19. Grupiranje

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

Jan Šnajder, predavanja, v2.1

Do sada smo se na ovom kolegiju bavili nadziranim strojnim učenjem. Kod nadziranog strojnog učenja, primjeri za učenje su označeni, tj., to su parovi $(\mathbf{x}, y)$, i za svaki nam je primjer poznata ciljna klasa y (ako radimo klasifikaciju) ili ciljna vrijednost y (ako radimo regresiju). Informacija o ciljnoj klasi ili ciljnoj vrijednosti je vrlo jak signal koji imamo iz podataka.

Danas (i idući put) bavit ćemo se drugom stranom strojnog učenja: **nenadziranim strojnim učenjem** (engl. *unsupervised learning*). Kod nenadziranog strojnog učenja, nemamo označenih primjera. Očekivano, takav je problem teži, ali je i fascinantniji – kao naučiti nešto bez eksplicitnog signala koji bi nas vodio u pravom smjeru? Vidjet ćemo da postoje različiti pristupi nenadziranom učenju. Mi ćemo se usredotočiti na **grupiranje** (engl. *clustering*), kog kojega neoznačene primjere skupljamo u grupe, ovisno o tome koliko su primjeri međusobno slični. Ima raznih algoritama grupiranja: danas ćemo se usredotočiti na one najjednostavnije. Za početak, pričat ćemo općenito o nenadziranom strojnom učenju. Onda ćemo pogledati **algoritam K-sredina** i njegovo poopćenje, **algoritam K-medoida**. Na kraju, pričat ćemo o vrednovanju grupiranja, odnosno tzv. **provjeri grupiranja**. 1

1 Nenadzirano učenje

Osnovna motivacija za nenadzirano učenje jest što u mnogim slučajevima analize podataka jednostavno nemamo informaciju o tome koji primjer pripada kojoj klasi. To znači da umjesto skupa označenih primjera:

$$\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$$

raspolažemo skupom **neoznačenih primjera** (engl. *unlabeled instances*): 2

$$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$$

Primjeri mogu biti neoznačeni iz dva razloga: (1) ne znamo ih označiti (ne znamo unaprijed koje klase postoje) ili (2) znamo koje klase postoje, ali označavanje je teško izvedivo (npr., preskupo je). Evo nekih zadataka u kojima bismo tipično koristili nenadzirano strojno učenje:

- Grupiranje klijenata prema ponašanju (segmentacija korisnika) – ne znamo unaprijed koji segmenti korisnika postoje, ali znamo da svi korisnici nisu isti i pretpostavljamo da postoji nekoliko vrsta prototipnih korisnika (korisnika koji se, uz neka manja odstupanja, u prosjeku slično ponašaju);
- Grupiranje novinskih članaka prema temama – ne znamo unaprijed koje se sve teme javljaju u novinskim tekstovima, ali znamo da razni novinski tekstovi obrađuju različite teme (pri čemu jedan novinski tekst može istovremeno obrađivati više tema), pa želimo grupirati novinske tekstove po temama;
- Grupiranje tekstova prema autorima – raspolažemo zbirkom tekstova anonimnih autora za koje pretpostavljamo da ih je pisalo nekoliko autora. Tekstove želimo grupirati tako da tekstovi istog autora završe u istoj grupi, a tekstovi različitih autora u različitim grupama; 3

- Analiza sličnosti rješenja laboratorijskih vježbi – zanima nas koja su rješenja laboratorijskih vježbi međuosbno slična. Ovdje želimo dobiti grupe rješenja koja su međusobno slična, tako da svi oni koji su međusobno prepisivali ili su prepisivali od istog autora završe u istoj grupi i mogu biti zajednički sankcionirani (neugodan primjer, ali događa se);
- Grupiranje gena sa sličnom izražajnošću (funkcionalnošću) – ne znamo unaprijed koje su sve moguće funkcionalnosti, ali znamo da neki geni imaju sličnu funkcionalnost;
- Klasifikacija objekata na fotografiji (engl. *content-based image retrieval*) – ako ne postoje unaprijed definirane klase objekata, bolje nam je da rezultate pretraživanja slika jednostavno grupiramo po sličnosti, u nadi da će slike s istim objektima završiti u istim grupama;
- Klasifikacija poruka s Twittera prema iskazanoj emociji – označavanje emocija u porukama Twittera naporno je i skupo. Mnogo je lakše poruke grupirati na način da one poruke koje izražavaju iste ili slične emocije završe u istim grupama. Štoviše, možda niti ne znamo unaprijed koje sve emocije su izražene u skupu podataka kojim raspolažemo (postoje različite teorije i modeli emocija), pa je možda bolje pustiti da te emocije "isplivaju" same od sebe putem grupiranja; 4
- Otkrivanje napada korisnika na mreži (engl. *intrusion detection*) – ovo je primjer **otkrivanja vrijednosti koje odskaču** (engl. *outlier detection*). Zanima nas je li ponašanje korisnika na mreži bitno drugačije od ponašanja svih drugih korisnika. Ako je to slučaj, onda je to sumnjivo i moguće štetno ponašanje.

Kao što smo napomenuli u uvodu, nenadzirano učenje obuhvaća niz metoda, a svima njima je zajedničko to što rade na neoznačenih podatcima. Međutim, različite se metode koriste za različite zadatke. Četiri su osnovna zadatka nenadziranog učenja:

1. **Grupiranje** (engl. *clustering*) – nalaženje grupa sličnih primjera;
2. **Procjena gustoće** (engl. *density estimation*) – nalaženje gustoće vjerojatnosti $p(\mathbf{x})$ koja opisuje distribuciju neoznačenih podataka;
3. **Otkrivanje novih/stršećih vrijednosti** (engl. *novelty/outlier detection*) – nalaženje primjera koji po svojim značajkama bitno odskaču od većine ostalih primjera u skupu podataka;
4. **Smanjenje dimenzionalnosti** (engl. *dimensionality reduction*) – smanjenje broja značajki, odnosno preslikavanje podataka iz izvornog ulaznog prostora u prostor smanjenih dimenzija, s ciljem otkrivanja manjeg broja novih (tzv. latentnih) značajki odnosno dimenzija koje dovoljno dobro opisuju različitosti u podatcima.

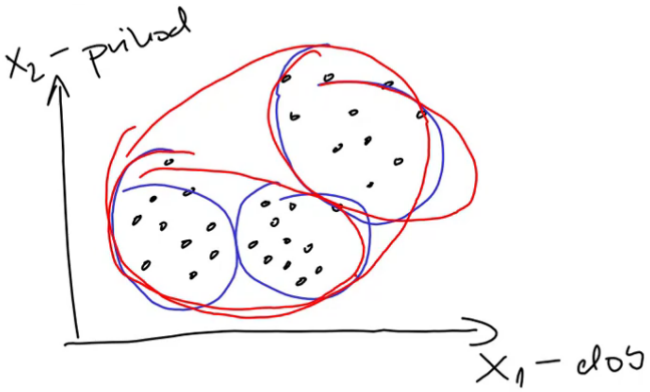
U nastavku ćemo se baviti isključivo grupiranjem podataka. Taj se oblik nenadziranog učenja u praksi daleko najčešće koristi.

2 Grupiranje

Grupiranje (engl. *clustering*) jest postupak razdjeljivanja primjera u grupe (engl. *clusters*), tako da **slični** primjeri (slični po nekom svojstvu) budu svrstani u istu grupu, a različiti primjeri u različite grupe. Svrha grupiranja jest nalaženje "prirodnih" (intrinzičnih) grupa u skupu neoznačenih podataka. Ili, pjesnički rečeno, svrha grupiranja jest "pustiti podatke da govore sami za sebe".

▸ **PRIMJER**

Želimo grupirati korisnike prema prihodu i dobi. Dakle, ulazni prostor je dvodimenzijski, i svaki je korisnik prikazan dvodimenzijskim vektorom $\mathbf{x} = (x_1, x_2)$, gdje je značajka x_1 prihod a značajka x_2 je dob. Skup podataka neka je ovakav:



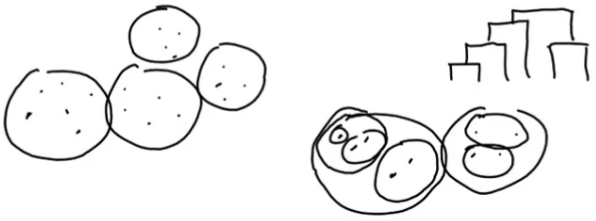
Neka je sličnost između primjera definirana kao inverz (euklidske) udaljenosti između vektora u dvodimenzijskome ulaznom prostoru, tj. što su primjeri u ulaznom prostoru bliži jedan drugome, to ih smatramo sličnijima. (Općenito, sličnost možemo definirati i na neki drugi način, no ideju je najlakše ilustrirati ako koristimo euklidsku udaljenost odnosno bliskost.) S obzirom na sličnost između primjera, ovaj skup podataka možemo grupirati u tri grupe (označene plavom bojom). Međutim, primjere bismo isto tako mogli grupirati u dvije grupe (označene crvenom bojom), ako bismo primjere u donje dvije plave grupe smatrali dovoljno sličnima (dovoljno bliskima) da čine jednu grupu. Ovo ilustrira općenit problem kod grupiranja, koji će nas pratiti cijelo vrijeme: često postoji više "prirodnih grupiranja", odnosno primjere je moguće grupirati u različit broj grupa, ovisno o tome kada primjere smatramo "dovoljno sličnim" da bismo ih smjestili u istu grupu.

2.1 Vrste grupiranja

Postoje različite vrste grupiranja. Razmotrit ćemo dvije podjele. Prva podjela odnosi se na to generira li algoritam grupe primjera koje imaju nekakvu internu strukturu (podgrupe primjera) ili generira "plošne" grupe koje nemaju nikakvu internu strukturu. Tako razlikujemo:

- **Particijsko grupiranje** – grupe su plošne (engl. *flat*), tj. ne postoje podgrupe;
- **Hijerarhijsko grupiranje** – grupe imaju podgrupe, koje imaju svoje podpodgrupe, i tako dalje, rekurzivno. Drugim riječima, postupak rezultira hijerarhijom grupa.

Na donjoj slici lijevo prikazan je rezultat particijskog grupiranja, a na slici desno rezultat hijerarhijskog grupiranja:

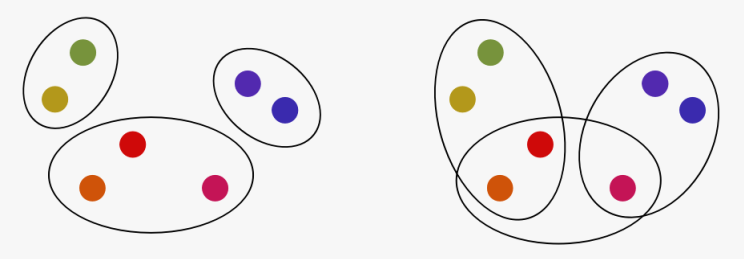


Hijerarhijsko grupiranje rezultira strukturom koje se može prikazati stablom, tzv. **dendrogramom**, gdje su u listovima tog stabla primjeri koje grupiramo. Više o tome idući put.

Neovisno o tome je li grupiranje particijsko ili hijerarhijsko, moguće je grupiranje ostvariti tako da jedan te isti primjer pripada uvijek samo jednoj (pod)grupi, ili dopustiti da primjer

istovremeno pripada u više (pod)grupa. U tom smislu razlikujemo:

- **Čvrsto grupiranje** (engl. *hard clustering*) – jedan primjer može pripadati jednoj i samo jednoj (pod)grupi;
- **Meko grupiranje** (engl. *soft clustering*) – jedan primjer može istovremeno pripadati u više grupa (npr. algoritam fuzzy k-means) ili može biti pridijeljen u više grupa s nekom vjerojatnošću, što modelira našu nesigurnost o njegovoj stvarnoj pripadnosti (npr. **probabilističko grupiranje** algoritmom GMM, o čemu ćemo pričati idući put). Na donjoj slici prikazana je razlika između čvrstog grupiranja (lijevo) i mekog grupiranja (desno):



2.2 Grupiranje kao predobrada za nadzirano učenje

Ako su primjeri označeni, onda možemo koristiti nadzirano učenje i ne treba nam grupiranje. Međutim, ponekad je korisno koristiti grupiranje kao predobradu podataka, dakle prije primjene nadziranog učenja. U tom slučaju na grupiranje možemo gledati kao na metodu smanjenja dimenzionalnosti. Tu imamo dvije mogućnosti. Prva je da grupiramo primjere, a druga da grupiramo značajke.

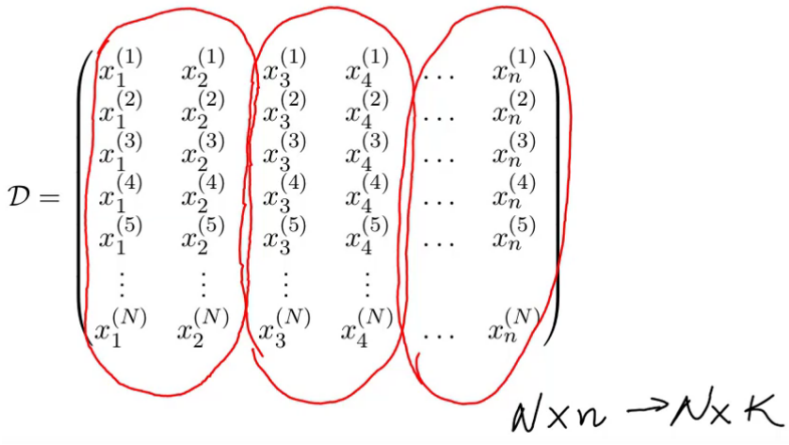
Grupiranje primjera svodi se na grupiranje redaka matrice dizajna. Na primjer:

$$\mathcal{D} = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} & \dots & x_n^{(2)} \\ x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & x_4^{(3)} & \dots & x_n^{(3)} \\ x_1^{(4)} & x_2^{(4)} & x_3^{(4)} & x_4^{(4)} & \dots & x_n^{(4)} \\ x_1^{(5)} & x_2^{(5)} & x_3^{(5)} & x_4^{(5)} & \dots & x_n^{(5)} \\ \vdots & \vdots & \vdots & \vdots & & \\ x_1^{(N)} & x_2^{(N)} & x_3^{(N)} & x_4^{(N)} & \dots & x_n^{(N)} \end{pmatrix}$$

$N \times n \rightarrow K \times n$

Rezultat će biti matrica dizajna s manje redaka (točno onoliko redaka koliko imamo grupa). To jest, u postupak ulazimo s matricom dizajna dimenzija $N \times n$, a dobivamo matricu dizajna dimenzija $K \times n$, gdje je K broj grupa i $K < N$. Svaku grupu sada možemo prikazati kao jedan primjer, npr., centroid (središnji vektor) svih primjera u toj grupi. Na taj smo način efektivno proveli **zaglađivanje primjera** (engl. *smoothing*). Možete se pitati zašto bismo ovo htjeli raditi, jer time očito smanjujemo broj primjera. No, neki algoritmi nadziranog učenja možda će bolje raditi ako im na ulaz dovedemo manji broj doista reprezentativnih primjera, nego ako ih obasipamo primjerima koji se razlikuju u šumu. Također, imajte na umu da ovime efektivno kombiniramo dvije induktivne pristranosti: one algoritma grupiranja s onom algoritma nadziranog učenja. U nekim slučajevima moglo bi biti da takva kombinacija daje dobre rezultate.

Alternativno, umjesto da grupiramo primjere (retke matrice dizajna), možemo grupirati značajke (stupce matrice dizajna). Na primjer:



Time matricu dizajna dimenzija $N \times n$ smanjujemo na matricu $N \times K$, gdje $K < n$. Ponovno možemo za svaki stupac izračunati centroid, čime više značajki svodimo na jednu reprezentativnu značajku. Ovdje, dakle, zadržavamo sve primjere iz izvornog skupa podataka, ali smanjujemo broj značajki, tako da značajke koje su međusobno vrlo slične zapravo svodimo na jednu reprezentativnu značajku, što opet moguće smanjuje šum.

U nastavku pogledajmo napokon jedan konkretan algoritam grupiranja.

3 Algoritam K-sredina

Najjednostavniji i najpoznatiji algoritam grupiranja jest **algoritam k-sredina** (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 (broj grupa) zadaje unaprijed. Primijetite da se ovdje radi o **particijskom grupiranju**, jer imamo čvrste grupe (svaki primjer pripada samo jednoj grupi). 5

Ideja algoritma jest da svaka grupa ima svoju srednju vrijednost (centroid) koja predstavlja grupu. Svaki primjer pripada grupi čiji mu je centroid najbliži (po euklidskoj udaljenosti). Postupak grupiranja je iterativan. Krenuvši od K slučajno odabranih sredina (centroida grupa), svi se primjeri svrstavaju u onu grupu čiji im je centroid najbliži. To može dovesti do pomaka centroida, stoga se u idućem koraku, nakon stvrstavanja svih primjera u njima najbližu grupu, ponovno izračunavaju novi centroidi za svaku grupu. No, nakon što su se izračunali novi centroidi, to može dovesti do promjene u pripadanju primjera grupama, pa se primjeri ponovo svrstavaju u grupe tako da svaki primjer bude u grupi čiji im je centroid najbliži. To novo razvrstavanje primjera ponovo može dovesti do promjena centroida, pa se ponovno izračunavaju centroidi za svaku grupu, i tako dalje. Postupak ponavlja ova dva koraka (pridjeljivanje primjera grupama i izračun centroida) sve do konvergencije (dok nema promjene u pripadnosti primjera grupama, odnosno dok nema promjene u centroidima grupa).

3.1 Formalna definicija

Pogledajmo to sada formalno. Mnogi algoritmi grupiranja mogu se formalizirati (i izvesti) tako da se definira **funkcija pogreške** koju minimiziraju. Ta se funkcija u kontekstu algoritama grupiranja često naziva **kriterijska funkcija**. Označit ćemo je sa J. Za algoritam K-sredina, kriterijska je funkcija definirana ovako:

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

Intuitivno, ova funkcija zbraja koliko primjeri unutar svake grupe odstupaju od centroida dotične grupe (primijetite da imamo dvije sume, jedna ide po grupama, a druga po primje-

rima). Preciznije, svaka grupa predstavljena je svojim centroidom, $\{\boldsymbol{\mu}_k\}_{k=1}^K$. Oznaka $\|\cdot\|$ je L_2-norma (tj. euklidska norma), definirana kao $\|\mathbf{x}-\boldsymbol{\mu}\|^2 = (\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}(\mathbf{x}-\boldsymbol{\mu})$, pa je dakle $\|\mathbf{x}^{(i)}-\boldsymbol{\mu}\|^2$ kvadrat euklidske udaljenosti između primjera $\mathbf{x}^{(i)}$ i centroida $\boldsymbol{\mu}$. Vrijednost $b_j^{(i)}$ je binarna indikatorska varijabla koja indicira pripada li primjer $\mathbf{x}^{(i)}$ grupi k: ako $b_k^{(i)}=1$, onda primjer $\mathbf{x}^{(i)}$ pripada grupi k, inače joj ne pripada. Prema tome, ukupna pogreška jednaka je zbroju, po svim primjerima, kvadrata euklidske udaljenosti primjera $\mathbf{x}^{(i)}$ od središta $\boldsymbol{\mu}_k$ grupe u koju je taj primjeri svrstan (za grupe u koje primjer $\mathbf{x}^{(i)}$ nije svrstan vrijedi $b_k^{(i)}=0$ i tu nemamo doprinosa pogrešci).

Mi, naravno, želimo pronaći ono grupiranje koje minimizira pogrešku. Grupiranje je definirano dvama parametrima: indikatorskim varijablama $b_k^{(i)}$ (one definiraju kojoj grupi pripada koji primjer) i centroidima grupa $\boldsymbol{\mu}_k$ (oni definiraju gdje u prostoru primjera se grupe nalaze). Formalno, dakle, tražimo parametre takve da:

$$\underset{b_1,\ldots,b_K;\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_K}{\operatorname{argmin}} \quad J$$

Očito, pogreška će biti to veća što su primjeri dalje od centroida svoje grupe. To znači da, 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:

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

Algoritam K-sredina radi tako da minimizira upravo kriterijsku funkciju J, ali to ne radi analitički jer ovaj optimizacijski problem nema rješenje u zatvorenoj formi. Naime, ako pokušamo derivirati funkciju J po oba parametra, nailazimo na problem jer su ti parametri međusobno ovisni: $b_k^{(i)}$ ovise o $\boldsymbol{\mu}_k$ i, obrnuto, $\boldsymbol{\mu}_k$ ovise o $b_k^{(i)}$. Umjesto toga, algoritam K-sredina optimizaciju provodi iterativno. Algoritam započinje sa slučajno odabranim srednjim vrijednostima $\boldsymbol{\mu}_k$. Zatim se u svakoj iteraciji temeljem izraza za $b_k^{(i)}$ 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 izraz za kriterijsku funkciju J. Konkretno, postavljenjem $\nabla_{\boldsymbol{\mu}_k} J = \mathbf{0}$ 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

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

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 za $b_k^{(i)}$ i ponovno izračunati koji primjeri pripadaju grupi k. Ova se dva koraka ponavljaju sve dok se ne dosegne stacionarno stanje, odnosno stanje u kojemu nema daljnjih promjena vrijednosti $\boldsymbol{\mu}_k$.

Kombinirajmo sada sve ovo u algoritam:

▸ **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

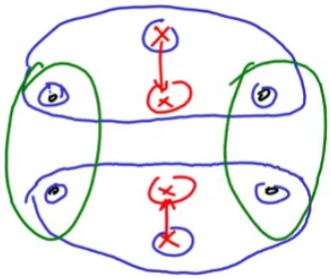
3.2 Svojstva algoritma

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), budući da je svaki primjer pridodan samo jednome centroidu (a za sve ostale centroide k vrijedi $b_k^{(i)} = 0$), to znači da ćemo efektivno iterirati samo jednom kroz cijeli skup primjera, pa je složenost $\mathcal{O}(nN)$. Ako još uvedemo T kao ukupan broj iteracija, onda je ukupna vremenska složenost algoritma $\mathcal{O}(TnNK)$. Složenost je, dakle, linearna u svim relevantnim parametrima, i to je vrlo dobro jer znači da će se algoritam dobro nositi i s velikim skupovima podataka i sa primjerima s mnogo značajki, a i sa zadatcima u kojima je potrebno grupirati u velik broj grupa. U praksi je broj iteracija T redovito mnogo manji od broja primjera N, pa broj iteracija nije ključan faktor pri razmatranju složenosti. 6

Osim složenosti, zanimat će nas je li ovaj algoritam deterministički, konvergira li i je li optimalan. Razmotrimo prvo determinističnost. Je li ovaj algoritam **deterministički**, u smislu da će dati uvijek iste izlaze za iste ulaze? Primijetite da unutar algoritma treba odabrati početna središta grupa. Očito, ako se taj odabir radi nekako stohastički, onda će i algoritam biti stohastički i time nedeterministički. No, čak i ako je odabir početnih središta deterministički, algoritam ima potencijalni dodatni 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). Zaključujemo, dakle, da algoritam može biti deterministički, ako se početna središta određuju nekim deterministčkim postupkom, a inače će biti stohastički.

Drugo pitanje jest konvergira li algoritam. Za početak, primijetimo da algoritam zapravo pretražuje prostor stanja. Svako stanje je jedno grupiranje (jedna particija primjera + raspored središta). Koliko različitih stanja postoji? Različitih stanja ima K^N (to je broj particija N primjera u K skupova). Dakle, konvergira li algoritam? Da, jer je broj konfiguracija konačan (konfiguracija = particija primjera + raspored središta), a optimizacijski je postupak definiran tako da se kriterijska funkcija J nužno smanjuje kroz iteracije. Iz ta dva svojstva slijedi da algoritam nikada ne posjećuje istu konfiguraciju više puta, iz čega slijedi da mora konvergirati.

Konačno, treće pitanje je nalazi li algoritam, jednom kada konvergira, optimalnu particiju. Odgovor na ovo je negativan: algoritam je pohlepan i nalazi lokalno optimalno rješenje, koje ne mora biti globalno optimalno. Rezultat grupiranja, naime, ovisi o odabiru početnih središta. Na primjer:



Ovdje imamo četiri primjera (crne točke). Ako su početno odabrana dva centroida označena plavim križićima, algoritam će konvergirati do centroida označenih crvenim križićima, i grupirati četiri primjera u dvije grupe označene plavom. Međutim, to je suboptimalna particija. Bolje grupiranje bilo bi ono u dvije grupe označene zelenom bojom. To grupiranje bi imalo manji iznos J, jer bi primjeri bili bliže svojim centroidima.

Možemo, dakle, utvrditi da je algoritam K-sredina pohlepan i da pronalazi lokalno optimalno rješenje. Hoće li to rješenje biti i globalno optimalno, ovisi o izboru početnih sredina $\boldsymbol{\mu}_k$. To onda pitanje optimalnosti algoritma prebacuje na pitanje optimalnog odabira početnih sredina. Pogledajmo koje tu sve mogućnosti imamo.

3.3 Odabir početnih sredina

- **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 to bi lako moglo dogoditi ako središta izaberemo posve nasumično), ali se ne rješava problem zaglavljivanja u lokalnom optimumu. Problem također predstavljaju **stršeći primjeri** (engl. *outliers*), koji lako mogu završiti izolirani svaki u svojoj grupami. 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 vektoru $\boldsymbol{\mu}$ dodavati manje **slučajne vektore** i tako dobiti K vektora $\boldsymbol{\mu}_k$. Ovo rješava problem izoliranih primjera, jer će središta biti centrirana oko "centra mase" podataka (oko točaka u ulaznom prostoru gdje je gustoća vjerojatnosti primjera najveća), no ovo ne rješava problem zaglavljivanja u lokalnome optimumu;
- Izračunati prvu glavnu komponentu skupa primjera metodom **analize glavnih komponenti** (engl. *principal component analysis, PCA*) (prva komponenta definira smjer najveće varijance podataka), razdijeliti raspon na K jedakih intervala, čime se primjeri razdjeljuju u K grupa, a zatim uzeti srednje vrijednosti tih grupa kao početne vrijednosti $\boldsymbol{\mu}_k$. Ovo u načelu također ne rješava problem zaglavljivanja u lokalnom optimumu; 7
- Slučajno odabrati jedno početno središte $\boldsymbol{\mu}_k$, a zatim svako iduće središte odabrati tako da je što dalje od ostalih središta. Algoritam koji implementira ovakav pristup poznat je pod nazivom **k-means++**. Kod tog je algoritma vjerojatnost da primjer $\mathbf{x}^{(i)}$ bude odabran kao novo središte $\boldsymbol{\mu}_{k+1}$ proporcionalna kvadratu udaljenosti tog primjera od njemu najbližeg, već odabranog središta $\boldsymbol{\mu}_k$:

$$P(\boldsymbol{\mu}_{k+1} = \mathbf{x}^{(i)} | \mathcal{D}, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k) = \frac{\min_k \|\boldsymbol{\mu}_k - \mathbf{x}^{(i)}\|^2}{\sum_j \min_k \|\boldsymbol{\mu}_k - \mathbf{x}^{(j)}\|^2}$$

  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, npr., algoritam k-means++, običaj je algoritam K-sredina pokrenuti više puta i uzeti rezultat sa što manjom pogreškom grupiranja J.

4 Algoritam K-medoida

Algoritam K-sredina jednostavan je i učinkovit algoritam. Međutim, ima jedno veliko ograničenje: primjenjiv je samo u slučajevima kada je primjere moguće prikazati u **vektorskome prostoru**, pa je između njih moguće izračunati euklidsku udaljenost i srednju vrijednost (centroid) grupe. No, kao što smo već ustanovili kada smo pričali o jezgrenim funkcijama, nekada primjere ne možemo prikazati u vektorskom prostoru. Npr., ako želimo grupirati riječi tako da ortografski slične riječi završe u istim grupama, nije skroz jasno kako bismo najbolje prikazali riječi kao vektore. Umjesto toga, mnogo je jednostavije izračunati sličnost između riječi nekom funkcijom koja za par riječi vraća njihovu sličnost (npr., na temelju Levenshteinove udaljenosti). Slično je i ako želimo grupirati ljude na temelju poznanstva, tako da ljudi koji se međusobno poznaju završe u istim grupama. Mnogo je lakše za svaku osobu prikupiti informaciju o tome koliko dobro poznaje neku drugu osobu, nego osobe prikazati kao vektore tako da euklidska udaljenost između tih vektora odgovara upravo mjeri u kojoj se te osobe međusobno poznaju.

Zajedničko ovakvim situacijama jest da primjeri nisu elementi vektorskog prostora, pa između njih ne možemo izračunati euklidsku udaljenost niti možemo izračunati centroid primjera. Umjesto toga, raspolažemo općenitom **mjerom sličnosti** (engl. *similarity measure*) ili njezinim komplementom, **mjerom različitosti** (engl. *dissimilarity measure*), definiranima između svih parova primjera. Mjera sličnosti može se izraziti **matricom sličnosti**.

▸ **PRIMJER**

Matrica sličnosti je simetrična kvadratna matrica koja definira sličnost između svih parova primjera u skupu primjera. Npr.:

$$S = \begin{pmatrix} 1 & 0.1 & 0.9 & 0.4 \\ 0.1 & 1 & 0.7 & 0.3 \\ 0.9 & 0.7 & 1 & 0.2 \\ 0.4 & 0.3 & 0.2 & 1 \end{pmatrix}$$

Osim što je simetrična, sličnost je također i refleksivna relacija, tj. svaki primjer je sličan samome sebi sa sličnošću 1, stoga su na dijagonali matrice sličnosti jedinice. Mjeru različitosti možemo dobiti jednostavno kao komplement mjere sličnosti, $1 - S$.

Primijetite da je mjera sličnosti zapravo isto što i jezgrena funkcija, pa je matrica sličnosti zapravo isto što i **jezgrena matrica**.

Algoritam K-sredina ne može, dakle, raditi s primjerima koji nisu vektori. Međutim, princip tog algoritma – nalaženje prototipa grupa (centroida) i zatim pridjeljivanje primjera najsličnijoj (najbližoj) grupi – može se poopćiti na slučaj kada primjeri nisu vektori. Upravo to radi **algoritam K-medoida**. Taj algoritam je poopćenje algoritma K-sredina koji može raditi s općenitom mjerom različitosti između (moguće nevektorskih) primjera. Kod tog je algoritma kriterijska funkcija definirana pomoću općenite mjere različitosti $\nu(\mathbf{x}, \mathbf{x}')$ između dvaju primjera:

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

Mjera različitosti ν (odnosno njoj komplementarna mjera sličnosti) općenitija je od euklidske udaljenosti i od bilo koje druge mjere udaljenosti budući da ne mora ispunjavati uvjete metrike (uključivo nejednakost trokuta).

Kod algoritma K-medoida prototipe grupa čine tzv. **medoidi**, odnosno reprezentativni primjeri iz skupa $\mathcal{D}$, a ne centroidi. Važno je napomenuti da medoid može biti samo neki od primjera iz skupa za učenje. To je drugačije od centroida, koji ne mora odgovarati niti jednom primjeru iz skupa za učenje, tj. to može biti neki nikad viđeni primjer. Razlog zašto medoid može biti samo jedan od primjera iz skupa za učenje je taj što primjeri nisu nužno vektorizirani, pa dakle ne možemo izračunati centroid, već primjere shvaćamo kao nedjeljive entitete, i jedino što o njima znamo jesu sličnosti/različitosti između primjera.

Tipična izvedba algoritma K-medoida jest **algoritam PAM** (engl. *partitioning around medoids*), čiji je pseudokod dan u nastavku.

▸ **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 $\boldsymbol{\mu}_k \in \mathcal{M}$
    $\boldsymbol{\mu}_k \leftarrow \operatorname{argmin}_{\boldsymbol{\mu}_j \in \mathcal{D} \backslash \mathcal{M} \cup \{\boldsymbol{\mu}_k\}} \sum_i b_k^{(i)} \nu(\mathbf{x}^{(i)}, \boldsymbol{\mu}_j)$
dok $\boldsymbol{\mu}_k$ ne konvergiraju

Kao i algoritam K-sredina, algoritam PAM također se 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 razmatraju se kao kandidati za medoid umjesto trenutačnog medoida, izračunava se zbroj mjere ν između $N-K$ primjera i kandidata za medoid te se odabire onaj medoid za koji je ta vrijednost najmanja. To iziskuje $\mathcal{O}(K(N-K)^2)$ izračuna mjere različitosti ν. Posljedično, algoritam PAM ukupno iziskuje $\mathcal{O}(TK(N-K)^2)$ izračuna mjere ν, gdje je T ukupan broj iteracija. Vidimo dakle da sada imamo kvadratnu složenost u N, dok smo kod algoritma K-sredina imali linearnu složenost. Visoka vremenska složenost glavni je nedostatak algoritma PAM, pa su predložena različita poboljšanja, u koja međutim sada nećemo ulaziti.

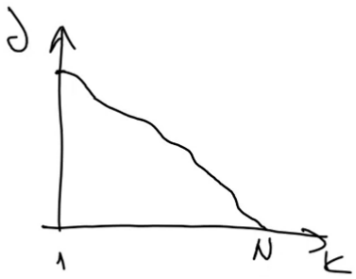
8

Umjesto toga, pozabavit ćemo se jednom važnijom temom, a to je odabir broja grupa, odnosno problemom "provjere grupiranja".

5 Provjera grupa

Kod oba algoritma koje smo razmotrili – algoritma K-sredina i algoritma K-medoida – broj grupa, odnosno **hiperparametar** K, potrebno je odrediti unaprijed. To je slučaj kod mnogih algoritama grupiranja (iznimka su tzv. neparametarski algoritmi grupiranja, ali njih nećemo raditi). Odabir optimalnog broja grupa dio je većeg problema koji se naziva **provjera grupa** (engl. *cluster validation*): *koliko je dobro naše grupiranje?* Odabir grupa jedan je od glavnih problema kod grupiranja. Idealno, broj grupa odgovarat će broju "prirodnih grupa" u skupu podataka, no taj nam je najčešće nepoznat. Pitanje je onda: kako odrediti optimalan broj grupa K, kada ne znamo unaprijed koliko grupa postoji u podatcima?

Prva stvar koja bi nam možda mogla pasti na pamet jest jednostavno odabrati K koji minimizira pogrešku J. Međutim, to nije dobra ideja. Naime, vrijednost pogreške J u ovisnosti o broju grupa K općenito izgleda ovako:



tj. pogreška J monotono pada s brojem grupa K. Pogreška J biti će jednaka nuli tek onda kada se svaki primjer nalazi sam u svojoj grupi, tj. onda kada je broj grupa jednak broju primjera. No, takvo "grupiranje" potpuno je beskorisno. Problem odabira broja grupa zapravo je analogan **odabiru složenosti modela** kod nadziranog učenja: grupiranje s velikim brojem grupa odgovara složenom modelu. I kod nadziranog smo učenja imali situaciju da najsloženiji model daje najmanju pogrešku.

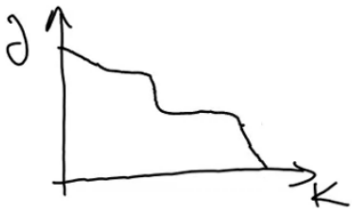
Dakle, minimizacija funkcije J nije opcija. Pogledajmo koje tehnike stvarno imamo na raspolaganju.

5.1 Tehnike provjere grupa

Ručna provjera kvalitete grupa. Inspiciramo primjere u pojedinim grupama i pokušamo dokučiti ima li takvo grupiranje smisla, pozivajući se na znanje o samom problemu koji rješavamo. Ovo je ograničeno upotrebljivo, jer ako primjera ima puno, teško da ćemo išta smisleno moći uočiti.

Smanjenje dimenzionalnosti plus vizualna provjera. Ovdje je ideja da primjere iz visokodimenzijskog ulaznog prostora (dimenzije n) preslikamo u dvodimenzijski prostor i da ih tamo vizualiziramo te očno utvrdimo optimalan broj grupa. Nakon toga primjere grupiramo u dotični broj grupa, ali u izvornom (n-dimenzijskom prostoru). Za projiciranje primjera u dvodimenzijski prostor može se upotrijebiti bilo koja tehnika za smanjenje dimenzionalnosti; tipično se koristi **analiza glavnih komponenti** (PCA), **višedimenzijsko skaliranje** (engl. *multidimensional scaling, MDS*), **analiza korespondencije** (engl. *correspondence analysis, CA*) ili metoda **t-SNE** (ova posljednja je najpopularnija). 9

Metoda koljena (engl. *elbow method*). Grafički prikažemo ovisnost kriterijske funkcije o parametru K i tražimo "koljeno" krivulje (mjesto gdje funkcija naprije naglo pada i zatim stagnira ili pada vrlo sporo). S porastom broja grupa K, vrijednost kriterijske funkcije će padati, međutim taj pad općenito neće biti ujednačen. Naime, za neke vrijednosti K algoritam će početi razdjeljivati prirodne grupe. Ako povećanjem broja grupa kriterijska funkcija brzo opadne i onda neko vrijeme stagnira, to je signal da smo upravo "uhvatili" neko prirodno grupiranje u podatcima. Naime, ako s porastom K vrijednost J naglo opada, to znači da s malim povećanjem broja grupa dobivamo znatno bolje grupiranje. U trenutku kada s povećanjem broja grupa K više ne dobivamo znatno bolje grupiranje, tj. kada J počne vrlo sporo opadati, to onda znači da novododane grupe ne hvataju baš neke prirodno grupirane primjere. Takvih koljena funkcije J može biti više, pogotovo ako grupe unutar sebe imaju podgrupe. Dakle, dobar odabir za broj grupa K jesu one vrijednosti koje odgovaraju točkama neposredno nakon koljena funkcije J, jer smo na tim mjestima upravo pogodili neki "prirodni" broj grupa. Npr., na donjoj slici imamo jedno takvo koljeno: 10



Ako je algoritam grupiranja nedeterministički, npr., algoritam K-sredina sa slučajno odabranim početnim središtima, za svaki izbor vrijednosti K treba napraviti više mjerenja za J, pa uzeti srednju vrijednost.

Analiza siluete. Slično kao metoda koljena, analiza siluete je grafička metoda. Metoda se provodi tako da se grafički prikaže "vrijednost siluete" za svaki primjer iz skupa $\mathcal{D}$, i zatim se na temelju toga očno odredi optimalan broj grupa. Vrijednost siluete za primjer $\mathbf{x}^{(i)}$ definirana je ovako:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Vrijednost $a(i)$ je prosječna udaljenost primjera $\mathbf{x}^{(i)}$ do svih drugih primjera iz iste grupe u kojoj se nalazi primjer $\mathbf{x}^{(i)}$. Vrijednost $b(i)$ je također prosječna udaljenost primjera $\mathbf{x}^{(i)}$ do svih drugih primjera iz grupe, ali se sada razmatraju sve druge grupe u kojima se ne nalazi primjer $\mathbf{x}^{(i)}$, te se uzima grupa za koju je prosječna udaljenost najmanja. Dakle, $b(i)$ će biti najmanja prosječna udaljenost između primjera $\mathbf{x}^{(i)}$ i svih primjera najbliže susjedne grupe. Vrijednost siluete je u intervalu $[-1, +1]$. Ako je $s(i) = 1$, to znači da je primjer jako udaljen od primjera iz susjedne grupe, a vrlo blizu primjerima iz svoje grupe. Obrnuto, ako je $s(s) = -1$, to bi značilo da je primjer pogrešno grupiran jer je bliže primjerima iz druge grupe nego primjerima iz svoje grupe (to se ne može dogoditi kod algoritma K-sredina, ali se možda može dogoditi kod nekih drugih algoritama). Primjer zi koje $s(i) = 0$ nalaze se negdje na granici između dviju ili više grupa.

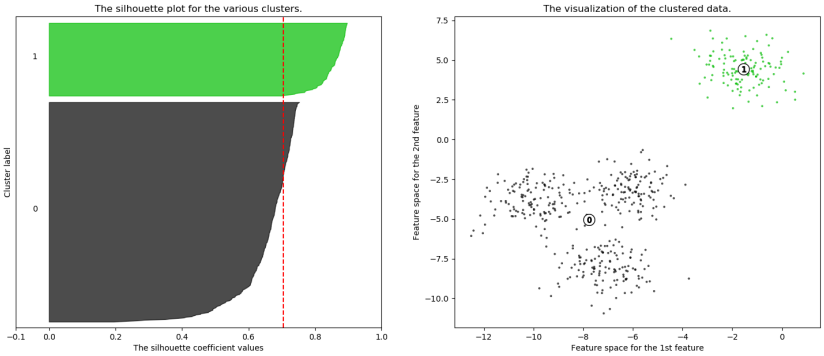
Ideja analize siluete jest da izračunavamo $s(i)$ za sve primjere iz $\mathcal{D}$ i grafički prikazujemo te vrijednosti. Također, izračunavamo prosječnu vrijednosti siluete za sve primjere iz skupa podataka. Pogledajmo primjer.

▸ **PRIMJER**

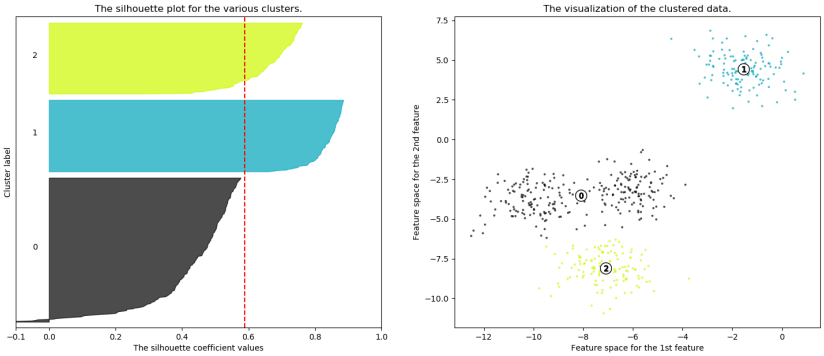
U nastavku je prikazana analiza siluete za algoritam K-sredina, za različit broj grupa, $K \in \{2, 3, 4, 5, 6\}$. Na desnoj strani prikazan je rezultat grupiranja u dvodimenzijskome ulaznom prostoru. Primjeru su, naravno, za svaki K jednako raspoređeni u ulaznom prostoru, ali su moguće drugačije grupirani. Boja primjera odgovara oznaci grupe u koju je primjer dodijeljen. (Primijetite da oznaka grupe kojoj primjer pripada nije poznata unaprijed, već se dobiva grupiranjem; kod algoritma K-sredina to je jednostavno grupa čijemu je centroidu primjer najbliži). Na lijevoj strani prikazana je silueta, i to tako da su na x-osi prikazane vrijednosti siluete za svaki primjer. Primjeri su grupirani prema oznaci grupe u koju pripadaju, i to u padajućem redoslijedu prema vrijednosti siluete. Okomita crvena iscrtkana linija označava prosjek vrijednosti siluete na cijelom skupu primjera.

11

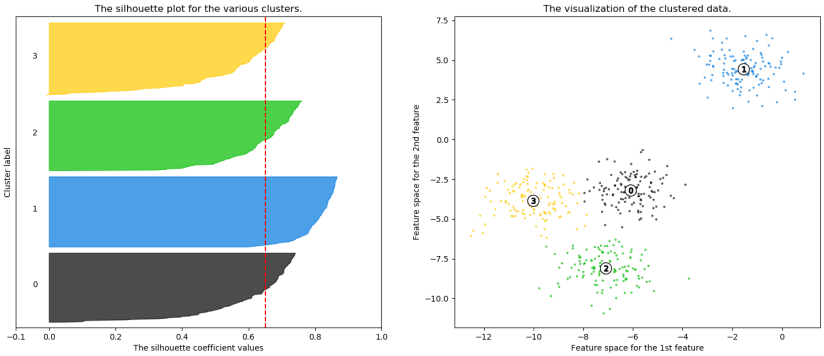
Silhouette analysis for KMeans clustering on sample data with n_clusters = 2



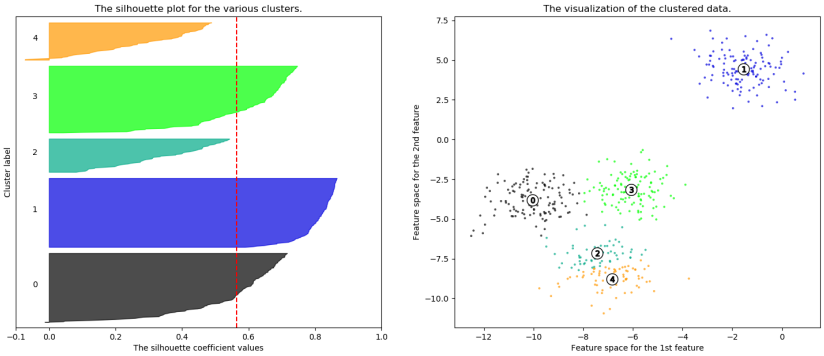
Silhouette analysis for KMeans clustering on sample data with n_clusters = 3

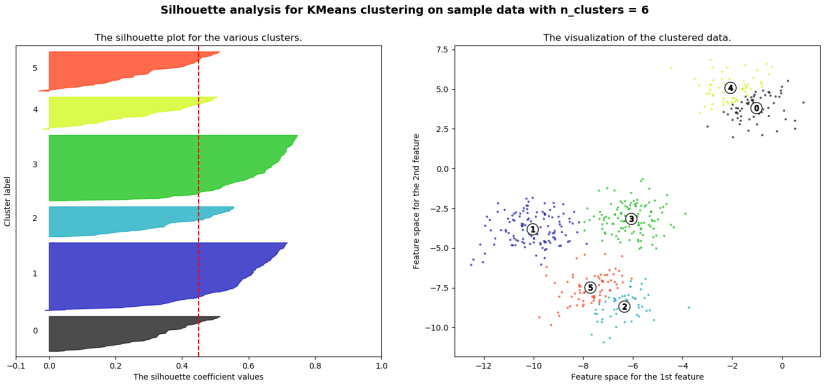


Silhouette analysis for KMeans clustering on sample data with n_clusters = 4



Silhouette analysis for KMeans clustering on sample data with n_clusters = 5





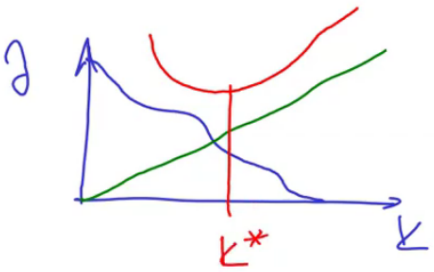
Lošim grupiranjima smatraju se ona grupiranja kod kojih su sve vrijednosti siluete za neku grupu manje od prosječne vrijednosti siluete za cijeli skup primjera. Ovdje su to grupiranja sa $K = 3$ i $K = 5$. Također, kod tih grupiranja postoji značajna varijanca u iznosima vrijednosti siluete. Npr., sa $K = 5$, kao što se vidi iz prikaza ulaznog prostora, narančasta grupa i tirkizna grupa preblizu su jedna drugoj i razdjeljivanje primjera u dvije grupe ovdje nije prirodno, što za posljedicu ima niske vrijednosti siluete za primjere koji su bliži drugoj grupi nego svojoj grupi. Za razliku od grupiranja sa $K = 3$ i $K = 5$, grupiranja sa $K = 2$, $K = 4$ i $K = 6$ su u redu. Zapravo, grupiranje sa $K = 4$ je možda najbolje jer su grupe približno jednakih veličina, što se lijepo vidi iz prikaza siluete (debljine silueta za svaku grupu su ujednačene).

U ovom konkretnom primjeru, budući da su primjeri u dvodimenzijskome ulaznom prostoru, grupe smo mogi provjeriti i izravno u ulaznom prostoru, međutim imajte na umu da metodu analize siluete možemo primijeniti i za grupiranje u visokodimenzijskom prostoru.

Minimizacija regularizirane funkcije pogreške. Ideja ovog postupka za odabir grupa slična je ideji **regularizacije**, koju smo kod nadziranog učenja koristili za sprječavanje prenaučenosti modela. Ideja je da proširimo kriterij grupiranja koji minimiziramo, tako da on kombinira izvornu kriterijsku funkciju i složenost modela te na neki način kažnjavamo modele s prevelikim brojem grupa. Općenit oblik tog kriterija je:

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

gdje je $J(K)$ vrijednost kriterijske funkcije za model s K grupa, a hiperparametar λ je faktor regularizacije. Veće vrijednosti faktora λ favoriziraju rješenja s manjim brojem grupa, budući da će kazna za veći broj grupa biti veća ako je λ veći. Za $\lambda = 0$ povećanje broja grupa uopće se ne kažnjava i optimalan broj grupa tada je $K^* = N$. Ovdje, dakle, tražimo minimum zbroja dviju funkcija, jedne koja monotono opada (funkcija J) i druge koja raste linearno (λK). To izgleda ovako:



Što je λ veći, to je veći nagib druge funkcije, i minimum K^* će biti to manji, tj. odabrat ćemo manji broj grupa. Ideja je slična regularizaciji utoliko što funkciju pogreške kombiniramo s regularizacijskim izrazom koji ovisi o složenosti modela. Razlika u odnosu na

regularizaciju kod nadziranog učenja je u tome što se regularizacija tamo provodi u sklopu optimizacije, dok je ovdje provodimo nakon optimizacije (tj. nakon grupiranja).

Naravno, tu je sad problem prebačen s odabira optimalnog K na odabir optimalne λ. Kako bismo odabrali optimalnu λ? (Unakrsna provjera ovog puta nije točan odgovor, jer nemamo označenih primjera, pa dakle niti nemamo ispitni skup primjera na kojima bismo mogli ispitati da li model dobro generalizira.) Ideja je da je odabir hiperparametra λ možda lakši problem nego odabir hiperparametra K. Odabir parametra λ može se temeljiti na našem iskustvu stečenome na grupiranju sličnih skupova podataka – skupova koji su možda imali drugačiji broj grupa, ali svi imaju približno istu λ, jer se odnose na isti ili sličan problem. Npr., zamislimo da želimo grupirati filmove tako da slični filmovi (po glumačkoj postavi, žanru, produkciji, sadržaju itd.) završe u istim grupama, i da to radimo na malom skupu podataka i zatim na velikom skupu podataka. Očekujemo da će broj grupa u ta dva slučaja biti različit, ali λ bi trebala biti više-manje ista, jer λ indirektno određuje što znači da su filmovi dovoljno slični da pripadaju u istu grupu, pa na taj način λ indirektno određuje broj grupa K u ovisnosti o veličini skupa primjera. Dakle, jednom određeni λ možemo koristiti za grupiranje na raznim skupovima podataka, dok god rješavamo približno sličan problem. Drugim riječima, na odabir hiperparametara λ možemo gledati kao na "meki odabir" broja grupa (odabir koji je prilagođen konkretnim podatcima). 12

Alternativa optimizaciji hiperparametra λ jest da koristimo neku teorijsku utemeljenu regulariziranu funkciju pogreške, poput **Akaikeova informacijskog kriterija (AIC)**. Konkretan oblik kriterija AIC za algoritam K-sredina je: 13

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

Ovdje koristimo $\lambda = 2n$, što znači da se za složenost modela uzima da je proporcionalna umnošku broja značajki n i broja grupa K.

Točnost na podskupu primjera. Ovaj se pristup temelji na usporedbi dobivenog grupiranja s grupiranjem koje smatramo točnim, tzv. **referentno grupiranje**. Naravno, mi ne znamo koje je točno grupiranje – kada bismo to znali, ne bismo ni trebali raditi grupiranje. No, referentno grupiranje možemo utvrditi na manjem uzorku primjera: za manji, slučajno odabran broj primjera odredit ćemo njihovu točnu grupu (ovdje pretpostavljamo da su nam grupe unaprijed poznate, ali nam nije poznata raspodjela primjera u grupe). Jednom kada smo to napravili, možemo grupirati zajedno označene i neoznačene primjere, a onda napraviti provjeru samo na označenim primjerima. Kao mjeru pogreške možemo koristiti bilo koju mjeru za točnost grupiranja: npr., **Randov indeks** (engl. *Rand index*), mjera **normalizirane uzajamne informacije** (engl. *normalized mutual information*, NMI) ili mjera F_1. O Randovom indeksu pričat ćemo u nastavku, a o ostalim mjerama u predavanju posvećenome vrednovanju modela. 14 15

5.2 Randov indeks

Pogledajmo sada **Randov indeks**. Randov indeks je mjera koja izračunava u kojoj mjeri dobiveno grupiranje odgovara referentnom grupiranju (grupiranju koje smatramo točnim). Randov indeks zapravo nije ništa drugo nego mjera **točnosti** izračunata na razini **parova primjera**. Što to znači? To znači da za svaki mogući par iz skupa primjera gledamo jesu li ta dva primjera završila u istoj grupi ili nisu. Ako su završili u istoj grupi, onda par primjera smatramo pozitivnim, inače ga smatramo negativnim. Ako su oba primjera iz para završila u istoj grupi, i ako su doista trebala završiti u istoj grupi, onda je taj par primjera **stvarno pozitivan** (engl. *true positive*, TP). Obrnuto, ako su primjeri iz para razdvojeni, tj. svaki je završio u drugoj grupi, a 16

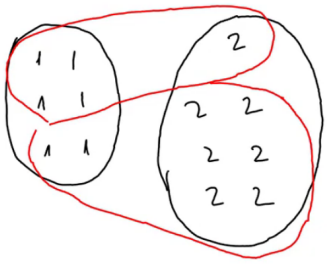
doista nisu trebala završiti u istoj grupi, onda je taj par primjera **stvarno negativan** (engl. *true negative*, TN). Randov indeks definiran je onda ovako:

$$R = \frac{a+b}{\binom{N}{2}}$$

gdje je a broj jednako označenih parova u istim grupama (tj. broj stvarno pozitivnih parova, TP), a b je broj različito označenih parova u različitim grupama (tj. broj stvarno negativnih parova, TN). Primijetite da u brojniku imamo $a + b = \text{TP} + \text{TN}$, što je ukupan broj točno grupiranih parova, dok je u nazivniku ukupan broj parova, kojih je $\binom{N}{2}$. Prema tome, Randov indeks je zapravo udio točno grupiranih parova mjera u ukupnom broju primjera, odnosno, kao što smo već rekli, to je točnost grupiranja izračunata na razini parova primjera.

▸ **PRIMJER**

Imamo $N = 13$ primjera. Referentno grupiranje sadrži dvije grupe – dakle, u stvarnosti primjeri zapravo dolaze iz dvije grupe. Primjere grupiramo u $K = 2$ grupe. Ovdje broj grupa K odgovara stvarnom broju grupa (broju grupa u referentnom grupiranju), no to ne mora uvijek biti tako (naprotiv, u stvarnosti ćemo rijetko pogoditi baš pravi broj grupa). Grupiranje je rezultiralo ovakvim dvjema grupama:



Crnom bojom označene su dvije referentne grupe (grupiranje koje smatramo točnim), a crvenom bojom dvije označene su grupe dobivene algoritmom grupiranja. Ovakav rezultat grupiranja možemo sažeto napisati ovako:

$$\{\{1,1,1,1,2\},\{1,1,2,2,2,2,2,2\}\}$$

gdje prvi i drugi skup odgovaraju prvoj odnosno drugoj dobivenoj grupi, a brojke označavaju iz koje je referentne grupe svaki od primjera. Vidimo da dobiveno grupiranje nije savršeno točno: u prvu grupu dobivenu algoritmom grupiranja ušuljao nam se jedan primjer iz druge referentne grupe, dok su u drugu grupu dobivenu algoritmom grupiranja upala dva primjera iz prve referentne grupe.

Na temelju ovih podataka lako možemo izračunati Randov indeks. Budući da imamo $N = 13$ primjera, to znači da je ukupan broj parova primjera jednak $\binom{N}{2} = \binom{13}{2} = 78$. U prvoj grupi imamo četiri primjera iz prve referentne grupe. Dakle, svi ti parovi primjera su ispravno grupirani. Ukupan broj tih točnih parova je $\binom{4}{2}$. Analogno računamo za ostale primjere u obje dobivene grupe.

Slično, vidimo da je naše grupiranje ispravno razdvojilo neke parove primjera. U prvoj našoj grupi su 4 primjera iz prve referentne grupe, a u drugoj grupi je 6 primjera iz druge referentne grupe. Dakle, $4 \cdot 6$ parova je ispravno razdvojeno. Nadalje, u prvoj grupi imamo 1 primjer iz druge referentne grupe, dok u drugoj grupi imamo 2 primjera iz prve referentne grupe, i ti su primjer također ispravno razdvojeni, što daje još $1 \cdot 2$ ispravno razvojenih parova. Imamo, dakle:

$$a = \binom{4}{2} + \binom{1}{2} + \binom{2}{2} + \binom{6}{2} = 6 + 1 + 0 + 15 = 22$$

$$b = 4 \cdot 6 + 1 \cdot 2 = 26$$

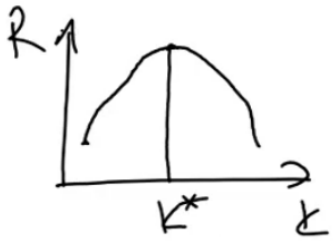
$$R = \frac{22 + 26}{78} = 0.62$$

Dobili smo Randov indeks od 0.62, odnosno 62%, što baš i ne zvuči tako dobro.

U ovom je primjeru odabrani broj grupa K i broj grupa u referentnom grupiranju bio identičan. No to općenito ne mora biti tako. Randov indeks funkcionira i kada je broj grupa različit od

referentnog broja grupa, npr., kada u referentnom grupiranju postoje tri grupe, a mi grupiramo u samo dvije grupe.

Randov indeks je mjera točnosti, pa će biti u intervalu $[0, 1]$. Što je grupiranje točnije, tj. što se više podudara s referentnim grupiranjem, to će Randov indeks biti veći. Korisno je razmotriti kako se vrijednost Randovog indeksa mijenja u ovisnosi broju grupa K. Ako je broj grupa premalen, onda će grupe biti velike, pa ćemo u iste grupe smještati mnoge parove primjera koji ne bi trebali biti u istoj grupi, tj. imat ćemo mnogo lažno pozitivnih parova primjera. S druge strane, ako je broj grupa prevelik, onda će grupe biti malene, pa ćemo razdvojiti mnoge parove primjera koji bi trebali biti smješteni u istu grupu, tj. imat ćemo mnogo lažno negativnih parova primjera. U oba slučaja, ako je K prevelik ili premalen, Randov indeks bit će manji nego kada odaberemo optimalan broj grupa, tj. broj grupa koji najbolje odgovara referentnom grupiranju. Zaključujemo, dakle, da je Randov indeks unimodalna funkcija broja grupa K: 17



Randov indeks, dakle, uspoređuje grupiranje s referentnim grupiranjem. Možemo ga koristiti ako nam je poznato referentno grupiranje. To je korisno u situacijama kada, npr., imamo standardni skup podataka za koji je poznato koje su grupe u njemu, pa isprobavamo različite algoritme grupiranja, uključivo neki novi algoritam koji smo razvili. Međutim, kao što smo već napomenuli, u stvarnim primjenama grupiranja najčešće ipak ne znamo koji primjeri pripadaju u koju grupu, pa onda Randov indeks računamo na manjem, ručno označenom uzorku primjera. 18

U praksi se umjesto Randovog indeksa češće koristi **prilagođen Randov indeks** (engl. *adjusted Rand index*), koji mjeru korigira s obzirom na očekivanu sličnost između slučajno grupiranih parova primjera. 19

Sažetak

- Ako primjeri nisu označeni (ne znamo ih označiti ili je preskupo) moramo koristiti **nenadzirano strojno učenje**
- Tipičan zadatak je **grupiranje**, kojim se primjeri razdjeljuju u grupe prema udaljenosti/sličnosti
- Grupiranje može biti **tvrdo** ili **meko**, **particijsko** ili **hijerarhijsko**
- **Algoritam K-sredina** računa središta grupa i primjere razdjeljuje u grupe s najbližim središtima, smanjujući postepeno funkciju pogreške. **Odabir početnih središta** utječe na grupiranje
- **Algoritam K-medoida (PAM)** je poopćenje na proizvoljnu mjeru različitosti (ne nužno udaljenost u vektorskom prostoru)
- Ključan problem grupiranja jest **odabir optimalnog broja grupa**, za što postoji nekoliko metoda
- **Randov-indeks** je mjera za **provjeru grupa** koja računa točnost grupiranja parova primjera u odnosu na referentno grupiranje

Bilješke

[1] Ovo predavanje zasniva se na poglavljima 7.1, 7.3 i 7.8 iz [Alpaydin, 2020] te poglavlju 9.1 iz [Bishop, 2006], ali je značajno prošireno. Ipak, u odnosu na nadzirano strojno učenje, nenadziranom strojnom učenju, odnosno konkretno grupiranju, posvetit ćemo zapravo vrlo malo pažnje. Ako će vas zainteresirati algoritmi grupiranja, predlažem da produbite svoje znanje tako da krenete od nekih od sljedećih preglednih radova: [Xu and Wunsch, 2005, Berkhin, 2006, Xu and Tian, 2015].

[2] Diskutabilno je bismo li **neoznačene primjere** željeli zvati "primjeri". Točnije bi bilo "instance". Međutim, u nastavku nećemo komplicirati, zvat ćemo ih i dalje "primjeri", ali imajte na umu da nemamo oznaku y, nego samo vektor $\mathbf{x}$.

[3] Zadatak otkrivanja autora teksta naziva se **atribucija autorstva** (engl. *authorship atribution*). Nenadzirane metode, uključivo grupiranje, tipično se koriste da bi se otkrilo je li neki tekst (tipično se radi o povijesnim tekstovima), za koji se prethodno možda smatralo da je djelom jednog autora, zapravo napisalo više autora. U tu se svrhu tipično analiziraju stilometrijske značajke teksta te se zatim provodi grupiranje. Npr., u Savoy [2019] koristi se hijerarhijsko grupiranje za analizu autorstva Poslanica apostola Pavla. Standardni skup podataka za testiranje metoda atribucije autorstva su *Federalistički spisi* (engl. *The Federalist Papers*), nastali uoči ratifikacije ustava Sjedinjenih Američkih Država krajem 18. stoljeća. Za dvanaest spisa od njih 88 ne zna se puzdano je li autor Alexander Hamilton ili James Madison. To je motiviralo razvoj metoda za analizu autorstva, najprije ručnih, a kasnije računalnih metoda, npr., [Holmes and Forsyth, 1995], a sve to u okviru područja koje se danas naziva **računalna lingvistička forenzika**. Za dobar pregled metoda atribucije autorstva, pogledajte [Stamatatos, 2009].

[4] Kao primjere znanstvenih radova koji grupiranje koriste za detekciju emocija iskazanih u tekstu, pogledati [Aragón et al., 2019, Mohammad and Kiritchenko, 2015, Chatzakou et al., 2013].

[5] Algoritam je osmislio Stuart Lloyd 1957. godine, ali je objavljen tek tridesetak godina kasnije u [Lloyd, 1982]. Naziv ovog rada ("Kvantizacija metodom najmanjih kvadrata za pulsno-kodnu manipulaciju") reflektira činjenicu da algoritam minimizira kvadratno odstupanje primjera od centroida grupa i da je izvorno bio namijenjen za pulsno-kodnu modulaciju.

[6] Premda algoritam K-sredina linearno ovisi o svim relevantnim parametrima, problem u praksi ipak predstavljaju visokodimenzijski ulazni prostori, dakle velik n. To je pogotovo problem u nekim područjima primjene grupiranje, npr., kod obrade prirodnog jezika, gdje prostor može imati onoliko dimenzija koliko ima riječi u jeziku. U takvim se slučajevima mogu koristiti varijante algoritma K-sredina, npr., inačica sa skraćenim (engl. *truncated*) centroidima; v. `https://nlp.stanford.edu/IR-book/html/htmledition/k-means-1.html`. U drugim, pak, primjenama problem može predstavljati velik broj primjer N, koji ne stanu svi odjednom u memoriju. Tada se može koristiti **algoritam K-sredina s mini-grupama** (engl. *mini-batch K-means*) [Sculley, 2010]. Popis različitih varijanti algoritma K-sredina, neke od kojih ciljaju smanjiti računalnu složenost, dok druge mijenjaju induktivne pretpostavke algoritma, možete naći na `https://en.wikipedia.org/wiki/K-means_clustering#Variations`.

[7] **Analiza glavnih komponenti** (engl. *principal component analysis*, PCA) važna je tehnika za smanjenje dimenzionalnosti koja se koristi i u strojnom učenju i statistici. Mi se u ovom predmetu nažalost ne bavimo metodama za smanjenje dimenzionalnosti. Ako se slučajno dogodi da se nigdje na FER-u ne susretnete sa PCA, svakako preporučam da tu metodu savladate sami. U tu svrhu preporučam [Shlens, 2014] i [Smith, 2002].

[8] Jedno poboljšanje algoritma PAM, razvijeno upravo s ciljem smanjenja vremenske složenosti, jest nedavno predložen **algoritam BanditPAM** [Tiwari et al., 2020]. Zahvaljujem Domagoju Alagiću i Marijanu Smetku na ovoj informaciji.

[9] **t-SNE** je kratica od **t-distribuirana stohastička ugradnja susjeda** (engl. *t-distributed stohastic neighbor embedding*). Metodu su izvorno predložili u [Hinton and Roweis, 2002], a nadogradili u [Maaten and Hinton, 2008]. Pristupačno objašnjenje metode t-SNE možete naći na `https://towardsdatascience.com/t-sne-clearly-explained-d84c537f53a`. Metoda t-SNE je vrlo moćna, no ponekad generira neintuitivne vizualizacije u kojima je lako vidjeti više nego što doista pos-

toji u podatcima, pa, ako želite koristiti t-SNE za vizualizaciju podataka, svakako pročitajte ovo: `https://distill.pub/2016/misread-tsne/`. Pregled ostalih često korištenih tehnika za smanjenje dimenzionalnosti možete naći u [Van Der Maaten et al., 2009] i [Lee and Verleysen, 2010].

[10] **Metoda koljena** ili **metoda lakta**.

[11] Primjer preuzet sa `https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html`.

[12] Postoje algoritami grupiranja koji, umjesto broja grupa K, imaju neki hiperparametar, poput hiperparametra λ, koji indirektno određuje broj grupa, i to već pri samom grupiranju. Primjer je, npr., **algoritam propagacije afiniteta** (engl. *affinity propagation*) [Frey and Dueck, 2007] ili **algoritam Markovljevog grupiranja** (engl. *Markov Cluster Algorithm*) [Van Dongen, 2000]. Ove algoritme ima smisla koristiti ako ne znamo unaprijed koji je optimalan broj grupa (a najčešće to ne znamo), a želimo provoditi grupiranje na skupovima podataka različitih veličina. Na primjer, ako želimo grupirati slične fotografije u rezultatima tražilice, onda ne možemo unaprijed znati koliko različitih grupa fotografija postoji za pojedini upit postavljen preko tražilice. Naprotiv, broj grupa bit će općenito različit za svaki upit.

[13] **Akaikeov informacijski kriterij** osmislio je japanski statističar Hirotogu Akaike. Kriterij je zasnovan na teoriji informacije i izvorno je korišten kod modela **linearne regresije** za korekciju log-izglednosti uslijed prenaučenosti, no kasnije je našao općenitu primjenu u strojnom učenju za **odabir modela**. Zanimljivost Akaikeovog kriterija jest da je agnostičan na duboku podjelu u statistici između frekventističke i bayesovske statistike, i da zapravo sâm može biti korišten kao teorijski temelj statistike. Inače, pitanje **temelja statistike** je otvoreno (filozofsko ali i praktično) pitanje o prirodi i načinu statističkog zaključivanja, u kojemu se, pored ostalog, debatira između frekventističke i bayesovske statistike. Više na `https://en.wikipedia.org/wiki/Foundations_of_statistics`, ili u [Savage, 1972] i [Efron, 2005].

[14] U ovakvom kontekstu, treba razlikovati **vanjski (ekstrinzični) kriterij** i **unutarnji (intrinzični) kriterij** grupiranja: mjera koju koristimo za mjerenje točnosti grupiranja predstavlja **vanjski kriterij**, dok kriterij koji algoritam interno koristi za samo grupiranje (npr., kriterijska funkcija kod algoritma K-sredina) predstavlja **unutarnji kriterij**. Pretpostavka je da je unutarnji kriterij, po kojem se provodi grupiranje, dobro usklađen s vanjskim kriterijem, koji mjerimo i koji nam je bitan. Ako to nije slučaj – ako unutarnji i vanjski kriterij grupiranja nisu dobro usklađeni – onda nas ne treba čuditi ako algoritam grupiranja u smislu vanjskog kriterija daje suboptimalne rezultate.

[15] Dobar pregled raznih mjera za provjeru grupa možete naći u [Amigó et al., 2009].

[16] Randov indeks predložio je 1971. godine američki statističar William Rand sa MIT-a [Rand, 1971].

[17] U idealnom slučaju, za neki odabrani algoritam grupiranja i neko referentno grupiranje, Randov indeks bit će u pravilu maksimalan ako je odabrani broj grupa K jednak pravom ("prirodnom") broju grupa u podatcima. Međutim, ova tvrdnja dolazi s jednom važnom napomenom, a to je da K koji maksimizira Randov indeks ipak ne mora uvijek odgovarati "prirodnom" broju grupa u podatcima. To, naime, ipak ovisi i o algoritmu grupiranja. Konkretnije, svaki algoritam grupiranja ima svoju induktivnu pristranost, i moguće je da algoritam jednostavno nije u stanju naći grupiranje koje doista odgovara prirodnom grupiranju. Kao i uvijek u strojnom učenju, induktivna pristranost algoritma treba biti usklađena s onime što imamo u podatcima, inače se ne trebamo čuditi da algoritam ne radi dobro. Dobar primjer za neusklađenost induktivne pristranosti algoritma grupiranja i podataka jest kada algoritam K-sredina koristimo za grupiranje primjera koji u ulazom prostoru čine grupe koje nisu sferične (npr., izdužene grupe, ili grupe koje se isprepliću i slično). Naime, time što koristi euklidsku udaljenost za pridjeljivanje primjera u najbližu grupu, algoritam K-sredina zapravo pretpostavlja da su grupe sferične. Ako one to nisu, algoritam K-sredina neće nam moći pronaći "prirodne" grupe. U takvim se situacijama može dogoditi da algoritam K-sredina daje bolje grupiranje (ostvaruje veći Randov indeks) s nekim brojem grupa K koji se ne podudara s pravim brojem grupa.

[18] Korisno je primijetiti da Randov indeks možemo koristiti i onda kada ne znamo koji primjer pripada kojoj grupi, ali znamo da li dva primjera trebaju pripadati istoj grupi. Naime, budući da Ran-

dov indeks točnost izračunava nad parovima primjera, dovoljno je označiti **uzorak parova primjera** tako da za svaki par označimo je li pozitivan (primjeri trebaju biti grupirani zajedno) ili negativan (primjeri trebaju biti razdvojeni). Iz takvog uzorka možemo onda izračunati sve što nam treba za Randov indeks: znamo ukupan broj parova u uzorku, a broj stvarno pozitivnih i stvarno negativnih parova dobit ćemo usporedbom rezultata grupiranja sa parovima iz našeg uzorka. Dakle, referentno grupiranje ne mora nužno doista grupirati primjere u grupe, nego može samo odrediti koji parovi primjera trebaju biti zajedno grupirani, a koji ne. Primijetite da je to mnogo lakši zadatak nego pridijeliti grupu svakom primjeru, jer ne pretpostavlja da znamo koje grupe postoje niti koliko ih je ukupno. Npr., ako grupiramo riječi nekog jezika u grupe značenjski sličnih riječi, onda možemo relativno dobro za neki par riječi odrediti znače li te riječi isto ili ne, dok bi puno teži zadatak bio da unaprijed odredimo sva moguća značanja riječi nekog jezika i onda riječi razvrstavamo u takve grupe.

19 Za **prilagođen Randov indeks**, pogledati: `https://en.wikipedia.org/wiki/Rand_index#Adjusted_Rand_index`. Za razliku od Randovog indeksa, prilagođeni Randov indeks može poprimiti i negativne vrijednosti.

Literatura

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

E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information retrieval*, 12(4):461–486, 2009.

M. E. Aragón, A. P. López-Monroy, L. C. González-Gurrola, and M. Montes. Detecting depression in social media using fine-grained emotions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1481–1486, 2019.

P. Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

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

D. Chatzakou, V. Koutsonikola, A. Vakali, and K. Kafetsios. Micro-blogging content analysis via emotionally-driven clustering. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pages 375–380. IEEE, 2013.

B. Efron. Bayesians, frequentists, and scientists. *Journal of the American Statistical Association*, 100(469):1–5, 2005.

B. J. Frey and D. Dueck. Clustering by passing messages between data points. *science*, 315 (5814):972–976, 2007.

G. E. Hinton and S. Roweis. Stochastic neighbor embedding. *Advances in neural information processing systems*, 15:857–864, 2002.

D. I. Holmes and R. S. Forsyth. The federalist revisited: New directions in authorship attribution. *Literary and Linguistic computing*, 10(2):111–127, 1995.

J. A. Lee and M. Verleysen. Unsupervised dimensionality reduction: overview and recent advances. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2010.

S. Lloyd. Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2): 129–137, 1982.

L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

S. M. Mohammad and S. Kiritchenko. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence*, 31(2):301–326, 2015.

W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

L. J. Savage. *The foundations of statistics*. Courier Corporation, 1972.

J. Savoy. Authorship of pauline epistles revisited. *Journal of the Association for Information Science and Technology*, 70(10):1089–1097, 2019.

D. Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178, 2010.

J. Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

L. I. Smith. A tutorial on principal components analysis. Technical report, 2002.

E. Stamatatos. A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, 60(3):538–556, 2009.

M. Tiwari, M. J. Zhang, J. Mayclin, S. Thrun, C. Piech, and I. Shomorony. BanditPAM: Almost linear time k-medoids clustering via multi-armed bandits. In *NeurIPS*, 2020.

L. Van Der Maaten, E. Postma, and J. Van den Herik. Dimensionality reduction: a comparative review. *J Mach Learn Res*, 10(66-71):13, 2009.

S. M. Van Dongen. *Graph clustering by flow simulation*. PhD thesis, 2000.

D. Xu and Y. Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015.

R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.