Fakultet elektrotehnike i računarstva Sveučilišta u Zagrebu
Strojno učenje, ak. god. 2011/12. – **bilješka za predavanja** (v1.1)

Bilješka 5
Grupiranje podataka

Do sada smo se bavili problemima nadziranog učenja, odnosno problemima klasifikacije i regresije. U oba ta slučaja na raspolaganju nam je bio skup označenih primjera za učenje, $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}$, kod kojega je svakome primjeru $\mathbf{x}^{(i)}$ pridružena diskretna ili kontinuirana ciljna vrijednosti $y^{(i)}$. U mnogim slučajevima, međutim, skup primjera nije označen, dakle na raspolaganju imamo samo neoznačene primjere, $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$. To može biti zato što je označavanje preskupo, ili pak zato što oznake (odnosno klase) nisu unaprijed poznate i potrebno je najprije otkriti kakve se zakonitosti kriju u podatcima. U takvim slučajevima moramo se osloniti na **nenadzirano učenje** (engl. *unsupervised learning*). Tri tipična zadatka nenadziranog učenja su **grupiranje** (engl. *clustering*) podataka, otkrivanje novih vrijednosti ili vrijednosti koje odskaču (engl. *novelty/outlier detection*) i **smanjenje dimenzionalnosti** (engl. *dimensionality reduction*).

U nastavku se usredotočujemo na grupiranje podataka. Grupiranje podataka je postupak razdjeljivanja primjera u grupe (klastere) primjera, tako da su slični primjeri (slični po nekom svojstvu) svrstani u istu grupu, a različiti primjeri u različite grupe. Svrha grupiranja jest nalaženje "prirodnih" (intrinzičnih) grupa u skupu neoznačenih podataka.

1 Vrste grupiranja

Postoje dvije osnovne vrste grupiranja. Kod **particijskog grupiranja** (engl. *partitional clustering*), skup primjera particionira se u grupe sličnih primjera. **Hijerarhijsko grupiranje** (engl. *hierarchical clustering*) skup primjera razdjeljuje u ugniježđene grupe koje sačinjavaju hijerarhiju grupa.

Grupiranje također možemo podijeliti prema "čvrstoći" granica između grupa. Kod **čvrstog grupiranja** (engl. *hard clustering*) svaki primjer može pripadati isključivo jednoj grupi. Kod **mekog grupiranja** (engl. *soft clustering*) jedan primjer može pripadati u više grupa, i to eventualno s različitim stupnjem ili različitom vjerojatnošću pripadanja.

Možemo također govoriti o različitim pristupima s obzirom na kriterij koji se koristi za grupiranje. Kriterij može biti minimizacija kriterijske funkcije (npr. kod algoritma k-srednjih vrijednosti), maksimizacija izglednosti (kod EM-algoritma) ili se grupiranje može raditi prema nekoj funkciji udaljenosti ili mjeri sličnosti (kod hijerarhijskog grupiranja).

2 Primjene grupiranja

Primjene grupiranja mogu biti različite, od kojih neke tipične navodimo u nastavku.

- Grupiranje se često koristi za **istraživanje podataka** (engl. *data exploration*), s ciljem pronalaženja skrivene strukture u podatcima. Jednom kada se primjeri grupiraju, dobivene grupe mogu se ručno označiti, središta grupa mogu se tumačiti kao prototipni predstavnici grupa, a za svaku se grupu mogu utvrditi tipični rasponi vrijednosti značajki. To omogućava da se podatci opišu na jednostavniji način, da se uoče pravilnosti i sličnosti u podatcima te da se otkriju odnosi između grupa.

- Grupiranje se može koristiti za **kompresiju podataka**, uključivo za preslikavanje kontinuiranih vrijednosti u diskretne vrijednosti. Npr., boje 24-bitne digitalne slike mogu se grupirati u 256 grupa, a zatim se svaka boja može predstaviti centroidom grupe. Time se ostvaruje kompresija sa 24 na 8 bita po slikovnom elementu. Takav postupak nazivamo **kvantizacija vektora** (engl. *vector quantization*).
- Kod nadziranog učenja, grupiranje se može koristiti kao tehnika **predobrade** (engl. *preprocessing*) s ciljem **smanjenja dimenzionalnosti** (engl. *dimensionality reduction*) prostora primjera, odnosno smanjenja broja značajki. Smanjenje dimenzionalnosti dovodi do ušteda u smislu prostora i vremena izvođenja te smanjuje utjecaj šuma.
- U situaciji kada je u skupu primjera za učenje samo manji dio njih označen, grupiranje se može koristiti u kombinaciji s nadziranim učenjem kako bi se automatski označili svi primjeri za učenje. Osnovna ideja jest da se primjeri (i označeni i neoznačeni) najprije grupiraju, a zatim se neoznačeni primjeri unutar pojedine grupe označavaju prema označenim primjerima koji su se našli u istoj grupi. Pritom je najjednostavnije kao oznaku grupe odabrati onu koja se u grupi najčešće pojavljuje. Ova se tehnika naziva **grupiraj i označi** (engl. *cluster and label*) i tipičan je primjer **polunadziranog** (engl. *semi-supervised*) pristupa učenju.

2.1 Smanjenje dimenzionalnosti grupiranjem

Grupiranje se može koristiti za smanjenje dimenzionalnosti prostora primjera. U izvornom prostoru primjera $\mathcal{X}$, primjere možemo prikazati matricom $N \times n$, čiji retci odgovaraju primjerima $\mathbf{x}^{(i)}$, a stupci značajkama $x_1, \ldots, x_n$. Grupiranjem u K grupa, $K < n$, dobivamo matricu smanjenih dimenzija $N' \times K$. Ovu matricu možemo dobiti na dva načina, ovisno o tome što grupiramo.

- Prva mogućnost jest grupiranje redaka matrice, odnosno grupiranje primjera. Grupiranjem N primjera u K grupa ostvarujemo preslikavanje u nov, K-dimenzijski prostor, u kojemu svakom primjeru odgovara vektor čije komponente indiciraju kojoj grupi dotični primjer pripada. Kod čvrstog grupiranja samo je jedna komponenta tog vektora jednaka jedinici, dok su ostale jednake nuli; kod mekog grupiranja više komponenti vektora može biti različito od nule. Posljedično, kod čvrstog grupiranja gubitak informacija bit će razmjerno velik jer će se mnogi primjeri preslikati u istu točku K-dimenzijskog prostora, pa će dobivena matrica imati manje od N redaka. Primijetite da se izvorne značajke koriste samo za grupiranje u n-dimenzijskog prostoru, a nakon toga te se značajke više ne koriste i zamjenjuju se značajkama u K-dimenzijskom prostoru. Također primijetite da, ako je broj primjera veći od dimenzije izvornog prostora primjera, $N > n$, onda – ovisno o tome kako se provodi grupiranje – dimenzija K novog prostora također može biti veća od n, čime se zapravo ostvaruje povećanje dimenzionalnosti, što može biti korisno u nekim primjenama.
- Drugi način grupiranja jest da se zamijene uloge primjera i značajki: umjesto da grupiramo primjere (tj. retke matrice), grupiramo njihove značajke (tj. stupce matrice). Ovdje nam je cilj u istu grupu smjestiti značajke koje su međusobno slične. Dvije značajke smatramo to sličnijima što se više podudaraju njihove vrijednosti kroz pojedinačne primjere. Smanjenje dimenzionalnosti ostvaruje se zamjenom svih značajki koje su grupirane u zajedničku grupu jednom novom, reprezentativnom značajkom, npr. centroidom grupe. Ovaj pristup naziva se **grupiranje značajki** (engl. *feature clustering*).

3 Algoritam k-srednjih vrijednosti

Najjednostavniji i najpoznatiji algoritam grupiranja jest **algoritam k-srednjih vrijednosti** (engl. *k-means algorithm*). Algoritmom se primjeri iz neoznačenog skupa primjera $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ grupiraju u K čvrstih grupa, gdje se parametar K zadaje unaprijed. Svaka grupa predstavljena je svojim centroidom, $\{\boldsymbol{\mu}_k\}_{k=1}^K$. Grupiranjem primjera u grupe predstavljene vektorom $\boldsymbol{\mu}_k$ nastaje određena pogreška. Pogrešku grupiranja izražava **kriterijska funkcija**:[1]

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} b_k^{(i)} \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_k\|^2 \tag{1}$$

gdje je $\|\cdot\|$ **euklidska norma**, tj.

$$\|\mathbf{x} - \boldsymbol{\mu}\|^2 = (\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu})\,.$$

Vrijednost $b_j^{(i)}$ u (1) je indikatorska varijabla koja indicira kojoj grupi pripada primjer $\mathbf{x}^{(i)}$: ako $b_k^{(i)} = 1$, onda primjer $\mathbf{x}^{(i)}$ pripada grupi k. Pogreška je jednaka zbroju kvadratnih pogreški, odnosno kvadratima euklidske udaljenosti primjera $\mathbf{x}^{(i)}$ od središta $\boldsymbol{\mu}_k$ grupe u koju su ti primjeri svrstani. Želimo li minimizirati pogrešku J, svaki primjer $\mathbf{x}^{(i)}$ trebamo svrstati u grupu čije je središte $\boldsymbol{\mu}_k$ tom primjeru najbliže, to jest

$$b_k^{(i)} = \begin{cases} 1 & \text{ako } k = \operatorname*{argmin}_j \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_j\| \\ 0 & \text{inače} \end{cases} \tag{2}$$

Očito je da bi bilo kakvo drugačije razvrstavanje primjera u grupe rezultiralo samo još većom pogreškom.

Sada se međutim postavlja pitanje kako odabrati srednje vrijednosti $\boldsymbol{\mu}_k$, a da one minimiziraju pogrešku J. Budući da vrijednosti $b_k^{(i)}$ također ovise o $\boldsymbol{\mu}_k$, optimalne vrijednosti za $\boldsymbol{\mu}_k$ nije moguće izraziti u zatvorenoj formi. Umjesto toga, algoritam k-srednjih vrijednosti optimizaciju provodi iterativno. Algoritam započinje sa slučajno odabranim srednjim vrijednostima $\boldsymbol{\mu}_k$. Zatim se u svakoj iteraciji temeljem (2) za svaki primjer $\mathbf{x}^{(i)}$ izračunava vrijednost $b_k^{(i)}$, odnosno svaki se primjer pridjeljuje grupi čijem je centroidu najbliži. Nakon toga – budući da sad imamo fiksirane vrijednosti $b_k^{(i)}$ – možemo izravno minimizirati pogrešku (1). Postavljanjem $\nabla_{\boldsymbol{\mu}_k} J = \mathbf{0}$ i rješavanjem po $\boldsymbol{\mu}_k$ dobivamo:

$$2\sum_{i=1}^{N} b_k^{(i)}(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k) = \mathbf{0}$$

iz čega slijedi

$$\boldsymbol{\mu}_k = \frac{\sum_i b_k^{(i)} \mathbf{x}^{(i)}}{\sum_i b_k^{(i)}}\,. \tag{3}$$

Vektor $\boldsymbol{\mu}_k$ jednak je dakle srednjoj vrijednosti vektora svih primjera koji su svrstani u grupu k. Budući da je ovime ostvarena promjena vektora $\boldsymbol{\mu}_k$ u odnosu na njegovu prethodnu vrijednost, sada treba opet primijeniti izraz (2) i ponovno izračunati koji primjeri pripadaju grupi k. Ova dva koraka ponavljaju se sve dok se ne dosegne stacionarno stanje, odnosno stanje u kojemu nema daljnjih promjena vrijednosti $\boldsymbol{\mu}_k$.

[1] Također: *ciljna funkcija* (engl. *objective function*), *mjera distorzije* (engl. *distortion measure*) ili (u kontekstu kvantizacije vektora) *pogreška rekonstrukcije*.

Algoritam 1. Algoritam k-srednjih vrijednosti

inicijaliziraj centroide $\boldsymbol{\mu}_k$, $k = 1, \ldots, K$ (npr. na slučajno odabrane $\mathbf{x}^{(i)}$)
ponavljaj
  za svaki $\mathbf{x}^{(i)} \in \mathcal{D}$
    $b_k^{(i)} \leftarrow \begin{cases} 1 & \text{ako } k = \text{argmin}_j \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_j\| \\ 0 & \text{inače} \end{cases}$
  za svaki $\boldsymbol{\mu}_k$, $k = 1, \ldots, K$
    $\boldsymbol{\mu}_k \leftarrow \sum_{i=1}^N b_k^{(i)} \mathbf{x}^{(i)} / \sum_{i=1}^N b_k^{(i)}$
dok $\boldsymbol{\mu}_k$ ne konvergiraju

Razmotrimo računalnu složenost algoritma. Složenost izračuna euklidske udaljenosti je $\mathcal{O}(n)$, gdje je n broj značajki. U prvom koraku (pridjeljivanje primjera grupama) izračunavamo KN udaljenosti, pa je složenost prvog koraka $\mathcal{O}(nNK)$. U drugom koraku (izračun centroida), svaki se primjer pridodaje jednome centroidu (za sve ostale centroide k vrijedi $b_k^{(i)} = 0$), pa je složenost $\mathcal{O}(nN)$. Ukupna vremenska složenost algoritma je linearna po svim parametrima, $\mathcal{O}(TnNK)$, gdje je T broj iteracija. U praksi je broj iteracija T redovito mnogo manji od broja primjera N.

Uočite da algoritam, osim odabira početnih središta, ima dodatan izvor nedeterminističnosti, a to je razrješavanje izjednačenja udaljenosti dvaju primjera od centroida. Pri implementaciji treba voditi računa da se razrješavanje provodi na proizvoljan, ali konzistentan način (u suprotnom se može dogoditi da algoritam zaglavi u beskonačnoj petlji).

Algoritam k-srednjih vrijednosti naziva se također i **Lloydov algoritam**. U kontekstu kvantizacije vektora, algoritam je poznat pod nazivom algoritam **Linde-Buzo-Gray** (LBG).

3.1 Odabir početnih središta

Algoritam k-srednjih vrijednosti u stvari pretražuje prostor stanja kojih ima onoliko koliko ima različitih particija od N primjera u K skupova. Nameću su sljedeća pitanja: doseže li algoritam uvijek stacionarno stanje te je li grupiranje koje algoritam nalazi optimalno u smislu pogreške (1)? Lako je dokazati da će algoritam k-srednjih vrijednosti zajamčeno doseći stacionarno stanje. Broj mogućih particija iznosi K^N i konačan je, pa je konačan i broj konfiguracija u kojima se svaki $\boldsymbol{\mu}_k$ nalazi u središtu svoje grupe. Budući da se u svakoj iteraciji pogreška smanjuje, to u svakoj iteraciji algoritam nalazi novu konfiguraciju (nikada ne posjećuje dva puta istu konfiguraciju), a takvih je konfiguracija konačno mnogo, pa se algoritam nužno zaustavlja u konačnom broju koraka.

S druge strane, lako je pokazati da algoritam ne nalazi uvijek optimalno grupiranje. Algoritam k-srednjih vrijednosti je pohlepan i pronalazi lokalno optimalno rješenje. Hoće li to rješenje biti i globalno optimalno, ovisi o izboru početnih srednjih vrijednosti $\boldsymbol{\mu}_k$. Postoje razni načini kako odabrati početne srednje vrijednosti:

- Nasumično odabrati K primjera kao početne vrijednosti $\boldsymbol{\mu}_k$. Ovime se doduše izbjegava postavljanje centroida na mjesta u prostoru primjera u kojemu uopće nema primjera (a koje se lako može dogoditi ako se središta izabiru posve nasumično), ali se ne rješava problem zaglavljivanja u lokalnom optimumu. Problem također predstavljaju primjeri koji odskaču (engl. *outliers*), koji lako mogu završiti u zasebnim grupama. Na prvi pogled možda se čini da je dobro da takvi primjeri završe u

zasebnim grupama, ali to nije tako jer je broj grupa K ograničen i one se trebaju poklapati s "prirodnim" (većinskim) grupama koje postoje u podatcima;

- Izračunati srednju vrijednost (centroid) sviju primjera, $\boldsymbol{\mu}$, a zatim srednjoj vrijednosti $\boldsymbol{\mu}$ dodavati manje slučajne vektore i tako dobiti K vektora $\boldsymbol{\mu}_k$. Ovo rješava problem izoliranih primjera, ali ne rješava problem zaglavljivanja u lokalnome optimumu;

- Izračunati prvu glavnu komponentu skupa primjera metodom PCA, razdijeliti raspon na K jedakih intervala, čime se primjeri razdjeljuju u K grupa, i zatim uzeti srednje vrijednosti tih grupa kao početne vrijednosti $\boldsymbol{\mu}_k$;

- Slučajno odabrati jedno početno središte $\boldsymbol{\mu}_k$, a zatim svako iduće središte odabrati tako da je što dalje od ostalih središta. Primjer ovakvog pristupa je algoritam **k-means++**, kod kojega je vjerojatnost da primjer $\mathbf{x}^{(i)}$ bude odabran kao novo središte $\boldsymbol{\mu}_i$ proporcionalna kvadratu udaljenosti tog primjera od njemu najbližeg, već odabranog središta $\boldsymbol{\mu}_k$.

$$P(\boldsymbol{\mu}_i = \mathbf{x}^{(i)}|\mathcal{D}) = \frac{\|\mathbf{x}^{(i)} - \boldsymbol{\mu}_k\|^2}{\sum_j \|\mathbf{x}^{(j)} - \boldsymbol{\mu}_k\|^2}\,. \tag{4}$$

  Premda na ovaj način vrijednosti koje odskaču imaju veću vjerojatnost da budu odabrane za središte, njih je u pravilu manje, pa je ipak vjerojatniji odabir nekog od prosječnih primjera, koji su brojniji. Pokazano je da ovaj način odabira početnih središta znatno smanjuje pogrešku grupiranja, a također ubrzava konvergenciju algoritma.

U slučajevima kada se početna središta određuju nedeterministički, preporuča se algoritam pokrenuti više puta kako bi se dobio rezultat sa što manjom pogreškom grupiranja.

Osim o izboru početnih središta grupa, ishod grupiranja ovisi očito i o odabranom broju grupa, odnosno vrijednosti parametra K. Problem odabira broja grupa zajednički je svim algoritmima grupiranja i razmotrit ćemo ga u odjeljku 7.

3.2 Algoritam k-medoida

Algoritam k-srednjih vrijednosti upotrebljava euklidsku udaljenost kao mjeru različitosti između primjera i prototipnih vektora. To algoritam čini vrlo osjetljivim na vrijednosti koje odskaču. Algoritam je također ograničen na slučajeve kada je primjere moguće prikazati u vektorskome prostoru, pa je između njih moguće izračunati euklidsku udaljenost i srednju vrijednost (centroid) grupe. Međutim, u nekim slučajevima raspolažemo samo informacijom o međusobnoj sličnosti parova primjera. To je slučaj ako, primjerice, želimo grupirati riječi na temelju sličnosti znakovnih nizova, tako da dobijemo grupe grafijski sličnih riječi, ili grupirati ljude na temelju jakosti njihova poznanstava, tako da dobijemo grupe ljudi koji se međusobno dobro poznaju. U ovakvim situacijama ne raspolažemo mjerom udaljenosti, već **mjerom sličnosti** (engl. *similarity measure*) ili njezinim komplementom, **mjerom različitosti** (engl. *dissimilarity measure*), izračunatom između svih parova primjera.

Poopćenje algoritma k-srednjih vrijednosti predstavlja **algoritam k-medoida**, kod kojega je kriterijska funkcija definirana pomoću općenite mjere različitosti $\nu(\mathbf{x}, \mathbf{x}')$ između dvaju primjera:

$$\tilde{J} = \sum_{i=1}^{N} \sum_{k=1}^{K} b_k^{(i)} \nu(\mathbf{x}^{(i)}, \boldsymbol{\mu}_k)\,. \tag{5}$$

Mjera različitosti ν (odnosno mjera sličnosti) općenitija je od euklidske udaljenosti i od bilo koje druge mjere udaljenosti budući da ne mora ispunjavati uvjete metrike (v. odjeljak 5.1).

Kod algoritma k-medoida, prototipe grupa čine medoidi, odnosno reprezentativni primjeri iz skupa $\mathcal{D}$, a ne centroidi. Tipična izvedba je **algoritam PAM** (engl. *partitioning around medoids*).

Algoritam 2. Algoritam PAM

inicijaliziraj medoide $\mathcal{M} = \{\boldsymbol{\mu}_k\}_{k=1}^{K}$ na odabrane $\mathbf{x}^{(i)}$
ponavljaj
  za svaki $\mathbf{x}^{(i)} \in \mathcal{D} \backslash \mathcal{M}$
    $b_k^{(i)} \leftarrow \begin{cases} 1 & \text{ako } k = \operatorname{argmin}_j \nu(\mathbf{x}^{(i)}, \boldsymbol{\mu}_j) \\ 0 & \text{inače} \end{cases}$
  za svaki $\boldsymbol{\mu}_k \in \mathcal{M}$
    $\boldsymbol{\mu}_k \leftarrow \operatorname{argmin}_{\boldsymbol{\mu}_j \in \mathcal{D} \backslash \mathcal{M} \cup \{\boldsymbol{\mu}_k\}} \sum_i \sum_k b_k^{(i)} \nu(\mathbf{x}^{(i)}, \boldsymbol{\mu}_j)$
dok $\boldsymbol{\mu}_k$ ne konvergiraju

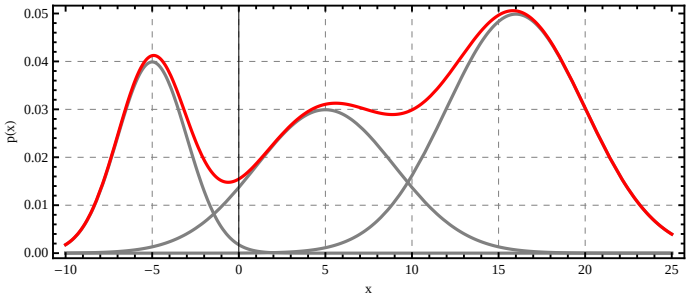
Algoritam se također izvršava s po dva koraka u svakoj iteraciji. U prvome koraku, primjeri se svrstavaju u grupu za koju je vrijednost mjere različitosti najmanja. To iziskuje $\mathcal{O}(K(N-K))$ izračuna mjere različitosti ν. U drugom koraku prototipi grupa odabiru se tako da minimiziraju $\tilde{J}$. Za svaku od K grupa, svaki od $N-K$ primjera koji trenutno nisu odabrani kao medoidi zamjenjuje se s trenutno odabranim medoidom, izračunava se zbroj mjere ν između $N-K$ primjera i predloženog medoida te se odabire onaj medoid za koji je ta vrijednost najmanja. To iziskuje $\mathcal{O}(K(N-K)^2)$ izračuna mjere različitosti ν. Posljedično, ukupno algoritam iziskuje $\mathcal{O}(TK(N-K)^2)$ izračuna mjere ν, gdje je T broj iteracija.

Visoka vremenska složenost glavni je nedostatak PAM-algoritma, pa su predložena različita poboljšanja, npr. algoritmi **CLARA** (engl. *clustering i n large appications*) ili **CLARANS** (engl. *CLARA based upon radnomized search*).

4 Model miješane gustoće

Sada ćemo razmotriti probabilistički pristup grupiranju. Za razliku od algoritma k-srednjih vrijednosti, kod kojega su granice između grupa čvrste, kod probabilističkog pristupa primjeri grupama pripadaju s određenom vjerojatnošću. Drugim riječima, jedan te isti primjer može pripadati u dvije ili više grupa, što dovodi do mekih granica između grupa.

Probabilistički pristup grupiranju je generativni i parametarski: pretpostavljamo da se primjeri iz svake grupe pokoravaju nekoj teorijskoj razdiobi. Kod klasifikacije nam je unaprijed bilo poznato koji primjeri pripadaju kojoj klasi, pa smo svaku klasu mogli modelirati zasebnom gustoćom vjerojatnosti (npr. Gaussovom). Kod grupiranja nemamo informaciju o tome koji primjer pripada kojoj klasi i naš je pristup zato drugačiji. Umjesto da izravno



Slika 1: Mješavina Gaussovih gustoća sačinjena od komponenti $\mathcal{N}(-5, 2)$, $\mathcal{N}(5, 4)$ i $\mathcal{N}(16, 5)$ s koeficijentima $\pi_1 = 0.2$, $\pi_2 = 0.3$ i $\pi_3 = 0.5$.

modeliramo izglednosti za pojedinačne grupe, rješavat ćemo općenitiji i teži problem modeliranja gustoće vjerojatnosti $p(\mathbf{x})$. Funkcija gustoće $p(\mathbf{x})$ općenito je složenog oblika, pa ćemo koristiti **miješani model** (engl. *mixture model*) sačinjen od linearne kombinacije K osnovnih razdioba. Svaka od tih razdioba odgovara jednoj od K grupa, pa će naš cilj zapravo biti utvrditi parametre tih razdioba i tako ustanoviti s kojom vjerojatnošću primjeri pripadaju pojedinim grupama.

Miješana gustoća (engl. *mixture density*) je linearna kombinacija K funkcija gustoća vjerojatnosti:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \, p(\mathbf{x}|\boldsymbol{\theta}_k) \tag{6}$$

gdje su $p(\mathbf{x}|\boldsymbol{\theta}_k)$ **komponente mješavine** (engl. *mixture components*), svaka s parametrima $\boldsymbol{\theta}_k$. Ako se za komponente koriste Gaussove gustoće vjerojatnosti, $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, miješanu gustoću nazivamo **mješavina Gaussovih gustoća** (engl. *mixture of Gaussians*); primjer takve mješavine za jednodimenzijski slučaj dan je na slici 1.

Parametri π_k u (6) su **koeficijenti mješavine** (engl. *mixture coefficients*). Ako integriramo (marginaliziramo) obje strane izraza (6) po $\mathbf{x}$, onda, budući da su pojedinačne komponente normalizirane, mora vrijediti $\sum_{k=1}^{K} \pi_k = 1$. Također, budući da $p(\mathbf{x}|\boldsymbol{\theta}_k) \geqslant 0$ i $p(\mathbf{x}) \geqslant 0$, to mora vrijediti $0 \leqslant \pi_k \leqslant 1$. Vidimo dakle da se koeficijenti π_k mogu tumačiti kao vjerojatnosti. Sukladno pravilima o zbroju i umnošku vjerojatnosti, marginalnu gustoću $p(\mathbf{x})$ možemo izraziti kao

$$p(\mathbf{x}) = \sum_{k=1}^{K} P(\mathcal{G}_k) p(\mathbf{x}|\mathcal{G}_k) \tag{7}$$

gdje je $\pi_k = P(\mathcal{G}_k)$ apriorna vjerojatnost odabira komponente k, dok je $p(\mathbf{x}|\boldsymbol{\theta}_k) = p(\mathbf{x}|\mathcal{G}_k)$ gustoća od $\mathbf{x}$ uz odabranu komponentu k. Možemo primijeniti Bayesovo pravilo kako bismo izračunali aposteriorne vjerojatnosti $P(\mathcal{G}_k|\mathbf{x})$:

$$P(\mathcal{G}_k|\mathbf{x}) = \frac{P(\mathcal{G}_k) p(\mathbf{x}|\mathcal{G}_k)}{p(\mathbf{x})} = \frac{P(\mathcal{G}_k) p(\mathbf{x}|\mathcal{G}_k)}{\sum_j P(\mathcal{G}_j) p(\mathbf{x}|\mathcal{G}_j)} = \frac{\pi_k p(\mathbf{x}|\boldsymbol{\theta}_k)}{\sum_j \pi_j p(\mathbf{x}|\boldsymbol{\theta}_j)} \equiv h_k \,. \tag{8}$$

Ovu veličinu označavamo s h_k i nazivamo **odgovornost**. Odgovornost h_k iskazuje kolika je vjerojatnost da primjer $\mathbf{x}$ pripada komponenti $\mathcal{G}_k$.

Usporedimo li izraz (8) s izrazom za Bayesov klasifikator, možemo zaključiti da je generativni klasifikacijski model u biti miješani model kod kojega grupe $\mathcal{G}_k$ odgovaraju klasama

$\mathcal{C}_j$, gustoće komponenti $p(\mathbf{x}|\mathcal{G}_k)$ odgovaraju izglednostima $p(\mathbf{x}|\mathcal{C}_j)$, vjerojatnosti komponenti $P(\mathcal{G}_k)$ odgovaraju apriornim vjerojatnostima klasa $P(\mathcal{C}_j)$, a odgovornost $P(\mathcal{G}_k|\mathbf{x})$ odgovara posteriornoj vjerojatnosti $P(\mathcal{C}_j|\mathbf{x})$. Razlika je međutim u tome što primjeri iz skupa za učenje $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ nisu označeni, pa ne znamo koji primjer pripada kojoj komponenti.

Kao i kod klasifikacije, naš je zadatak odrediti parametre modela

$$\boldsymbol{\theta} = \{P(\mathcal{G}_k), \boldsymbol{\theta}_k\}_{k=1}^K .$$

Na primjer, kod mješavine univarijatnih Gaussovih gustoća, parametri koje moramo odrediti su

$$\boldsymbol{\theta} = \{P(\mathcal{G}_k), \mu_k, \sigma^2\}_{k=1}^K$$

odnosno ukupno $3K$ parametara. U općenitijem slučaju viševarijatne Gaussove gustoće, parametri su

$$\boldsymbol{\theta} = \{P(\mathcal{G}_k), \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K .$$

a to je ukupno $(\frac{1}{2}n(n+1)+n+1)K$ odnosno $\mathcal{O}(n^2K)$ parametara. Kao i kod generativnih klasifikacijskih modela, i ovdje se možemo odlučiti za jednostavniji model (model s dijeljenom, dijagonalnom ili kovarijacijskom matricom) i tako smanjiti ukupan broj parametara.

Kao i kod klasifikacije, parametre možemo procijeniti metodom najveće izglednosti. No, budući da ne znamo koji primjer pripada kojoj komponenti, procjenu ne možemo raditi zasebno za svaku klasu, pa je procjena parametara sada složenija. Funkcija log-izglednosti za gustoću (6) jednaka je

$$\ln \mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = \ln \prod_{i=1}^N p(\mathbf{x}^{(i)}) = \ln \prod_{i=1}^N \sum_{k=1}^K \pi_k \, p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k) = \sum_{i=1}^N \ln \sum_{k=1}^K \pi_k \, p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k) . \tag{9}$$

Nažalost, maksimizacija izraza (9) nema rješenja u zatvorenoj formi. Problem predstavlja sumacija koja se nalazi između logaritamske funkcije i funkcije gustoće vjerojatnosti. Rješenje zbog toga moramo potražiti u iterativnim metodama. Jedna mogućnost je optimizacija gradijentnom metodom. Češće upotrebljavana je metoda maksimizacije očekivanja, koju opisujemo u nastavku.

4.1 Algoritam maksimizacije očekivanja

Algoritam maksimizacije očekivanja (engl. *expectation maximization algorithm*) ili **EM-algoritam** je iterativan optimizacijski postupak za rješavanje problema najveće izglednosti kod modela s **latentnim varijablama**. Latentna varijabla je slučajna varijabla čije realizacije ne opažamo izravno, već o njoj zaključujemo na temelju drugih, opaženih varijabli. Latentna varijabla može biti uvedena samo kao sredstvo apstrakcije, s ciljem pojednostavljenja modela. Također, latentna varijabla može modelirati nešto što je stvarno, no nedostupno, i takvu latentnu varijablu nazivamo **skrivena varijabla**.

Primjer 1 (Latentna varijabla) Imamo dva novčića, A i B. Vjerojatnost da bacanjem novčića A dobijemo glavu neka je μ_A, dok je kod novčića B to μ_B. Dakle, $P(A) = \mu_A$ i $P(B) = \mu_B$.

Parametri μ_A i μ_B su nam nepoznati te ih želimo procijeniti na temelju uzorka. Pretpostavimo da smo uzorak dobili tako da smo slučajno odabrali jedan od dvaju novčića,

bacali ga 10 puta, te sve to ponavljali ukupno 5 rundi (ukupno dakle imamo 50 bacanja). Neka je $X_i \in \{0, 10\}$ broj koliko smo puta dobili glavu u i-toj rundi. Neka slučajna varijabla $Z_i \in \{A, B\}$ određuje koji od dviju novčića je korišten u i-toj rundi. Naš uzorak $\{(x_i, z_i)\}_{i=1}^5$ je

$$\{(5, B), (9, A), (8, A), (4, B), (7, A)\}.$$

Gornji problem je problem s potpunim podatcima jer nam je poznato sve što nam treba da bismo mogli izračunati procjenu parametara: znamo koliko smo puta u svakoj rundi dobili glavu (x_i) i znamo koji je novčić korišten u kojoj rundi (z_i). Parametare lako možemo procijeniti metodom najveće izglednosti:

$$\hat{\mu}_A = 24/30 = 0.8 \qquad \hat{\mu}_B = 9/20 = 0.45$$

Problem postaje teži kada ne znamo koji je novčić korišten u kojoj rundi, odnosno kada na raspolaganju imamo samo uzorak $\mathbf{X} = \{x_i\}_{i=1}^5$. Varijabla Z_i sada je skrivena te je riječ o problemu s nepotpunim podatcima. U tom slučaju parametre ne možemo izračunati izravno metodom najveće izglednosti.

Veza ovog primjera s problemom grupiranja podataka je sljedeća: svaka runda odgovara jednom neoznačenom primjeru $\mathbf{x}^{(i)}$. Ukupno imamo pet primjera. Primjer može pripadati prvoj grupi (novčić A) ili drugoj grupi (novčić B). Ove su grupe modelirane Bernoullijevom razdiobom s parametrom μ_A odnosno μ_B. Varijabla Z_i određuje kojoj od dviju grupa pripada primjer $\mathbf{x}^{(i)}$, ali je nama ta informacija skrivena.

U nastavku ćemo najprije dati općenitu formulaciju EM-algoritma. Nakon toga pokazat ćemo primjenu algoritma na model miješane gustoće, a zatim konkretno na Gaussovu mješavinu. Treba napomenuti da EM-algoritam ima široku primjenu i da je grupiranje podataka samo jedna od mogućih primjena ovog algoritma.

4.2 Općenita formulacija algoritma

Cilj algoritma maksimizacije očekivanja jest nalaženje parametara $\boldsymbol{\theta}$ koji maksimiziraju log-izglednost $\ln \mathcal{L}(\boldsymbol{\theta}|\mathbf{X})$, gdje su $\mathbf{X}$ podatci odnosno primjeri. Model $p(\mathbf{X}|\boldsymbol{\theta})$ proširujemo skupom latentnih varijabli $\mathbf{Z}$ i radimo sa zajedničkom gustoćom $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$. Marginalna gustoća $p(\mathbf{X}|\boldsymbol{\theta})$ uvijek se može rekonstruirati marginalizacijom zajedničke gustoće po latentnim varijablama:

$$p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}).$$

Skup $\{\mathbf{X}, \mathbf{Z}\}$ nazivamo potpunim, a $\mathbf{X}$ nepotpunim skupom podataka. Analogno, $\ln \mathcal{L}(\boldsymbol{\theta}|\mathbf{X}, \mathbf{Z})$ zovemo **potpuna log-izglednost**, a $\ln \mathcal{L}(\boldsymbol{\theta}|\mathbf{X})$ zovemo **nepotpuna log-izglednost**. Nepotpuna log-izglednost jednaka je

$$\ln \mathcal{L}(\boldsymbol{\theta}|\mathbf{X}) = \ln p(\mathbf{X}|\boldsymbol{\theta}) = \ln \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) \tag{10}$$

dok je potpuna log-izglednost jednaka

$$\ln \mathcal{L}(\boldsymbol{\theta}|\mathbf{X}, \mathbf{Z}) = \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}). \tag{11}$$

Važna razlika između izraza (10) (optimizacija marginalne gustoće) i (11) (optimizacija zajedničke gustoće) jest u tome što u prvome slučaju logaritam djeluje na zbroj gustoća,

dok u drugome djeluje izravno na gustoću vjerojatnosti. Ta je razlika ključna: u prvome slučaju analitičko rješenje nije moguće, dok u drugom jest.

Nažalost, budući da nemamo pristup potpunom skupu podataka, ne možemo izravno raditi s potpunom log-izglednošću. Vrijednosti latentnih varijabli $\mathbf{Z}$ su nam nepoznate. Umjesto da izravno radimo s potpunom log-izglednošću, radit ćemo s očekivanjem potpune log-izglednosti, $\mathbb{E}[\ln \mathcal{L}(\boldsymbol{\theta}|\mathbf{X},\mathbf{Z})]$. Osnovna ideja EM-algoritma jest iterativno ugađati parametre $\boldsymbol{\theta}$ kako bi se maksimiziralo to očekivanje. Može se pokazati da maksimizacija očekivanja potpune log-izglednosti ujedno dovodi do povećanja nepotpune log-izglednosti $\ln \mathcal{L}(\boldsymbol{\theta}|\mathbf{X})$, što je zapravo naš cilj.

Maksimizacija očekivanja $\mathbb{E}[\ln \mathcal{L}(\boldsymbol{\theta}|\mathbf{X},\mathbf{Z})]$ ostvaruje se alterniranjem između dva koraka: E-koraka i M-koraka. U **E-koraku** (korak procjene) računamo očekivanje potpune log-izglednosti uz fiksirane trenutne vrijednosti parametara $\boldsymbol{\theta}^{(t)}$. To očekivanje označavamo sa $\mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ i računamo kao

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) &\equiv \mathbb{E}_{\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{(t)}}[\ln \mathcal{L}(\boldsymbol{\theta}|\mathbf{X},\mathbf{Z})] \\ &= \mathbb{E}_{\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{(t)}}[\ln p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta})] \qquad (12)\\ &= \sum_{\mathbf{Z}} P(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{(t)}) \ln p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta})\,. \qquad (13)\end{aligned}$$

Očekivanje smo izrazili po slučajnoj varijabli $\mathbf{Z}$ uz fiksirane varijable $\mathbf{X}$ i parametre $\boldsymbol{\theta}^{(t)}$, tj. po varijabli $\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{(t)}$. Vjerojatnost $P(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{(t)})$ je aposteriorna vjerojatnost latentne varijable uz trenutne vrijednosti parametara. Nju možemo izračunati (primjenom Bayesovog teorema) jer su nam $\mathbf{X}$ i $\boldsymbol{\theta}^{(t)}$ poznati. Jedino što je u izrazu (12) slobodno jesu parametri $\boldsymbol{\theta}$, i to su parametri koje trebamo optimirati.

U **M-koraku** (korak maksimizacije) odabiremo nove parametre $\boldsymbol{\theta}^{(t+1)}$ koji maksimiziraju (12):

$$\boldsymbol{\theta}^{(t+1)} = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})\,. \qquad (14)$$

Ovo je manje težak optimizacijski problem od onog od kojeg smo krenuli i u pravilu (ovisno o odabiru parametarskog modela) može se riješiti analitički.

U nastavku je dan pseudokod općenitog EM-algoritma.

Algoritam 3. Općenit EM-algoritam

inicijaliziraj parametre $\boldsymbol{\theta}^{(0)}$
$t \leftarrow 0$
ponavljaj:
  E-korak: izračunaj $P(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{(t)})$
  M-korak: $\boldsymbol{\theta}^{(t+1)} \leftarrow \operatorname{argmax}_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$
    gdje $\mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \sum_{\mathbf{Z}} P(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{(t)}) \ln p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta})$
  $t \leftarrow t+1$
do konvergencije (nepotpune) log-izglednosti ili konvergencije parametara

Krenuvši od nekih početno odabranih parametara $\boldsymbol{\theta}^{(0)}$, E-korak i M-korak se izmjenjuju sve dok $\boldsymbol{\theta}^{(t)}$ ne konvergira. Konvergencija je zajamčena budući da algoritam u svakoj iteraciji povećava očekivanje izglednosti. S druge strane, algoritam ne nalazi nužno globalni optimum izglednosti. Zbog toga je važno odabrati dobre početne vrijednosti parametara. Kod grupiranja se taj problem tipično rješava tako da se za inicijalizaciju koristi algoritam

k-srednjih vrijednosti: grupiranje u prvih nekoliko koraka provodi se algoritmom k-srednjih vrijednosti (ili algoritmom k-means++), a onda se tako dobivene srednje vrijednosti koriste kao početne vrijednosti EM-algoritma.

4.3 Primjena na model miješane gustoće

Primijenimo sada EM-algoritam na optimizaciju parametara modela miješane gustoće. U slučaju modela miješane gustoće, latentne varijable modeliraju izvore primjera, odnosno modeliraju koji primjer pripada kojoj komponenti. Očito, kada bismo unaprijed znali koji primjer pripada kojoj komponenti, kao što je to slučaj kod klasifikacije, ne bismo imali potrebe za takvim varijablama i točno bismo znali koja je to komponenta čije parametre ugađamo.

Neka je $\mathbf{z} = (z_1, \ldots, z_K)$ vektor indikatorskih varijabli, takvih da $z_k = 1$ ako je primjer generiran iz grupe $\mathcal{G}_k$, a inače $z_k = 0$. Ovdje pretpostavljamo da je primjer generiran od samo jedne grupe, ali će se postupak u konačnici svesti na izračun vjerojatnosti pripadanja primjera svakoj od grupa. Vektor $\mathbf{z}$ je u stvari slučajan vektor s k međusobno isključivih stanja. Svako stanje odnosno grupa ima određenu apriornu vjerojatnost

$$P(z_k = 1) = \pi_k \,.$$

Budući da vrijedi $\sum_k z_k = 1$ (sve indikatorske varijable osim jedne su jednake nuli), to za vjerojatnost $P(\mathbf{z})$ vrijedi

$$P(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k} \,. \tag{15}$$

Slično možemo izraziti izglednost $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$:

$$p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) = \prod_{k=1}^{K} p(\mathbf{x}|\boldsymbol{\theta}_k)^{z_k} \,. \tag{16}$$

Izglednost $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$ nam kazuje koja je vjerojatnost primjera $\mathbf{x}$, ako pretpostavimo da je primjer generiran komponentom koja je predstavljenom vektorom $\mathbf{z}$. Zajedničku gustoću $p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})$ sada možemo izraziti pomoću (15) i (16) kao

$$p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) = P(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k} \prod_{k=1}^{K} p(\mathbf{x}|\boldsymbol{\theta}_k)^{z_k} = \prod_{k=1}^{K} \pi_k^{z_k} p(\mathbf{x}|\boldsymbol{\theta}_k)^{z_k} \,. \tag{17}$$

Posljednja jednakost vrijedi jer je $z_k = 0$ za svaki k, osim za jedan i to isti k.

Izrazimo sada potpunu log-izglednost, $\ln \mathcal{L}(\boldsymbol{\theta}|\mathcal{D}, \mathcal{Z})$. Raspolažemo skupom neoznačenih primjera $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ i skupom latentnih varijabli $\mathcal{Z} = \{\mathbf{z}^{(i)}\}_{i=1}^N$. Svakom primjeru $\mathbf{x}^{(i)}$ odgovara po jedna latentna varijabla $\mathbf{z}^{(i)}$, budući da različiti primjeri mogu biti generirani od različitih komponenti. Za model (17), potpuna log-izglednost je

$$\begin{aligned} \ln \mathcal{L}(\boldsymbol{\theta}|\mathcal{D}, \mathcal{Z}) &= \ln \prod_{i=1}^{N} p(\mathbf{x}^{(i)}, \mathbf{z}^{(i)}|\boldsymbol{\theta}) = \ln \prod_{i=1}^{N} \prod_{i=1}^{K} \pi_k^{z_k^{(i)}} \, p(\mathbf{x}|\boldsymbol{\theta}_k)^{z_k^{(i)}} \\ &= \sum_{i=1}^{N} \sum_{k=1}^{K} z_k^{(i)} \Big(\ln \pi_k + \ln p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k) \Big) \,. \end{aligned} \tag{18}$$

Usporedimo li izraz (18) s izrazom (9) za nepotpunu log-izglednost, uočavamo da se izrazi razlikuju po tome što su redoslijed sumacije $\sum_k$ i logaritamska funkcija zamijenjeni.

E-korak

Za E-korak algoritma trebamo izraziti očekivanje $\mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$. Dobivamo

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) &= \mathbb{E}_{\mathcal{Z}|\mathcal{D},\boldsymbol{\theta}^{(t)}}[\ln \mathcal{L}(\boldsymbol{\theta}|\mathcal{D},\mathcal{Z})] \\ &= \mathbb{E}_{\mathcal{Z}|\mathcal{D},\boldsymbol{\theta}^{(t)}}\Big[\sum_{i=1}^{N}\sum_{k=1}^{K} z_k^{(i)}\big(\ln \pi_k + \ln p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k)\big)\Big] \\ &= \sum_{i=1}^{N}\sum_{k=1}^{K} \mathbb{E}[z_k^{(i)}|\mathcal{D},\boldsymbol{\theta}^{(t)}]\big(\ln \pi_k + \ln p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k)\big) \end{aligned} \tag{19}$$

gdje smo iskoristili linearnost očekivanja i činjenicu da drugi član umnoška nije slučajna varijabla (jedino $z_k^{(i)}$ je slučajna varijabla). Sada treba uočiti da za slučajnu varijablu $z_k^{(i)}$ vrijedi

$$\mathbb{E}[z_k^{(i)}|\mathcal{D},\boldsymbol{\theta}^{(t)}] = \mathbb{E}[z_k^{(i)}|\mathbf{x}^{(i)},\boldsymbol{\theta}^{(t)}] = P(z_k^{(i)}=1|\mathbf{x}^{(i)},\boldsymbol{\theta}^{(t)})\,. \tag{20}$$

Prva jednakost vrijedi zato što od svih primjera iz $\mathcal{D}$ samo primjer $\mathbf{x}^{(i)}$ uvjetuje varijablu $\mathbf{z}^{(i)}$, a time i varijablu $z_k^{(i)}$. Druga jednakost vrijedi zato što je $z_k^{(i)}$ Bernoullijeva varijabla, za koje općenito vrijedi $\mathbb{E}[z] = P(z=1)$.

Očekivanje latentne varijable jednako je dakle njezinoj aposteriornoj vjerojatnosti. Nju možemo izračunati primjenom Bayesovog pravila:

$$P(z_k^{(i)}|\mathbf{x}^{(i)},\boldsymbol{\theta}^{(t)}) = \frac{p(\mathbf{x}^{(i)}|z_k^{(i)},\boldsymbol{\theta}^{(t)})\pi_k^t}{\sum_{j=1}^{N} p(\mathbf{x}^{(i)}|z_j^{(i)},\boldsymbol{\theta}^{(t)})\pi_j^t} \equiv h_k^{(i)}\,. \tag{21}$$

Ovu veličinu već smo bili izrazili u (8) i nazvali je **odgovornost**. Odgovornost $h_k^{(i)}$ je vjerojatnost da je primjer $x^{(i)}$ generirala komponenta k. Budući da je to vjerojatnost, dakle broj između 0 i 1, granica između grupa je meka, za razliku od čvrste granice kakvu nalazi algoritam k-srednjih vrijednosti.

Uvrštavanjem $h_k^{(i)}$ u (19), za očekivanje potpune log-izglednosti dobivamo

$$\mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \sum_{i=1}^{N}\sum_{k=1}^{K} h_k^{(i)}\Big(\ln \pi_k + \ln p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k)\Big)\,. \tag{22}$$

Radi lakšeg izračuna u nastavku, pribrojnike možemo razdvojiti:

$$\mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \sum_{i=1}^{N}\sum_{k=1}^{K} h_k^{(i)} \ln \pi_k + \sum_{i=1}^{N}\sum_{k=1}^{K} h_k^{(i)} \ln p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k)\,. \tag{23}$$

M-korak

U M-koraku izračunavamo (14), odnosno maksimiziramo (23) kako bismo dobili nove parametre $\boldsymbol{\theta}^{(t+1)}$. Optimume nalazimo analitički, rješavanjem jednadžbe $\nabla_{\boldsymbol{\theta}}\mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = 0$. Primijetite da optimizaciju možemo raditi nezavisno za svaki π_k i za svaki $\boldsymbol{\theta}_k$, budući da se svaki od njih u (23) pojavljuje zasebno u linearnoj kombinaciji.

Za određivanje koeficijenata mješavine, $\pi_k^{(t+1)}$, trebamo riješiti $\nabla_{\pi_k}\mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = 0$. Drugi pribrojnik u (23) je neovisan o π_k, pa ga možemo zanemariti. U obzir moramo uzeti uvjet

$\sum_k \pi_k = 1$, pa za nalaženje uvjetnog ekstrema koristimo metodu Lagrangeovih multiplikatora:

$$\nabla_{\pi_k}\Big(\sum_{i=1}^{N}\sum_{k=1}^{K} h_k^{(i)} \ln \pi_k + \lambda\big(\sum_k \pi_k - 1\big)\Big) = 0$$

gdje je λ Lagrangeov multiplikator. Deriviranjem dobivamo

$$\frac{1}{\pi_k}\sum_{i=1}^{N} h_k^{(i)} + \lambda = 0\,. \tag{24}$$

Ako obje strane jednadžbe pomnožimo s π_k i zatim lijevu stranu sumiramo po svim k i izjednačimo s nulom (što vrijedi budući da, ako su svi pojedinačni pribrojnici jednaki nuli, onda je i njihov zbroj jednak nuli), dobivamo

$$\sum_{k=1}^{K}\Big(\pi_k \frac{1}{\pi_k}\sum_{n=1}^{N} h_k^{(i)} + \pi_k\lambda\Big) = \sum_{i=1}^{N}\sum_{k=1}^{K} h_k^{(i)} + \sum_{k=1}^{N}\pi_k\lambda = N + \lambda = 0$$

pri čemu smo iskoristili $\sum_k h_k^{(i)} = \sum_k P(z_k^{(i)}|\mathbf{x}^{(i)}, \boldsymbol{\theta}) = 1$ (marginalizacija) i $\sum_k \pi_k = 1$. Iz gornje jednadžbe za Lagrangeov multiplikator dobivamo $\lambda = -N$. Uvrštavanjem u (24) konačno dobivamo

$$\pi_k^{(t+1)} = \frac{1}{N}\sum_{i=1}^{N} h_k^{(i)}\,. \tag{25}$$

Usporedimo li ovaj izraz s izrazom za procjenu apriorne vjerojatnosti klase $P(\mathcal{C}_j)$ kod generativnog klasifikatora, uočavamo da je jedina razlika u tome što se umjesto oznaka klase $y_j^{(i)}$, koje su nam ovdje nepoznate, koriste trenutne procjene odgovornosti $h_k^{(i)}$. Drugim riječima, umjesto procjene relativnim frekvencijama, procjenu radimo temeljem težinskog zbroja.

Da bismo odredili parametre pojedinih komponenata, $\boldsymbol{\theta}_k^{(t+1)}$, trebamo riješiti jednadžbu $\nabla_{\boldsymbol{\theta}_k}\mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = 0$. Prvi pribrojnik u (23) ne ovisi o $\boldsymbol{\theta}_k$ i možemo ga zanemariti. Tako dobivamo

$$\nabla_{\boldsymbol{\theta}_k}\sum_{i=1}^{N}\sum_{k=1}^{K} h_k^{(i)} \ln p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k) = \nabla_{\boldsymbol{\theta}_k}\sum_{i=1}^{N} h_k^{(i)} \ln p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k) = 0\,. \tag{26}$$

Rješenje dalje ovisi o tome kako su modelirane komponente mješavine, $p(\mathbf{x}|\boldsymbol{\theta}_k)$.

4.4 Primjena na Gaussovu mješavinu

Za mješavinu univarijatnih (jednodimenzijskih) Gaussovih gustoća, uvrštavanjem $p(x|\boldsymbol{\theta}_k) \sim \mathcal{N}(\mu_k, \sigma_k^2)$ u (26), dobivamo

$$\nabla_{\boldsymbol{\theta}_k}\sum_{i=1}^{N} h_k^{(i)} \ln\Big(\frac{1}{\sqrt{2\pi}\sigma_k}\exp\Big\{-\frac{(x^{(i)}-\mu_k)^2}{2\sigma_k^2}\Big\}\Big) = 0 \tag{27}$$

gdje $\boldsymbol{\theta}_k = (\mu_k, \sigma_k^2)$. Deriviranjem i rješavanjem po μ_k odnosno σ_k^2, dobivamo:

$$\mu_k^{(t+1)} = \frac{\sum_i h_k^{(i)} x^{(i)}}{\sum_i h_k^{(i)}} \tag{28}$$

$$(\sigma^2)_k^{(t+1)} = \frac{\sum_i h_k^{(i)}\big(x^{(i)} - \mu_k^{(t+1)}\big)^2}{\sum_i h_k^{(i)}} \tag{29}$$

Slično, za multivarijatnu (višedimenzijsku) Gaussovu gustoću $p(\mathbf{x}|\boldsymbol{\theta}_k) \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ dobivamo:

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{\sum_i h_k^{(i)} \mathbf{x}^{(i)}}{\sum_i h_k^{(i)}} \tag{30}$$

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{\sum_i h_k^{(i)} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k^{(t+1)})(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k^{(t+1)})^{\mathrm{T}}}{\sum_i h_k^{(i)}} \tag{31}$$

Opet možemo uočiti da su ove jednadžbe analogne onima za ML-procjenitelje generativnog klasifikacijskog modela, s tom razlikom da se umjesto oznaka $y_j^{(i)}$ koriste trenutne procjene odgovornosti $h_k^{(i)}$.

Pseudokod algoritma dan je u nastavku.

Algoritam 4. EM-algoritam za Gaussovu mješavinu

1: **inicijaliziraj** parametre Gaussove mješavine, $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$

2: ponavljaj:

3: **E-korak**: Izračunaj odgovornosti temeljem trenutnih vrijednosti parametara. Za svaki primjer $\mathbf{x}^{(i)} \in \mathcal{D}$ i svaku komponentu $k = 1, \ldots, K$:

$$h_k^{(i)} \leftarrow \frac{p(\mathbf{x}^{(i)}|z_k^{(i)}, \boldsymbol{\theta})\pi_k}{\sum_{j=1}^N p(\mathbf{x}^{(i)}|z_j^{(i)}, \boldsymbol{\theta})\pi_j}$$

4: **M-korak**: Izračunaj procjene parametara temeljem trenutnih odgovornosti. Za svaku komponentu $k = 1, \ldots, K$:

$$\boldsymbol{\mu}_k \leftarrow \frac{\sum_i h_k^{(i)} \mathbf{x}^{(i)}}{\sum_i h_k^{(i)}}$$

$$\boldsymbol{\Sigma}_k \leftarrow \frac{\sum_i h_k^{(i)} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^{\mathrm{T}}}{\sum_i h_k^{(i)}}$$

$$\pi_k \leftarrow \frac{1}{N} \sum_{i=1}^N h_k^{(i)}$$

5: Izračunaj trenutnu vrijednost log-izglednosti

$$\ln \mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = \sum_{i=1}^N \ln \sum_{k=1}^K \pi_k \, p(\mathbf{x}^{(i)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

6: do konvergencije log-izglednosti ili parametara

Usporedimo li EM-algoritam s algoritmom k-srednjih vrijednosti, možemo zaključiti da su algoritmi zapravo vrlo slični. Algoritam k-srednjih vrijednosti provodi čvrsto grupiranje, dok EM-algoritam provodi meko grupiranje. Može se pokazati da je algoritam k-srednjih vrijednosti ustvari poseban slučaj EM-algoritma. Konkretno, ako komponente Gaussove mješavine imaju dijeljenu i izotropnu kovarijacijsku matricu $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$ te ako se vrijednosti $h_k^{(i)}$ zaokružuju na 0 ili 1, dobivamo algoritam k-srednjih vrijednosti.

5 Hijerarhijsko grupiranje

Hijerarhijsko grupiranje, za razliku od particijskog, rezultira hijerarhijom grupa. Hijerarhija grupa može se pregledno prikazati **dendrogramom**:[2] stablom kod kojega listovi odgovaraju primjerima, a vodoravne linije odgovaraju povezivanjima na određenoj udaljenosti. Dendrogram se može presjeći na bilo kojoj željenoj udaljenosti, čime se dobivaju grupe kakve bi se dobile particijskim grupiranjem na toj određenoj udaljenosti. Hijerarhijsko grupiranje može biti **aglomerativno** ili **divizivno**. Aglomerativno grupiranje kreće od grupa koje sadrže svaka po samo jedan primjer i zatim postepeno stapa grupe dok sve primjere ne stopi u jednu grupu. Suprotno, divizivno grupiranje kreće od jedne grupe koju postepeno razdjeljuje. Hijerarhijsko grupiranje, za razliku od algoritma k-srednjih vrijednosti i EM-algoritma, nema nekakvu teorijsku osnovu i zapravo je heuristički postupak.

5.1 Funkcija udaljenosti

Hijerarhijsko grupiranje provodi se temeljem funkcije udaljenosti ili mjere sličnosti, s ciljem da se pronađu grupe primjera koji su najsličniji jedan drugome. **Funkcija udaljenosti** (engl. *distance function*) ili **metrika** je funkcija $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ za koju vrijede sljedeća svojstva:

(i) $d(\mathbf{x}^a, \mathbf{x}^b) \geqslant 0$ (nenegativnost),

(ii) $d(\mathbf{x}^a, \mathbf{x}^b) = 0$ ako i samo ako $\mathbf{x}^a = \mathbf{x}^b$ (strogost),

(iii) $d(\mathbf{x}^a, \mathbf{x}^b) = d(\mathbf{x}^b, \mathbf{x}^a)$ (simetričnost),

(iv) $d(\mathbf{x}^a, \mathbf{x}^b) + d(\mathbf{x}^b, \mathbf{x}^c) \geqslant d(\mathbf{x}^a, \mathbf{x}^c)$ (nejednakost trokuta).

Svojstva (i) i (ii) zajedno nazivaju se pozitivna definitnost. Najčešće korištena je **Minkowskijeva udaljenost**, odnosno Minkowskijev razred metrika:

$$d(\mathbf{x}^a, \mathbf{x}^b) = \Big(\sum_{j=1}^{n} (x_j^a - x_j^b)^p \Big)^{1/p} .$$

Za $p = 1$ dobivamo **L1-udaljenost**,[3] a za $p = 2$ euklidsku udaljenost.

Poopćenje euklidske udaljenosti za općenit slučaj koreliranih značajki s različitim varijancama jest **Mahalanobisova udaljenost**:

$$d^2(\mathbf{x}^a, \mathbf{x}^b) = (\mathbf{x}^a - \mathbf{x}^b)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}^a - \mathbf{x}^b)$$

gdje je $\boldsymbol{\Sigma}$ kovarijacijska matrica. Prednost Mahalonbisove udaljenosti nad euklidskom jest to što ne ovisi o razlikama u rasponima vrijednosti pojedinih dimenzija. Euklidska udaljenost sve dimenzije tretira jednako, što nije dobro ako su rasponi vrijednosti vrlo neujednačeni. U načelu, korištenje euklidske udaljenosti opravdano je samo ako je matrica kovarijacije izotropna, $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}$, tj. ako su značajke nekorelirane i primjeri su jednoliko raspršeni po svim dimenzijama. Mahalanobisova udaljenost degradira na euklidsku za $\boldsymbol{\Sigma} = \mathbf{I}$, tj. za slučaj sferičnih Gaussovih gustoća s jediničnom varijancom.

[2] Od grč. *dendron* – stablo, *gramma* – crtež. Često se u literaturi koristi pogrešan naziv *dendogram*.
[3] Također: *Manhattan distance*, *city block distance*, *taxicab distance*.

5.2 *Mjera sličnosti

Općenitiji pojam od udaljenosti je **mjera sličnosti** (engl. *similarity measure*), odnosno njoj komplementarna **mjera različitosti** (engl. *dissimilarity measure*). Udaljenost se može tumačiti kao geometrijska interpretacija sličnosti, odnosno različitosti. Bitna razlika jest što mjera sličnosti (odnosno mjera različitosti) nije metrika. Mjera sličnosti je funkcija $s : \mathcal{X} \times \mathcal{X} \to [0, 1]$ za koju se obično podrazumijeva da zadovoljava sljedeće:

(i) $s(\mathbf{x}, \mathbf{x}) = 1$,

(ii) $0 \leqslant s(\mathbf{x}^a, \mathbf{x}^b) \leqslant 1$,

(iii) $s(\mathbf{x}^a, \mathbf{x}^b) = s(\mathbf{x}^b, \mathbf{x}^a)$.

Iz navedenih je svojstava očigledno da mjera udaljenosti i mjera sličnosti nisu suprotne u smislu da je jedna komplement druge, već su suprotne u smislu da je jednu moguće preslikati u drugu nekom monotono padajućom funkcijom. Na primjer, za preslikavanje udaljenosti d u sličnost s često se koristi

$$s(\mathbf{x}^a, \mathbf{x}^b) = \frac{1}{1 + d(\mathbf{x}^a, \mathbf{x}^b)} .$$

Obrnuti smjer, pretvorba sličnosti u udaljenosti, nešto je teži zbog uvjeta nejednakosti trokuta. U većini slučajeva, udaljenosti i sličnosti (odnosno različitosti) mogu se koristiti jednako. U nekim slučajevima međutim sličnosti pružaju dodatnu fleksibilnost.

5.3 Hijerarhijsko aglomerativno grupiranje

Algoritam **hijerarhijskog aglomerativnog grupiranja** (engl. *hierarchical agglomerative clustering*, HAC) najčešće je upotrebljavan algoritam hijerarhijskog grupiranja. Algoritam započinje tako da se svaki primjer nalazi u svojoj zasebnoj grupi, a zatim u svakom koraku stapa dvije najbliže grupe, sve dok se ne dosegne unaprijed zadani broj grupa K.

Algoritam 5. Hijerarhijsko aglomerativno grupiranje

inicijaliziraj K, $k \leftarrow N$, $\mathcal{G}_i \leftarrow \{\mathbf{x}^{(i)}\}$ za $i = 1, \ldots, N$
ponavljaj
  $k \leftarrow k - 1$
  $(\mathcal{G}_i, \mathcal{G}_j) \leftarrow \operatorname{argmin}_{\mathcal{G}_a, \mathcal{G}_b} d(\mathcal{G}_a, \mathcal{G}_b)$
  $\mathcal{G}_i \leftarrow \mathcal{G}_i \cup \mathcal{G}_j$
dok $k > K$

Za $K = 1$ algoritam će rezultirati potpunim dendrogramom koji onda možemo naknadno presijecati na željenim udaljenostima.

Uočite da algoritam u svakom koraku pronalazi par najbližih grupa. Udaljenost između grupa tipično se definira na jedan od sljedeća dva načina:

$$d_{min}(\mathcal{G}_i, \mathcal{G}_j) = \min_{\mathbf{x} \in \mathcal{G}_i, \mathbf{x}' \in \mathcal{G}_j} d(\mathbf{x}, \mathbf{x}') \tag{32}$$

$$d_{max}(\mathcal{G}_i, \mathcal{G}_j) = \max_{\mathbf{x} \in \mathcal{G}_i, \mathbf{x}' \in \mathcal{G}_j} d(\mathbf{x}, \mathbf{x}') \tag{33}$$

Mjera d_{min} udaljenost između dviju grupa definira kao najmanju udaljenost između pojedinačnih primjera u tim grupama. Takvo grupiranje nazivamo grupiranje temeljem **jednostruke povezanosti** (engl. *single-link clustering*).[4] Tomu suprotna je mjera d_{max}, koja udaljenost između dviju grupa definira kao najveću udaljenost između pojedinačnih primjera u tim grupama. Takvo grupiranje nazivamo grupiranje **potpunom povezanošću** (engl. *complete-link clustering*).[5] Ako su grupe kompaktne i prirodno dobro odvojene, onda ove dvije udaljenosti ne daju značajno različite rezultate. Međutim, ako to nije slučaj, razlike mogu biti značajne. Tada jednostruko povezivanje rezultira dugim, ulančanim grupama, dok potpuno povezivanje rezultira manjim, zbijenijim grupama.

Rezultati ove dvije vrste grupiranja imaju lijepo tumačenje u teoriji grafova. Stapanje dviju grupa, $\mathcal{G}_i$ i $\mathcal{G}_j$, odgovara uvođenju brida između odgovarajućih primjera u tim dvjema grupama. Kod jednostrukog povezivanja, to su dva najbliža primjera iz svake grupe. Budući da se bridovi uvijek uvode između primjera različitih grupa, a nikad između primjera iz iste grupe, rezultirajući graf je stablo (tj. nema ciklusa). Ako $K = 1$, algoritam HAC generira **minimalno razapinjuće stablo** (engl. *minimal spanning tree*) (stablo sa stazom između svaka dva brida kod kojega je ukupan težinski zbroj bridova minimalan). Suprotno, kod potpunog povezivanja, stapanje dviju grupa odgovara uvođenju bridova između svih parova primjera, pa algoritam HAC rezultira **potpuno povezanim grafom**.

Jednostruko i potpuno povezivanje su dva krajnja slučaja izračuna udaljenosti između grupa i dosta su osjetljivi na šum. Kompromisno rješenje je grupiranje temeljem **prosječne povezanosti** (engl. *average-linkage clustering*):

$$d_{avg}(\mathcal{G}_i, \mathcal{G}_j) = \frac{1}{N_i N_j} \sum_{\mathbf{x} \in \mathcal{G}_i} \sum_{\mathbf{x}' \in \mathcal{G}_j} d(\mathbf{x}, \mathbf{x}') \tag{34}$$

gdje je N_i odnosno N_j broj primjera u grupi $\mathcal{G}_i$ odnosno $\mathcal{G}_j$.

Alternativu predstavlja mjera koja udaljenost između grupa definira kao udaljenost između njihovih centroida:

$$d_{mean}(\mathcal{G}_i, \mathcal{G}_j) = \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\| \,. \tag{35}$$

To je računalno najjednostavnija mjera, ali je ograničena na udaljenosti definirane u vektorskome prostoru. Kada udaljenosti nije moguće izračunati u vektorskome prostoru, u prednosti je mjera d_{avg}, koja je primjenjiva na bilo kakvu mjeru sličnosti.

Razmotrimo sada složenost algoritma hijerarhijskog aglomerativnog grupiranja. Algoritam treba izračunati udaljenost između svih $\binom{N}{2}$ parova primjera. Taj se izračun može učiniti jednokratno i pohraniti u tzv. **matricu udaljenosti** (odnosno matricu sličnosti) dimenzija $N \times N$. Prostorna složenost zbog toga je $\mathcal{O}(N^2)$. Tijekom grupiranja, pri svakom stapanju grupa, ova se matrica ažurira i smanjuje (izbacuje se jedan redak i jedan stupac), dok naposlijetku ne dosegne dimenziju $K \times K$. Nakon svakog stapanja potrebno je (ovisno o vrsti povezivanja) izračunati udaljenosti između stopljene grupe i svih preostalih grupa, što iziskuje $\mathcal{O}(N)$ izračuna mjere udaljenosti (ako je riječ o vektorskom prostoru dimenzije n, složenost se može preciznije izraziti kao $\mathcal{O}(nN)$). Nalaženje minimalno ili maksimalno udaljenog para grupa iziskuje $\mathcal{O}(N^2)$ usporedbi u svakom od ukupno $N - K$ koraka algoritma, pa je ukupna vremenska složenost algoritma $\mathcal{O}((N - K)N^2)$. Tipično je $K \ll N$ (za izgradnju potpunog dendrograma vrijedi $K = 1$), pa je vremenska složenost zapravo $\mathcal{O}(N^3)$. Kubna vremenska složenost očit je nedostatak ovog algoritma (za razliku od npr. algoritma k-srednjih vrijednosti čija je složenost u svim parametrima

[4] Također: grupiranje metodom najbližeg susjeda (engl. *nearest-neighbor algorithm*).

[5] Također: grupiranje metodom najdaljeg susjeda (engl. *farthest-neighbor algorithm*).

linearna). Postoje implementacije koje umjesto matrice udaljenosti koriste neke druge strukture podataka (npr. prioritetni red), čime je vremensku složenost moguće smanjiti na $\mathcal{O}(N^2 \log N)$. Ipak, kod vrlo velikog broja primjera, kvadratna protorna složenost može već biti nepremostiv problem.

6 Predgrupiranje

Kod grupiranja velikog broja primjera, velik nedostatak hijerarhijskog grupiranja predstavlja kvadratna prostorna složenost algoritma. Kod vrlo velikih količina podataka, čak je i linearna složenost algoritama k-srednjih vrijednosti ili EM-algoritma problematična.

Moguće rješenje jest provođenje **predgrupiranja** (**predstrukturiranja**), odnosno razdjeljivanja prostora primjera na particije temeljem nekog računalno jednostavnije postupka. Nakon što se primjeri razdijele u particije, grupiranje se može provesti nekom od složenijih metoda unutar svake takve particije zasebno. Na primjer, ako se N primjera razdijeli u particije s najviše po M primjera, $M \ll N$, prostorna složenost aglomerativnog hijerarhijskog grupiranja reducira se na $\mathcal{O}(M^2)$, a vremenska na $\mathcal{O}((M-K)NM)$, odnosno $\mathcal{O}(NM^2)$ za izgradnju potpunoga dendrograma. Daljnja ubrzanja moguće je ostvariti paralelizacijom.

Primjer takvog postupka je **algoritam krošnji** (engl. *canopy algorithm*). Algoritam prostor primjera razdjeljuje na preklapajuće particije. Za particioniranje se koriste dva praga, T_1 i T_2, gdje $T_1 > T_2$. Algoritam kreće s listom svih primjera te slučajno (ili po nekom odabranome kriteriju) odabire jedan primjer $\mathbf{x}$ kao središte prve krošnje. Zatim za sve preostale primjere $\mathbf{x}'$ računa udaljenost $d(\mathbf{x}, \mathbf{x}')$. Taj izračun može biti aproksimativan; ideja je da ne bude računalno zahtjevan. Primjeri za koje $d(\mathbf{x}, \mathbf{x}') < T_1$ smještaju se pod istu krošnju kao i primjer $\mathbf{x}$, dok se primjeri za koje $d(\mathbf{x}, \mathbf{x}') < T_2$ dodatno uklanjaju iz liste i oni više ne mogu biti izabrani za središta novih krošnji. Postupak se zatim ponavlja za preostale primjere iz liste, sve dok se lista ne isprazni. Uočite da su primjeri koji se nađu u pojasu između T_1 i T_2 pridjeljeni nekoj krošnji, ali se također mogu koristiti kao središta novih krošnji. To rezultira preklapajućim tj. mekim particijama, što algoritam čini robusnijim na šum u podatcima i na pogreške zbog eventualnog korištenja aproksimacije funkcije udaljenosti.

7 Provjera grupa

Kod svih je metoda grupiranja broj grupa, odnosno parametar K, potrebno odrediti unaprijed. (Kod hijerarhijskog grupiranja može se odabrati $K = 1$ i izgraditi čitav dendrogram, no onda je naknadno potrebno odlučiti gdje napraviti njegovo presijecanje.) Odabir broja grupa jedan je od glavnih problema kod grupiranja. Idealno, broj grupa odgovarat će broju "prirodnih grupa" u skupu podataka, no on nam je najčešće nepoznat.

Parametar K je zapravo **hiperparametar**: parametar složenosti modela koji se ne ugađa učenjem. Taj parametar ne možemo optimirati na način da minimiziramo kriterijsku funkciju (odnosno, kod probabilistički modela, da maksimiziramo log-izglednost). Razlog je isti kao što, primjerice, parametar K kod algoritma k-NN ne možemo optimirati na temelju empirijske pogreške učenja. Kriterijska funkcija grupiranja monotono opada s porastom K te doseže svoj minimum za $K = N$, ali to očito dovodi do prenaučenosti modela. Odabrati optimalan K znači dakle odabrati optimalnu složenost modela, odnosno onu složenost kod koje je sposobnost generalizacije najveća. Problem je analogan problemu provjere modela

kod nadziranog učenje, pa ga nazivamo **provjera (validacija) grupa** (engl. *cluster validation*). Poteškoću predstavlja činjenica da kod grupiranja u pravilu primjeri nisu označeni, pa nije moguće primijeniti metodu unakrsne provjere.

Postoje razni načini kako se može napraviti provjera grupa:

- Kod nekih je primjena, kao npr. kvantizacije boja, broj grupa K unaprijed poznat;
- Broj grupa može se odrediti tako da se, primijenom neke od tehnika redukcije dimenzionalnosti, podatci prikažu dvodimenzijskom prostoru, pa se temeljem toga pokuša odrediti prirodan broj grupa;
- Iterativan pristup: unaprijed definiramo maksimalan dozvoljeni iznos kriterijske funkcije (odnosno minimalni dozvoljeni iznos log-izglednosti) i K postepeno povećavamo sve dok kriterijska funkcija ne padne ispod te vrijednosti;
- Kod nekih je primjena rezultat grupiranja moguće provjeriti ručno te utvrditi ima li grupiranje sa zadanim K smisla;
- Možemo označiti manji podskup primjera, zatim grupirati zajedno označene i neoznačene primjere, a onda napraviti provjeru samo na označenim primjerima. Kao mjeru pogreške treba koristiti neku mjeru koja je temeljena na izračunu gubitaka, npr. **Randov indeks** (engl. *rand index*), mjera **normalizirane uzajamne informacije** (engl. *normalized mutual information*, NMI) ili mjera F_1. U ovakvom kontekstu, treba razlikovati vanjski i unutarnji kriterij grupiranja: mjera F_1 predstavlja **vanjski kriterij**, dok kriterij koji se koristi za samo grupiranje (kriterijska funkcija, log-izglednost ili udaljenost/sličnost) predstavlja **unutarnji kriterij**. Grupiranje se provodi prema unutarnjem kriteriju, za koji pretpostavljamo da je dobro usklađen s vanjskim kriterijem;
- Grafički prikažemo ovisnost kriterijske funkcije o parametru K i tražimo "koljeno" krivulje. S porastom vrijednosti K vrijednost kriterijske funkcije će padati. Kod dovoljno velikog K algoritam će početi razdjeljivati prirodne grupe, pa daljnje smanjenje kriterijske funkcije više neće biti tako značajno. (Ako je algoritam nedeterministički, npr. k-means sa slučajno odabranim početnim središtima, za svaki izbor vrijednosti K treba napraviti više mjerenja.) Slično, kod hijerarhijskog grupiranja možemo prikazati broj dobivenih grupa u ovisnosti o udaljenosti; na mjestima gdje taj broj stagnira (odnosno gdje su razlike između razina dendrograma velike) grupiranje dobro odražava prirodne grupe koje postoje u podatcima;
- Minimiziramo kriterij koji kombinira kriterijsku funkciju i složenost modela te na neki način kažnjavamo modele s prevelikim brojem grupa. Općenit oblik tog kriterija je

$$K^* = \operatorname*{argmin}_K \big(J(K) + \lambda K\big) \tag{36}$$

  gdje je $J(K)$ vrijednost kriterijske funkcije za model s K grupa, a λ težinski faktor. Veće vrijednosti faktora λ favoriziraju rješenja s manjim brojem grupa. Za $\lambda = 0$ povećanje broja grupa se ne kažnjava i optimalan broj grupa tada je $K^* = N$.

  Odabir parametra λ može se temeljiti na našem iskustvu stečenom na grupiranju sličnih skupova podataka. Alternativa jest da koristimo neki teorijski utemeljen kriterij, poput Akaikeova informacijskog kriterija (AIC):

$$K^* = \operatorname*{argmin}_K \big(-2 \ln \mathcal{L}(K) + 2q(K)\big) \tag{37}$$

gdje je $-\ln \mathcal{L}(K)$ negativna log-izglednost podataka za K grupa, a $q(K)$ je broj parametara modela s K grupa. Konkretno, u slučaju grupiranja algoritmom k-srednjih vrijednosti, kriterij AIC svodi se na:

$$K^* = \operatorname*{argmin}_K \big(J(K) + 2nK \big) \tag{38}$$

gdje je $J(K)$ vrijednost kriterijske funkcije za model s K grupa, a n je dimenzija prostora primjera. Izraz (38) može se izvesti iz (37) uzevši u obzir činjenicu da za model s K grupa vrijedi $q(K) = nK$ i $\ln \mathcal{L}(K) \propto -\frac{1}{2} J(K)$. Posljednje slijedi iz pretpostavke da su podatci generirani Gaussovom mješavinom s uniformnim koeficijentima mješavina te dijeljenom izotropnom kovarijacijskom matricom.