20. Grupiranje II

Strojno učenje 1, UNIZG FER, ak. god. 2021./2022.

Jan Šnajder, predavanja, v2.1

Prošli smo puta počeli pričati o **nenadziranom strojnom učenju**, dakle učenju iz podataka koji nemaju oznaku y. Rekli smo da postoje razni pristupi nenadziranom strojnom učenju, ali da se najčešće koristi **grupiranje**, kod kojega primjere koji su međusobno slični želimo smjestiti u iste grupe, a oni koji su različiti u različite grupe. Spomenuli smo i da pristupa grupiranju ima raznih: grupiranje može biti **particijsko** ili **hijerarhijsko**, te može biti **čvrsto** ili **meko**. Zatim smo pričali o algoritmima K-sredina i K-medoida, koji su oba algoritmi za čvrsto particijsko grupiranje.

Danas nastavljamo s grupiranjem, no bavit ćemo se dvama algoritmima koji ne rade particijsko čvrsto grupiranje. Prvo ćemo razmotriti algoritam koji nazivamo **model Gaussove mješavine (GMM)**, koji radi **meko particijsko grupiranje**. Kao što ćemo vidjeti, taj je algoritam zapravo probabilističko poopćenje algoritma K-sredina, odnosno algoritam K-sredina možemo shvatiti kao poseban slučaj modela Gaussovih mješavina. Također ćemo vidjeti da je model Gaussove mješavine zapravo generativan model s latentnim varijablama, za čije će nam treniranje trebati nov algoritam, **algoritam maksimizacije očekivanja** ili **EM-algoritam**. Na kraju ćemo razmotriti jedan posve drugačiji, a vrlo jednostavan algoritam: **hijerarhijsko aglomerativno grupiranje**, koji služi za **čvrsto hijerarhijsko grupiranje**, dakle algoritam koji iz podataka inducira hijerarhiju grupa. 1

1 Model Gaussove mješavine

Prošli put bavili smo se algoritmima K-sredina i K-medoida. To su algoritmi čvrstog particijskog grupiranja. Međutim, u nekim problemima primjere ne želimo grupirati u čvrste grupe. Naime, ponekad primjeri legitimno mogu pripadati u više grupa, tj. može postojati preklapanje između grupa. Npr., ako radimo grupiranje ljudi u grupe po crtama ličnosti, gdje jedna grupa odgovara jednoj crti ličnosti, onda će svaka osoba imati kombinaciju više crta ličnosti, što znači da treba pripadati više grupa. Slično ako radimo grupiranje riječi po značenjima, gdje jedna grupa odgovara jednom značenju, onda jedna riječ može imati više značenja, pa može istovremeno pripadati u vi[e grupa. Ovdje se dakle radi se tome da primjer pripada u više grupa istovemeno. Druga je mogućnost da svaki primjer ipak pripada uvijek u samo jednu grupu, ali da mi ne znamo koju, pa bismo željeli nekako modelirati tu neizvjesnost i to modeliramo tako da za svaku grupu definiramo vjerojatnost da joj primjer pripada. U oba slučaja – bilo da primjer pripada u više grupa istovremeno, ili da pripada samo jednoj, ali ne znamo točno kojoj – radi se o **mekom grupiranju**.

Konkretno, mi ćemo razmotriti **probabilističko grupiranje**, gdje svaki primjer svakoj grupi pripada s nekom vjerojatnošću. Alternativa bi bila **neizrazito grupiranje** (engl. *fuzzy clustering*), gdje svaki primjer svakoj grupi pripada s nekom mjerom pripadnosti. Vratit ćemo se kasnije na ovu razliku. 2

Najjednostavniji i najčešće korišten algoritam probabilističkog grupiranja je **model Gausove mješavine** (engl. *Gaussian mixture model*, GMM), koji se također naziva **mješavina Gaussova** (engl. *mixture of Gaussians*, MoG). Taj model tipično treniramo optimizacijskim algoritmom koji se naziva **algoritam maksimizacije očekivanja** (engl. *expectation maximization algorithm,*

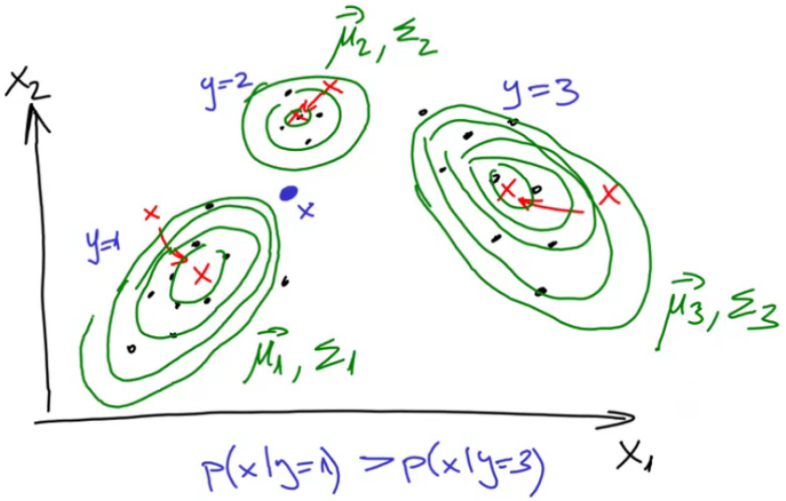
EM-algorithm), koji ćemo isto pogledati. Jednostavnosti radi, u nastavku ćemo umjesto o "modelu GMM učenom EM-algoritmom" jednostavno govoriti o "algoritmu GMM".

Algoritam GMM može se shvatiti kao poopćenje algoritma K-sredina. Razmotrimo primjer.

▸ **PRIMJER**

Grupiramo dvodimenzijske primjere u $K = 3$ grupe. Za razliku od algoritma K-sredina, kod algoritma GMM osim triju centroida želimo za svaku grupu procijeniti i njezinu **kovarijacijsku matricu**. Drugim riječima, svaka od triju grupa će biti jedna Gaussova gustoća vjerojatnosti $p(\mathbf{x}|\boldsymbol{\mu}, \Sigma)$. Ta Gaussova gustoća vjerojatnosti definira vjerojatnost da primjer $\mathbf{x}$ pripada dotičnoj grupi (preciznije, ta gustoća vjerojatnosti definira distribuciju vjerojatnosti primjera iz dotične grupe).

Dakle, ako imamo tri grupe, morat ćemo procijeniti parametre $\boldsymbol{\mu}_1, \Sigma_1$, $\boldsymbol{\mu}_2, \Sigma_2$, $\boldsymbol{\mu}_3$ i Σ_3. Radimo iterativno, kao i kod algoritma K-sredina: krenuvši od tri slučajno odabrana centroida (premda i ovdje možemo na pametniji način inicijalizirati početna središta), polako ćemo ugađati centroide i kovarijacijske matrice svake od triju grupa, dok ne dođemo do stacionarnog stanja, tj. dok postupak ne konvergira. Početne kovarijacijske matrice možemo postaviti na jedinične matrice. Kod algoritma K-sredina primjere smo stvrstavali u grupu čijem su centroidu najbliži, i zatim smo ponovno izračunavali centroide. Slično radimo kod algoritma GMM: za svaki primjer izračunavamo koja je vjerojatnost da primjer pripada dotičnoj grupi, i zatim ponovno izračunavamo centroide kao težinski prosjek primjera (u ovisnosti o vjerojatnostima da primjer pripada grupi). To ponavljamo do konvergencije. Na kraju dobivamo mješavinu od tri Gaussove gustoće vjerojatnosti. Na primjer, ako je ulazni prostor dvodimenzijski (imamo samo značajke x_1 i x_2), grupiranje bi moglo izgledati ovako:



Crvenom je označeno kretanje centroida kroz iteracije algoritma od početnog položaja do krajnjeg položaja centroida. Prema izokonturama ovih gustoća vidimo da postoji pozitivna korelacija u šumu između značajki x_1 i x_2 u grupi $y = 1$ (elipse izokontura su pozitivno nagnute) te negativna korelacija u šumu u grupi $y = 3$ (elipse izokontura su negativno nagnute). U grupi $y = 2$ nemamo korelacije u šumu (izokonture su kružnice). Za svaku od triju grupa y možemo zatim izračunati koja je vjerojatnost da je dotična grupa generirala primjer $\mathbf{x}$, tj. možemo izračunati **izglednost grupe**. U ovom slučaju, vrijedilo bi $p(\mathbf{x}|y = 1) > p(\mathbf{x}|y = 3)$ jer je primjer $\mathbf{x}$ bliži grupi $y = 1$ nego grupi $y = 3$, tj. "više je u gustoći vjerojatnosti" prve grupe nego treće grupe. Također ćemo iz ovoga, primjenom Bayesovog pravila, lako moći izračunati i obrnuto (a to je ono što nas zapravo zanima): za svaku grupu možemo izračunati kolika je **vjerojatnost grupe** za dani primjer $\mathbf{x}$, tj. vjerojatnost $p(y|\mathbf{x})$.

Drugi pogled na probabilističko grupiranje modelom Gaussove mješavine jest taj da je naš zadatak zapravo procijeniti gustoću vjerojatnosti $p(\mathbf{x})$. Za tu gustoću vjerojatnosti pretpostavljamo da se interno sastoji od K komponenti – K Gaussovih gustoća vjerojatnosti. Nakon

što procijenimo parametre tako definirane gustoće vjerojatnosti $p(\mathbf{x})$, dobit ćemo zapravo meko grupiranje, jer ćemo za svaki primjer moći izračunati s kojom vjerojatnošću pripada kojoj od tih K komponenti.

Pogledajmo odmah kako izgleda konačan algoritam, a onda ćemo ga izvesti. Najprije, prisjetimo se kako izgleda algoritam K-sredina:

▸ **Algoritam K-sredina**

1: **inicijaliziraj** centroide $\boldsymbol{\mu}_k$, $k = 1, \ldots, K$
2: **ponavljaj**
3: za svaki $\mathbf{x}^{(i)} \in \mathcal{D}$
4: $b_k^{(i)} \leftarrow \begin{cases} 1 & \text{ako } k = \operatorname{argmin}_j \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_j\| \\ 0 & \text{inače} \end{cases}$
5: za svaki $\boldsymbol{\mu}_k$, $k = 1, \ldots, K$
6: $\boldsymbol{\mu}_k \leftarrow \sum_{i=1}^N b_k^{(i)} \mathbf{x}^{(i)} / \sum_{i=1}^N b_k^{(i)}$
7: **dok** $\boldsymbol{\mu}_k$ ne konvergiraju

Da je ovdje riječ o algoritmu čvrstog grupiranja vidi se po tome što je pripadanje primjera grupi modelirano binarnom indikatorskom varijablom $b_k^{(i)}$, tj. $b_k^{(i)} \in \{0, 1\}$. Drugim riječima, primjer $\mathbf{x}^{(i)}$ ili pripada ili ne pripada grupi k. Prisjetimo se: algoritam K-sredina u svakoj iteraciji obavlja dva koraka: u prvom koraku primjere dodijeljuje grupama s najbližim centroidima, a u drugom koraku izračunava nove centroide grupa.

Očekivano, ključna promjena kod algoritma GMM bit će ta da primjer grupi pripada s nekom vjerojatnošću. Tu ćemo vjerojatnost označiti sa $h_k^{(i)} \in [0, 1]$. Vjerojatnost $h_k^{(i)}$ naziva se **odgovornost** – vjerojatnost da primjer $\mathbf{x}^{(i)}$ pripada grupi k. Evo kako izgleda algoritam GMM:

▸ **Algoritam GMM (model GMM + EM-algoritam)**

inicijaliziraj parametre $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$
ponavljaj do konvergencije log-izglednosti ili parametara
E-korak:
Za svaki primjer $\mathbf{x}^{(i)} \in \mathcal{D}$ i svaku komponentu $k = 1, \ldots, K$:

$$h_k^{(i)} \leftarrow \frac{p(\mathbf{x}^{(i)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\pi_k}{\sum_{j=1}^K p(\mathbf{x}^{(i)}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\pi_j}$$

M-korak:
Za svaku komponentu $k = 1, \ldots, K$:

$$\boldsymbol{\mu}_k \leftarrow \frac{\sum_i h_k^{(i)} \mathbf{x}^{(i)}}{\sum_i h_k^{(i)}}, \quad \boldsymbol{\Sigma}_k \leftarrow \frac{\sum_i h_k^{(i)} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^{\mathrm{T}}}{\sum_i h_k^{(i)}}, \quad \pi_k \leftarrow \frac{1}{N} \sum_{i=1}^N h_k^{(i)}$$

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

Primijetite najprije da algoritam K-sredina i algoritam GMM imaju istu strukturu: oba su algoritma iterativna i oba algoritma u svakoj iteraciji provode dva koraka. U prvome koraku primjeri se dodijeljuju grupama, a u drugome koraku ažuriraju se parametri svake grupe. No, razlika je u tome što kod algoritma K-sredina primjeri čvrsto pripadaju grupama (binarna varijabla $b_k^{(i)}$), dok kod algoritma GMM primjeri grupama pripadaju "mekano", s nekom vjerojatnošću (odgovornost $h_k^{(i)}$). Konkretno, u prvom koraku algoritma (tzv. E-korak), primjeri se dodijeljuju grupama na "mekan" način, tako da se izračunava odgovornost $h_k^{(i)}$, tj. za svaki se primjer $\mathbf{x}^{(i)}$ izračunava vjerojatnost da primjer pripada grupi k. U drugom koraku (tzv. M-korak) računaju

se novi parametri grupa, na temelju trenutačne (meke) dodijele primjera grupama. Najbolje se to može uočiti kod izračuna centroida grupa μ_k, gdje vidimo da se centroid za svaku grupu k izračunava tako da se vektori primjera zbrajaju težinski, u ovisnosti o tome kolika je vjerojatnost $h_k^{(i)}$ da primjer $\mathbf{x}^{(i)}$ pripada dotičnoj grupi k. Suprotno tomu, algoritam K-sredina jednostavno računao centroid grupe tako da uprosječi sve vektore primjera koji toj grupi pripadaju, bez ikakvog množenja primjera težinama.

U nastavku ćemo, korak po korak, izvesti algoritam GMM. Premda je osnovna ideja algoritma jednostavna, i zapravo već smo ju opisali, izvod algoritma iziskuje nešto matematičke artiljerije, pa budite strpljivi. Idemo redom. Pogledajmo najprije model.

1.1 Model miješane gustoće

Model Gaussove mješavine (GMM) zapravo je samo jedan specifičan slučaj **modela miješane gustoće** (engl. *mixture model*). Model miješane gustoće generativni je model, koji modelira gustoću $p(\mathbf{x})$ kao linearnu kombinaciju K komponenti. Krećemo od pretpostavke da postoji zajednička vjerojatnost $p(\mathbf{x}, y)$, gdje je y oznaka grupe kojoj primjer pripada, $y = \{1, \ldots, K\}$. Dakle, mi zapravo pretpostavljamo da primjeri $\mathbf{x}$ pripadaju grupama y, pa načelno možemo govoriti o zajedničkoj vjerojatnosti $p(\mathbf{x}, y)$. Međutim, budući da mi ovdje radimo nenadzirano učenje, oznaka y nam nije dostupna. To znači da nam vjerojatnost $p(\mathbf{x}, y)$ nije poznata, već nam je poznata samo vjerojatnost $p(\mathbf{x})$. Formalno, ovu drugu možemo dobiti iz ove prve tako da zajedničku vjerojatnost marginaliziramo upravo po varijabli y, koristeći pravilo zbroja vjerojatnosti:

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(\mathbf{x}, y = k)$$

Primijenimo li još pravilo umnoška vjerojatnosti, onda model miješane gustoće onda možemo napisati kao linearnu kombinaciju od K **komponenti**:

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(\mathbf{x}, y = k) = \sum_{k=1}^{K} \underbrace{P(y = k)}_{\pi_k} p(\mathbf{x}|y = k) = \sum_{k=1}^{K} \pi_k \, p(\mathbf{x}|\boldsymbol{\theta}_k)$$

gdje smo za apriornu vjerojatnost grupe $p(y = k)$ uveli oznaku π_k, koju nazivamo **koeficijent mješavine**. Očito, $\sum_k \pi_k = 1$, jer su to zapravo vjerojatnosti kategoričke slučajne varijable. Vjerojatnosti $p(\mathbf{x}|\boldsymbol{\theta}_k)$, koje su zapravo izglednosti grupa, nazivamo **gustoće komponenti**.

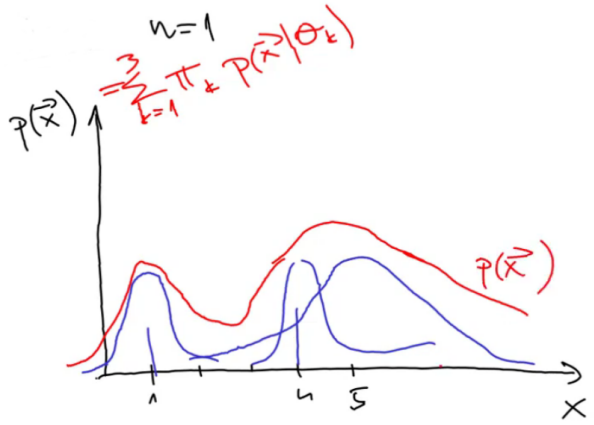
Ovo što smo napisali je općenit model miješane gustoće. Konkretno, kod GMM, gustoće komponenti su Gaussove gustoće vjerojatnosti sa srednjim vektorom $\boldsymbol{\mu}_k$ i kovarijacijskom martricom Σ_k, pa dakle imamo:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k)$$

Pogledajmo primjer.

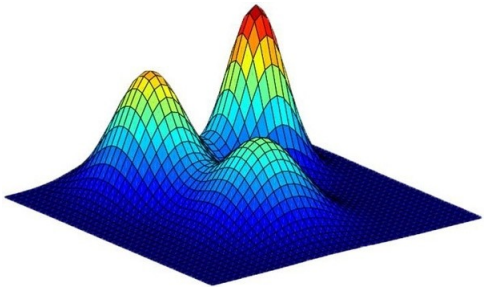
▸ **PRIMJER**

Razmotrimo kako bi izgledao Gaussov model miješane gustoće $p(x)$ za jednodimenzijski ulazni prostor i za $K = 3$ grupe. Neka $\mu_1 = 1, \mu_2 = 4, \mu_3 = 5$, $\sigma_1 = \sigma_2 = 1, \sigma_3 = 6$, $\pi_1 = \pi_2 = 0.2$ i $\pi_3 = 0.6$. Gustoća vjerojatnosti $p(\mathbf{x})$ onda izgleda (vrlo otprilike) ovako:



Gustoća vjerojatnosti $p(x)$ (ovdje označena crvenom) dobiva se zbrajanjem svih triju gustoća komponenti, pomnoženih s njima pridruženim koeficijentom mješavine.

U ovom jednostavnom primjeru zadržali smo se na jednodimenzijskome ulaznom prostoru, pa smo za gustoće komponenti koristili univarijatne Gussove gustoće vjerojatnosti. No, naravno, u stvarnosti će primjeri $\mathbf{x}$ imati više od jedne značajke, tj. ulazni će prostor biti višedimenzijski, i tada ćemo za gustoće komponenti koristiti multivarijatnu Gaussovu gustoću vjerojatnosti. Na primjer, za dvodimenzijski ulazni prostor koristili bismo **bivarijatnu Gaussovu gustoću vjerojatnosti**:



Ovdje imamo tri bivarijatne gustoće vjerojatnosti, dakle modeliramo $K = 3$ grupe.

GMM je **generativni model**, budući da marginalnu gustoću vjerojatnosti $p(\mathbf{x})$ definira preko zajedničke gustoće vjerojatnosti $p(\mathbf{x}, y)$. Kao i svaki generativni model, tako i GMM ima i svoju **generativnu priču**: skup podata je generiran tako da je prvo slučajno (po kategoričkoj distribuciji $P(y)$) odabrana jedna komponenta k, a onda je po njoj odgovarajućoj gustoći vjerojatnosti $p(\mathbf{x}|\boldsymbol{\theta}_k)$ uzorkovan primjer $\mathbf{x}$.

Međutim, u praksi, budući da mi želimo grupirati podatke, ono što nas zapravo zanima nije toliko sama gustoća $p(\mathbf{x})$, niti uzorkovanje primjera iz te gustoće, koliko **vjerojatnost da primjer pripada grupi** k, tj. vjerojatnost $P(y = k|\mathbf{x})$, za koju smo već rekli da se naziva **odgovornost**. Ako znamo tu vjerojatnost za svaki primjer, onda zapravo imamo meko grupiranje. Srećom, budući da se ovdje radi o generativnom modelu, tu vjerojatnost možemo lako izraziti iz vjerojatnosti koje smo već definirali, i to pomoću Bayesovog pravila. Evo kako:

$$P(y = k|\mathbf{x}^{(i)}) = \frac{P(y = k)p(\mathbf{x}^{(i)}|y = k)}{p(\mathbf{x}^{(i)})} = \frac{P(y = k)p(\mathbf{x}^{(i)}|y = k)}{\sum_j P(y = j)p(\mathbf{x}^{(i)}|y = j)}$$
$$= \frac{\pi_k p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k)}{\sum_j \pi_j p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_j)} = h_k^{(i)}$$

Vjerojatnost $P(y = k|\mathbf{x}^{(i)})$ je vjerojatnost da je grupa k odgovorna za nastanak (generiranje) primjera $\mathbf{x}$. I to je upravo ono što nas kod mekog grupiranja zanima!

Ovime smo definirali naš model. Sastoji se, dakle, od koeficijenata mješavine i od gustoće komponenata, kojih imamo onoliko koliko imamo grupa. Iz toga onda možemo izračunati odgovornosti grupa za svaki primjer. Pogledajmo sada koji su **parametri** modela koje moramo procijeniti. Parametri su:

$$\boldsymbol{\theta} = \{\pi_k, \underbrace{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k}_{\boldsymbol{\theta}_k}\}_{k=1}^{K}$$

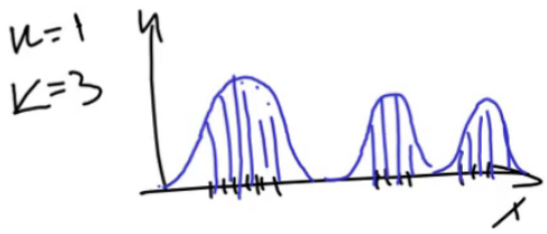
gdje su π_k koeficijenti mješavine a $\boldsymbol{\theta}_k$ su parametri gustoće komponenti (srednja vrijednosti i kovarijacijska matrica). Koliko je to ukupno parametara? Ukupno ih je $K \cdot \frac{n}{2}(n+1)+nK+K-1$.

1.2 Log-izglednost modela miješane gustoće

Kako bismo odredili parametre GMM-a, odnosno kako bismo naučili model na temelju podataka? Prisjetimo se: raspolažemo skupom **neoznačenih primjera** $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$. Kao i kod svih generativnih modela, učenje modela svodi se na statističku procjenu parametara iz podataka. Najlakše bi bilo da parametre probamo procijeniti sa MLE. Kada koristimo MLE za procjenu parametara, mi zapravo namještamo K komponenti tako da maksimiziramo izglednost parametara, odnosno, ekvivalentno, maksimiziramo vjerojatnost podataka pod modelom. To ilustrira sljedeći primjer.

▸ **PRIMJER**

Razmotrimo slučaj s jednodimenzijskim ulaznim prostorom i $K = 3$. MLE zapravo namješta tri Gaussove gustoće (određuje parametre srednje vrijednosti i kovarijacijske matrice) tako da primjeri "pokupe" što više gustoće vjerojatnosti:



Naime, uz tako namještene parametre izglednost parametara (tj. vjerojatnost primjera) biti će za zadani skup primjera najveća moguća.

Sada kada smo se toga prisjetili, primijenimo uobičajeni recept za izvođenje procjenitelja parametara probabilističkih modela, isti onu koju smo već bili primijenili za linearnu regresiju i logističku regresiju. Napišimo najprije funkciju log-izglednosti parametara na (u ovom slučaju neoznačenom) skupu primjera $\mathcal{D}$:

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

Budući da tražimo parametre koji maksimiziraju izglednost, mogli bismo sada pokušati derivirati ovaj izraz po parametrima $\boldsymbol{\theta}$, odnosno izračunati $\nabla_{\boldsymbol{\theta}} \ln \mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = 0$, i na taj način pronaći maksimizator funkcije u zatvorenoj formi. Nažalost, to neće funkcionirati. Maksimizacija ovog

izraza nema rješenje u zatvorenoj formi. Konkretno, problem je u logaritmu koji je "zapeo" između dviju sumacija, pa ne možemo derivirati zasebno po svakoj od komponenata, odnosno izglednost se **ne faktorizira po komponentama**.

Ako ne postoji rješenje u zatvorenoj formi, što nam je alternativa? Alternativa je iterativna optimizacija. Na primjer, **gradijentni spust** (kao kod logističke regresije, gdje optimizacija također nije imala rješenje u zatvorenoj formi, premda doduše iz drugog razloga – nelinearnosti sigmoidne funkcije), **metode MCMC** (spomenuli smo ih u predavanju o probabilističkim grafičkim modelima, npr. Gibbsovo uzorkovanje) 3

ili **algoritam maksimizacije očekivanja**. Mi ćemo u nastavku pogledati ovaj potonji.

2 Algoritam maksimizacije očekivanja

Osnovna ideja algoritma maksimizacije očekivanja (engl. *expectation maximization algoritam,* **EM-algoritam**) jest da se generativni model proširi **skrivenim (latentnim) varijablama**. Nakon takvog proširenja, maksimizaciju log-izglednost moći ćemo provesti iterativnim postupkom. Pogledajmo najprije što to znači da model proširujemo latnentnim varijablama, a onda ćemo pokazati kako upotrijebiti EM-algoritam da bismo naučili takav model. 4

2.1 Model miješane gustoće s latentnim varijablama

Prisjetite se da smo o skrivenim (latentnim) varijablama pričali u predavanju o probabilističkim grafički modelima. Rekli smo da su skrivene varijable one varijable čije vrijednosti ne opažamo u podatcima, ali u model ih uvodimo jer to može značajno pojednostaviti modeliranje, npr. onda kada skrivene varijable modeliraju neke zajedničke uzroke više opaženih varijabli. Ako skrivene varijable modeliraju neki apstraktni koncept, onda obično umjesto naziva skrivene varijable koristimo naziv **latentne varijable**. Tako će to biti u našem slučaju, gdje ćemo latentne varijable uvesti kako bismo modelirali vezu između primjera i grupa: latentne varijable će definirati koji primjer pripada kojoj grupi. Preciznije, svaki primjer $\mathbf{x}^{(i)}$ imat će pridruženu svoju latentnu varijablu $\mathbf{z}^{(i)}$ koja opisuje kojoj grupi taj primjer pripada. Ta varijabla je zapravo kategorička varijabla, i ima onoliko vrijednosti koliko ima grupa, tj. ima K vrijednosti. Kao što smo radili i do sada, tu kategoričku varijablu predstavit ćemo kao K-dimenzijski vektor binarnih indikatorskih varijabli:

$$\mathbf{z}^{(i)} = (z_1^{(i)}, \ldots, z_k^{(i)}, \ldots, z_K^{(i)})$$

pri čemu koristimo kodiranje s jednom jedinicom (engl. *one-hot encoding*): ako primjer $\mathbf{x}^{(i)}$ pripada grupi k, sve komponente varijable $\mathbf{z}^{(i)}$ bit će na nuli, osim jedne, $z_k^{(i)}$, koja će biti postavljena na jedinicu.

Jednostavnosti radi, u nastavku ćemo umjesto $\mathbf{z}^{(i)}$ pisati $\mathbf{z}$, no imajte na umu da imamo po jednu varijablu $\mathbf{z}$ za svaki primjer $\mathbf{x}^{(i)}$. Ideja je sada da te latentne varijable $\mathbf{z}$ ugradimo u model GMM, tj. da ga proširimo s latentnim varijablama. Prethodno je model bio definiran tako da modelira gustoću vjerojatnosti $p(\mathbf{x})$, a model proširen s latentnim varijablama treba definirati zajedničku gustoću vjerojatnosti $p(\mathbf{x}, \mathbf{z})$. Pokušajmo izraziti tu zajedničku gustoću vjerojatnosti.

Krenimo od toga da uočimo da je vjerojatnost $P(\mathbf{z} = k)$ zapravo jednaka apriornoj vjerojatnosti grupe k, tj. vjerojatnosti da primjer pripada grupi k neovisno o tome o kojem se primjeru radi. Apriornu vjerojatnost grupe k već smo bili označili kao π_k. Budući da je $\mathbf{z}$ kategorička varijabla prikazana kao binarni vektor, to onda znači da vjerojatnost da $\mathbf{z} = k$ možemo napisati kao:

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

Ovakav zapis vjerojatnosti za kategoričke varijable smo već bili koristili. Na ovaj način zapravo dohvaćamo π_k za onaj k za koji $z_k = 1$. Na isti način možemo definirati izglednost grupe (vjerojatnost da grupa $\mathbf{z}$ generira primjer $\mathbf{x}$):

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

Važno je da ovdje ne zaboravimo da mi naravno ne znamo kojoj grupi primjer pripada, i zato je varijabla $\mathbf{z}$ skrivena (latentna). Međutim, mi pretpostavljamo da u stvarnosti svaki primjer pripada jednoj grupi, samo što mi ne znamo kojoj. Pimijetite, dakle, da mi pretpostavljamo da svaki primjer pripada *jednoj* grupi. To sad možda izgleda čudno, jer smo ranije rekli da radimo meko grupiranje i da je $h_k^{(i)}$ vjerojatnost, što kao da sugerira da pojedini primjer može pripadati u više grupa istovremeno, tj. svim onim grupama za koje $h_k^{(i)} > 0$. No, to nije tako. Naime, ovdje trebamo razlikovati dvije stvari, koje smo već bili spomenuli. Mi u modelu GMM pretpostavljamo da svaki primjer izvorno pripada jednoj i samo jednoj grupi (tj. da ga je generirala jedna komponenta). Međutim, mi ne znamo kojoj grupi primjer pripada, i to izražavamo pomoću vjerojatnosti $h_k^{(i)}$. Dakle, vjerojatnost pripadanja primjera grupi k nije posljedica toga što primjer istovremeno pripada u više grupa, nego toga što mi ne znamo kojoj točno grupi primjer pripada. Drugim riječima, vjerojatnost modelira naše neznanje, a ne miješanu pripadnost primjera grupama. To je bitna razlika između mješavinskih modela i nekih drugih modela mekog grupiranja (npr., onih temeljenih na neizrazitoj logici).

Sada kada smo uveli latentne varijable koje opisuju vezu primjera i grupa, možemo napisti **zajedničku gustoću** $p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})$, koja nam govori koja je vjerojatnost generiranja primjera $\mathbf{x}$ takvog da on bude generiran iz grupe kodirane sa $\mathbf{z}$. Jednostavno ćemo primijeniti pravilo umnoška i zatim uvrstiti gore definirane vjerojatnosti:

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

Dakle, ovdje smo najprije primijenili pravilo umnoška, a onda smo dva umnoška niza kombinirali u jedan umnožak niza (što možemo zbog komutativnosti množenja).

Time smo izveli **model miješane gustoće s latentnim varijablama**. Usporedimo sada izvorni model miješane gustoće (bez latentnih varijabli) i ovaj upravo izvedeni model (s latentnim varijablama). Model miješane gustoće bez latentnih varijabli je:

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

dok je model miješane gustoće s latentnim varijablama:

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

Ako ćemo biti iskreni, moramo priznati da nismo napravili neku strašno pametnu stvar: model miješane gustoće s latentnim varijablama zapravo funkcionira tako da, *ako znamo kojoj grupi primjer pripada* (to je grupa k za koju $z_k = 1$), onda model "dohvaća" i množi dotične π_k i $p(\mathbf{x}|\boldsymbol{\theta}_k)$, što nam daje zajedničku vjerojatnost primjera i grupe za taj primjer. Problem je, naravno, što ne znamo kojoj grupi primjer pripada. Sad se možda čini da nismo ništa napravili. No, ipak jesmo. Bitna razlika između ova dva modela jest što kod modela s latentnim varijablama imamo modeliranu vezu između primjera i grupe, i to je parametar koji možemo procijeniti u podatcima, budući da je to upravo informacija koja nas kod grupiranja zapravo

zanima. Kao posljedica toga, druga bitna razlika jest da kod latentnog modela umjesto zbroja vjerojatnosti po komponentama imamo umnožak vjerojatnosti po komponentama. Da je ta razlika ključna postaje jasno nakon što napišemo log-izglednost modela miješane gustoće s latentnim varijablama:

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

gdje $\mathbf{Z}$ označava skup svih varijabli $z^{(i)}$. Ovako definiranu log-izglednost nazivamo **potpuna log-izglednost** (engl. *complete log-likelihood*), dok log-izglednost prvog modela, onog bez latentnih varijabli, nazivamo **nepotpuna log-izglednost** (engl. *incomplete log-likelihood*). Prisjetite se da smo o tome smo već bili pričali u predavanju o probabilističkim grafičkim modelima.

Usporedimo sada nepotpunu log-izglednost i potpunu log-izglednost. Vidimo da se, za razliku od nepotpune log-izglednosti, potpuna log-izglednost lijepo faktorizirala po komponentama. To znači da ćemo pri maksimizaciji moći derivirati svaku komponentu zasebno po njezinim parametrima, a to onda znači da ćemo imati rješenje u zatvorenoj formi. To je dobra vijest. Loša vijest jest da mi i dalje ne znamo koji primjeri pripadaju kojoj grupi, dakle ne znamo vrijednosti varijabli $\mathbf{z}^{(i)}$. Sve druge vrijednosti su nam poznate, ali te nisu. To onda znači da ipak ne možemo analitički provesti maksimizaciju log-izglednosti (nalaženje nultočke).

2.2 Algoritam maksimizacije očekivanja (napokon!)

I tu napokon dolazimo do **algoritma maksimizacije očekivanja**. Ideja je ova: budući da ne znamo vrijednosti za $\mathbf{z}^{(i)}$, ono što možemo napraviti jest izračunati **očekivanje** potpune log-izglednosti uz neke fiksirane vrijednosti za parametre π_k (koeficijenti mješavine) i $\boldsymbol{\theta}_k$ (parametri gustoće komponenti), a nakon toga ažurirati parametre π_k i $\boldsymbol{\theta}_k$ tako da maksimiziraju to očekivanje. I ta dva koraka onda možemo iterativno ponavljati: svaki puta izračunati očekivanje potpune log-izglednosti po latentnim varijablama, a onda odabrati parametre koji maksimiziraju to očekivanje. U ovom drugom koraku sve varijable će već biti fiksirane, pa će postojati rješenje u zatvorenoj formi.

Može se pokazati – a u to sada nećemo ulaziti – da maksimizacija očekivanja potpune log-izglednosti ujedno dovodi do povećanja nepotpune log-izglednosti, što je zapravo početni problem koji smo željeli riješiti. Treba, međutim, napomenuti da ovaj postupak ne mora nužno dovesti do globalnog maksimuma nepotpune log-izglednosti. To će, kao i kod algoritma K-sredina, ovisiti o odabiru početnih vrijednosti parametara. 5

Algoritam, dakle, ima **dva koraka**: izračun očekivanja potpune log-izglednosti i zatim njegovu maksimizaciju. Prvi korak (izračun očekivanja potpune log-izglednosti) naziva se **E-korak** (od "expectation"). Označimo sa $\mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ očekivanje potpune log-izglednosti uz fiksirane parametre $\boldsymbol{\theta}^{(t)}$ u iteraciji t. To očekivanje je jednako:

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

U ovom izrazu, jedino je latentna varijabla $z_k^{(i)}$ slučajna varijabla, dok su sve ostale vrijednosti fiksne. Dakle, očekivanje zapravo računamo po varijabli $z_k^{(i)}$ (pomnoženoj s nekom konstantom). Varijabla $z_k^{(i)}$ je binarna indikatorska varijabla, dakle Bernoullijeva varijabla, pa je njezino očekivanje zapravo jednako vjerojatnosti da ona bude jednaka 1. A to je upravo jednako odgovornosti $h_k^{(i)}$! Formalno:

$$\mathbb{E}[z_k^{(i)}|\mathcal{D}, \boldsymbol{\theta}^{(t)}] = \mathbb{E}[z_k^{(i)}|\mathbf{x}^{(i)}, \boldsymbol{\theta}^{(t)}] = P(z_k^{(i)} = 1|\mathbf{x}^{(i)}, \boldsymbol{\theta}^{(t)}) = \frac{p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k^{(t)})\pi_k^t}{\sum_{j=1}^K p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_j^{(t)})\pi_j^t} = h_k^{(i)}$$

Dakle, očekivanje potpune log-izglednosti uz fiksirane parametre $\boldsymbol{\theta}^{(t)}$ u iteraciji t jednako je (samo uvrstimo $h_k^{(i)}$ umjesto $z_k^{(i)}$):

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) &= \sum_{i=1}^N \sum_{k=1}^K h_k^{(i)} \Big(\ln \pi_k + \ln p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k) \Big) \\ &= \sum_{i=1}^N \sum_{k=1}^K h_k^{(i)} \ln \pi_k + \sum_{i=1}^N \sum_{k=1}^K h_k^{(i)} \ln p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k)\end{aligned}$$

To je prvi korak algoritma maksimizacije očekivanja. U drugom koraku, koji nazivamo **M-korak** (od "maximization"), želimo maksimizirati očekivanje potpune log-izglednosti po parametrima π_k i $\boldsymbol{\theta}_k$. Budući da u izrazu više nemamo latentnih varijabli, to sada možemo napraviti analitički, deriviranjem i nalaženjem nultočke:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) &= 0 \\ \nabla_{\pi_k} \Big(\sum_{i=1}^N \sum_{k=1}^K h_k^{(i)} \ln \pi_k + \lambda \big(\sum_k \pi_k - 1 \big) \Big) &= 0 \\ \nabla_{\boldsymbol{\theta}_k} \sum_{i=1}^N h_k^{(i)} \ln p(\mathbf{x}^{(i)}|\boldsymbol{\theta}_k) &= 0\end{aligned}$$

Vidimo da se maksimizacija može raditi odvojeno za svaki od dva pribrojnika. Ovdje ćemo sada preskočiti sam postupak maksimizacije. Za prvi pribrojnik radili bismo maksimizaciju metodom Lagrangeovih multiplikatora, budući da postoji ograničenje da se koeficijenti mješavine, koji zapravo definiraju kategoričku varijablu, moraju zbrajati u jedinicu. Za drugi pribrojnik moramo najprije odabrati konkretnu gustoću vjerojatnosti (u našem slučaju to su Gaussove gustoće vjerojatnosti) pa tek onda provesti maksimizaciju. Krajni rezultat za komponente mješavina je:

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

a za parametre Gaussovih komponenata:

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

I to nas onda napokon dovodi do EM-algoritma za GMM, koji smo već bili pogledali:

▸ **Algoritam GMM (model GMM + EM-algoritam)**

inicijaliziraj parametre $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$
ponavljaj do konvergencije log-izglednosti ili parametara
E-korak:
Za svaki primjer $\mathbf{x}^{(i)} \in \mathcal{D}$ i svaku komponentu $k = 1, \ldots, K$:

$$h_k^{(i)} \leftarrow \frac{p(\mathbf{x}^{(i)}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\pi_k}{\sum_{j=1}^K p(\mathbf{x}^{(i)}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\pi_j}$$

M-korak:
Za svaku komponentu $k = 1, \ldots, K$:

$$\boldsymbol{\mu}_k \leftarrow \frac{\sum_i h_k^{(i)}\mathbf{x}^{(i)}}{\sum_i h_k^{(i)}}, \quad \boldsymbol{\Sigma}_k \leftarrow \frac{\sum_i h_k^{(i)}(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^{\mathrm{T}}}{\sum_i h_k^{(i)}}, \quad \pi_k \leftarrow \frac{1}{N}\sum_{i=1}^N h_k^{(i)}$$

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

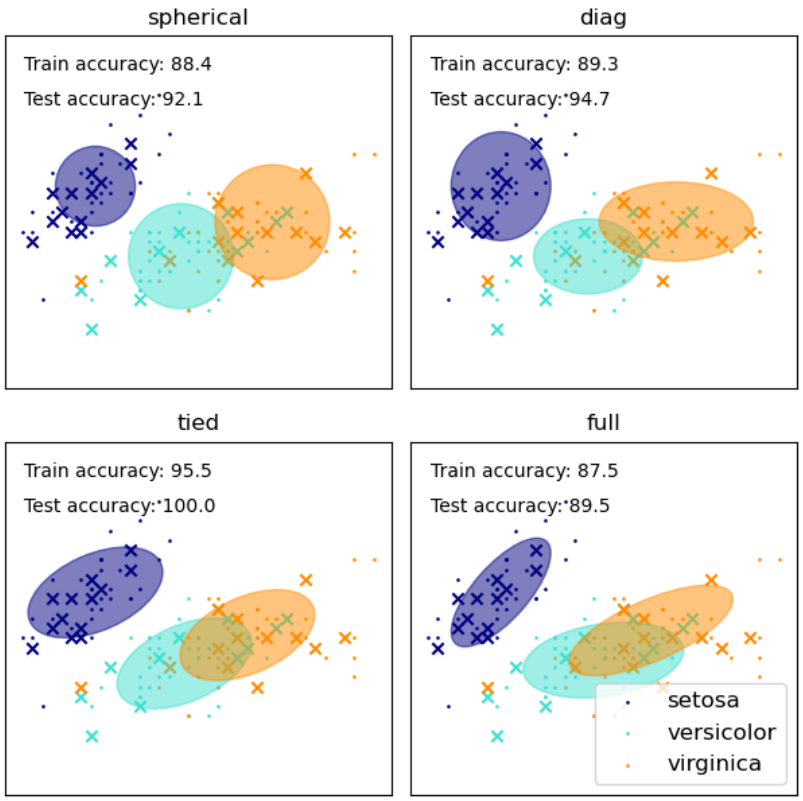
Nepotpuna log-izglednost $\ln \mathcal{L}(\boldsymbol{\theta}|\mathcal{D})$ (isto kao i potpuna log-izglednost) rasti će kroz iteracije, sve do konvergencije. U tom trenutku zaustavljamo postupak grupiranja. Rezultat algoritma su naučeni parametri $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$, tj. koeficijenti mješavine te srednji vektor i kovarijacijska matrica za svaku od K grupa. Iz toga lako, primjenom Bayesovog pravila, izračunamo odgovornosti $h_k^{(i)}$, koje opisuju vjerojatnost da primjer $\mathbf{x}^{(i)}$ pripada grupi (odnosno da ga je generirala grupa) K. I time smo zapravo ostvarili meko grupiranje primjera u K grupa.

Na ovom mjestu zgodno je napraviti usporedu GMM-a i **Gaussovog Bayesovog klasifikatora**, o kojemu smo već bili pričali. Usporedba ima smisla, jer oba modela koriste Gaussovu gustoću za modeliranje izglednosti te Bayesovo pravilo za izračun aposteriorne vjerojatnosti. Naravno, očita razlika između GMM i Gaussovog Bayesovog klasifikatora je u tome što je prvi nenadzirani, a drugi nadzirani algoritam. Posljedično, kod ovog drugog imali smo označene primjere, i mogli smo u zatvorenoj formi izračunati MLE. Kod GMM-a nismo imali označene primjere i MLE nema rješenja u zatvorenoj formi, pa smo zato uveli latentne varijable i postupak optimizacije proveli smo iterativno. Ta se razlika onda svodi na to da kod Gaussovog Bayesovog klasifikatora kod procjene parametara Gaussove gustoće za svaku klasu znamo točno koji primjeri pripadaju toj klasi i samo njih koristimo kod procjene parametara, dok kod GMM-a to ne znamo, pa sve primjere moramo uzeti u obzir s nekom vjerojatnošću (što modeliramo odgovornošću).

Osim ove razlike, postoji i jedna sličnost, a to je da oba modela koriste **kovarijacijsku matricu** kako bi modelirala korelacije između značajki. Isto kao i kod Gaussovog Bayesovog klasifikatora, uvođenjem pretpostavki odnosno ograničenja nad kovarijacijskom matricom možemo definirati modele **manje složenosti** od modela s punom kovarijacijskom matricom. Pogledajmo primjer.

▸ **PRIMJER**

Razmotrimo grupiranje dvodimenzijskih primjera iz poznatog skupa podataka Iris u $K = 3$ grupe. Koristimo algoritam GMM, pri čemu razmatramo četiri složenosti modela Gaussove mješavine. Te su složenosti određene ograničenjima na kovarijacijske matrice Σ_k za svaku od K komponenti. Razmatramo punu, dijeljenu, dijagonalnu i izotropnu kovarijacijsku matricu. Rezultati grupiranja za ta četiri modela izgledaju ovako: 6



Na slici gore ljevo imamo grupiranje dobiveno modelom s izotropnom kovarijacijskom matricom, pa su izokonture Gaussove gustoće kružnice odnosno sferične su. Na slici gore desno imamo grupiranje dobiveno modelom s dijagonalnom kovarijacijskom matricom, pa su izokonture elipse poravnate s osima ulaznog prostora. Na slici dolje dolje lijevo imamo dijeljenu (ali ne i dijagonalnu, budući da su elipse zakošene!) konvarijacijsku matricu, pa su sve tri gustoće jednako poravnate. Konačno, dolje desno imamo punu (i nedijeljenu) kovarijacijsku matricu, pa je svaka gustoća različita i epise nisu poravnate. Najsloženiji model je upravo ovaj posljednji, gdje imamo punu kovarijacijsku matricu, pa možemo modelirati (linearnu) zavisnost u šumu između značajki svih primjera iste grupe. Da je taj model najsloženiji vidi se i po tome što je točnost na skupu za učenje najveća. Međutim, taj model nije optimalan, jer točnost na ispitnome skupu nije maksimalna: maksimalnu točnost ostvaruje model s dijeljenim kovarijacijskim matricama. Gornja dva modela imaju znatno manju točnost i na skupu za učenje i na ispitnom skupu. 7

Uvođenjem različitih ograničenja na kovarijacijsku matricu možemo, dakle, upravljati složenošću modela. No, kako onda odrediti koji je model najbolji? Primijetite da ne možemo koristiti unakrsnu provjeru na ispitnom skupu, budući da se ovdje ne radi o modelu nadziranog učenja koji bismo mogli primijeniti za predikciju na neviđenim primjerima. Problem odabira složenosti modela GMM zapravo je istovjetan problemu određivanja broja grupa, pa dakle možemo koristiti neke od metoda koje smo bili spomenuli prošli put, npr., AIC ili (tipičnije) BIC (engl. *Bayesian information criterion*). Alternativno, možemo označiti podskup primjera pa na tom podskupu provjeriti grupe, tako da izračunamo Randov indeks ili neku drugu mjeru točnosti grupiranja. 8

Krajnje pojednostavljenje modela jest da pretpostavimo da sve komponente imaju **dijeljenu i izotropnu kovarijacijsku matricu**, $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$. Ako pored toga odgovornosti $h_k^{(i)}$ zaokružimo na 0 ili 1, tj. napravimo čvrsto grupiranje, onda ćemo imati $h_k^{(i)} = b_k^{(i)}$ i algoritam GMM degenerira u algoritam K-sredina. Tada također vrijedi $\ln \mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) \propto -J$, tj. log-izglednost je proporcionalna

negativnoj pogrešci. Prisjetite se da smo sličnu vezu imali između log-izglednosti poopćenih linearnih modela i njihovih funkcija pogreški.

2.3 Napomene

Zaključimo naše izlaganje algoritma GMM s nekoliko praktičnih napomena:

- EM-algoritam nužno konvergira, ali u **lokalni optimum** log-izglednosti. To znači da će konačni rezultat vrlo ovisiti o inicijalizaciji parametara;
- Poznato je da algoritam GMM sporo konvergira. Kako bi se algoritam ponešto ubrzao, inicijalizacija središta $\boldsymbol{\mu}_k$ može se provesti algoritmom K-srednjih vrijednosti. Algoritam GMM onda će se moći fokusirati na procjenu parametara kovarijacijskih matrica te na fino ugađanje centroida;
- Kao što smo pokazali, možemo uvesti ograničenja na kovarijacijsku matricu Σ (npr., dijeljena, dijagonalna ili izotropna matrica). Takva ograničenja daje jednostavnije modele, koji su manje skloni prenaučenosti;
- Kao i kod svih algoritama grupiranja, broj grupa K je hiperparametar koji treba nekako unaprijed odrediti (metodama koje smo spominjali prošli put). U tu svrhu može se koristiti Akaikeov informacijski kriterij (AIC):

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

  gdje je $q(K)$ broj parametara modela sa K grupa.
- EM-algoritam je općenit algoritam za **optimizaciju parametara modela s latentnih varijablama**. Ovdje smo ga primijenili na model GMM.

Algoritam GMM razlikuje se od algoritma K-sredina po tome što daje meko a ne čvrsto grupiranje. Međutim, oba su algoritma particijska. Pogledajmo sada jedan algoritam grupiranja koji nije particijski već hijerarhijski.

3 Hijerarhijsko grupiranje

Na kraju današnjeg predavanja razmotrit ćemo hijearhijski algoritam grupiranja, i to konkretno algoritam **hijerarhijskog aglomerativnog grupiranja** (engl. *hierarchical agglomerative clustering*, HAC). Radi se o jednom vrlo jednostavnom ali i vrlo dobrom i široko korištenom algoritmu grupiranja, pogotovo u dubinskoj analizi podataka. HAC, naravno, nije jedini algoritam za hijerarhijsko grupiranje, no mi nećemo razmatrati druge algoritme. 9

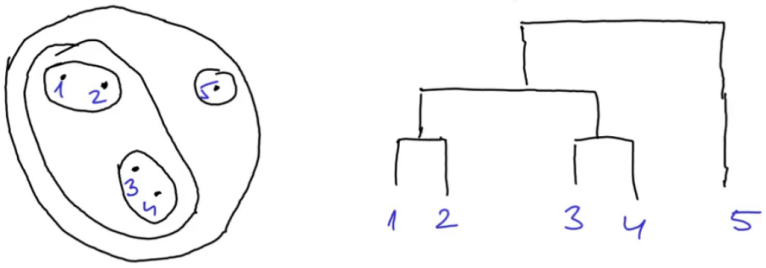
3.1 Dendrogram

Krenimo od toga da razmotrimo na koji se način može prikazati rezultat hijerarhijskog grupiranja. Za razliku od particijskog grupiranja, koje rezultira grupama koje odgovaraju skupovima primjera, hijerarhijsko grupiranje, pa tako i algoritam HAC, rezultira grupama koje se sastoje od podgrupa, odnosno **hijerarhijom grupa**. Takva se hijerarhija može prikazati tzv. **dendrogramom**. Pogledajmo primjer. 10

▸ **PRIMJER**

Neka je pet primjera u dvodimenzijskome ulaznom prostoru raspoređeno kao na slici dolje lijevo. Te primjere po euklidskoj udaljenosti (odnosno bliskosti) možemo grupirati tako da su primjeri 1 i 2 u jednoj grupi, primjeri 3 i 4 u drugoj grupi, a primjeri 5 u trećoj grupi. Time smo primjere grupirali na najnižoj hijerarhijskoj razini. Na idućoj razini možemo grupirati grupe primjera: možemo spojiti

grupe 1 i 2 u jednu nadgrupu te grupe 3 i 4 u drugu nadgrupu. Konačno, na najvišoj razini, ove dvije nadgrupe možemo spojiti u jednu nadnadgrupu. Takvom grupiranju onda odgovara dendrogram prikazan na slici dolje desno.



Dendrogram je zapravo stablasti prikaz hijerarhije. U listovima dendrograma su primjeri koje grupiramo, a grane povezuju primjere koji su u istoj grupi te grupe u nadgrupe. Na najnižoj razini (listovi) svaki primjer je u svojoj zasebnoj grupi. Kako se uspinjemo uz dendrogram, grupe od po jednog primjera najprije spajamo u grupe s po dva primjera, a zatim u grupe s po tri primjera, itd. Na najvišoj razini (korijen) svi primjeri čine jednu grupu. Vertikalna os dendrograma indicira na kojoj se udaljenosti događa spajanje grupa: što se više uspinjemo od listova prema korijenu dendrograma, to je veća udaljenost između grupa koje spajamo u jednu nadgrupu. Drugim riječima, visine grana indiciraju koliko su daleko grupe koje su spojene u nadgrupu: ako su grane dugačke, znači da su grupe dosta udaljene i da se spajaju na većoj udaljenosti. Ako su grane vrlo kratke, onda to znači da su grupe vrlo blizu i da se spajaju na maloj udaljenosti.

3.2 Povezivanje grupa

Hijerarhijsko se grupiranje općenito provodi na temelju neke **mjere udaljenosti** $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ između primjera ili na temelju **mjere sličnosti (ili različitosti)** $s : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ primjera. U tom je smislu hijerarhijsko grupiranje slično algoritmu K-medoida, o kojem smo pričali prošli put, a koji isto tako može koristiti općenitu mjeru sličnosti odnosno različitosti. Prisjetimo se da je mjera sličnosti/različitosti općenitija od mjere udaljenosti. Naime, mjera udaljenosti mora zadovoljavati svojstva metrike, uključivo nejednakost trokuta, dok mjera sličnosti/različitosti to ne mora. Tipično korištene mjere udaljenosti su euklidska udaljenost (L2), Manhattan udaljenost (L1) i Mahalanobisova udaljenost, dok za mjeru različitosti postoji više mogućnosti, sve dok su zadovoljena sljedeća svojstva:

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

(2) $0 \leqslant s(\mathbf{x}^a, \mathbf{x}^b) \leqslant 1$,

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

Prisjetimo se: to su ista ona svojstva koja smo imali za **jezgrene funkcije**, koje nisu ništa drugo nego funkcije sličnosti.

Hijerarhijsko grupiranje može biti aglomerativno ili divizivno. **Aglomerativno grupiranje** kreće od grupa koje sadrže svaka po samo jedan primjer, i zatim postepeno stapa najbliže grupe dok sve primjere ne stopi u jednu veliku grupu. To znači da se dendrogram gradi odozdo prema gore (od pojedinačnih primjera do zajedničke grupe). **Divizivno grupiranje** ide obrnuto: cijeli skup primjera dijeli se na grupe i podgrupe, tj. dendrogram se gradi odozgo prema dolje. Mi ćemo se u nastavku baviti isključivo aglomerativnim grupiranjem.

Kod aglomerativnog grupiranja, u svakom koraku treba stopiti dvije najbliže odnosno najsličnije grupe $\mathcal{G}_i$ i $\mathcal{G}_j$. Ako te dvije grupe imaju svaka po jedan primjer, onda lako možemo izračunati udaljenost (odnosno sličnost) između takvih grupa jer se udaljenost (odnosno sličnost) svodi na udaljenost (odnosno sličnost) između dotični primjera. Međutim, postavlja se pitanje kako izračunati udaljenost (odnosno sličnost) između grupa koje sadrže više od jednog primjera? Tu imamo više mogućnosti. U nastavku ćemo govoriti o mjeri udaljenosti, no razmatranja vrijede analogno i za mjeru sličnosti.

Prva mogućnost jest da udaljenost između dviju grupa definiramo kao **najmanju udaljenost** između pojedinačnih primjera u tim grupama. Formalno:

$$d_{min}(\mathcal{G}_i, \mathcal{G}_j) = \min_{\mathbf{x}^{(i)} \in \mathcal{G}_i, \mathbf{x}^{(j)} \in \mathcal{G}_j} d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

Takvo grupiranje nazivamo grupiranje **jednostruke povezanosti** (engl. *single-link clustering*). Grupiranje nazivamo jednostrukim jer se dvije grupe povezuju na temelju udaljenosti samo jednog para primjera iz dviju grupa (svi ostali parovi primjera su više udaljeni). Alternativno, udaljenost između grupa možeom idefinirati kao **najveću udaljenost** između pojedinačnih primjera u tim grupama:

$$d_{max}(\mathcal{G}_i, \mathcal{G}_j) = \max_{\mathbf{x}^{(i)} \in \mathcal{G}_i, \mathbf{x}^{(j)} \in \mathcal{G}_j} d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

Takvo grupiranje nazivamo grupiranje **potpunim povezivanjem** (engl. *complete-link clustering*). Grupiranje nazivamo potpunim jer, ako smo primjere povezali na temelju udaljenosti dva najdalja primjera iz dviju grupa, onda su svi drugi parovi primjera iz dviju grupa još manje udaljeni, odnosno možemo reći da su grupe povezane preko svih parova primjera (potpuno). Nameće se pitanje: koje povezivanje odabrati u praksi? Ako su grupe kompaktne i prirodno dobro odvojene, onda ove dvije metode ne daju značajno različite rezultate. Međutim, ako to nije slučaj, razlike mogu biti značajne. Tada jednostruko povezivanje rezultira dugačkim, ulančanim grupama, dok potpuno povezivanje rezultira manjim, zbijenijim grupama. 11

Jednostruko i potpuno povezivanje predstavljaju dva krajnja slučaja izračuna udaljenosti između grupa i obje su metode dosta osjetljive na šum u podatcima. Kompromisno rješenje jest grupiranje na temelju **prosječne povezanosti** (engl. *average-linkage clustering*):

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

Konačno, četvrta mogućnost je povezivanje na temelju **centroida**:

$$d_{cent}(\mathcal{G}_i, \mathcal{G}_j) = \left\| \frac{1}{N_i} \sum_{\mathbf{x} \in \mathcal{G}_i} \mathbf{x} - \frac{1}{N_j} \sum_{\mathbf{x} \in \mathcal{G}_j} \mathbf{x} \right\|$$

Kod povezivanja na temelju centroida imamo ograničenje da prostor mora biti vektorski. Primijetite da prosječno povezivanje i povezivanje centroida nisu identične mjere udaljenosti.

Često je korištena i **Wardova metoda minimalne varijance**, koja udaljenost između grupa izračunava kao povećanje kvadrata udaljenosti između primjera i centroida prije i poslije stapanja dviju grupa. Ovo podsjeća na kriterijsku funkciju K-sredina. Ovdje nećemo u detalje. 12

Kako odabrati optimalnu metodu povezivanja? Različite metode unose sa sobom različite pretpostavke. Kako to inače biva, optimalna će biti ona metoda čije su pretpostavke najbolje usklađene sa stvarim stanjem na terenu, tj. s našim podatcima. Npr., potpuno povezivanje rezultirat će grupama kompaktnih rubova, dok Wardova metoda daje grupe koje mogu imati raspšene rubove. U praksi se odabir ipak nerijetko svodi na eksperimenitiranje. 13

3.3 Hijerarhijsko aglomerativno grupiranje (HAC)

Složimo sada sve ovo u **algoritam hijerarhijskog aglomerativnog grupiranja** (engl. *hiearachical agglomerative clustering*, HAC). U nastavku je prikazan pseudokod.

▸ **Algoritam hijerarhijskog aglomerativnog grupiranja (HAC)**

1: **inicijaliziraj** K, $k \leftarrow N$, $\mathcal{G}_i \leftarrow \{\mathbf{x}^{(i)}\}$ za $i = 1, \ldots, N$
2: **ponavljaj**
3: $\quad k \leftarrow k - 1$
4: $\quad (\mathcal{G}_i, \mathcal{G}_j) \leftarrow \underset{\mathcal{G}_a, \mathcal{G}_b}{\operatorname{argmin}}\, d(\mathcal{G}_a, \mathcal{G}_b)$
5: $\quad \mathcal{G}_i \leftarrow \mathcal{G}_i \cup \mathcal{G}_j$
6: **dok je** $k > K$

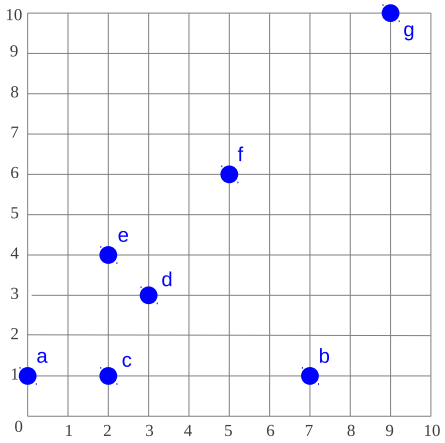
Hiperparametar K je unaprijed zadani broj grupa. Primijetite da ovaj pseudokod zapravo ne gradi eksplicitan dendrogram. No, to lako možemo ostvariti tako da umjesto unije skupova u koraku 5 kreiramo čvor dendograma i u njegove listove stavimo grupe koje spajamo.

Za $K = 1$ dobivamo potpun dendrogram koji možemo naknadno **presijecati**. To znači da možemo odabrati neku udaljenost (odnosno razinu sličnosti, odnosno razinu različitosti) i na tom mjestu (na toj visini dendrograma) uzdužno presijeći grane dendrograma. To će nam dati particijsko grupiranje. Odabir mjesta presijecanja istovjetan je odabiru broja grupa.

Pogledajmo jedan jednostavan primjer.

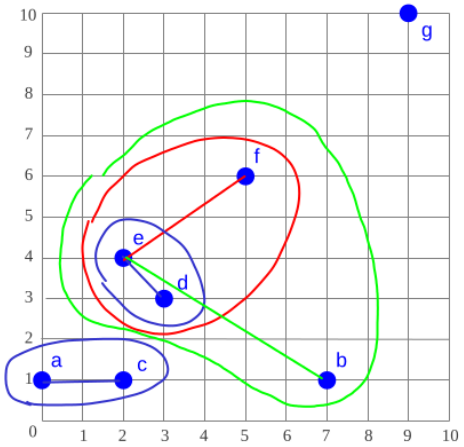
▸ **PRIMJER**

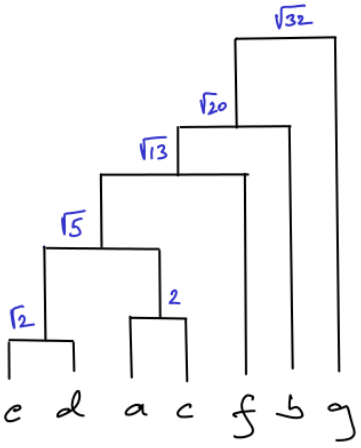
Algoritmom HAC grupiramo primjere prema euklidskoj udaljenosti u dvodimenzijskome prostoru. Skup se sastoji od sljedećih sedam primjera:



Razmotrimo najprije grupiranje s jednostrukom povezanošću, što znači da ćemo udaljenost između grupa računati kao minimalnu udaljenost između svih parova primjera iz tih dviju grupa. Prisjetite se da radimo algomerativno grupiranje: dakle, krećemo s najvećim brojem grupa, a kako grupe spajamo u nadgrupe, tako smanjujemo broj grupa, sve dok sve grupe ne spjimo u jednu zajedničku grupu. Na početku grupiranja svaki primjer primjer tretiramo kao zasebnu grupu, dakle imamo $K = 7$ grupa s po jednim primjerom. Zatim spajamo one grupe koje su najbliže, računajući udaljenosti između grupa kao maksimalnu udaljenost između primjera u tim grupama. U prvom koraku to je jednostavno, budući da trebamo izračunati udaljenosti između grupa koje sadrže samo po jedan primjer, a to je isto kao da računamo udaljenosti između tih primjera. Dakle, u prvom koraku najbliže u su grupe $\{d\}$ i $\{e\}$ (njihova udaljenost je udaljenost je $\sqrt{2}$), pa te dvije grupe spajamo u jednu grupu, $\{d, e\}$, i to spajanje se događa na udaljenosti $d_{min} = \sqrt{2}$. Time je ukupan

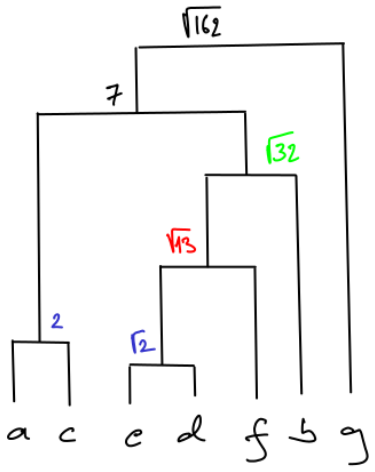
broj grupa smanjen na $K = 6$. U drugom koraku, najbliže su grupe $\{a\}$ i $\{c\}$, pa njih spajamo u zajedničku grupu, $\{a, c\}$, i to na udaljenosti $d_{min} = 2$. Time je ukupan broj grupa smanjen na $K = 5$. U trećem koraku najbliže su upravo te dvije novostvorene grupe: naime, udaljenost između grupe $\{a, c\}$ i $\{d, e\}$ je minimalna udaljenost između 4 parova primjera između tih grupa (to su parovi (a, d), (a, e), (c, d) i (c, e)), i ta minimalna udaljenost iznosi $d_{min} = \sqrt{5}$ (za par (c, d)). Uvjerite se da je to doista najmanja udaljenost između svih parova grupa u ovom koraku, tj. da je udaljenost između svih drugih parova od 5 grupa koje u ovom koraku imamo veća od $\sqrt{5}$. Dakle, u trećem koraku na udaljenosti $d_{min} = \sqrt{5}$ spajamo grupe $\{a, c\}$ i $\{d, e\}$ te dobivamo nadgrupu $\{a, c, d, e\}$. Time je ukupan broj grupa smanjen na $K = 4$. U četvrtom koraku ispostavlja se da su najbiže grupe $\{f\}$ i grupa $\{a, c, d, e\}$, i to na udaljenosti $d_{min} = \sqrt{13}$. Spajanjem tih dviju grupa dobivamo grupu a, c, d, e, f te ukupno onda imamo $K = 3$ grupe. U iduća dva koraka još ćemo tu grupu spojiti s grupom $\{b\}$ i zatim s grupom $\{g\}$, te završiti s jednom velikom grupom, $K = 1$, čime se postupak grupiranja zaustavlja. Na slici dolje lijevo prikazan je rezultat grupiranja kao ugniježđene grupe (prikazan je predzadnji korak, prije stapanja s grupom koja sadrži primjer g).





Paralelno sa postupkom grupiranja možemo crtati dendrogram, pri čemu kod svakog spajanja grupa naznačavamo na kojoj je udaljenosti spajanje načinjeno. Dendrogram za ovo grupiranje prikazan je na slici gore desno. Budući da uvijek spajamo par grupa, dendrogram će uvijek biti binarno stablo.

Gornje grupiranje napravili smo s jednostrukim povezivanjem. Za vježbu, ponovite postupak grupiranja s potpunim povezivanjem. Postupak je sličan, samo što udaljenost između parova grupa treba računati kao maksimalnu udaljenost između parova primjera iz tih grupa, a ne kao minimalnu udaljenost. Grupiranje s potpunim povezivanjem daje ovakav dendrogram:



Ako sada iz hijerarhijskog grupiranja želimo dobiti particijsko grupiranje, onda ove dendrogram možemo presijecati na nekoj odabranoj udaljenosti ili na nekom odabranom broju grupa. Na primjer, ako gornji dendrogram presjećemo na udaljenosti između $\sqrt{13}$ i $\sqrt{32}$ (npr. na udaljenosti 5), onda ćemo dobiti $K = 4$ grupe, i to: $\{a, c\}$, $\{d, e, f\}$, $\{b\}$ i $\{g\}$. Dakle, grupe čija je točka u dendrogramu spajanja ispod točke presijecanja ostaju kao jedna grupa, dok se grupe čija je točka spajanja iznad točke presijecanja razlamaju u više grupa. Isti rezultat, naravno, dobili bismo da se odlučimo za $K = 4$ grupe. Kako odrediti optimalan broj grupa? Vrijede iste napomene kao i prošli put: ili unaprijed znamo koji bi broj grupa trebao biti, ili moramo upotrijebiti neku od metoda za nalaženje optimalnog broja grupa.

Primijetite da su primjeri ovdje bili definirani u vektorskome prostoru i da smo grupiranje radili na temelju euklidske udaljenosti. No, algoritam HAC može se primijeniti i za općenitiji slučaj gdje primjeri nisu u vektorskom prostoru i grupiranje provodimo na temelju neke općenite mjere sličnosti odnosno različitosti.

14

Što je sa složenošću algoritma HAC? Prostorna složenost određena je matricom udaljenosti/sličnosti, koja sadržava udaljenosti/sličnosti između $\binom{N}{2}$ parova primjera. To je $\mathcal{O}(N^2)$. U praksi, kod vrlo velikog broja primjera, ova kvadratna prostorna složenost može već biti nepremostiv problem. Što se vremenske složenosti tiče, najprije ustanovima da je broj koraka algoritma $N - K$ (odnosno N ako $K = 1$, tj. ako gradimo kompletan dendrogram). Za pronalaženje najbližeg/najsličnijeg para grupa potrebno je $\mathcal{O}(N^2)$ izračuna. Dakle, ukupna je vremenska složenost $\mathcal{O}((N - K)N^2) \approx \mathcal{O}(N^3)$. Međutim, moguća je implementacja s prioritetnom listom, čija je vremenska složeost $\mathcal{O}(N^2 \log N)$. Nadalje, specifično kod jednostrukog povezivanja, vremenska složenost je $\mathcal{O}(N^2)$.

15

Ovime završava naš kratak izlet u nenadzirano strojno učenje.

Sažetak

- Kod **probabilističkog grupiranja** primjeri pripadaju grupama s određenom vjerojatnošću
- Algoritam GMM je **poopćenje** algoritma K-srednjih vrijednosti gje su grupe modelirane Gaussovim gustoćama vjerojatnosti
- Probabilističko grupiranje možemo promatrati kao **optimizaciju log-izglednosti Gaussove mješavine**
- Procjenu parametara možemo napraviti **algoritmom maksimizacije očekivanja (EM-algoritam)**
- **Hijerarhijsko aglomerativno grupiranje (HAC)** postepeno stapa najbliže/najsličnije grupe i gradi **dendrogram**
- HAC može raditi s bilo kojom mjerom **sličnosti**, no glavni nedostatak je **kvadratna prostorna složenost**

Bilješke

1 Ovo predavanje zasniva se na poglavljima 7.2 i 7.4 iz [Alpaydin, 2020] te poglavlju 9.2 iz [Bishop, 2006] (za temu GMM) te poglavlju 7.4 iz [Alpaydin, 2020] (za temu hijerarhijskog grupiranja).

2 Algoritmima **neizrazitog grupiranja** (engl. *fuzzy clustering*) nećemo se baviti. Tipičan primjer je algoritam **fuzzy C-means (FCM)** [Dunn, 1973]. Više možete pročitati ovdje: `https://matteucci.faculty.polimi.it/Clustering/tutorial_html/cmeans.html`.

3 Za primjenu **Gibbsovog uzorkovanja** za učenje modela GMM, vidjeti [Diebolt and Robert, 1994]. U [Levine and Casella, 2001] je predložen hibridni pristup, **Monte Carlo EM-algoritam**, kod kojega se

u očekivanje izglednosti u E-koraku računa Monte Carlo simulacijom. Usporedbu ovih metoda za učenje modela GMM možete naći u [Zaheer et al., 2015].

[4] **EM-algoritam** osmislili su 1977. godine američki statističari Arthur Dempster, Nan Laird i Donald Rubin [Dempster et al., 1977]. EM-algoritam doista je jedan od važnijih i širokoprimjenjivih algoritama, čemu u prilog govori i da je izvorni članak iz 1977. godine do sada citiran više od 62k puta. Pristupačan opis EM-algoritma, s primjenom na GMM i na HMM (ovo drugo mi nećemo raditi) možete naći u [Bilmes et al., 1998]. Lijep pregled različitih pogleda na EM-algoritam te njegovih različitih varijanti možete naći u [Roche, 2011].

[5] Dokaz da maksimizacija očekivanja potpune log-izglednosti dovodi do povećanje nepotpune log-izglednosti može se naći u [Wu, 1983] ili na `https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm#Proof_of_correctness`

[6] Primjer je preuzet iz dokumentacije knjižnice scikit-learn, sa `https://scikit-learn.org/stable/auto_examples/mixture/plot_gmm_covariances.html`.

[7] Treba napomenuti da je ovaj primjer, preuzet sa web-stranice biblioteke scikit-learn, malo nerealan. Naime, sredine grupa $\boldsymbol{\mu}_k$ izračunate su na temelju oznaka primjera (koje su ovdje bile na raspolaganju), dok su samo parametri π_k i $\boldsymbol{\Sigma}_k$ učeni iz podataka. Zbog toga je ovdje i točnost grupiranja (ovdje vrlo netipično definirana kao podudaranje oznaka grupe i oznaka primjera) visoka. U stvarnosti ćemo rijetko vidjeti 100% točnost na ispitnome skupu. Također, u praksi nije tipično da ispitna točnost bude bolja od točnosti na skupu za učenje.

[8] **BIC** (engl. *Bayesian information kriterion*) je kriterij za odabir modela sličan **Akaikeovom kriteriju** (o kojem smo pričali prošli put). O razlici između BIC i AIC možete pročitati ovdje: `https://stats.stackexchange.com/q/577/93766`. Za primjer uporabe BIC-a za odabir broja grupa modela GMM, v. `https://stats.stackexchange.com/q/368560/93766`.

[9] Dobar pregled **algoritama hijerarhijskog grupiranja** možete naći u [Murtagh and Contreras, 2012].

[10] **Dendrogram** je složenica od grčkih riječi *dendron* (stablo) i *gramma* (crtež). Često ćete vidjeti da se (pogrešno) koristi naziv *dendogram* (bez "r").

[11] Grupiranje **jednostrukim povezivanjem** odnosno **potpunim povezivanjem** 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**.

[12] **Wardovu metodu minimalne varijance** predložio je 1963. godine Joe Ward [Ward Jr, 1963]. Sažet opis postupka možete naći na `https://en.wikipedia.org/wiki/Ward%27s_method`. Lijep primjer primjene Wardove metode možete naći na `https://jbhender.github.io/Stats506/F18/GP/Group10.html`.

[13] Ovdje možete naći dobar pregled različitih **metoda povezivanja** `https://stats.stackexchange.com/a/217742/93766` i opis toga kakvo grupiranje možete očekivati dobiti sa svakom od metoda.

[14] Kod dendrograma imamo zgodnu mogućnost da na temelju visine grana dendrograma odlučimo o optimalnom mjestu **presjecanja dendrograma** (a time onda i optimalnom broju grupa): na mjestima gdje su sve grane razmjerno visoke možemo napraviti presjecanje jer je tamo grupiranje konzistentno. Naime, ako su sve grane dugačke, to onda znači da bi za bilo kakva daljnja stapanja trebalo stopiti poprilično udaljene (različite) grupe, kao i da bi za daljnja razdvajanja trebalo razdvojiti poprilično bliske (slične) grupe. Kvantifikaciju ove ideje pružaju **koeficijenti nekonzistentnosti**, koji se temelje na duljini grane i duljini grana poslije razdvajanja. Za detalje, pogledati `https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.inconsistent.html`

15 Grupiranje jednostrukim povezivanjem istovjetno je izračunavanju minimalnog razapinjućeg stabla. Vremenska složenost tog algoritma (**Prim-Jarníkov algoritam**) je $\mathcal{O}(N^2)$, gdje je N broj čvorova. V. `https://en.wikipedia.org/wiki/Prim%27s_algorithm`

Literatura

E. Alpaydin. *Introduction to machine learning*. MIT press, 2020.

J. A. Bilmes et al. A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute*, 4(510):126, 1998.

C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1): 1–22, 1977.

J. Diebolt and C. P. Robert. Estimation of finite mixture distributions through bayesian sampling. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(2):363–375, 1994.

J. C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. 1973.

R. A. Levine and G. Casella. Implementations of the Monte Carlo EM algorithm. *Journal of Computational and Graphical Statistics*, 10(3):422–439, 2001.

F. Murtagh and P. Contreras. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.

A. Roche. EM algorithm and variants: An informal tutorial. *arXiv preprint arXiv:1105.1476*, 2011.

J. H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

C. J. Wu. On the convergence properties of the EM algorithm. *The Annals of statistics*, pages 95–103, 1983.

M. Zaheer, M. Wick, S. Kottur, and J.-B. Tristan. Comparing gibbs, em and sem for map inference in mixture models. 2015.