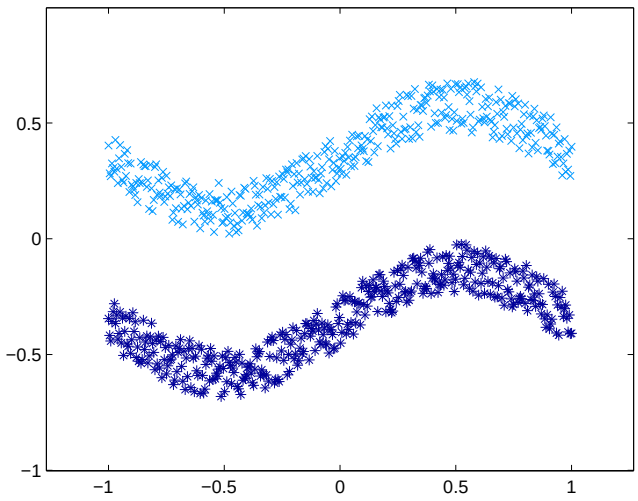
3.3.4 Luni

514 dvorazsežnih vzorcev je v množici *Luni* razporejenih v štiri gruče izrazito nepravilnih oblik, ki spominjajo na krajec lune ali polkrog. Gruče imajo po 104, 110, 150 in 150 vzorcev. Množica vzorcev je vzeta iz [9], prikazuje pa jo slika 3.5.

Značilnost: Gruče so izrazito nepravilnih oblik in so med seboj težko ločljive.

3.3.5 Iris

Iris je najbrž najbolj znana in citirana množica vzorcev v literaturi na področju razpoznavanja vzorcev. 150 štirirazsežnih vzorcev je v množici *Iris* razporejenih v tri gruče po 50 vzorcev. Leta 1936 jo je objavil Sir R. A. Fisher na



Slika 3.3: Izris množice vzorcev *Sinus2*, ki vsebuje 2 gruči v obliki sinusnih krivulj – prikazani sta z različnimi barvnimi znaki.

podlagi podatkov, ki jih je zbral E. Anderson, ko je preučeval geografske vplive na lastnosti perunik (lat. iris). Meril je dolžino in širino venčnih (angl. petal) in čašnih (angl. sepal) listov treh različnih vrst perunike: *Perunika setosa*, *Perunika versicolour*, *Perunika virginica*. Množica vzorcev je prosto dostopna na spletu v zbirki podatkov UCI[6].

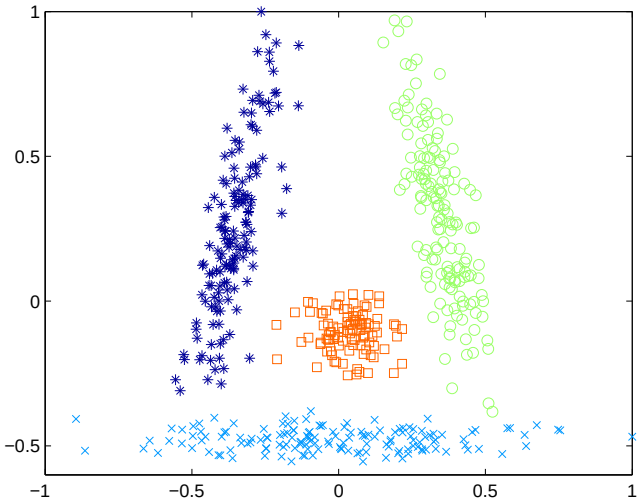
Ker je množica vzorcev 4-razsežna, jo je nemogoče enostavno prikazati v dveh dimenzijah. Zato sem uporabil metodo za transformacijo večrazsežnih podatkov na nižje stopnje razsežnosti, katere rezultat prikazuje slika 3.6. Imenuje se analiza glavnih komponent ali PCA (angl. Principal Components Analysis) in deluje na principu maksimizacije variance podatkov, ko jih ortogonalno projeciramo na nižje dimenzije [2, 21]. Na projecirane podatke tako gledamo iz najbolj informativne perspektive.

Značilnost: Ena gruča je linearno ločljiva od drugih dveh, medtem ko slednji med seboj nista.

3.3.6 Wine

Množica vzorcev *Wine* vsebuje podatke o kemični analizi sestavin treh vrst vina, pridelanega na istem področju v Italiji. Izmerjenih je 13 kemičnih last-

[6]UCI Machine learning repository, dostopno na: http://archive.ics.uci.edu/ml/datasets .



Slika 3.4: Izris množice vzorcev *TriSpot*, ki vsebuje 4 gruče, prikazane z različnimi barvnimi znaki.

nosti (raven alkohola, kislost, intenziteta barve, raven magnezija, itd.) na 173 vzorcih vina. Množica vzorcev je prosto dostopna na spletu v zbirki podatkov UCI. Za prikaz na sliki 3.7 je, kot v prejšnjem primeru, uporabljena analiza PCA.

Značilnost: Vzorci imajo veliko atributov – veliko stopnjo razsežnosti.

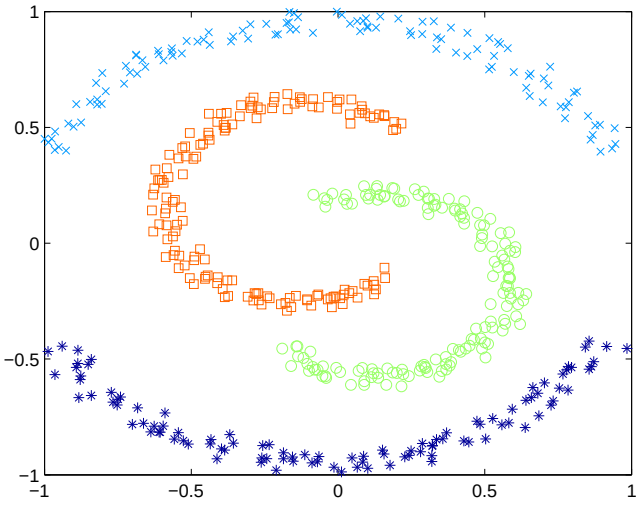
3.3.7 Lung cancer

O tej množici vzorcev je malo znanega, zgolj to, da opisuje 3 patološke vrste pljučnega raka. Vzorcev je samo 27, atributov oziroma razsežnosti pa kar 56, kar predstavlja za vse algoritme razvrščanja precejšen zalogaj. V izvirni množici je bilo vzorcev 32, vendar so imeli pri določenih atributih manjkajoče vrednosti, zato sem jih 5 odstranil iz obravnave. Množica vzorcev, prikazana na sliki 3.8, je prosto dostopna na spletu v zbirki podatkov UCI.

Značilnost: Število atributov (razsežnost vzorcev) je veliko večja od števila vzorcev.

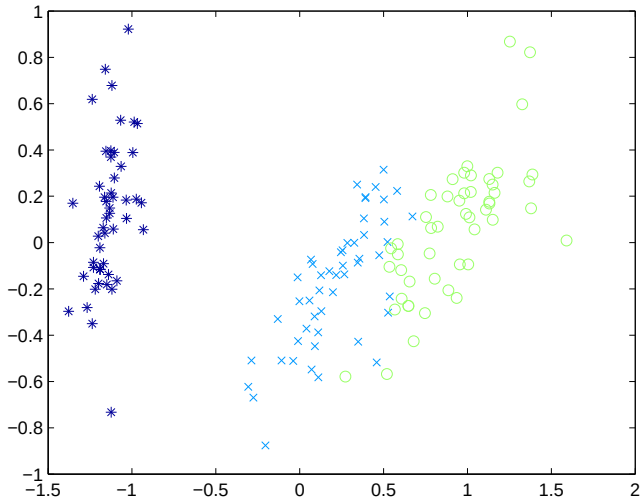
Lastnosti vseh množic vzorcev so pregledno zbrane v tabeli 3.3.

Sedaj imamo vse pripravljeno za izvedbo primerjave med algoritmi za razvrščanje. V naslednjem poglavju bom najprej predstavil postopek primerjanja, nato pripravo podatkov (množic vzorcev in vhodnih parametrov v algo-

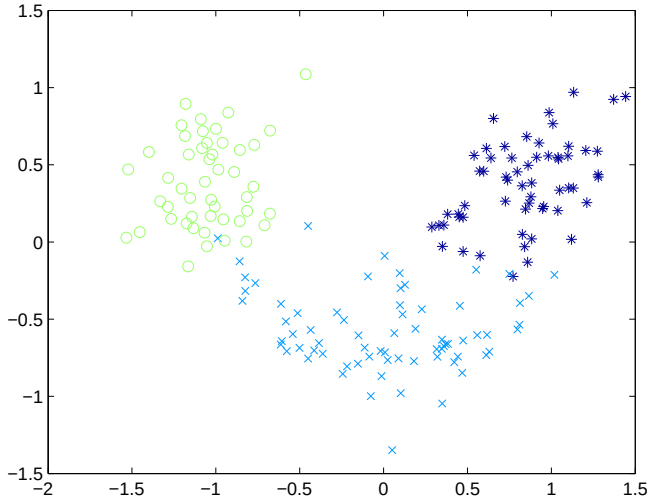


Slika 3.5: Izris množice vzorcev *Luni*, ki vsebuje 4 gruče, prikazane z različnimi barvnimi znaki.

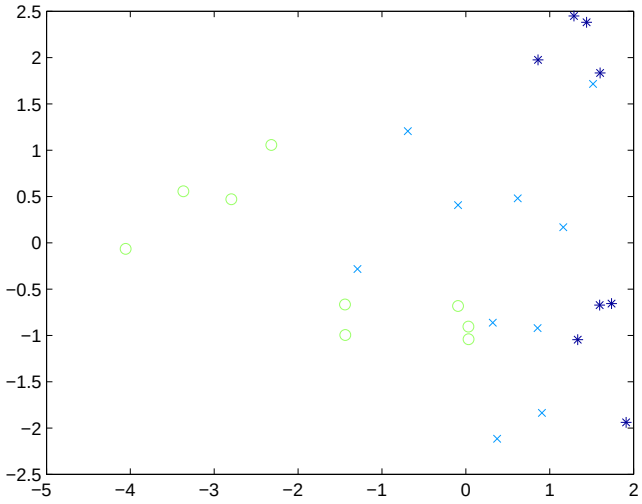
ritme) in končno rezultate primerjave.



Slika 3.6: Izris množice vzorcev *Iris* z uporabo metode PCA. Prikazana je projekcija glede na dva najpomembnejša lastna vektorja.



Slika 3.7: Izris množice vzorcev *Wine* z uporabo metode PCA. Prikazana je projekcija glede na dva najpomembnejša lastna vektorja.



Slika 3.8: Izris množice vzorcev *Lung cancer* z uporabo metode PCA. Prikazana je projekcija glede na dva najpomembnejša lastna vektorja.

Tabela 3.3: Tabela prikazuje lastnosti opisanih množic vzorcev. N pomeni število vzorcev, D njihovo razsežnost in K število gruč v pravilni (idealni) razvrstitvi.

ime	N	D	K	vrsta
Gauss4	400	2	4	umetna
Sinus2	800	2	2	umetna
TriSpot	550	2	4	umetna
Luni	514	2	4	umetna
Iris	150	4	3	realna
Wine	178	13	3	realna
L. cancer	27	56	3	realna

Poglavje 4

Rezultati

4.1 Postopek primerjave

Primerjavo algoritmov med seboj sem izvedel kot tekmovanje, kjer glavno vlogo sodnikov igrajo zunanji kriteriji za ocenjevanje uspešnosti razvrstitve. Za vsako množico vzorcev posebej poznamo pravilni rezultat, se pravi želeno razvrstitev. Ta je v primeru umetnih podatkov (*Gauss4, Sinus2, TriSpot, Luni*) pridobljena kar iz načina, kako so bili vzorci ustvarjeni. V primeru realnih množic vzorcev je pravilna razvrstitev podana s strani avtorjev teh množic. Če pravilnih razvrstitev ne bi poznali, bi bilo ocenjevanje uspešnosti z zunanjimi kriteriji nemogoče. V takih primerih nam preostanejo še notranji kriteriji, ki so v mojem primeru služili za identifikacijo pravilnega ali optimalnega števila gruč v množici vzorcev. Ker sem pravo oziroma optimalno število gruč določil, je bila predmet mojega zanimanja uspešnost notranjih kriterijev pri napovedovanju le-teh.

Vsak algoritem je na vsaki množici vzorcev izvedel razvrščanje na k gruč, kjer je $k = 2, 3, \ldots, 10$. Vsako razvrščanje se je izvedlo desetkrat, razen pri algoritmu ECMC, ki se bom posvetil nekaj odstavkov naprej. Izmed desetih razvrstitev vsakega algoritma za določen k, je bila kot najboljša izbrana tista, ki je izkazovala najboljši rezultat glede na kriterijsko funkcijo, ki je lastna vsakemu algoritmu. Tako je bila denimo izmed 10 razvrstitev vzorcev v 4 gruče algoritma KMC izbrana tista, ki je dala najmanjšo vrednost vsote kvadratov razdalj J (glej enačbo 3.2). Tako je vsak algoritem predlagal svojo najboljšo rešitev problema, ki je bila nato ocenjena z notranjimi kriteriji *DB*, *CH*, *Hom & Sep* in *Wtertra*.

Ker poznamo pravilne razvrstitve in pravilno število gruč K, sem torej v ocenjevanje z zunanjimi kriteriji dal vseh deset razvrstitev posameznega algo-

ritma pri razvrščanju v K gruč in izračunal srednjo vrednost ter standardni odklon. Posebej sem izpostavil tudi rezultat razvrstitve, ki jo algoritem razume kot najboljšo. Na koncu sem rezultate ocenjevanja po zunanjih kriterijih točkoval glede na uvrstitev (3 točke za prvo mesto, 2 za drugo, 1 za tretje in 0 za zadnje). V primeru, da ni bilo enoumnega rezultata – zunanji kriteriji se niso ujemali – sem upošteval večinsko mnenje kriterijev. Med izvajanjem algoritmov sem meril tudi njihov povprečni čas preko vseh razvrščanj (9 x 10 za vsak algoritem).

Omenil sem, da je algoritem ECMC izjema, kar se izvedbe primerjave tiče, saj je njegovo delovanje precej drugačno od ostalih treh. Njegova bistvena lastnost je, da deluje sprotno, iz česar sledi, da za enak vrstni red vzorcev dobimo vedno isti rezultat razvrščanja. Algoritem namreč jemlje vzorce iz množice po vrsti, enega po enega. Zato je nesmiselno za isto sekvenco podatkov postopek večkrat ponavljati. Sicer načeloma obstaja možnost, da vrstni red vzorcev ob vsaki ponovitvi premešamo in s tem spremenimo vhodno zaporedje, vendar se tu pojavi druga omejitev. Ker želimo, da so izbrani algoritmi med seboj čimbolj primerljivi, moramo zagotoviti razvrščanje, pogojeno z želenim številom gruč. Torej, če pravilna razvrstitev vzorcev vsebuje K gruč, jih mora točno toliko vsebovati tudi vsaka razvrstitev, ki je primerjana s pravilno. To je pogoj, ki ga zahtevajo zunanji kriteriji ocenjevanja. Metoda ECMC pa kot vhodni parameter ne vzame želenega števila gruč, temveč največjo dovoljeno razdaljo med posameznim vzorcem in središčem njegove gruče, označeno kot D_{thr}. Zato je potrebno nekaj dodatnega naprezanja, da ugotovimo, katera vrednost D_{thr} povzroči razvrstitev v želeno število gruč. V ta namen sem implementiral postopek, ki s poskušanjem išče najmanjšo mogočo razdaljo D_{thr}, ki še zagotavlja želeni k – število gruč. Vrednosti D_{thr} si za vsak $k = 2, 3, \ldots, 10$ zapomnimo in uporabimo v primerjavi. Problem, ki se pojavi pri tem je, da ni zagotovila, da obstaja taka vrednost D_{thr}, ki bi povzročila razvrstitev v želeno število gruč. Iz tega izhaja dejstvo, da poljubno mešanje podatkov ni priporočljivo, ker mogoče za nek k ne dobimo rezultata in bi bila primerjava z zunanjimi kriteriji neveljavna. S tem utemeljujem svojo odločitev, da algoritem ECMC za vsako razvrščanje v k gruč poženem zgolj enkrat.

Če povzamem pravila tekmovanja med metodami razvrščanja: nad vsako od sedmih množic podatkov se izvede razvrščanje vzorcev v 2, 3,..., 10 gruč. Vsako razvrščanje se ponovi 10-krat; vseh 10 razvrščanj se ovrednoti z zunanjimi kriteriji in izračuna povprečno vrednost. Posebej se primerjajo najboljše razvrstitve (najboljše glede na kriterijsko funkcijo posameznega algoritma). Algoritme nato točkujemo glede na mesto, ki so ga zasedli na naslednji način: 3 točke za 1. mesto, 2 za 2., 1 za 3. in 0 točk za zadnje mesto. Posebej

točkujemo uvrstitev glede na povprečno vrednost zunanjih kriterijev (kategorija povprečje) in tisto pri najboljši razvrstitvi (kategorija prvi). V primeru, da rezultati zunanjih kriterijev soglasno ne določijo vrstnega reda, se ravnamo po večinskem mnenju. Na primer: zmaga tisti, za katerega se opredeli večina od 6 kriterijev. Vzporedno opazujemo notranje kriterije, ki napovedujejo pravilno število gruč določene razvrstitve in jih glede na uspeh pri tem komentiramo.

4.2 Priprava podatkov in vhodnih parametrov

Pred izvedbo samega tekmovanja med algoritmi je potrebno opraviti standardizacijo vhodnih množic vzorcev kot tudi zagotoviti čimbolj optimalne vhodne parametre algoritmov. Zlasti se je potrebno posvetiti nastavljanju parametrov algoritmov ECMC in CSC.

4.2.1 Standardizacija podatkov

Vse podatke iz množic vzorcev je pred razvrščanjem potrebno standardizirati oziroma zagotoviti, da ima vsak atribut ali razsežnost pri razvrščanju enako težo. Vsi atributi vzorcev so bili preslikani na interval $[-1, 1]$ s srednjo vrednostjo 0. Slednje za d-to razsežnost i-tega vzorca matematično zapišemo kot

$$x_i^{(d)\prime} = \frac{x_i^{(d)} - \mu_d}{\sigma_d} \quad , \tag{4.1}$$

kjer je μ_d srednja vrednost d-te razsežnosti, σ_d pa njen standardni odklon.

Vzorce nadalje preslikamo na interval $[-1, 1]$ tako, da za vsako d-to razsežnost vzorcev najdemo njeno absolutno največjo vrednost in z njo delimo vse ostale:

$$x_i^{(d)\prime} = \frac{x_i^{(d)}}{\max_i(|x_i^{(d)}|)} \quad . \tag{4.2}$$

4.2.2 Nastavljanje vhodnih parametrov

Z željo, da bi vsak algoritem dal od sebe najboljši rezultat, je bilo potrebno precej predpriprav v smislu nastavljanja parametrov, ki določajo sam proces razvrščanja. Tabela 4.1 prikazuje parametre, ki jih kot vhod zahtevajo izbrani algoritmi.

Tabela 4.1: Vhodni parametri za izbrane algoritme.

	parametri
KMC	K, $numIter$
ECMC	D_{thr}, $numIter$
EM GMM	K, $numIter$
CSC	K, $K_{\text{zač}}$, $N_{\text{zač}}$, σ

Vsi parametri, razen $numIter$, so opisani v poglavju 3.1. Parameter $numIter$ pomeni največje število iteracij, ki jih algoritem lahko napravi. Postavljen je bil fiksno na 100 in izkazalo se je, da jih v nobenem primeru algoritmi ne dosežejo, zato to ni bila omejitev za izhod.

Zadrego s parametrom D_{thr} algoritma ECMC sem že omenil in opisal. Naj samo povzamem, da sem s poglobljenim poizkušanjem za vsak $k = 2, 3, \ldots, 10$ našel najmanjši še ustrezni D_{thr}. Natančnost najdenega D_{thr} je reda velikosti 10^{-5}.

Naslednji problem, ki se je pojavil pred samim primerjanjem algoritmov je določitev parametrov metode CSC. Po priporočilih avtorjev metode (Jenssen idr.) sem parameter $K_{\text{zač}}$ postavil na vrednost 20 in $N_{\text{zač}}$ na 10, razen kadar je bilo število vzorcev v množici manjše od 200. V takih primerih sem število $N_{\text{zač}}$ ustrezno zmanjšal, da je bila pokritost vzorcev z začetnimi gručami približno 50 %. Eksperimentiranje z različnimi vrednostmi teh dveh parametrov je pokazala, da sam rezultat razvrščanja niti ni toliko odvisen od njiju, če le nista premajhna. Večji vpliv ima parameter σ, ki določa širino Gaussovih jedrnih funkcij. Določanja σ sem se lotil tako, da sem interval [0,01, 0,7] razdelil na podintervale s korakom 0,01 in tako dobljene vrednosti uporabil kot σ pri razvrščanju vsake množice vzorcev. Tak interval je bil izbran na podlagi prejšnjih izkušenj in se je izkazal za dovolj širokega, da je zajel vse optimalne vrednosti. Za vsako vrednost σ je bilo opravljenih 5 razvrščanj istih vzorcev. Rezultat (vrednost kriterijske funkcije D_{CS}) sem nato povprečil in primerjal s tistimi pri ostalih vrednostih σ. Končna vrednost parametera σ pri določeni množici vzorcev je bila tista, ki je v povprečju dala višjo vrednost kriterijske funkcije D_{CS}, ni pa nujno, da je to najboljša izbira iz zornega kota zunanjih kriterijev za ocenjevanje.

4.3 Rezultati in ocenjevanje razvrščanja

V tem razdelku so podani rezultati primerjave oziroma tekmovanja med algoritmi KMC, ECMC, EM GMM in CSC. Razdeljeni so po sklopih glede na množico vzorcev. Znotraj vsakega sklopa so rezultati predstavljeni kot:

- tabela ocen zunanjih kriterijev,
- grafični prikaz najboljšega razvrščanja za vnaprej določeno optimalno število gruč,
- tabela napovedanega optimalnega števila gruč s strani notranjih kriterijev,
- grafični prikaz notranjih kriterijev [1].

Rezultati posameznih algoritmov se točkujejo od 0 (najslabši) do 3 (najboljši) za dve kategoriji: prvi in povprečje. Prvi pomeni rezultat pri tisti razvrstitvi v optimalno število gruč, ki je za algoritem najboljša. Povprečje je povprečje rezultatov vseh desetih tekov algoritma za določeno število gruč. Tabeli, ki prikazujeta končni skupni rezultat točkovanja in povprečne izvajalne čase, sta podani na koncu poglavja.

4.3.1 Gauss4

Ocene zunanjih kriterijev so podane v tabeli 4.2, kjer so najboljše vrednosti v mastnem tisku. Algoritem KMC je zmagal v kategoriji prvi kot tudi v kategoriji povprečje. Rezultati so si na splošno blizu skupaj, saj ni šlo za težaven problem. Napake v razvrščanju so očitno posledica prekrivanja gruč med seboj.

Najboljše razvrstitve so grafično prikazane na sliki 4.1, potek notranjih kriterijev za algoritem KMC pa na sliki 4.2. V poglavju 3.2.1 so opisani postopki za določitev optimalnega števila gruč glede na potek vrednosti notranjih indeksov. Na podlagi tega je narejena tabela 4.3, ki prikazuje napovedi pri vseh algoritmih. Vidimo, da so napovedi v veliki večini primerov točne, kar očitno pomeni, da je bila množica vzorcev dovolj enostavna v smislu kompaktnosti gruč.

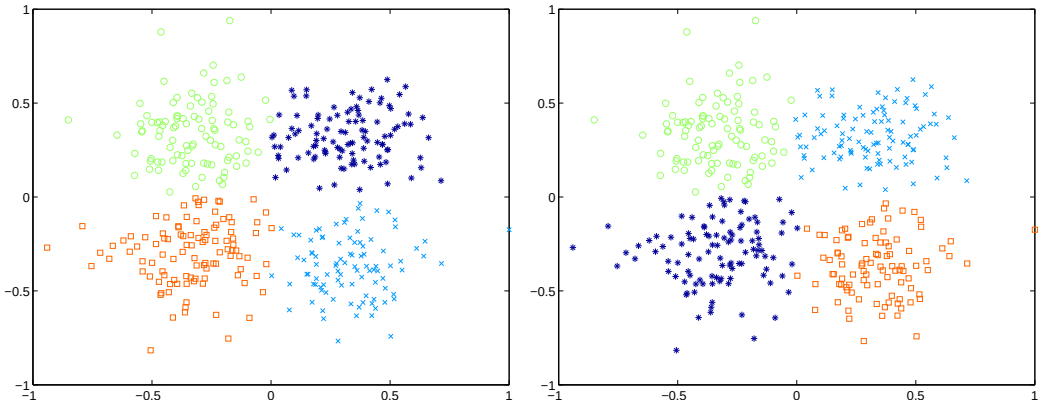
[1] Ker bi bilo slik s prikazi potekov notranjih indeksov za vsak algoritem in vsako množico vzorcev posebej preveč, bom prikazal samo tiste za najboljši algoritem.

Tabela 4.2: Rezultati ocenjevanja razvrščanja z zunanjimi kriteriji – množica *Gauss4*. Prikazana je vrednost kriterijev za najboljšo razvrstitev (prvi), povprečje preko 10 poizkusov (povpr.) in standardni odklon od povprečja (odklon).

		RI	*ARI*	*JI*	*CE*	*VI*	*ADCO*
KMC	prvi	**0,0483**	**0,129**	**0,177**	**0,0500**	**0,106**	**0,00535**
	povpr.	**0,0483**	**0,129**	**0,177**	**0,0500**	**0,106**	**0,00535**
	odklon	0	0	0	0	0	0
ECMC	prvi	0,0506	0,135	0,185	0,0525	0,110	0,00599
	povpr.	0,0506	0,135	0,185	0,0525	0,110	0,00599
	odklon	0	0	0	0	0	0
EM GMM	prvi	0,0529	0,142	0,192	0,0550	0,113	0,00661
	povpr.	0,0529	0,142	0,192	0,0550	0,113	0,00661
	odklon	0	0	0	0	0	0
CSC	prvi	0,0506	0,135	0,185	0,0525	0,110	0,00598
	povpr.	0,0524	0,140	0,190	0,0545	0,112	0,00587
	odklon	0,00461	0,0123	0,0150	0,00497	0,00757	0,00107

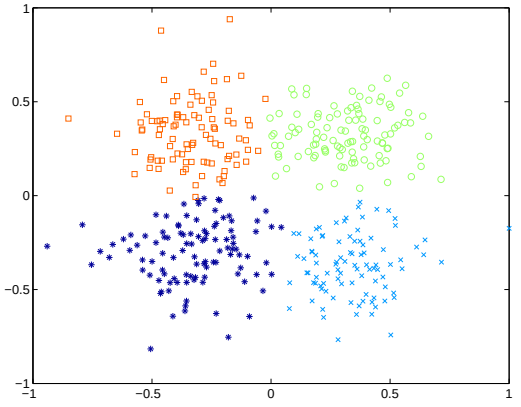
Tabela 4.3: Z notranjimi kriteriji napovedano optimalno število gruč za množico vzorcev *Gauss4*. Pravilno število gruč je 4.

	DB	*CH*	*Hom & Sep*	*Wtertra*
KMC	4	4	4	3
ECMC	4	4	4	5
EM GMM	4	4	4	4
CSC	4	4	4	4

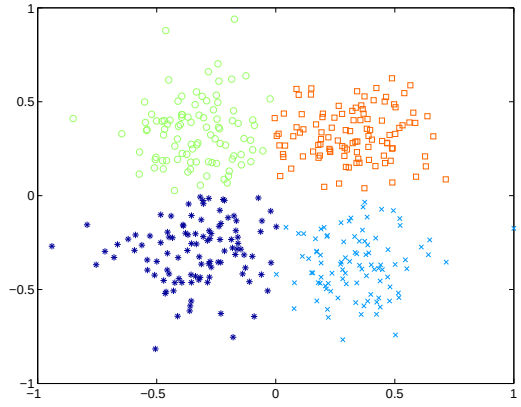


(a) KMC

(b) ECMC

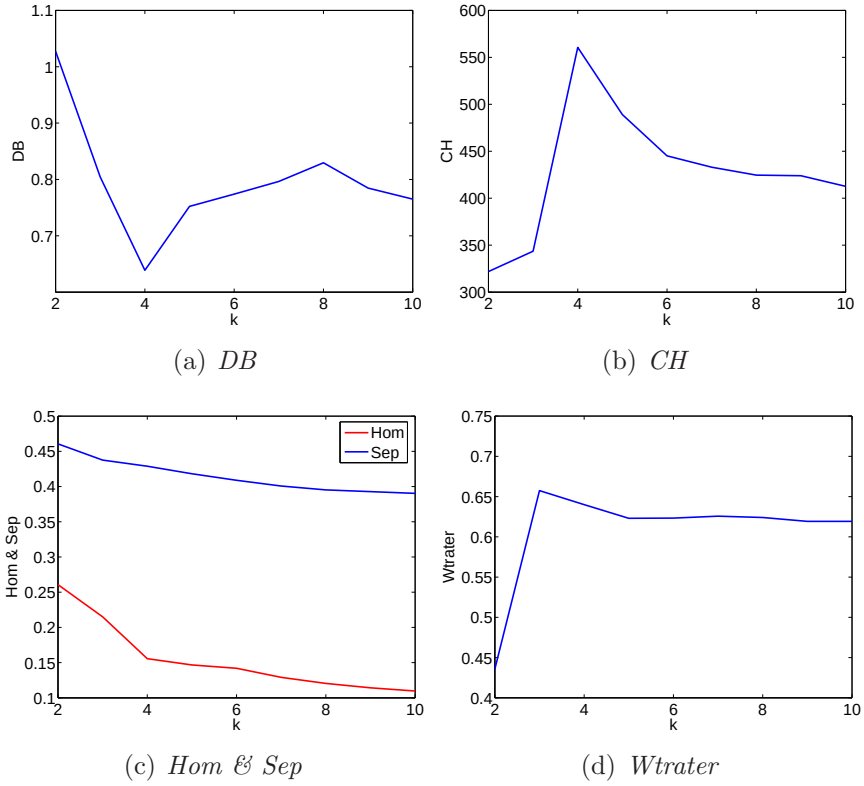


(c) EM GMM



(d) CSC

Slika 4.1: Rezultati razvrščanj vzorcev iz množice *Gauss4*. Prikazana je najboljša razvrstitev izbranih algoritmov.



(a) *DB* (b) *CH*

(c) *Hom & Sep* (d) *Wtrater*

Slika 4.2: Vrednosti notranjih kriterijev za razvrščanje vzorcev iz množice *Gauss4* v k gruč z algoritmom KMC. Za vsak k je prikazan rezultat pri najboljši razvrstitvi.

4.3.2 Sinus2

Ocene zunanjih kriterijev so podane v tabeli 4.5, kjer so najboljše vrednosti v mastnem tisku. Situacija je bila v kategoriji prvi med EM GMM in CSC zelo izenačena, je pa CSC zmagal v povprečju, ker ni naredil nobene napake v vseh desetih poizkusih. Odklon od povprečja pri metodi EM GMM je precejšen, kar nakazuje na večjo nestabilnost algoritma pri tej množici vzorcev. Najslabše se je odrezal algoritem KMC, ki mu podolgovate gruče niso povšeči. V nasprotju s pričakovanji pa je rezultat algoritma ECMC zelo obetaven, glede na to, da minimizira podobno kriterijsko funkcijo kot algoritem KMC.

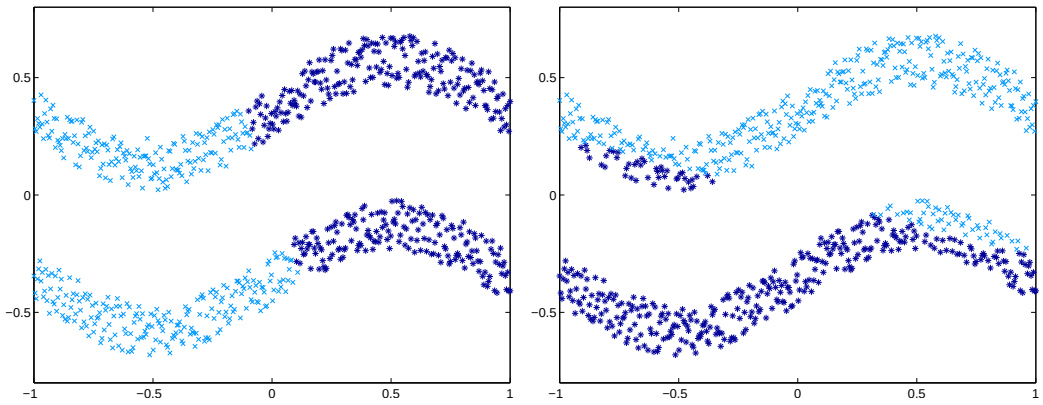
Najboljše razvrstitve so grafično prikazane na sliki 4.3, potek notranjih kriterijev za algoritem CSC pa na sliki 4.4. Tabela 4.4 prikazuje napovedi optimalnega števila gruč in kot vidimo, so napovedi v vseh primerih napačne, kar implicira na dejstvo, da je človeku na videz trivialen problem lahko stroju precejšnja uganka. V primeru indeksa *CH* je celo videti, da bi rasel še naprej, če bi bili testi narejeni za $k > 10$. Vse to kaže na to, da na podlagi notranjih indeksov ne moremo sklepati na strukturo vzorcev, oziroma, če bi jim verjeli, bi najbrž za optimum vzeli $k = 4$.

Tabela 4.4: Z notranjimi kriteriji napovedano optimalno število gruč za množico vzorcev *Sinus2*. Pravilno število gruč je 2.

	DB	*CH*	*Hom & Sep*	*Wtertra*
KMC	6	10	4	4
ECMC	4	10	4	6
EM GMM	4	10	4	5
CSC	5	10	4	4

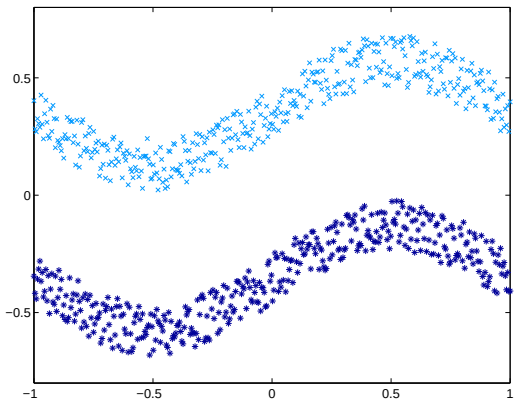
Tabela 4.5: Rezultati ocenjevanja razvrščanja z zunanjimi kriteriji – množica *Sinus2*. Prikazana je vrednost kriterijev za najboljšo razvrstitev (prvi), povprečje preko 10 poizkusov (povpr.) in standardni odklon od povprečja (odklon).

		RI	*ARI*	*JI*	*CE*	*VI*	*ADCO*
KMC	prvi	0,496	0,997	0,664	0,454	0,297	0,408
	povpr.	0,496	0,997	0,664	0,454	0,297	0,408
	odklon	0	0	0	0	0	0
ECMC	prvi	0,496	0,993	0,664	0,454	0,297	0,408
	povpr.	0,496	0,993	0,664	0,454	0,297	0,408
	odklon	0	0	0	0	0	0
EM GMM	prvi	**0**	**0**	**0**	**0**	**0**	**0**
	povpr.	0,447	0,894	0,598	0,409	0,268	0,368
	odklon	0,149	0,298	0,199	0,136	0,0892	0,123
CSC	prvi	**0**	**0**	**0**	**0**	**0**	**0**
	povpr.	**0**	**0**	**0**	**0**	**0**	**0**
	odklon	0	0	0	0	0	0

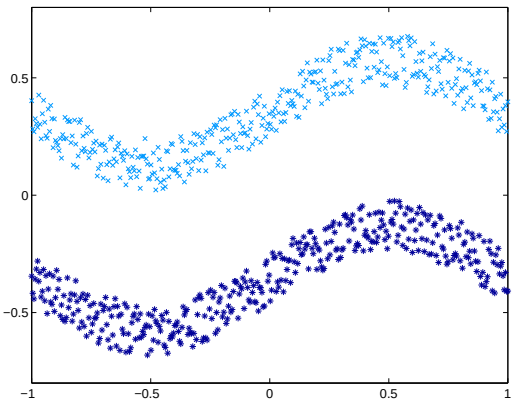


(a) KMC

(b) ECMC

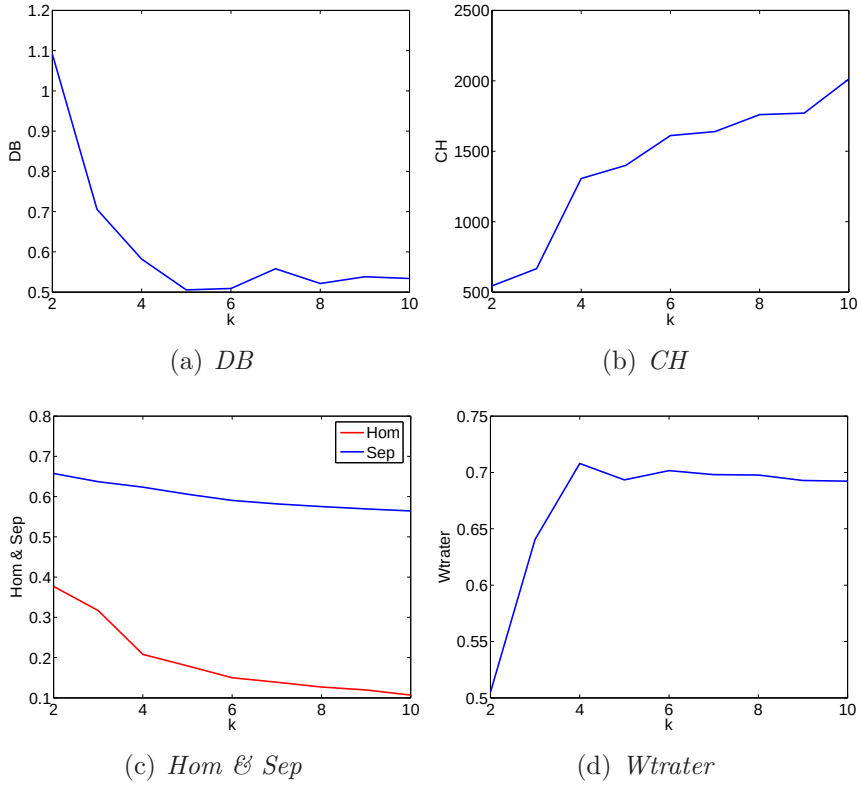


(c) EM GMM



(d) CSC

Slika 4.3: Rezultati razvrščanj vzorcev iz množice *Sinus2*. Prikazana je najboljša razvrstitev izbranih algoritmov.



(a) *DB* (b) *CH*

(c) *Hom & Sep* (d) *Wtrater*

Slika 4.4: Vrednosti notranjih kriterijev za razvrščanje vzorcev iz množice *Sinus2* v k gruč z algoritmom CSC. Za vsak k je prikazan rezultat pri najboljši razvrstitvi.

4.3.3 TriSpot

Iz tabele 4.7 je razvidna zanimiva situacija, vsaj kar se tiče vrednosti za algoritem CSC. Povprečna vrednost je namreč boljša od tiste, ki jo je algoritem razglasil za najboljšo. Tak rezultat nakazuje na splošni problem razvrščanja, kjer obstaja ogromno različnih pogledov na to, kaj je dobro – to, kar je dobro za metodo CSC, ni dobro za zunanje kriterije in obratno. V povprečju je algoritem CSC sicer zmagal, vendar ga s popolnoma pravilnim rezultatom v rubriki prvi premaga algoritem EM GMM. Pomembno pa je opaziti, da je standardni odklon od povprečne vrednosti pri algoritmu EM GMM zelo velik, kar nakazuje na večjo nestabilnost kot pri algoritmu CSC.

Približno enako slabo sta razvrščala metodi KMC in ECMC. Zanimivo pri njiju je še eno opažanje, in sicer primerjava indeksov *ADCO* in *CE* z ostalimi. Ti kažejo prednost algoritma KMC, razen indeksa *ADCO* in *CE* pokažeta obratno. Kaj je bolj relevantno, je težko reči, lahko pa upoštevamo dejstvo, da mera *ADCO* zazna strukturne razlike v razvrstitvah in mu zato lahko bolj zaupamo, vendar sem v tem delu vse zunanje kriterije upošteval kot enakovredne.

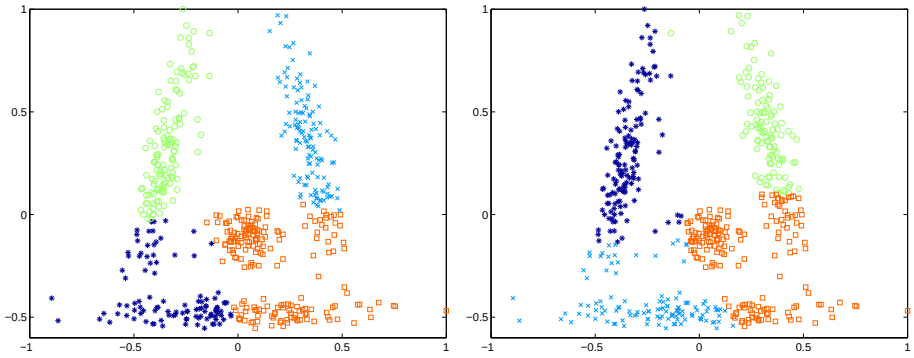
Najboljše razvrstitve so grafično prikazane na sliki 4.5, potek notranjih kriterijev za algoritem EM GMM pa na sliki 4.6. Napovedi optimalnega števila gruč, kot kaže tabela 4.6, so še najbližje pravilnemu pri indeksih *Hom & Sep* in *Wtertra*.

Tabela 4.6: Z notranjimi kriteriji napovedano optimalno število gruč za množico vzorcev *TriSpot*. Pravilno število gruč je 4.

	DB	*CH*	*Hom & Sep*	*Wtertra*
KMC	7	10	4	4
ECMC	10	9	3	3
EM GMM	9	10	3	9
CSC	6	8	4	6

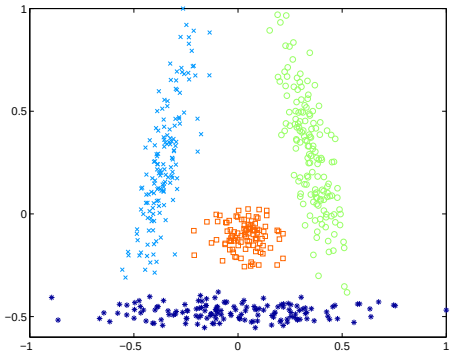
Tabela 4.7: Rezultati ocenjevanja razvrščanja z zunanjimi kriteriji – množica *TriSpot*. Prikazana je vrednost kriterijev za najboljšo razvrstitev (prvi), povprečje preko 10 poizkusov (povpr.) in standardni odklon od povprečja (odklon).

		RI	*ARI*	*JI*	*CE*	*VI*	*ADCO*
KMC	prvi	0,173	0,451	0,500	0,231	0,216	0,311
	povpr.	0,172	0,449	0,499	0,227	0,216	0,301
	odklon	0,00107	0,00250	0,00196	0,00445	0,000804	0,0127
ECMC	prvi	0,175	0,457	0,507	0,220	0,237	0,224
	povpr.	0,175	0,457	0,507	0,220	0,237	0,224
	odklon	0	0	0	0	0	0
EM GMM	prvi	**0**	**0**	**0**	**0**	**0**	**0**
	povpr.	0,0435	0,104	0,119	0,0765	0,0446	0,0518
	odklon	0,0667	0,160	0,182	0,117	0,0688	0,0796
CSC	prvi	0,00590	0,0155	0,0229	0,00545	0,0138	0,00447
	povpr.	**0,00447**	**0,0118**	**0,0174**	**0,00418**	**0,0103**	**0,000804**
	odklon	0,000806	0,00212	0,00310	0,000833	0,00224	0,00128

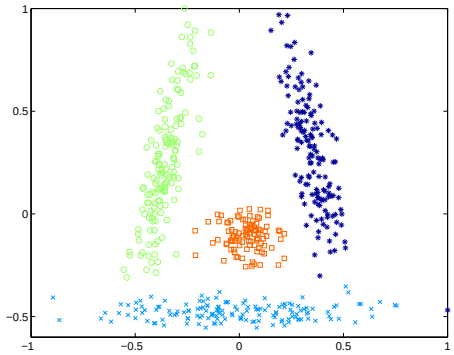


(a) KMC

(b) ECMC

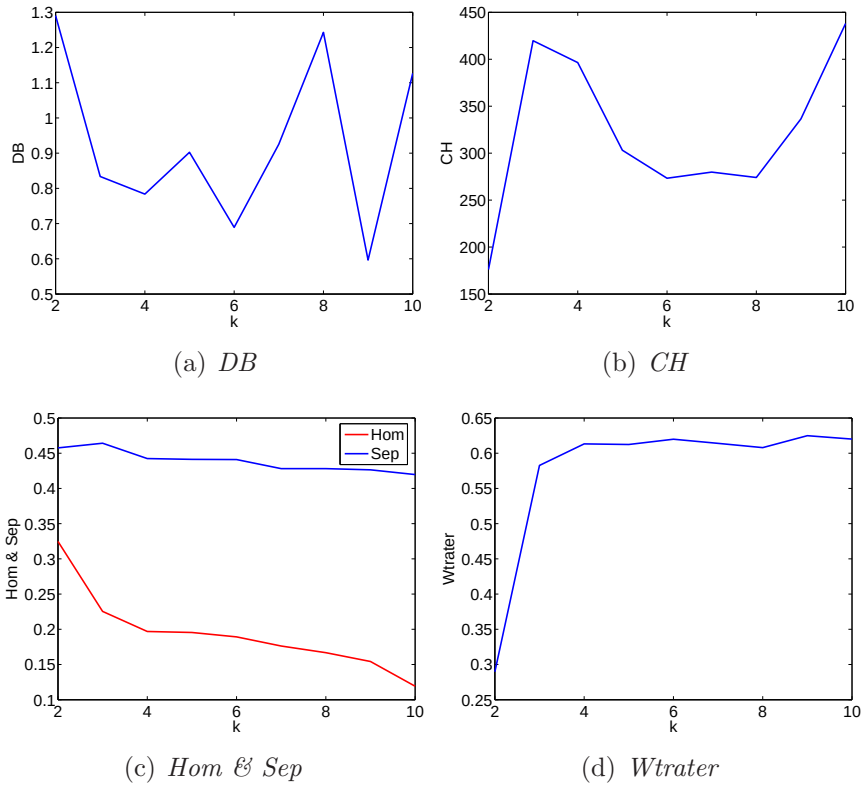


(c) EM GMM



(d) CSC

Slika 4.5: Rezultati razvrščanj vzorcev iz množice *TriSpot*. Prikazana je najboljša razvrstitev izbranih algoritmov.



(a) *DB* (b) *CH*

(c) *Hom & Sep* (d) *Wtrater*

Slika 4.6: Vrednosti notranjih kriterijev za razvrščanje vzorcev iz množice *TriSpot* v k gruč z algoritmom EM GMM. Za vsak k je prikazan rezultat pri najboljši razvrstitvi.

4.3.4 Luni

Zmagovalec v obeh kategorijah je znan; iz tabele 4.9 namreč vidimo, da je to metoda CSC. Samo temu algoritmu je uspelo opraviti razvrstitev brez napak in to v vseh desetih poizkusih. Ostali trije so bili tokrat občutno slabši in kar izenačeni med seboj. Množica vzorcev je bila tokrat res težja, saj gruče med seboj niso enostavno ločljive in je pričakovano, da algoritmi, ki izkoriščajo statistike drugega reda tej situaciji niso kos.

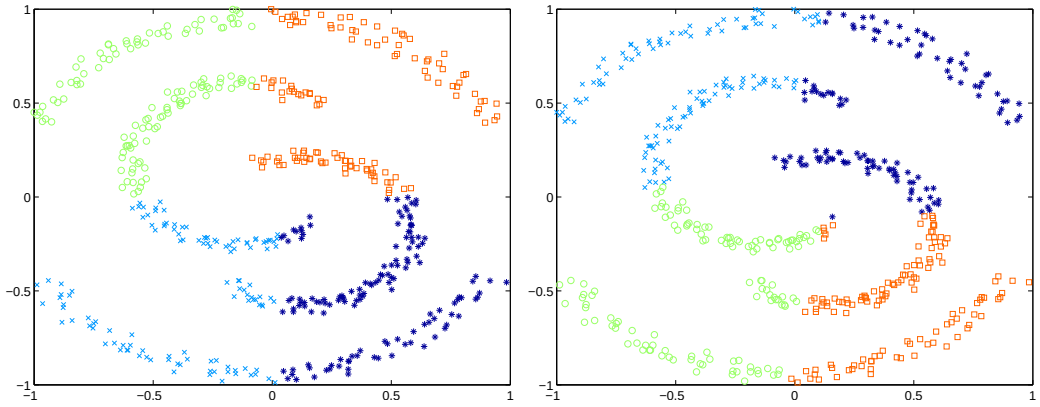
Najboljše razvrstitve so grafično prikazane na sliki 4.7, potek notranjih kriterijev za algoritem CSC pa na sliki 4.8. Notranja indeksa *DB* in *CH* sta zopet precej zgrešila pravo napoved števila gruč, kot kaže tabela 4.6. Kriterij *Hom & Sep* se je tu precej izkazal, vendar je razumevanje njegovih vrednosti lahko včasih dvoumno in napačno interpretirano. Nedvoumnost je prednost ostalih notranjih kriterijev.

Tabela 4.8: Z notranjimi kriteriji napovedano optimalno število gruč za množico vzorcev *Luni*. Pravilno število gruč je 4.

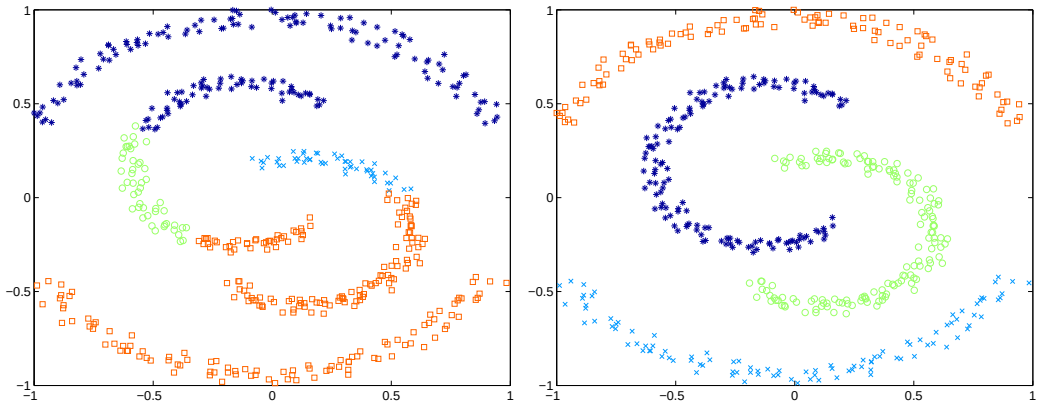
	DB	*CH*	*Hom & Sep*	*Wtertra*
KMC	10	10	4	3
ECMC	10	10	4	4
EM GMM	8	2	6	4
CSC	10	10	4	6

Tabela 4.9: Rezultati ocenjevanja razvrščanja z zunanjimi kriteriji – množica *Luni*. Prikazana je vrednost kriterijev za najboljšo razvrstitev (prvi), povprečje preko 10 poizkusov (povpr.) in standardni odklon od povprečja (odklon).

		RI	*ARI*	*JI*	*CE*	*VI*	*ADCO*
KMC	prvi	0,293	0,775	0,734	0,502	0,421	0,361
	povpr.	0,293	0,775	0,734	0,502	0,421	0,361
	odklon	0,00107	0,00250	0,00196	0,00445	0,000804	0,0127
ECMC	prvi	0,286	0,760	0,724	0,537	0,407	0,342
	povpr.	0,286	0,760	0,724	0,537	0,407	0,342
	odklon	0	0	0	0	0	0
EM GMM	prvi	0,286	0,667	0,639	0,391	0,279	0,427
	povpr.	0,279	0,703	0,678	0,429	0,344	0,293
	odklon	0,0119	0,0260	0,0218	0,0285	0,0359	0,0706
CSC	prvi	**0**	**0**	**0**	**0**	**0**	**0**
	povpr.	**0**	**0**	**0**	**0**	**0**	**0**
	odklon	0	0	0	0	0	0

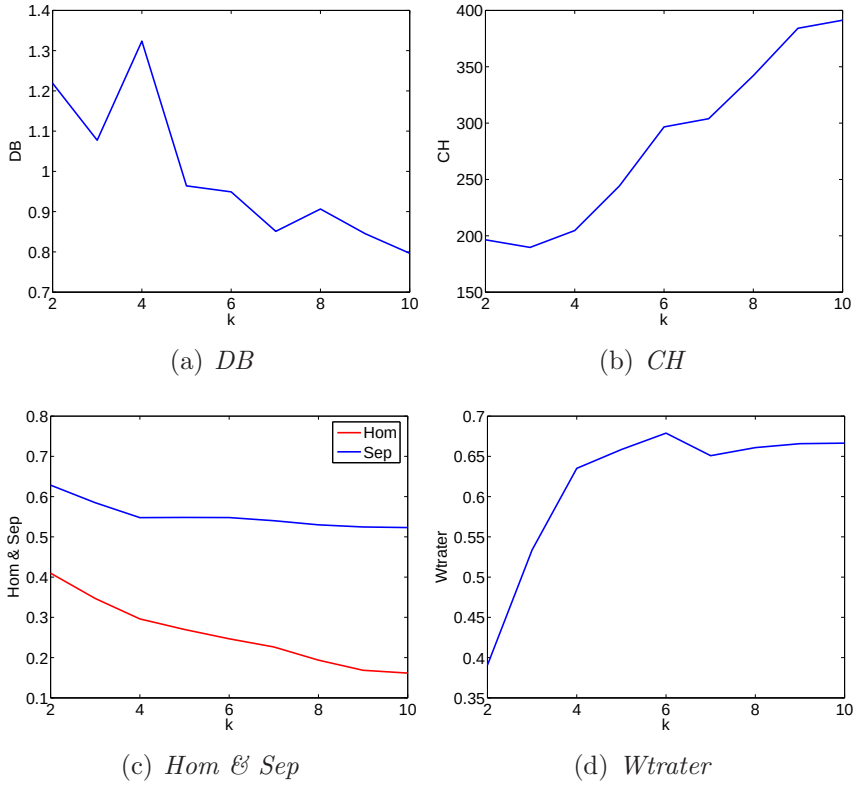


(a) KMC (b) ECMC



(c) EM GMM (d) CSC

Slika 4.7: Rezultati razvrščanj vzorcev iz množice *Luni*. Prikazana je najboljša razvrstitev izbranih algoritmov.



(a) *DB* (b) *CH*

(c) *Hom & Sep* (d) *Wtrater*

Slika 4.8: Vrednosti notranjih kriterijev za razvrščanje vzorcev iz množice *Luni* v k gruč z algoritmom CSC. Za vsak k je prikazan rezultat pri najboljši razvrstitvi.

4.3.5 Iris

Tabela 4.11 z rezultati zunanjih kriterijev razglaša algoritem EM GMM za najboljšega v kategoriji prvi in, če upoštevamo večinsko mnenje kriterijev, tudi v povprečju. Kriterija *CE* in *ADCO* sta sicer pokazala na prednost algoritma CSC, ki je v povprečju z njunega stališča boljši, kot končni rezultat pa se upošteva mnenje večine.

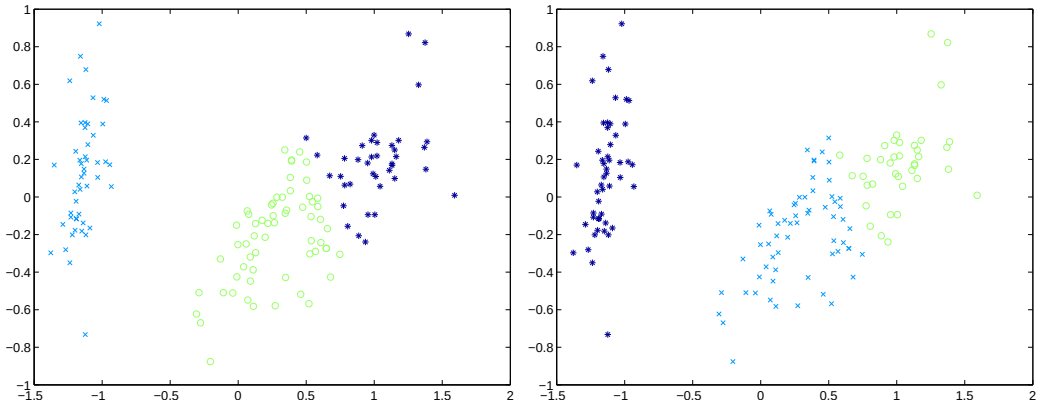
Najboljše razvrstitve algoritmov so grafično prikazane na sliki 4.9. Za prikaz vzorcev v dvorazsežnem prostoru je bila uporabljena metoda PCA. Potek notranjih kriterijev za zmagovalni algoritem EM GMM je na sliki 4.10. Za najbolj uporabna kazalca notranje uspešnosti razvrščanja sta se izkazala *Hom & Sep* in *Wtrater*. V grobem pa so vsi notranji kriteriji izbrali število gruč v bližnji okolici optimuma.

Tabela 4.10: Z notranjimi kriteriji napovedano optimalno število gruč za množico vzorcev *Iris*. Pravilno število gruč je 3.

	DB	*CH*	*Hom & Sep*	*Wtertra*
KMC	2	3	3	3
ECMC	2	3	3	3
EM GMM	2	2	3	3
CSC	2	2	3	3

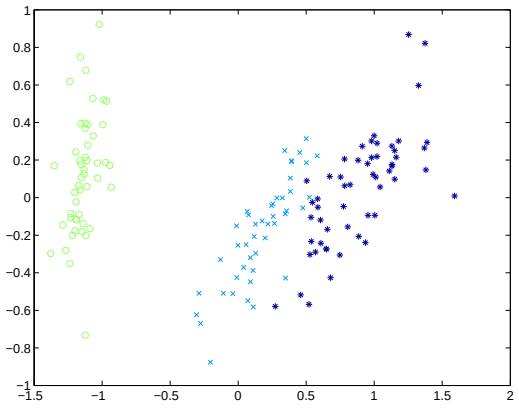
Tabela 4.11: Rezultati ocenjevanja razvrščanja z zunanjimi kriteriji – množica *Iris*. Prikazana je vrednost kriterijev za najboljšo razvrstitev (prvi), povprečje preko 10 poizkusov (povpr.) in standardni odklon od povprečja (odklon).

		RI	*ARI*	*JI*	*CE*	*VI*	*ADCO*
KMC	prvi	0,126	0,284	0,318	0,113	0,162	0,0320
	povpr.	0,157	0,341	0,358	0,176	0,177	0,108
	odklon	0,0616	0,115	0,0801	0,125	0,0300	0,151
ECMC	prvi	0,120	0,270	0,304	0,107	0,152	0,0272
	povpr.	0,120	0,270	0,304	0,107	0,152	0,0272
	odklon	0	0	0	0	0	0
EM GMM	prvi	**0,0425**	**0,0961**	**0,121**	**0,0333**	**0,0634**	**0,00698**
	povpr.	**0,0851**	**0,178**	**0,189**	0,105	**0,0859**	0,0560
	odklon	0,0853	0,163	0,138	0,144	0,0454	0,101
CSC	prvi	0,118	0,263	0,297	0,100	0,141	0,0367
	povpr.	0,0939	0,211	0,242	**0,0807**	0,141	**0,0333**
	odklon	0,0421	0,0929	0,0893	0,0473	0,0431	0,0321

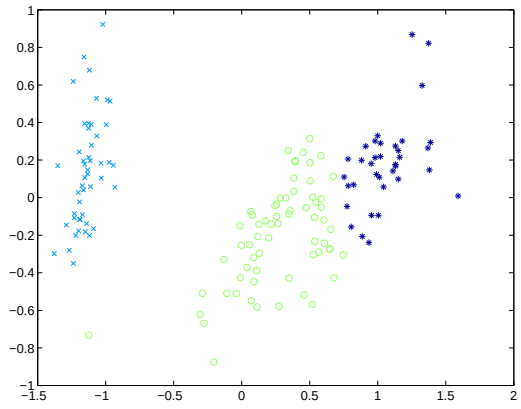


(a) KMC

(b) ECMC

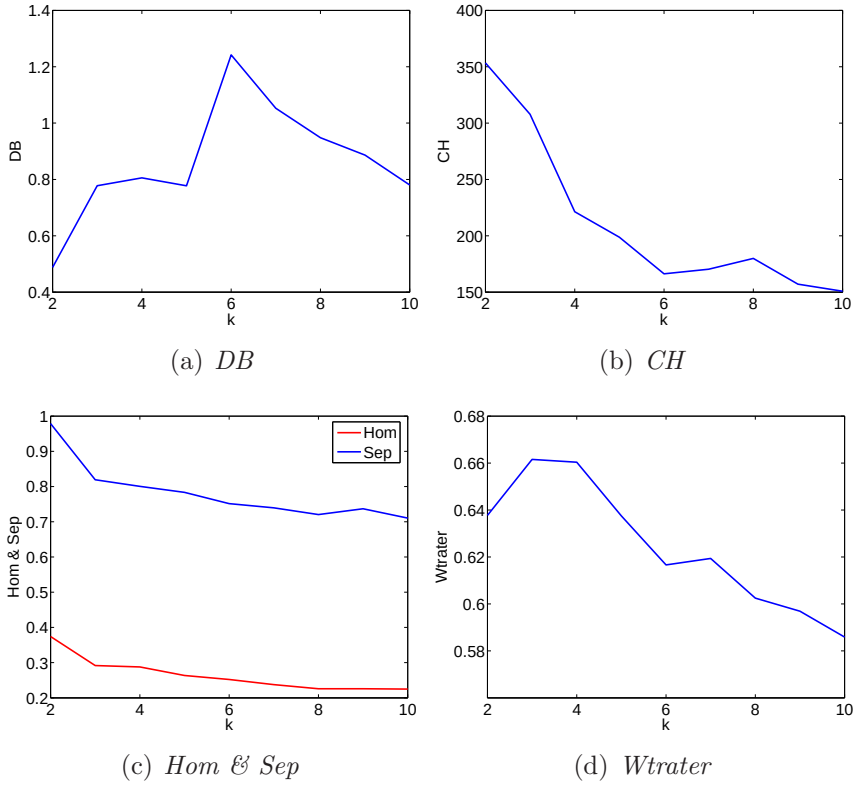


(c) EM GMM



(d) CSC

Slika 4.9: Rezultati razvrščanj vzorcev iz množice *Iris*, prikazani z metodo PCA. Prikazana je najboljša razvrstitev izbranih algoritmov.



(a) *DB* (b) *CH*

(c) *Hom & Sep* (d) *Wtrater*

Slika 4.10: Vrednosti notranjih kriterijev za razvrščanje vzorcev iz množice *Iris* v k gruč z algoritmom CSC. Za vsak k je prikazan rezultat pri najboljši razvrstitvi.

4.3.6 Wine

V tokratni preizkušnji se je zgodil zanimiv razplet, saj je algoritem CSC v povprečju nadvladal ostale, a se ni znal pravilno (s stališča ocenjevalnih kriterijev) odločiti, katera njegova razvrstitev je najboljša. Kot torej vidimo v tabeli 4.13, je algoritem EM GMM razglašen za najboljšega v kategoriji prvi. Pomembno je poudariti, da je razlika v rezultatih vseh štirih algoritmov precej majhna, zanimivo pa je, da metoda KMC v kategoriji povprečje premaga metodo EM GMM.

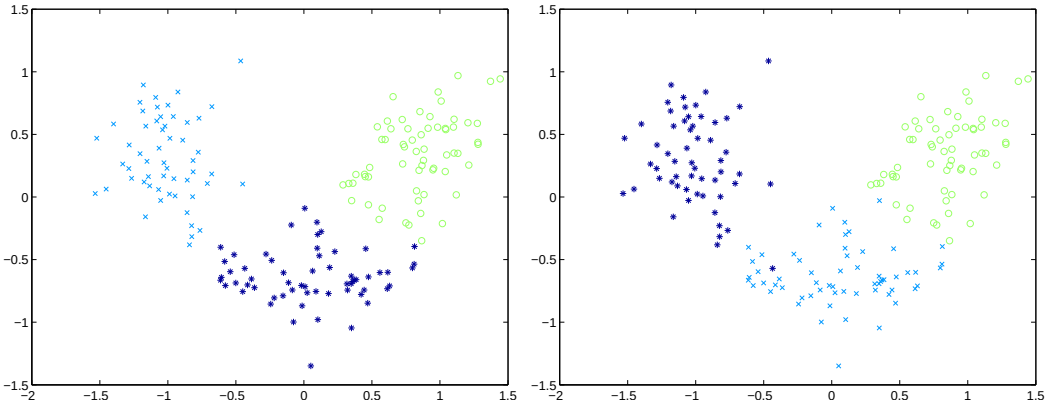
Najboljše razvrstitve algoritmov so z uporabo metode PCA grafično prikazane na sliki 4.11. Potek notranjih kriterijev za zmagovalni algoritem EM GMM je na sliki 4.12. Tudi tokrat je najbolj pravilno napovedoval kriterij *Hom & Sep*, zelo se je približal tudi indeks *DB*.

Tabela 4.12: Z notranjimi kriteriji napovedano optimalno število gruč za množico vzorcev *Wine*. Pravilno število gruč je 3.

	DB	*CH*	*Hom & Sep*	*Wtertra*
KMC	3	2	3	4
ECMC	3	2	3	3
EM GMM	3	3	3	5
CSC	2	2	3	3

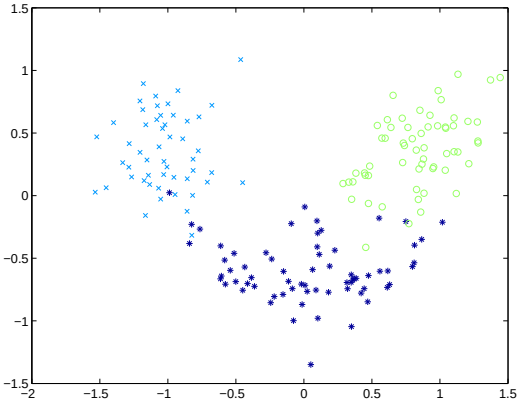
Tabela 4.13: Rezultati ocenjevanja razvrščanja z zunanjimi kriteriji – množica *Wine*. Prikazana je vrednost kriterijev za najboljšo razvrstitev (prvi), povprečje preko 10 poizkusov (povpr.) in standardni odklon od povprečja (odklon).

		RI	*ARI*	*JI*	*CE*	*VI*	*ADCO*
KMC	prvi	0,0800	0,180	0,214	0,0618	0,115	0,0462
	povpr.	0,0800	0,180	0,214	0,0618	0,115	0,0462
	odklon	0	0	0	0	0	0
ECMC	prvi	0,0935	0,210	0,246	0,0730	0,137	0,0466
	povpr.	0,0935	0,210	0,246	0,0730	0,137	0,0466
	odklon	0	0	0	0	0	0
EM GMM	prvi	**0,0300**	**0,0675**	**0,0858**	**0,0225**	**0,0534**	**0,0112**
	povpr.	0,0874	0,188	0,202	0,0860	0,106	0,0478
	odklon	0,0834	0,165	0,137	0,119	0,0629	0,0522
CSC	prvi	0,0641	0,144	0,175	0,0506	0,0910	0,0376
	povpr.	**0,0571**	**0,128**	**0,156**	**0,0438**	**0,0878**	**0,0313**
	odklon	0,0150	0,0338	0,0390	0,0123	0,0206	0,00938

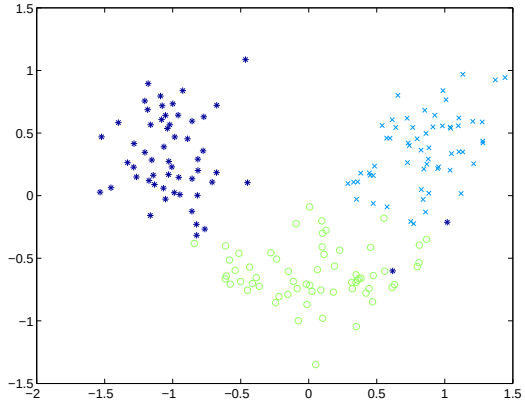


(a) KMC

(b) ECMC

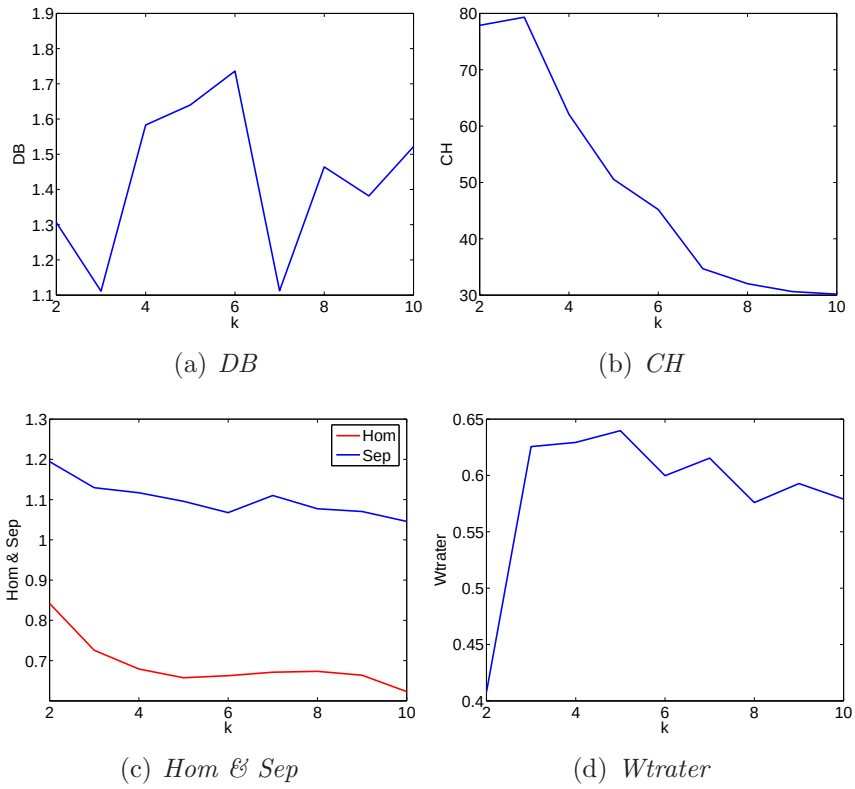


(c) EM GMM



(d) CSC

Slika 4.11: Rezultati razvrščanj vzorcev iz množice *Wine*, prikazani z metodo PCA. Prikazana je najboljša razvrstitev izbranih algoritmov.



(a) *DB* (b) *CH*

(c) *Hom & Sep* (d) *Wtrater*

Slika 4.12: Vrednosti notranjih kriterijev za razvrščanje vzorcev iz množice *Wine* v k gruč z algoritmom EM. Za vsak k je prikazan rezultat pri najboljši razvrstitvi.

4.3.7 Lung cancer

Množica vzorcev *Lung cancer* je za razvrščanje izjemno zahtevna, saj vsebuje malo vzorcev, ki pa imajo veliko razsežnost. Zato so temu primerni tudi rezultati v tabeli 4.15, kjer je povprečna napaka razvrščanja (*CE*) za najboljši algoritem v kategoriji prvi, torej CSC, kar 37,04 %. Zelo tesno za njim je metoda KMC, ki v povprečju celo zmaga.

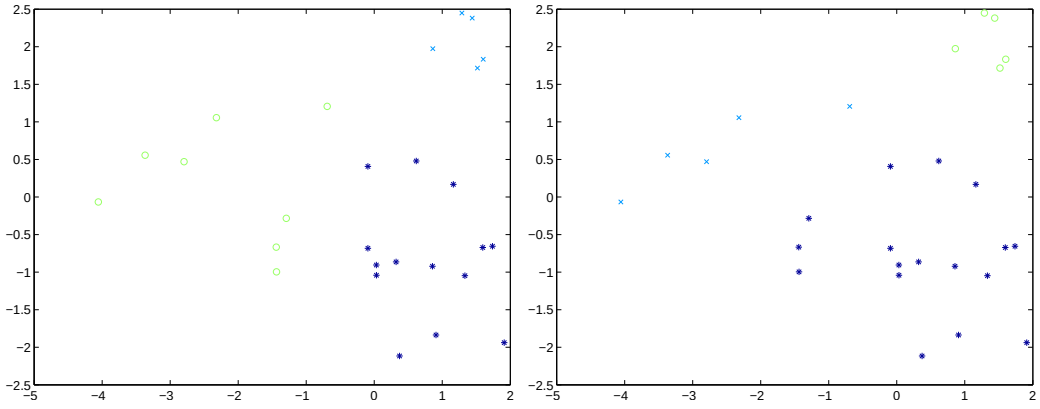
Najboljše razvrstitve algoritmov so z uporabo metode PCA grafično prikazane na sliki 4.13. Potek notranjih kriterijev za zmagovalni algoritem EM GMM je na sliki 4.14. Največ pravilnih napovedi ima kriterij *Wtertra*, ki je pri vseh algoritmih napovedal pravilno število gruč, to je 3. Polovično uspešnost si lasti kriterij *Hom & Sep*, ostala dva pa sta se od pravilnega rezultata precej oddaljila.

Tabela 4.14: Z notranjimi kriteriji napovedano optimalno število gruč za množico vzorcev *Lung cancer*. Pravilno število gruč je 3.

	DB	*CH*	*Hom & Sep*	*Wtertra*
KMC	10	2	3	3
ECMC	9	2	4	3
EM GMM	9	2	4	3
CSC	8	8	3	3

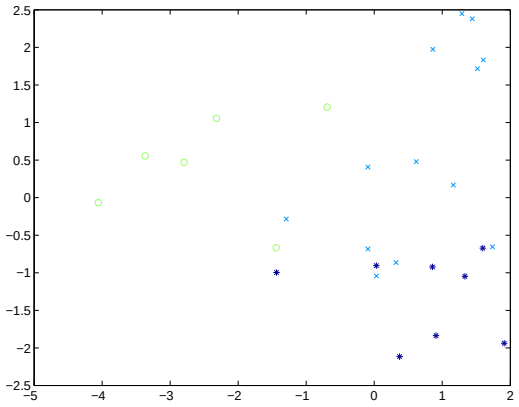
Tabela 4.15: Rezultati ocenjevanja razvrščanja z zunanjimi kriteriji – množica *Lung cancer*. Prikazana je vrednost kriterijev za najboljšo razvrstitev (prvi), povprečje preko 10 poizkusov (povpr.) in standardni odklon od povprečja (odklon).

		RI	*ARI*	*JI*	*CE*	*VI*	*ADCO*
KMC	prvi	0,382	0,849	0,720	0,370	**0,661**	**0,161**
	povpr.	**0,359**	**0,798**	**0,691**	**0,374**	**0,625**	**0,14945**
	odklon	0,0549	0,0996	0,0539	0,0802	0,0864	0,0830
ECMC	prvi	0,436	0,910	0,732	0,40741	0,666	0,265
	povpr.	0,436	0,910	0,732	0,407	0,666	0,265
	odklon	0	0	0	0	0	0
EM GMM	prvi	0,410	0,930	0,770	0,481	0,774	0,144
	povpr.	0,413	0,912	0,753	0,474	0,713	0,167
	odklon	0,0316	0,0358	0,0221	0,0323	0,0500	0,0890
CSC	prvi	**0,376**	**0,836**	**0,714**	**0,370**	0,668	0,163
	povpr.	0,417	0,923	0,757	0,452	0,745	0,170
	odklon	0,0200	0,0474	0,0278	0,0463	0,0746	0,0301

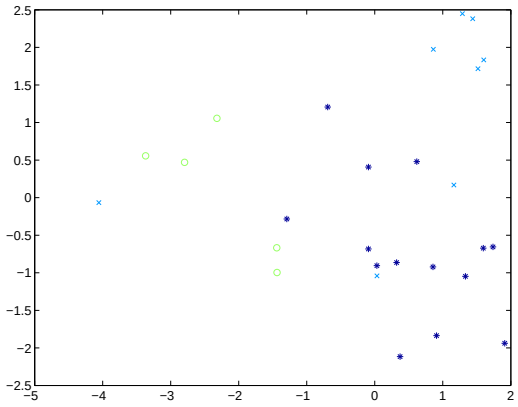


(a) KMC

(b) ECMC

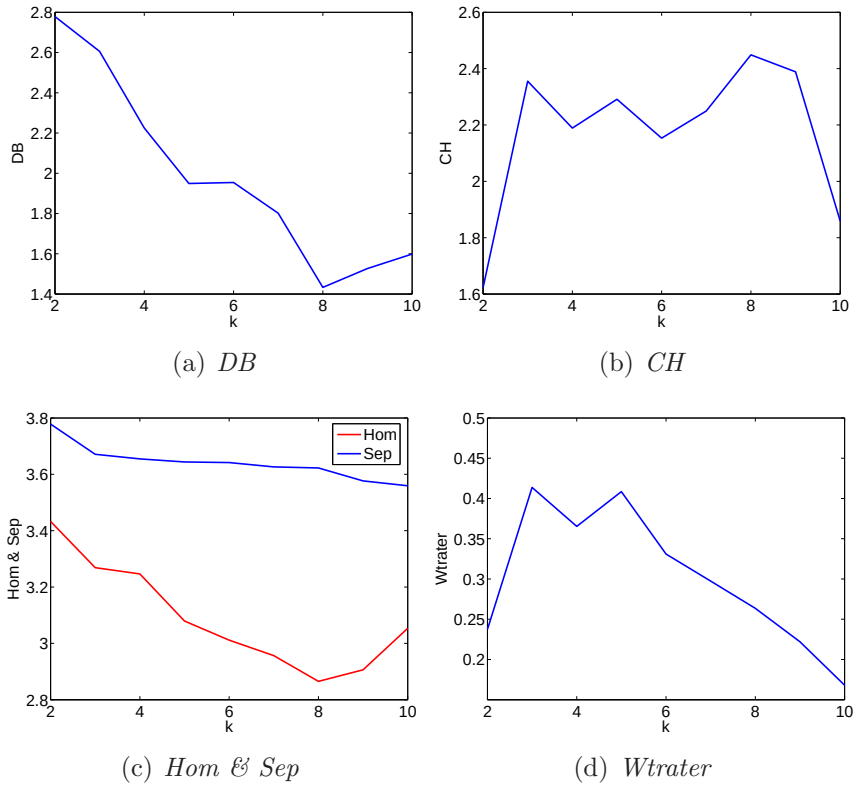


(c) EM GMM



(d) CSC

Slika 4.13: Rezultati razvrščanj vzorcev iz množice *Lung cancer*, prikazani z metodo PCA. Prikazana je najboljša razvrstitev izbranih algoritmov.



(a) *DB* (b) *CH*

(c) *Hom & Sep* (d) *Wtrater*

Slika 4.14: Vrednosti notranjih kriterijev za razvrščanje vzorcev iz množice *Lung cancer* v k gruč z algoritmom CSC. Za vsak k je prikazan rezultat pri najboljši razvrstitvi.

4.3.8 Skupni rezultati

Kot zaključek poglavja z rezultati si poglejmo končno točkovanje algoritmov v kategorijah prvi in povprečje ter kakšni so bili povprečni izvajalni časi algoritmov. Tabela 4.16 nam prikazuje rezultate točkovanja za najboljše razvrstitve (kategorija prvi), tabela 4.17 rezultate točkovanja v povprečju (kategorija povprečje) in tabela 4.18 povprečne izvajalne čase algoritmov glede na množice vzorcev. Sistem točkovanja je podrobno opisan v razdelku 4.1 in na začetku tega razdelka.

Tabela 4.16: Končni rezultat točkovanja v kategoriji prvi.

	Gauss4	*Sinus2*	*TriSpot*	*Luni*	*Iris*	*Wine*	*L. cancer*	**skupaj**
KMC	3	0	1	0	0	1	2	**7**
ECMC	2	1	0	1	1	0	1	**6**
EM GMM	0	3	3	2	3	3	0	**14**
CSC	2	3	2	3	2	2	3	**17**

Tabela 4.17: Končni rezultat točkovanja v kategoriji povprečje.

	Gauss4	*Sinus2*	*TriSpot*	*Luni*	*Iris*	*Wine*	*L. cancer*	**skupaj**
KMC	3	0	1	0	0	2	3	**9**
ECMC	2	2	0	1	1	0	2	**8**
EM GMM	0	1	2	2	3	1	1	**10**
CSC	1	3	3	3	2	3	0	**15**

Opazimo, da je končni vrstni red po seštevku točk v obeh kategorijah enak, in sicer si algoritmi sledijo po padajoči uspešnosti tako: CSC, EM GMM,

Tabela 4.18: Povprečni izvajalni časi v sekundah.

	Gauss4	*Sinus2*	*TriSpot*	*Luni*	*Iris*	*Wine*	*L. cancer*
KMC	0,016	0,022	0,018	0,020	0,007	0,012	0,004
ECMC	0,772	1,905	0,954	1,279	0,636	1,852	2,904
EM GMM	0,053	0,196	0,136	0,153	0,075	0,176	0,050
CSC	5,105	55,973	16,280	13,382	0,316	0,785	0,069

KMC, ECMC. Za najboljši algoritem izmed primerjanih torej lahko proglasimo algoritem CSC, gotovo pa ni zmagovalec v hitrosti, saj so izvajalni časi pri velikem številu vzorcev tudi za faktor 100 daljši od ostalih. Slabše se je v tem smislu izkazal tudi algoritem ECMC, ki je povrhu vsega nabral tudi najmanjše število točk. Hitrost izvajanja moramo seveda jemati s precejšnjo mero zadržanosti, ker so bila merjenja opravljena na osebnem računalniku z večopravilnim operacijskim sistemom, kjer se istočasno izvaja več procesov, ki si delijo procesorski čas. Poleg tega niso vse implementacije algoritmov enako optimalne. Vseeno pa je glavna razlika v času izvajanja med algoritmom CSC in ostalimi algoritmi jasno izražena.

Tabela 4.19: Priporočena algoritma za posamezno lastnost problemske domene. Prva možnost je izbrana glede na kategorijo prvi, druga pa glede na kategorijo povprečje.

značilnost problemske domene	algoritem
nepravilne, podolgovate oblike gruč	CSC, EM GMM
linearna neločljivost gruč	CSC, EM GMM
velika razsežnost vzorcev	EM GMM, CSC
razsežnost veliko večja od števila vzorcev	CSC, KMC
zahteva po hitrem razvrščanju	KMC, EM GMM

Vzporedna primerjava uspešnosti notranjih kriterijev pri napovedovanju optimalnega števila gruč kaže, da sta bila najbolj zanesljiva kriterija *Hom & Sep* in *Wtertra*, predvsem kadar je šlo za težje primere ločljivosti in razsežnosti. Poudariti pa velja, da je določevanje števila gruč s pomočjo kriterija *Hom & Sep* lahko precej naporno in nenatančno opravilo. Ostali trije kriteriji imajo namreč jasno izraženo pravilo za določevanje števila gruč (minimum ali maksimum vrednosti).

Kot sem že večkrat omenil, je na področju razvrščanja zelo težko potegniti črto in izpostaviti nek algoritem ali mero podobnosti kot najboljšo, ker je, podobno kot pri ljudeh, vsak lahko najboljši na svojem področju. Tako bi bila lahko končna razvrstitev algoritmov v mojem primeru drugačna, če bi izbral drugačne kriterije ocenjevanja ali mogoče druge množice vzorcev. Vseeno pa iz končnega rezultata lahko napravimo nekaj zaključkov; zbrani so v tabeli 4.19 – ta na podlagi točkovanja predlaga izbiro ustreznega algoritma. Podani sta dve možnosti glede na kategoriji prvi in povprečje.

Poglavje 5

Sklepne ugotovitve

Glavni problem razvrščanja vzorcev v gruče je izjemna širina tega pojma, ki vključuje malodane vsa področja človekovega udejstvovanja. Zato je nemogoče enkrat za vselej postaviti točna merila in razviti enotne postopke za reševanje razvrščanja. Iz tega izhaja tudi zahtevnost poštene primerjave razvitih algoritmov, predlaganih mer podobnosti in ocenjevalnih kriterijev, kar je bil namen mojega dela. Po najboljših močeh sem za primerjalne množice vzorcev izbral raznolike in vsestranske primere, prav tako tudi ocenjevalne kriterije.

Rezultati so pokazali, da je v skupnem seštevku najboljši algoritem CSC, po vrsti pa mu sledijo metode EM GMM, KMC in ECMC. Tak rezultat je posledica notranjih kriterijev, ki jih algoritmi uporabljajo. Algoritmi KMC, EM GMM in ECMC namreč uporabljajo mere podobnosti na osnovi statistik drugega reda – gre torej za minimizacijo variance znotraj gruč. Posledica tega je, da znajo med vzorci najti gruče, ki so hiper-eliptičnih (kroglastih) oblik, zaplete pa se, če struktura podatkov ni taka. To omejitev algoritem CSC presega. Slabost algoritma KMC je nezmožnost prilagajanja na podolgovate in zapletenejše oblike gruč (primer *Sinus2, Luni*), prednost pa izjemna hitrost in enostavnost uporabe. Izkazal se tudi pri razvrščanju vzorcev velikih razsežnosti. Njegova izpeljanka, metoda ECMC, je bolj zanimiva kot izvedba sprotnega razvrščanja (ECM), kot njena nesprotna različica, ki boleha za istimi slabostmi kot metoda KMC. Algoritem EM GMM je izboljšava algoritma KMC v smislu prilagajanja obliki gruč in zato doseže boljši rezultat. Dober je tudi v primerih vzorcev z veliko razsežnostmi, še vedno pa so njegova omejitev gruče zapletenejših oblik, ki med seboj niso enostavno ločljive. Algoritem CSC je naprednejši v smislu prilagajanja zapletenejšim strukturam, njegova največja slabost pa je počasnost, zlasti za število vzorcev večje od 400. Poleg tega zahteva veliko pozornosti pri nastavljanju ustreznih vhodnih parametrov.

V namen ocenjevanja kvalitete razvrščanja so bili v delo vključeni številni zunanji in notranji kriteriji, ki so, prav tako kot algoritmi, omejeni s svojo notranjo cenilno funkcijo ali mero. Kot zaključek naj velja načelo, da se ob odločanju o najboljši razvrstitvi ali o optimalnem številu gruč, vedno ravnamo po večjem številu kazalcev in ne zgolj po enem.

Primerjanje metod za razvrščanje vzorcev odpira veliko novih vprašanj in problemov, ki ostajajo tema nadaljnjega raziskovanja. Potrebno bi bilo raziskati možnosti pohitritve izvajanja algoritma CSC in iskati ustrezna matematična orodja za optimalno in učinkovito nastavljanje njegovih vhodnih parametrov. Zanimiva bi bila tudi študija in implementacija popolnoma avtonomnega sistema za razvrščanje vzorcev z več algoritmi hkrati, kjer bi se končni rezultat določal po načelu večine – glasovanje med algoritmi za razvrščanje.

Literatura

[1] E. Bae, J. Bailey, D. Guozhu, "Clustering Similarity Comparison Using Density Profiles", v zborniku *Proceedings of 19th Australian joint conference on artificial intelligence*, Hobart, Avstralija, dec. 2006, str. 342–351.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006, pogl. 2 in 9.

[3] G. Chen, N. Banerjee, S. A. Jaradat, T. S. Tanaka, M. S. H. Ko, M. Q. Zhang, "Evaluation and Comparison of Clustering Algorithms in Analyzing ES Cell Gene Expression Data," *Statistica Sinica*, št. 12, str. 241–262, 2002.

[4] M. H. Dunham, *Data Mining: Introductory and Advanced Topics*, Prentice Hall, 2003, pogl. 5.

[5] A. Ferligoj, *Razvrščanje v skupine – teorija in uporaba v družboslovju*, Metodološki zvezki 4, Ljubljana: FSPN, 1989, pogl. 2.

[6] E. Gokcay, J. C. Principe, "Information Theoretic Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, št. 24, zv. 2, str. 158–171, 2002.

[7] L. Hubert, P. Arabie, "Comparing partitions", *Journal of Classification*, št. 2, str. 193–218, 1985.

[8] N. Kasabov, Q. Song, "DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and its Application," *IEEE Transactions on Fuzzy Systems*, št. 2, zv. 10, str. 144–154, 2002.

[9] R. Jenssen, K. E. Hild, D. Erdogmus, J. C. Principe, T. Eltoft, "Clustering using Renyi's Entropy," v zborniku *Int'l. Joint Conference on Neural Networks*, Portland, USA, julij 2003, str. 523–528.

[10] R. Jenssen, J. C. Principe, T. Eltoft, "Cauchy-Schwarz pdf Divergence Measure for non-Parametric Clustering," v zborniku *IEEE Norway Section Int'l. Symposium on Signal Processing*, Bergen, Norway, okt. 2003.

[11] R. Jenssen, J. C. Principe, D. Erdogmus, T. Eltoft, "The Cauchy-Schwarz Divergence and Parzen Windowing: Connections to Graph Theory and Mercer Kernels," *Journal of the Franklin Institute*, št. 343, zv. 6, str. 614–629, 2006.

[12] U. Maulik, S. Bandyopadhyay, "Performance evaluation of some clustering algorithms and validity indices," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, št. 12, zv. 24, str. 1650–1654, 2002.

[13] M. Meila, "Comparing clusterings – an information based distance," *Journal of Multivariate Analysi*, št. 98, zv. 5, str. 873–895, 2007.

[14] *Optimization Toolbox User's Guide*, The Math Works, Inc., št. 3, Natick, MA, str. 19–24, 1996.

[15] S. Petrović, "A Comparison Between the Silhouette Index and the Davies-Bouldin Index in Labelling IDS Clusters," v zborniku *Proceedings of the 11th Nordic Workshop on Secure IT-systems*, Linkoping, Sweden, 2006, str. 53–64.

[16] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, št. 66, str. 846–850, 1971.

[17] R. Sharan, R. Elkon, R. Shamir, "Cluster analysis and its applications to gene expression data," na *Ernst Schering workshop on Bioinformatics and Genome Analysis*, 2001.

[18] Q. Song, N. Kasabov, "ECM – A Novel On-line, Evolving Clustering Method and Its Applications," v zborniku *Proceedings of the Fifth Biannual Conference on Artificial Neural Networks and Expert Systems*, Dunedin, New Zealand, nov. 2001, str. 87–92.

[19] A. Strehl, J. Ghosh, "Value-based customer grouping from large retail data-sets," v zborniku *Proceedings of the SPIE Conference on Data Mining and Knowledge Discovery*, Orlando, ZDA, 2000, str. 33–42.

[20] (2008) Wikipedia, *Jaccard index*. Dostopno na: http://en.wikipedia.org/wiki/Jaccard_index

[21] (2008) Wikipedia, *Principal Components Analysis*. Dostopno na: http://en.wikipedia.org/wiki/Principal_components_analysis

[22] K. Yeung, W. Ruzzo, "Details of the adjusted rand index and clustering algorithms. supplement to the paper "an experimental 801 study on principal component analysis for clustering gene expression data"," *Bioinformatics*, št. 17, str. 763—774, 2001.