

Istrazivanje Podataka 1 - Belekse

Andrija Urosevic

Contents

Uvod	3
Sta je istrazivanje podataka?	4
Istrazivanje podataka i otkrivanje znanja	4
Izazovi u istrazivanju podataka	4
Skalabilnost	4
Velika Dimenzionalnost	4
Heterogeni i kompleksni podaci	4
Pripadnost i distribucija podataka	4
Netradicionalna analiza	5
Nastanak istrazivanja podataka	5
Zadatak istrazivanja podataka	5
Podaci	5
Tipovi podataka	6
Atributi i Mere	6
Tipovi skupova podataka	7
Kvalitet podataka	9
Merenje i problemi pri sakupljanju podataka	9
Problemi pri primenama podataka	11
Predprocesiranje podataka	11
Agregacija	11
Uzorkovanje	12
Redukcija dimenzija	13
Diskretizacija i Binarizacija	13
Mere slicnosti i razlicitosti	15
Osnove	15
Slicnost i Razlicitost izmedju jednostavih atributa	16
Razlicitosti izmedju objekta podataka	16
Slicnosti izmedju objekta podataka	17
Primeri mera blizine	17
Problemi pri racunanju blizine	19
Pretrazivanje Podataka	19
Klasifikacija: Osnovni koncepti, drveta odlucivanja	19
Osnove definicije	20
Generalni pristup resavanja klasifikacionih problema	20
Drvo odlucivanja uvod	21
Kako radi drvo odlucivanja?	21
Kako napraviti drvo odlucivanja?	22
Metodi za izrazavanje uslova testiranja atributa	23
Mera za odabir najboljeng deljenja	24

Algoritam indukovanja drveta odlucivanja	26
Karakteristike indukovanja drveta odlucivanja	26
Preprilagodjavanje modela	27
Preprilagodjavanje zbog prisustva suma	27
Preprilagodjavanje zbog nedostatka reprezentativnih uzoraka	28
Preprilagodjavanje i procedura visestrukog poredjenja	29
Procenjivanje greske generalizacije	30
Preprilagodjavanje u indukovanje drveta odlucivanja	31
Racunanje performanse klasifikatora	32
Metod zadrzavanja	32
Nasumicno uzorkovanje	32
Unakrsna-Validacija	32
Bootstrap	33
Metodi za uporedjivanje klasifikatora	33
Procenjivanje intervala poverenja za tacnost modela	33
Racunanje performansi dva modela	34
Uperedjivanje performansi dva klasifikatora	34
Razni Algoritmi Zasnovani na Drvetima Odlucivanja	35
ID3	35
C4.5	35
CART	35
CHAID	36
QUEST	36
SLIQ	36
SPRINT	36
Klasifikacije: Alternativne tehnike	36
Klasifikator zasnovan na pravilima	36
Kako klasifikator zasnovan na pravilima radi?	37
Seme uredjenja-pravila	38
Kako napraviti klasifikator zasnovan na pravilima?	38
Direktni metodi za izdvajanje pravila	38
Indirektni metodi za izdvajanje pravila	41
Karakteristike klasifikatora zasnovanog na pravilima	41
Klasifikator: Najblizi sused	42
Algoritam	42
Karakteristike klasifikatora: najblizi sused	42
Bajasov klasifikator	43
Bajasova teorema	43
Bajasova teorema u klasifikaciji	43
Naivni Bajasov klasifikator	44
Bajasova greska	47
Bajasova mreza poverenja	48
Vestacke neuronske mreze	48
Perceptron	48
Viseslojne vestacke neuronske mreze	49
Karakteristike vestacke neuronske mreze	51
Metod potpunih vektora (Support Vector Machine — SVM)	51
Hiperravni maksimalne margine	51
Linearna SVM: Razdvojiv slucaj	52
Linearni SVM: Nerazdvojivi slucaj	54
Nelinearni SVM	55
Karakteristike SVM	57
Metod Ansambla	58

Problem: Nebalansiranih Klasnih Instanci	58
Alternativne Metrike	58
Receiver Operating Characteristic Krive	59
Učenje: Cena-Senzibilitet	60
Postupak Uzorkovanja	61
Problem: Viseklasnih Skupova Podataka	61
Klaster Analiza: Osnovni Koncepti i Algoritmi	62
Pregled	63
Sta je Klaster Analiza?	63
Razni Tipovi Klasterovanja	63
Razni Tipovi Klastera	63
K-sredina	64
Osnovni Algoritam K-sredina	64
K-sredina: Dodatni Problemi	66
Bisekcija K-sredina	66
K-sredina i Drugaciji Tipovi Klastera	67
Prednosti i Mane	67
K-sredina kao Optimizacioni Problem	67
Hijerarhijsko Klasterovanje	68
Osnovni Algoritam Sakupljajućeg Klasterovanja	68
Specifične Tehnike	69
Lens-Vilijams Formula za Klaster Blizine	74
Ključni Problemi u Hijerarhijskom Klasterovanju	74
DBSCAN	74
Tradicionalne Tehnike: Pristup Baziran na Centru	74
Algoritam: DBSCAN	74
Prednosti i Mane	75
Evaluacija Klastera	75
Pravila Pridruživanja: Osnovni Koncepti i Algoritmi	75
Problem Definicije	76
Generisanje Frekventnih Skupova Stavki	77
Apriori Princip	77
Generisanje Frekventnih Skupova Stavki i Apriori Algoritam	78
Generisanje Kandidata i Odsecanje	79
Brojanje Cvrstine	79
Generisanje Pravila	79
Potkresivanje na Osnovu Pouzdanosti	80
Generisanje Pravila u Apriori Algoritmu	80

Uvod

Sakupljanje podataka neverovatno brzo raste, u smislu kolicine, ali sta nedostaje jestu metodi za izvlacenje korisnih informacijama iz velikog skupa podataka. Zbog toga tradicionalni alati za analizu podataka nisu dovoljno sufisticirani, i novi moraju biti razvijeni.

Istrazivanje podataka je tehnologija koja spaja tradicionalnu analizu podataka sa sofisticiranim algoritmima za procesiranje velike zapremine podataka.

Biznisi. Postoje mnogi alati za prikupljanje podataka potrosaca u realnom vremenu. Pa proizvođači mogu da iskoriste te informacije za svoje potrebe, tako da naprave proizvod koji ce bolje odgovarati korisniku. Ove informacije mogu takodje da daju odgovore na neka od pitanja kao sto su: Ko su najprofitabilniji potrosaci? Koji proizvod se bolje prodaje, a koji losije? Kolika je zarada kompanije za tekucu godinu?

Medicina, Nauka i Inženjering. Prikupljaju se podaci koji su ključni za nova otkrića. Primer je NASA koja je postavila satelite oko planete Zemlje i meti kopno, okeane i atmosferu. Ali zbog količine podataka tradicionalni metodi nisu korisni za analizu ovakvih skupova podataka. Istraživanje podataka može da da odgovore na sledeća pitanja: Koja je relacija između frekvencije i intenziteta vremenskih neprilika kao što su poplave i tornadi? Kako je temperatura na kopnu u zavisnosti od temperature na površini okeana? Kako predvideti početak i kraj uzgajne sezone?

Sta je istraživanje podataka?

Istraživanje podataka je proces automatskog otkrivanja korisnih informacija u velikim skladistenim podacima. Pronalazi nove i korisne šablone koji bi možda ostali neotkriveni. Takođe imaju mogućnost da predvide buduća opazanja, kao što je predviđanje da li će novi potrošač potrositi više od 1000din u radnji.

Nisu sva otkrivanja informacija istraživanje podataka. Na primer, jednostavni upit data baze ili nalazanje određene Web stranice preko pretraživača su zadaci oblasti koja se naziva *pronalaženje informacija*. Oni jesu veoma korisni ali se oslanjaju na tradicionalne algoritme i strukture podataka.

Istraživanje podataka i otkrivanje znanja

Istraživanje podataka je deo otkrivanja znanje u bazi podataka(KDD), što je proces dobijanja korisnih informacija iz sirovih podataka.

```
Ulazni Podaci --> Predprocesiranje --> Istraživanje Podataka
--> Postprocesiranje --> Informacija
```

Uloga **predprocesiranja** je da transformise sirove podatke u radne podatke koji su spremni za analizu. Ovo uključuje spajanje podataka sa više izvora, čišćenje podataka od suma i duplikata, biranje karakteristika koji su relevantni za istraživanje podataka.

Takođe nakon istraživanja podataka potrebno je rezultat interpretirati, i ovaj proces se naziva **postprocesiranje**. Primeri je vizualizacija.

Izazovi u istraživanju podataka

Skalabilnost

Skupovi podataka se čuvaju u gigabajtima, terabajtima, pa čak i petabajtima. Zbog toga tehnike istraživanja podataka moraju biti skalabilne. Mnogi algoritmi koriste specijalne strategije pretrage, pa čak i implementacije novih struktura podataka koji pristupaju slogovima efikasno.

Velika Dimenzionalnost

Sada je često da se nadju skupovi podataka sa stotinama ili hiljadama atributa. Za neke tradicionalne algoritme podataka, njihova kompleksnost se povećava sa povećanjem dimenzija (broja atributa). Takođe, neki uopšte ne daju dobre rezultate.

Heterogeni i kompleksni podaci

Tradicionalni metodi analize podataka se primenjuju na skupove podataka koji imaju attribute istog tipa. Kako se uloga istraživanja podataka povećava, povećava se potreba za obradjivanjem heterogenih atributa. Takođe pojavljuju se i mnogi kompleksni podaci, kao što su XML dokumenti, grafovi...

Pripadnost i distribucija podataka

Podaci ne moraju biti smesteni na jednoj lokaciji, takođe, ne moraju ce ni da pripadaju jednoj organizaciji. Ovo zahteva distributivne tehnike istraživanja podataka, tj. smanjenje komunikacije za distribuirano izvršavanje, spajanje rezultata iz više izvora i sigurnosne probleme.

Netradicionalna analiza

Za razliku od tradicionalnih statističkih metoda koji se baziraju na hipotezi i testu, tj. iskaze se hipoteza, onda se dizajnira eksperiment koji prikuplja podatke, i onda se analiza sprovede po iskazanoj hipotezi, noviji metodi analize podataka generisu i evaluisu hiljade hipoteza, a i mnoge tehnike su napravljene tako da automatizuju ovaj proces.

Nastanak istraživanja podataka

Istraživanje podataka se oslanja na idejama kao što su

1. uzorkovanje, ocenjivanje, i testiranje hipoteza iz statistike
2. algoritmi pretrage, tehnike modelovanja, i teorija učenja iz veštačke inteligencije, prepoznavanje sablona, i masinsko učenje.

Takodje potrebne su i dodatne oblasti računarstva kao što su sistemi baza podataka, paralelnog izračunavanja, distributivno programiranje.

Zadatak istraživanja podataka

Zadatak predviđanja. Predviđa vrednost nekog atributa bazirano na vrednostima drugih atributa. Atribut koji se predviđa naziva se **target** ili **zavisna promenljiva**, dok atributi koji se koriste za predviđanje se nazivaju **opisni** ili **nezavisne promenljive**.

Zadatak opisivanja. Izvuci sablone koji sumiraju relacije između podataka.

Model predviđanja se odnosi na izgradnju modela za target promenljive kao funkcije koja prima ulazne promenljive. Postoje dva zadatka modela predviđanja: **klasifikacija** i **regresija**. Klasifikacije se koristi za diskretnu vrednost target promenljive, dok se regresija koristi za neprekidnu vrednost target promenljive. Cilj oba zadatka je da minimizuju gresku između predviđene vrednosti i istinite vrednosti target promenljive.

Primer (Predviđanje vrsta Irida). Za dati skup podataka koji predstavlja cvet irisa, možemo odrediti vrstu irisa na osnovu dužine i širine latica.

Asocijativna analiza se koristi za otkrivanje sablona koji opisuju pridružene karakteristike u podacima. Sabloni se predstavljaju kao implicitno pravilo ili kao podskup karakteristika.

Primer (Analiza korpe). Na osnovu podataka o kupovinama proizvoda možemo zaključiti da ako je potrošač kupio Pampres, onda je i kupio Mleko, pa imamo sledeće pravilo {Pampers} → {Mleko}.

Klaster analiza pronalazi grupe usko povezanih podataka tako da podaci koja pripadaju istom klasteru su sličnija međusobno nego podaci nekog dugog klastera.

Primer (Klasterovanje dokumenata). Možemo da klasterujemo artikle bazirano na njihovoj upotrebi. Na osnovu broja ponavljanja određene reči iz opisa artikla možemo da zaključimo svrhu tog artikla. Na primer, ako sadrži reči kao što je **medicinski**, **pacijent**, **lek**, **zdravlje**,... možemo ove artikle smestiti u jedan klaster.

Otkrivanje anomalija je zadatak identifikovanje podataka čije su karakteristike značajno drugačije od ostalih podataka. Takvi podaci se poznati kao *anomalije* ili *autlajeri*. Pri ovom procesu moramo što preciznije odrediti anomalije, u smislu da ne smemo označiti normalne objekte kao anomalije, i suprotno.

Primer (Kradja kreditne kartice). Banka skuplja podatke o transakcijama korisnika kreditne kartice, zajedno sa ličnim informacijama korisnika. Na osnovu toga, može zablokirati karticu ako dodje do transakcije koja je najmanje verovatna da se dogodi, jer predstavlja potencionalnog kradljivca.

Podaci

Postoje nekoliko probleme koji su vezani za podatke:

1. **Tipovi podataka.** Atributi koji opisuju podatke mogu biti drugacijeg tipa. Neki podaci mogu imati posebne karakteristike, pokazuju na druge objekte, ili sadrže neke vremenske nizove.
2. **Kvalitet podataka.** Podaci su daleko od perfektnog. Ako se poboljša kvalitet podataka vrlo često se boljša i rezultat analize. Treba otkloniti prisustvo suma, autlajere, duplikate, podatke zasnovane na sklonosti, ili druge fenomene.
3. **Koraci preprocesiranja kako bi napravili zgodnije podatke za istraživanje podataka.** Treba modifikovati podatke tako da se uklape u odgovarajući algoritam.
4. **Analiziranje podataka u smislu njegovih relacija.** Jedan pristup analiziranju podataka je pronalazanje relacija između podataka i primenjivanje analize nad tim relacijama, a ne na samim objektima.

Tipovi podataka

Skup podataka je kolekcija **objekta podataka** (*slogova, tacaka, sablona, događaja, slučaja, uzorka, posmatranja, ili pristupa*). **Objekti podataka.** Objekti podataka se opisuju **atributima** (*promenljivima, karakteristikama, poljima, osobinama, ili dimenzijama*).

Primer (Jednostavi skup informacija o studentu)

Student ID	Godina	Prosečna Ocena	...
mi18083	1	9.32	...
mi17083	4	6.21	...
...

Atributi i Mere

Sta je atribut?

Definicija: Atribut je osobina ili karakteristika objekta koja može da varira, ili iz jednog objekta u drugi ili iz jednog vremena u drugo.

Primer: Boja očiju varira od osobe do osobe (objekta), dok temperatura osobe varira vremenom.

Definicija: Merna skala je pravilo (funkcija) koja je pridružuje numeričku ili simboličku vrednost atributu objekta.

Tip atributa

Osobine nekog atributa ne moraju biti isti kao osobine vrednosti koje je ga mere, tj. vrednosti koje predstavljaju atribut mogu imati osobine koje nisu osobine samog atributa, i obrnuto.

Primer (Zaposleni: Godine i ID). Dva atributa su *ID* i *Godine* koja mogu da se pridruže zaposlenom. Ovi atributi se mogu predstaviti kao celi brojevi. Razumno je pričati o prosečnoj godini zaposlenih, ali nije razumno pričati o prosečnom IDu. Zapravo, jedino što hocemo da znamo pomoću ID atributa je da li su isti ili različiti, tj. jedina operacija koja može da se pridruži ID atributu je provera jednakosti.

Primer (Duzina linijskog segmenata). Svakom linijskom segmentu možemo da dodelimo neku vrednost koja će označavati njegovu dužinu. Postoji bar dva načina da ovo uradimo. Jedan je da ih mapiramo tako da se očuvamo poredak dužina. Drugi način je da očuvamo odnos između dužina. Drugi način jasno opisuje i prvi način, pa atribut možemo meriti na način na koji ne opisuje sve osobine atributa.

Tip atributa treba da nam kaže koje osobine atributa se reflektuju u vrednosti koje ga mere. Zbog toga se referise na tipove atributa kao **tipove merne skale**.

Različiti tipovi atributa

Sledeće osobine (operacije) brojeva se koriste za opisivanje atributa

1. **Razlicitost:** $= i \neq$

2. **Poredak:** $<, \leq, >, \geq$
3. **Sabiranje:** $+, -$
4. **Množenje:** $\cdot, /$

Na osnovu ovih operacija mozemo definisati razlicite tipove:

Tip Atributa	Opis	Primeri	Operacije
Nominalni (Imenski)	To su samo imena na kojima se moze primeniti <i>razlicitost</i>	boja ociju, id, postansk broj	mode, entropija, pripadnostna korelacija
Ordinalni (Redni)	Informacije koje nam pružaju i <i>poredak</i>	ocene, brojevi stanova	medijana, percentili, rank korelacija
Intervali	Razlike izmedju vrednosti su znacajne, tj. postoji merna jedinica	datumi, temperatura	ocekivana vrednost, standardno odstupanje, puasonova korelacija
Razmerni	Pored razlika znacajni su i odnosi	duzine, masa	geometrijsko ocekivanje, harmonijsko ocekivanje, disperzija

Nominalne i ordinalne attribute nazivamo **kategoricki** ili **kvalitativni** atributi i o njima mislimo kao o simbolima, dok intervale i razmerne attribute nazivamo **kvantitativni** ili **numericki** atributi i o njima mislimo kao o brojevima.

Opisivanje atributa po broju vrednosti

Diskretni. Diskretni atributi uglavnom imaju konacan ili prebrojivo beskonacan domen. Ovi atributi su obicno kategoricki, i predstavljaju se celim brojevima. **Binarni atributi** spadaju u diskretne attribute i uzimaju samo dve vrednosti 0 ili 1.

Neprekidni. Neprekidni atributi imaju uzimaju vrednosti realnih brojeva. Ovi atributi predstavljaju uglavnom duzine, temperaturu, itd.

Bilo koji od nominalnih, ordinalnih, intervala, i rezmerna mozemo da kombinujemo sa diskretnim ili neprekidnim atributima, samo sto neki nemaju smisla, ili se veoma retko koriste.

Asimetricni atributi

Kod asimetricnih atributa samo prisustvo ne-nula vrednosti se uzima kao znacajno. Na primer, ako posmatramo studente i kurseve koji su oni upisali, nije nam bitan broj upisanih kurseva, kako bi tada svi studenti bili veoma slicni, vec nam je bitno da li su ili nisu upisali odredjeni kurs. Binarni atributi kod kojih je jedino prisustvo ne-nula vrednosti vazno nazivaju se **asimetricno binarni atributi**. Takodje asimetricni atributi mogu biti i diskretni i neprekidni.

Tipovi skupova podataka

Tipove skupova podataka grupisemo u tri grupe: slogovni podaci, grafovski podaci, uredjeni podaci

Generalne karakteristike podataka:

1. **Dimenzionalnost** je broj atributa nekog skupa podataka.

2. **Retkost** ocenjuje koliki procenat skupa podataka ima ne-nula vrednosti. Retki skupovi podataka su korisni za mnoge algoritme, pa i za skladistenje
3. **Rezolucija** je bitna zbog rezultata koji mogu da nam daju podaci. Ako na primer posmatramo temeperature zemlje, na svakih $2m$, dobijamo veliki sum, dok ako je posmatramo na svakih $2km$, dobijamo glatke prelaskе. Takodje pritisak vazduha na je bitno da znamo svakog sata, kako on uticne na trenutne vetrove, dok ukoliko imamo pritisak za svaki mesec, ne dobijamo nista.

Slogovni podaci

Slogovni podaci predstavljaju skup podataka kao kolekciju slogova. Nemaju relacije izmedju slogova, ili polja, i svaki slog ima iste atribute. Cuvaju se u *flat* fajlovima ili u relacionim bazama podataka.

ID	Ime	Godine
123	Pera	32
221	Mara	23
321	Sara	43

Transakcije ili korpa podaci su slogovni skupovi podataka gde je svaki slog sadrzi skup stavki. Taj skup stavki moze da se asocira sa potrosackom korpom pa od tuda naziv. Takodje ovi podaci mogu da se predstave preko asimetričnih polja, gde su atributi sve moguće stavke i gde je polje prazno ako se ta stavka ne nalazi u skupu stavki.

ID	Korpa
211	kafa, mleko, sir
321	sok, jaja, mleko
353	kafa, jaja

Matricni podaci su slogovni skupovi podataka kod kojih svi slogovi (objekti podataka) imaju fiksiran broj atributa sa numerickim vrednostima. Ove skupove je zgodno predstavljati matricno, kako je svaka kolona jedan objekat podataka, a svaki red predstavlja jedan atribut.

X	Y	Temp(X, Y)
1	4	22
2	2	32
2	3	30
3	1	10

Retki matricni podaci su specijalan slucaj matricnih podataka gde su svi atributi istog tipa i asimetrični su, tj. bitne su samo ne-nula vrednosti.

ID	Tenis	Fudbal	Kosarka
Doc1	1	0	0
Doc2	0	1	0
Doc3	1	0	0
Doc4	0	0	1

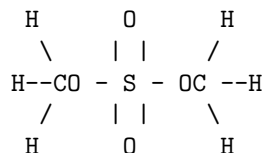
Grafovska podaci

Podaci sa relacijama izmedju objekata. Relacija izmedju objekata cesto cuva vazne informacije. Takve

informacije se predstavljaju pomocu grafa, gde su cvorovi objekti, a grane relacije izmedju njih. Na primer, jedan HTML dokument, moze imati linkove na ostale HTML dokumente, cesto pri pretrazivanju Web stranica koristni su i podaci koji se nalaze na stranicama ciji se link nalazi na nekoj stranici.

```
index1.html:
  Neki test
  link -----> index2.html
  Jos teksta
                        Neki tekst
                        link1 -----> index3.html
                        link2 -----> index4.html
                        Jos teksta
```

Podaci sa objektima koji su grafovi. Ako sami objekti imaju neko strukturu oni se predstavljaju pomocu grafova. Primer ovih podataka mogu biti molekuli, gde su cvorovi atomi, a grane, veze izmedju njih. Takodje svaka grana moze imati i labelu koja oznacava tip veze.



Uredjeni podaci

Nekada podaci imaju u sebi uredjenje kao sto je vremensko ili prostorno.

Sekvencijalni podaci su ekstenzija slogovnih podataka tako da se svakom broju pridruzuje atribut vremena. Time dobijamo informacije koje inace ne bismo mogli da dobijemo, kao sto je informacija o proizvodima koji ce potrosaci kupiti nakon sto su kupili neki poizvod ili ucestalost kupovine nekog proizvoda. Na primer, potrosac koji je kupio auto, verovatno ce kupiti i gorivo za njega, ili prodaja novogodisnjih poklona se povecava krajem decembra.

Diskretne sekvence ili Niske su skupovi podataka koji su sekvence individualnih entiteta, kao sto su reci ili slova. Kod ovih podataka je bitan redosled, a ne vremensko obelzje. Na primer, GNK predstavlja diskretne sekvence koje koriste slova A, T, G, i C.

Vremenske serije su skupovi podataka kod kojih je svaki slog vremenska serija, tj. serija merenje izmerena tokom nekog vremena. Neki primeri su dnevne cene na berzi ili prosečna dnevna temperatura tokom jednog meseca. Kod ovakvih skupova podataka mora postojati **vremenska autokorelacija**, tj. dva susedna sloga moraju biti u veoma slicna.

Prostorni podaci su skupovi podataka kod kojih svaki slog ima prostorne attribute. Primer su podaci o vremenu koji za svaku lokaciju i vreme imaju temperaturu, pritisak, brzinu vetra, itd. Takodje mora postojati **prostorna autokorelacija**, tj. dva susedna sloga koja su prostorno blizu moraju imati slicne ostale attribute.

Kvalitet podataka

Istrazivanje podataka se cesto primenjuje nad podacima koji su prikupljeni za druge svrhe ili za buduce nespecifikovane svrhe. Zbog toga istrazivanje podataka nema perfektan kvalitet podataka za obradu kao kod nekih statistickih pristupa, vec ima za cilj da detektuje i poboljska prolem kvalitet podataka (*ciscenje podataka*) i koristi algoritme koji mogu da obrade lose podatke.

Merenje i problemi pri sakupljanju podataka

Nije realisticno da pri prikupljanju podataka sakupimo savrsene podatke. Pri sakupljanju podataka dolazi do raznih gresaka kao sto su ljudske greske, greske pri merenju, gubitak ili dupliranje objekta podataka, itd. Takodje, podaci mogu biti i nekonzistentni, kao na primer covek je visok $2m$ i tezak $2kg$.

Greska pri merenju i prikupljanju podataka

Greska pri merenju se odnosi na limitacije uređaja za merenje da izmeri realni objekat precizno, ta razlika izmerene i stvarne vrednosti se naziva **greska**.

Greske pri prikupljanju podataka se odnose na greske kao što je ne popunjavanje određenog polja, atributa, ili čas i celog sloga.

Postoje i ostale greske kao što je pogresno unosenje vrednosti pri kucanju, ali za to postoji odgovarajuće metode za detekciju i otklanjanje takvih gresaka.

Sum i Artifakti

Sum je nasumična komponenta nekog merenja. Sum obično postoji u vremenskim serijama i prostornim podacima. Iako postoji mnogi merni uređaji u sebi imaju metod za otklanjanje sum, algoritmi istraživanja podataka se dizajniraju tako da mogu da se bore sa sumom.

Deterministične greske podataka, kao što je ogrebotina slike, nazivaju se artifakti.

Preciznost, Pristrasnost, i Tacnost

Definicija: Preciznost je približnost pri ponovljenom merenju.

Definicija: Pristrasnost je sistematska varijacija merenja od količine koja se meri.

Preciznost se obično meri standardnim odstupanjem, dok se pristrasnost meri razlikom očekivane vrednosti sa pravom vrednošću kvantiteta koji se meri. Na primer, ako merimo teg mase *1kg* $\pm 5\%$ puta, i dobijemo sledeće vrednosti $\{1.015, 0.990, 1.013, 1.001, 0.986\}$, tada je očekivanje 1.001 pa je pristrasnost 0.001 i preciznost je 0.013 kako je to standardno odstupanje.

Definicija: Tacnost je približnost merenja pravoj vrednosti kvantiteta koji se meri.

Za tacnost su nam bitne **znacajne cifre**, tj. cuvacemo onoliko cifara koliko je moguće dobiti mernim instrumentom.

Autlajeri (Nepodobni)

Nepodobni podaci su ili

1. objektni podaci koji imaju karakteristike koje su drugačije od svih ostalih objekata iz skupa podataka; ili
2. vrednosti atributa je neobična u odnosu na ostale vrednosti atributa.

Nedostajuce vrednosti

Nedostajuce vrednosti predstavljaju polja u skupu podataka koja su prazna. Prazna polja mozemo da imamo ukoliko ta vrednosti nije prikupljena, na primer, ako osoba nije htela da iskaze svoj broj godina. Takodje, prazna polja nastaju ukoliko, su bila uslovna u popunjavanju formi. Kako god ona se moraju uzeti u obzir.

Eliminisanje objekta podataka ili atributa. Jednostavan i efikasan način je eliminisati slogove ili attribute tamo gde imamo neku nedostajucu vrednost. Mana ovog pristupa je to što ukoliko imamo puno nedostajucih vrednosti nije moguće dobiti dobar rezultat analize kako gubimo puno informacija. Prednosti ove metode jeste to što ukoliko ima veoma malo nedostajucih vrednosti brisanjem nekoliko slogova ne utice na analizu, ali ovo se ipak treba raditi sa oprezom, jer čak i tada oni mogu imati ključne informacije za analizu.

Procena nedostajuce vrednosti. Umesto nedostajucih vrednosti mozemo jednostavno proceniti vrednost nekog polja. Kod vremenskih serija procenu mozemo izvršiti tako što interpoliramo između vrednosti u trenutku pre i trenutku posle datog atributa. Ako su podaci neprekidni mozemo koristiti aritmetičku sredinu između susedna dva objekta; ako su kategorički mozemo koristiti onaj koji se najcesce pojavljuje.

Ignorisanje nedostajuce vrednosti prilikom analize. Mnogi algoritmi istraživanje podataka mogu se modifikovati tako da rade sa skupovima podataka koji imaju nedostajuce vrednosti.

Nekoinzistentne vrednosti

Skup podataka može da ima nekoinzistentne vrednosti. Moguće je da je došlo do zamene dve cifre pri unosu podataka, ili je pogrešno seknirana ručno napisana cifra, itd. Za ovakve probleme moramo da imamo odgovarajuće metode pronalazenja i ispravljanje ovih gresaka. Neki nekoinzistentne vrednosti se lako otklanjaju, kao što je, na primer, broj godina neke osobe ne može biti negativan. Za lakše otkrivanje ovih gresaka dobro je znati domen svakog atributa. Za ispravljanje obično moramo imati dodatnu informaciju o vrednostima nekog atributa.

Duplikati

Skup podataka, takođe, može imati objekte podataka koji su duplikati, ili su skoro duplikati. Za pronalazenje duplikata, prvo se mora ispitati da li dva sloga koja imaju slice vrednosti atributa predstavljaju isti objekat, a drugo moramo biti sigurni da dva slična sloga zapravo predstavljaju dva različita objekta.

Problemi pri primenama podataka

Skup podataka je visokog kvaliteta ako se može koristiti za svoje nemene. Ovakav pristup se pokazao veoma korisnim. Ali, takođe i za ovakve skupove podataka postoje problemi:

Starost. Puno podataka postaje staro čim se prikupi, kao što je na primer, pretraživanje weba. Ako su podaci stari, onda je bilo kakav model ili šablon prepoznat nad njima takođe star.

Relevantnost. Dostupni skupovi podataka moraju biti relevantni za svoju primenu. Ako na primer ispitujemo saobraćajne nesreće, onda ukoliko nemamo informaciju o broju godina vozača i/ili o polu vozača, vrlo verovatno naša analiza neće biti toliko tačna. Takođe, kao što su atributi bitni, bitni su i slogovi, jer može doći do **pristrasnosti pri uzorkovanju**, tj. ako pri uzorkovanju dobijamo podatke od osoba koje hoće raditi anketu.

Znanje o Podacima. Najbolje bi bilo da skupovi podataka idu zajedno sa dokumentacijom, koja opisuje taj skup podataka, tipove njegovih atributa, i domene vrednosti atributa, skalu merenja, poreklo i preciznost podataka. Pa tako ukoliko -999 predstavlja nedostajuću vrednost, onda će naša analiza zasigurno biti pogrešna ukoliko nemamo tu informaciju.

Predprocesiranje podataka

Predprocesiranje podataka je široka oblast koja ima brojne tehnike i strategije, neke od kojih su:

- Agregacija
- Uzorkovanje
- Redukcija dimenzija
- Odabir podskupa karakteristika
- Kreiranje karakteristika
- Diskretizacija i binarizacija
- Transformacija promenljivih

Agregacija

Agregacija je proces u kome se dva ili više objekta spajaju u jedan objekat. Razmotrimo skup podataka koje predstavlja transakcije u prodavnicama u raznim gradovima za različite dane u godini. Jedan način da se izvrši agregacija jeste da se sve prodavnice iz jednog grada zamene sa jednom prodavnicom koja predstavlja ceo grad.

...	Grad	Cena	Datum	...
...	BG	590din	05/03/2021	...
...	NS	230din	05/03/2021	...
...	NI	540din	05/03/2021	...
...	BG	240din	05/03/2021	...
...	NI	100din	08/03/2021	...

...	Grad	Cena	Datum	...
...

Ovde dolazi do jednog očiglednog problema, šta će biti ostale vrednosti atributa, kao što je cena, i proizvod. Cene možemo sumirati, dok proizvode možemo spojiti u novi skup koji sadrži proizvode iz svih gradova. Kvantitativni atributi se spajaju sumiranjem ili prosekom, dok se kvalitativni atributi spajaju uniraju.

...	Grad	Cena	Datum	...
...	BG	830 <i>din</i>	05/03/2021	...
...	NS	230 <i>din</i>	05/03/2021	...
...	NI	540 <i>din</i>	05/03/2021	...
...	NI	100 <i>din</i>	08/03/2021	...
...

Prednosti agregacije su to što će istraživanje podataka da se vrsi na skupu podataka koji je dosta manji, pa će zauzimati manje memorijskog prostora i samim tim će izračunavanje biti brže. Takođe, agregacija može da posluži kao menjanje oblasti koje podaci pokrivaju, sa uskog na široko. Agregacija poboljšava stabilnost podataka. Mane agregacije su to što možemo izgubiti detalje koji mogu biti bitni.

Uzorkovanje

Uzorkovanje je odabir podskupa od skupa podataka nad kojim će se vrsiti analiza. Uzorkovanje u statistici i istraživanju podataka se razlikuje u tome što kod statističkih analiza vremenski je ograničeno sakupljanje podataka, dok je u istraživanju podataka to iz razloga zato što vremenski zahtevno procesuirati ogroman broj podataka.

Analizom uzorka dobijamo iste rezultate kao i analizom celog skupa podataka sve dok je uzorak reprezentativan. Uzorak je **reprezentativan** ako ima približne vrednosti osobina kao i originalan skup podataka. Ako nam je osobina očekivanja bitna, onda je uzorak reprezentativan ako ima približno očekivanje celom skupu podataka.

Pristupi uzorkovanju

Najjednostavnija tehnika uzorkovanja je **nasumično biranje uzorka**. Njegova karakteristika je to da svaki objekat skupa podataka može biti izabran sa istom verovatnoćom. Postoje dve varijante:

1. **Bez vraćanja** — kada izaberemo neki objekat ne vraćamo ga nazad u **populaciju**.
2. **Sa vraćanjem** — objekte ne izbacujemo iz populacije, pri odabiru.

Kada populacija sadrži objekte koji su drugacijeg tipa, i pri tome imamo veliku razliku u broju tipova, nasumično biranje uzorka neće lepo raditi, kako može da ne izabere objekte nekog tipa koji su značajni za analizu, na primer, pri klasifikaciji. Zato se koristi **stratifikovano uzorkovanje**, koje uzima u obzir grupe u kojima objekti pripadaju. Najjednostavnije je birati isti broj objekta iz svake grupe. Malo složenija varijacija je biranje objekata proporcionalno veličini grupe.

Primer (Uzorkovanje i Gubitak informacije). Kada se izabere tehnika, ostaje izabrati kolika će biti veličina uzoraka. Ako je veličina uzoraka velika gubimo lepa svojstva uzorkovanja, dok ukoliko je veličina uzoraka mala možemo izgubiti bitne informacije.

Progresivno uzorkovanje

Odgovarajuću veličinu uzorka je teško odrediti, pa se **adaptivno** ili **progresivno uzorkovanje** koristi. Ovaj pristup podrazumeva da se krene sa malim uzorkom, i da se veličina uzorka progresivno povećava vremenom, dok se ne dobije odgovarajuća veličina. Iako se ova tehnika čini jednostavnom, teško je odrediti kada stati sa povećavanjem veličine. Na primer, ako imamo prediktivni model, sa povećanjem veličine uzorka

dobijamo bolju tacnost, ali ako dodjemo do tacke preloma, tacnost modela ce se smanjivati, a model ce postati pretreniran. Zato je od kljucne vaznosti znati gde je prelomna tacka i gde treba prestati sa treniranjem.

Redukcija dimenzija

Postoji mnogo skupova podataka koji imaju mnogo karakteristika (dimenzija). Jedan on benefita smanjivanja dimenzije je to sto mnogi algoritmi rade bolje nad podacima koji imaju manje dimenzija, tj. mnoge dimenzije samo dodaju sum na podacima. Takodje smanjivanje dimenzija moze da se koristi pri vizuelizaciji podataka, a i ima memorijsku i vremensku optimalnost.

Redukcija dimenzija se odnosi na tehniku smanjivanja dimenzionalnosti skupa podataka tako sto se novi atributi kreirao kombinacijom starih.

Prokletstvo dimenzionalnosti

Izraz prokletstvo dimenzionalnosti se odnosi na fenomen da mnogi tipovi analiza postaju tezi kada se dimenzionalnost povecava. Ovo je najizrazitije kod klasifikacije, i klasterovanja.

Tehnike linearne algebre za redukciju dimenzija

Principal Components Analysis (PCA) je tehnika linearne algebre za neprekidne attribute koje nalaze nove attribute koji su:

1. linearna kombinacija originalnih atributa;
2. ortogonalni jedni na druge; i
3. opisuju maksimalno varijacije u podacima

Definicija. Za datu $\mathbf{D}_{m \times n}$ matricu podataka, kovarijansa matrice \mathbf{D} je matrica \mathbf{S} , cije su s_{ij} definisani kao

$$s_{ij} = \text{cov}(\mathbf{d}_{*i}, \mathbf{d}_{*j})$$

Kovarijansom dobijamo koliko su atributi zavisni jedni na druge.

Cilj PCA je da nadje transformaciju podataka tako da zadovoljava sledece osobine:

1. Svaki razliciti par novih atributa ima 0 kovarijance.
2. Atributi su uredjeni u odnosu na to koliko razlicitosti podataka oni opisuju (mera je disperzija).
3. Prvi atributu opisuje najvise razlicitosti moguće podatak (mera je disperzija).
4. Svaki sledeci atribut opisuje sto je vise moguće preostalih razlicitosti (mera je disperzija).

Ove osobine mozemo dobiti tako sto koristimo sopstvene vrednosti matrice kovarijanse. Neka su $\lambda_1, \dots, \lambda_n$ kao sopstvene vrednosti od \mathbf{S} . Sopstvene vrednosti su ne-negativne, i mogu se urediti tako da $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$. Neka je $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_n]$ matrica sopstvenih vektora \mathbf{S} , tako da i -ti sopstveni vektor odgovara i -toj sopstvenoj vrednosti. Konacno, predpostavimo da je matrica \mathbf{D} preprocesirana tako da je ocekivanje svakog atributa (kolone) jednako 0. Onda vazi sledece:

- Matrica podataka $\mathbf{D}' = \mathbf{D}\mathbf{U}$ je skup transformisanih podataka koji zadovoljavaju uslove navedene gore.
- Svaki novi atribut je linearna kombinacija originalnih atributa, cije su tezine za i -ti atributa i -ti sopstveni vektor, a to imamo iz definicije mnozenja matrica.
- Disperzija novog i -tog atributa je λ_i .
- Suma disperzija originalnih atributa je jednaka sumi disperzija novih atributa.
- Novi atributi se zovi **glavne komponenta**, tj. prvi novi atribut je prva glavna komponenta, drugi novi atribut je druga glavna komponenta, itd. . .

Diskretizacija i Binarizacija

Mnogi algoritmi istrazivanja podataka zahtevaju da podaci imaju kategoricke attribute (binarne attribute). Zbog toga je cesto neophodno konvertovati attribute koji su neprekidni u kategoricke (**diskretizacija**), ili neprekidne i kategoricke u binarne.

Binarizacija

Ako imamo m kategorickih atributa, onda svakom od atributa, dodelimo jedan ceo broj iz intervala $[0, m-1]$. Sada konvertujemo tih m celih brojeva u $n = \lceil \log_2(m) \rceil$ binarnih atributa.

Kategoricka vrednst	Celi Broj	x_1	x_2
dobar	0	0	0
low	1	0	1
zao	2	1	0

Kod ovakve transformacije može da dodje do probleme, i stvaranja veza imezju transformisanih atributa. Stavise, kod nekih analiza su nam potrebani asimetrični binarni atributi. Zbog toga kod asimetričnih binarnih atributa moramo da uvedemo atribut x_3 , kako bi svaki atribut predstavljao po jednu kategoricku vrednost.

Kategoricka vrednst	Celi Broj	x_1	x_2	x_3
dobar	0	1	0	0
low	1	0	1	0
zao	2	0	0	1

Diskretizacija neprekidnih atributa

Transformacija neprekidnih atributa u kategoricke attribute zahteva: odredjivanje broja kategorija i odredjivanje mapiranje vrednosti neprekidnih atributa u te kategorije. Kada se vrednosti neprekidnog atributa sortiraju, onda se oni dele na n intervala tako se se odrede $n-1$ **razdvojnih tacaka**. Onda se sve vrednosti jednog intervala mapiraju na istu kategoricku vrednost. Pa se diskretizacija svodi na odredjivanje koliko razdvojnih tacaka hocemo da imamo i gde da ih postavimo. Rezultat se predstavlja kao niz nejednakosti $x_0 < x_1 < \dots < x_n$.

Neinformisana diskretizacija. Diskretizacija u kojoj se ne koristi informacija klase. Na primer, pristup **jednake duzine** deli domen atributa u odredjeni broj intervala koji su iste duzine. Pristu **jednake frekvencije (jednake dubine)** se koristi kako bi se izbegli autlajeri pri pristupu jednakih duzina, tako sto u svakom intervala proba da stavi isti broj objekta. Jos jedna tehnika diskretizacije je preko algoritma **K-sredine**.

Informisana diskretizacija. Informisana diskretizacija koristi dodatne informacije o klasama, te cesto ima bolje rezultate. Primer je pristup diskretizacije sa entropijom. Neka je k broj razlicitih labele klasa, m_i broj vrednosti u i -tom intervalu particije, i m_{ij} broj vrednosti klase j u intervalu i . Onda je entropija e_i intervala i data sa

$$e_i = \sum_{j=1}^k p_{ij} \log_2(p_{ij}),$$

gde je $p_{ij} = m_{ij}/m_i$ verovatnoca da je klase j u intervalu i . Potpuna entropija e particije je tezinski prosek individualnih entropije intervala, tj.

$$e = \sum_{i=1}^n w_i e_i,$$

gde je $w_i = m_i/m$ odnost broja vrednosti u intervalu i i ukupnog broja vrednosti m , i n je broj intervala. Intuitivno, entropija intervala je mera cistoce tog intervala. Ako interval sadrzi samo vrednosti jedne klase

(onda je cist), tada je entropija 0 i ne doprinosi potpunoj entropiji. Ako su klase vrednosti u intervalu pojavljuju jednako cesto (onda je prljav), tada je entropija maksimalna.

Pristup particionisanja neprekidnog atributa pocinje tako sto se inicijalne vrednosti dele u 2 intervala sa minimalnom entropijom. Ova tehnika se nastavlja nad intervalom sa najvecom entropijom, sve dok se ne zadovolji neki kriterijum ili ne dodjemo do odredenog broja intervala.

Kategoricki atributi sa previse vrednosti

Ako su kategoricki atributi ordinalni(redni) atributi, onda mozemo koristiti tehnike slicne onim za neprekidne attribute. Ali ako imamo nominalne(imenske) attribute, onda su nam potrebni drugi pristupi. Na primer, hocemo da diskretizujemo fakultete nekog univerziteta. Znamo da mozemo da ih podelimo u vece grupe, kao sto su to *prirodne nauke*, *drustvene nauke*, i *umetnost*. Ako nemamo dodatna znanja o kategorijama, onda moramo koristiti neke empirijske tehnike kao sto je nasumicno grupisanje koje nam daje najbolji rezultat.

Mere slicnosti i razlicitosti

Mere slicnosti i razlicitosti su bitne za mnoge tehnike istrazivanja podataka, kao sto je klasterovanje, klasifikacije i otkrivanje anomalija. U mnogim slucajevima, inicijalni skup podataka nije bitan nakon sto se izracunaju slucnosti i razlicitost, tj. prelazi se sa prostora skupa podataka na prostor slicnosti i razlicitosti i na tom prostoru se primenjuju analize.

Osnove

Definicije

Slicnost izmedju dva objekta je numericka mera kojom se meri koliko 2 objekta lice jedan na drugi. Slicnost je *veca* ako 2 objekta vise lice jedan na drugi. Slicnost je obicno ne-negativna i izmedju 0 (nema slicnosti) i 1 (kompletna slicnost).

Razlicitost imedju dva objekta je numericka mera kojom se meri koliko 2 objekta imaju razlika. Razlicitost je *manja* ako 2 objekta vise lice jedan na drugi. Izraz **rastojanje (distanca)** se koristi kao sinonim razlicitosti, ali je ustvari on specijalna klasa razlicitosti. Razlicitost je obicno u intervalu od 0 do 1 ili od 0 do ∞ .

Blizina (Proximity) je mera koja oznacava slicnost i razlicitost.

Transformacije

Transformacije se obicno primenjuju za konvertovanje slicnosti u razlicitost, i obrnuto, ili da blizinu iz nekog intervala preslikaju u $[0, 1]$.

Transormacija slicnosti/razlicitost u interval $[0, 1]$ je data izrazima:

$$s' = (s - s_{min}) / (s_{max} - s_{min})$$

$$d' = (d - d_{min}) / (d_{max} - d_{min})$$

gde je s_{min}, s_{max} minimalna i maksimalna vrednost za slicnost, i d_{min}, d_{max} minimalna i maksimalna vrednost za razlicitost. Za mere blizine iz intervala $[1, \infty]$, moramo koristite neke ne-linearne transformacije kao sto je $d' = d / (1 + d)$. Pri ovoj transformaciji veliki brojevi se gomilaju oko 1, sto moze da smeta, ali i ne mora u zavisnosti da li to hocemo ili ne. Takodje, ako transformisemo iz intrvala $[-1, 1]$ u interval $[0, 1]$ apsolutnom vrednoscu, takodje moze doci do gubitka informacije.

Transformacija izmedju slicnosti i razlicitosti je jednostavna ako se nalaze u intervalu $[0, 1]$ i moze se definisati kao $d = 1 - s$ ($s = 1 - d$). U slucaju da ne upadaju u interval $[0, 1]$ mogu se primeniti neke druge transformacije kao sto su:

$$s = 1 / (d + 1), s = e^{-d}, s = 1 - (d - d_{min}) / (d_{max} - d_{min}).$$

Slicnost i Razlicitost izmedju jednostavih atributa

Blizina objekata sa vecim brojem atributa je tipicno kombinacija blizina individualnih atributa, pa zbog toga razmotrimo blizine imedju objekata koji imaju samo jedan atribut.

Neka su objekti opisani jednim nominalnim (imenskim) atributom. Sta onda znaci da su ta dva objekta slicna ili razlicita? Kako nominalni (imenski) atributi sadrze samo informaciju o tome da li su dva objekta ista ili razlicita, onda slucnost i razlicitost definisemo kao:

$$s = \begin{cases} 1 & \text{ako } x = y \\ 0 & \text{ako } x \neq y \end{cases}$$
$$d = \begin{cases} 0 & \text{ako } x = y \\ 1 & \text{ako } x \neq y \end{cases}$$

Za objekte sa jednim ordinalnim (rednim) atributom informacija o uredjenju se mora postovati. Razmotrimo primer *dobar, los, zao*. Razumno je ako je osoba *dobar* da se nece druziti sa osobom *zao*, ali da ce se mozda druziti sa osobom *los*, slicno i za osobu *zao*, dok ce se *los* mozda druziti sa osobom *dobar* ili *zao*. Zbog toga prvi korak je dodeliti cele brojeve ovom vrednostima atributa, tj. $dobar = 0, los = 1, zao = 2$. Onda je razlicitost izmedju ovih osoba data kao $d(zao, dobar) = (2-0)/2 = 1$, a slicnost je data kao $s = 1 - d = 0$ (*dobar* i *zao* su kompletno razliciti, tj. nema slicnosti). U opstem slucaju dobijamo:

$$d = |x - y|/(n - 1), s = 1 - d$$

Za intervale ili razmere, prirodna mera razlike izmedju dva objekta je apsolutna razlika njegovih vrednosti. Za ovakve attribute obicno se koristi interval $[1, \infty]$. Slicnost se dobija nekom transformacijom iz razlicitosti. Formalno:

$$d = |x - y|, s = -d; s = 1/(1 + d); s = e^{-d}; s = 1 - (d - d_{min})/(d_{max} - d_{min})$$

Razlicitosti izmedju objekta podataka

Rastojanja

Euklidsko rastojanje d , izmedju dve tacke \mathbf{x} , i \mathbf{y} , u n -dimenzionalnom prostoru je dato sa:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{k=1}^n (x_k - y_k)^2}$$

Rastojanje Minkovskog je generalizacija euklidskog rastojanja:

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{k=1}^n |x_k - y_k|^r \right)^{1/r}$$

- Za $r = 1$ imamo Manhetn rastojanje (L_1 norma)
- Za $r = 2$ imamo Euklidsko rastojanje (L_2 norma)
- Za $r \rightarrow \infty$ imamo Supremum rastojanje (L_{max} ili L_∞ norma)

Definicija Funkciju $d : X \times X \mapsto \mathbb{R}$ zovemo **metrikom** ako vazi $\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in X$:

1. $d(\mathbf{x}, \mathbf{y}) \geq 0$
2. $d(\mathbf{x}, \mathbf{y}) = 0$ akko $x = y$
3. $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$

$$4. d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{x})$$

Za mnoge razlicitosti, hocemo da vazi da su metrike, jer nam to garantuje tacnost nekih algoritama. Za rastojanje Minkovskog vazi da je metrika, dok mnoge razlicite ne zadovoljavaju jednu ili vise osobina metrike.

Primer (Ne-metricka razlicitost: Razlika skupova). Definise rastojanje d imedju dva skupa A i B kao $d(A, B) = |A - B|$. Ovako definisano rastojanje ne zadovoljava samo osobinu pozitivnosti. Ali za funkciju $d(A, B) = |A - B| + |B - A|$, vazi da je metrika.

Primer (Ne-metricka razlicitost: Vreme). Definise meru rastojanja izmedju casova u danu kao:

$$d(t_1, t_2) = \begin{cases} t_2 - t_1 & \text{ako } t_1 \leq t_2 \\ 24 + (t_2 - t_1) & \text{ako } t_1 \geq t_2 \end{cases}$$

Slicnosti izmedju objekta podataka

Ako je $s(\mathbf{x}, \mathbf{y})$ mera slicnosti izmedju dve tacke \mathbf{x} i \mathbf{y} , onda su njene tipicne osobine $\forall \mathbf{x}, \mathbf{y} \in X$:

1. $s(\mathbf{x}, \mathbf{y}) = 1$ akko $\mathbf{x} = \mathbf{y}$
2. $s(\mathbf{x}, \mathbf{y}) = s(\mathbf{y}, \mathbf{x})$.

Primer (Ne-simetricne mere slicnosti). Neka se vrsi eksperiment klasifikovanja napisanih slova nad ljudima. **Matrica konfuzije** sadrzi u sebi slogove koliko se puta neko slovo javlja i koliko se puta zamenilo sa nekim drugim karakterom. Na primer, '0' se pojavljuje 200 puta, ali je klasifikovana kao '0' 160 puta, i kao 'o' 40 puta, slicno, 'o' se pojavljuje 200 puta, ali je klasifikovano 170 puta kao 'o', i 30 puta kao '0'. Jasno je da ovde ne vazi simetrija. Zbog toga u ovakvim situacijama koristimo novu meru slicnosti

$$s'(\mathbf{x}, \mathbf{y}) = (s(\mathbf{x}, \mathbf{y}) + s(\mathbf{y}, \mathbf{x}))/2.$$

Primeri mera blizine

Mera slicnosti za binarne podatke

Mera slicnosti imedju objekta koji sadrže samo binarne atribute se nazivaju **keoficijenti slicnosti**, i tipicno imaju vrednosti imedju 0 i 1. Neka su \mathbf{x} i \mathbf{y} dva objekta koja imaju n binarnih atributa. Njihovim upoređivanjem dobijamo:

$$f_{00} = \text{broj atributa gde je } \mathbf{x} \text{ 0 i } \mathbf{y} \text{ je 0}$$

$$f_{01} = \text{broj atributa gde je } \mathbf{x} \text{ 0 i } \mathbf{y} \text{ je 1}$$

$$f_{10} = \text{broj atributa gde je } \mathbf{x} \text{ 1 i } \mathbf{y} \text{ je 0}$$

$$f_{11} = \text{broj atributa gde je } \mathbf{x} \text{ 1 i } \mathbf{y} \text{ je 1}$$

Jednostavno uparivanje keoficijenata. (Simple matching coefficient — SMC)

$$SMC = \frac{f_{11} + f_{00}}{f_{00} + f_{01} + f_{10} + f_{11}}$$

Zakardov keoficijent. Koristi se kada imamo asimetricne atribute, jer bi u tom slucaju SMC racunao i one koji nam nisu od znacaja.

$$J = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}$$

Primer (SMC i Zakardov keoficijent). Neka su $\mathbf{x} = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0)$ i $\mathbf{y} = (0, 0, 0, 0, 0, 0, 1, 0, 0, 1)$, onda imamo da je $f_{00} = 7, f_{01} = 2, f_{10} = 1, f_{11} = 0$, te sledi da je

$$SMC = \frac{0 + 7}{7 + 2 + 1 + 0} = 0.7$$

$$J = \frac{0}{2 + 1 + 0} = 0$$

Kosinusna slicnost

Ako posmatramo skupove podataka koji dokumenti, takodje kao kod Zakardovih koeficijenata ne posmatramo kada su uparnene dve nule, ali pored toga moramo da znamo da poredimo dva ne-binarna vektora. **Kosinusna slicnost** je mera slucnosti dokumenata definisana nad dva vektora \mathbf{x} i \mathbf{y} kao

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

Primer (Kosinusna slicnost imedju dva vektora dokumenta). Neka su $\mathbf{x} = (3, 2, 0, 5, 0, 0, 0, 2, 0, 0)$, i $\mathbf{y} = (1, 0, 0, 0, 0, 0, 0, 1, 0, 2)$, onda je

$$\begin{aligned}\mathbf{x} \cdot \mathbf{y} &= 5 \\ \|\mathbf{x}\| &= \sqrt{(3 \cdot 3 + 2 \cdot 2 + 5 \cdot 5 + 2 \cdot 2)} = 6.48 \\ \|\mathbf{y}\| &= \sqrt{(1 \cdot 1 + 1 \cdot 1 + 2 \cdot 2)} = 2.24 \\ \cos(\mathbf{x}, \mathbf{y}) &= 0.31\end{aligned}$$

Prosireni Zakardov keoficijent

Prosireni Zakardov koeficijenata se koristi za skup podataka koji je dokument i koji postaje Zakardov koeficijent u slucaju da je skup podataka binarnih atributa. Definisan je kao:

$$EJ(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - \mathbf{x} \cdot \mathbf{y}}$$

Korelacija

Korelacija izmedju dva objekta podataka koji imaju binarne ili neprekidne attribute je mera linearne zavisnosti izmedju atributa objekta. **Pirsonov koeficijent korelacije** izmedju dva objekta podataka \mathbf{x} i \mathbf{y} , je definisan kao

$$\rho_{\mathbf{x}, \mathbf{y}} = \frac{\text{cov}(\mathbf{x}, \mathbf{y})}{\sigma_{\mathbf{x}} \sigma_{\mathbf{y}}}$$

Primer (Savrsene Korelacija). Korelacija je uvek u intervalu $[-1, 1]$. Korelacije od 1 (ili -1) znaci da su \mathbf{x} i \mathbf{y} Savrseno pozitivne linearne kombinacije, tj. $x_k = ay_k + b$, gde su a i b konstante.

Primer (Savrseno Nekolerisane). Korelacija je **nekorelisana**, ako je $\rho_{\mathbf{x}, \mathbf{y}} = 0$, sto znaci da nema nikakve linearne zavisnosti izmedju dva objekta. Ali to ne znaci da ne postoji neka ne-linearna zavisnost.

Bregmanova divergencija

Bregmanova divergencija je familija funkcija blizine koji imaju neke zajednicke osobine. To su funkcije gubitka ili distorzije. Neka su \mathbf{x} i \mathbf{y} dve tacke, gde je \mathbf{y} originalna tacka i \mathbf{x} neka distorzija ili aproksimacija tacke \mathbf{y} . Cilj je odrediti meru distorzije ili gubitka koji se javlja kada se \mathbf{y} aproksimira sa \mathbf{x} .

Definicija (Bregmanova divergencija). Neka je data strogo konveksna funkcija ϕ , Bergmanova divergencija (funkcija gubitka) $D(\mathbf{x}, \mathbf{y})$ generisana funkcijom ϕ je data kao:

$$D(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) - \phi(\mathbf{y}) - \langle \nabla \phi(\mathbf{y}), (\mathbf{x} - \mathbf{y}) \rangle$$

gde je $\nabla \phi(\mathbf{y})$ gradijent funkcije ϕ u tacki \mathbf{y} , i $\langle \nabla \phi(\mathbf{y}), (\mathbf{x} - \mathbf{y}) \rangle$ je unutasnji proizvod izmedju $\nabla \phi(\mathbf{y})$ i $(\mathbf{x} - \mathbf{y})$.

$D(\mathbf{x}, \mathbf{y})$ može da se zapise kao $D(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) - L(\mathbf{x})$, gde je $L(\mathbf{x}) = \phi(\mathbf{y}) + \langle \nabla \phi(\mathbf{y}), (\mathbf{x} - \mathbf{y}) \rangle$ jednačina ravni koja je tangentna na funkciju ϕ u tački \mathbf{y} . Pa je Bregmanova divergencija samo razlika između funkcije i njene linearne aproksimacije.

Problemi pri racunanju blizine

1. Kako resiti slucaj kada atributi imaju drugacije domene i/ili su korelisani?
2. Kako izracunati blizinu objekta koji imaju drugacije tipove atributa?
3. Kako izracunati blizinu kada atributi imaju drugacije tezine, tj. kada svi atributi uticu drugacije na blizinu objekata?

Standardizacija i Korelacija za mere rastojanja

Problem može da nastane kada se meri rastojanje kada atributi nemaju isti opseg vrednosti. Na primer, jedan atribut ima domen u intervalu $[0, 100]$, dok drugi ima domen u intervalu $[1000, 100000]$. Pri racunanju Euklidskog rastojanja, veci uticaj ima drugi atribut.

Generalizacija Euklidovog rastojanja je **Mahalanobijevo rastojanje**, koje se koristi kada su atributi korelisani, imaju drugacije domene, i kada je distribucija podataka približna normalnoj (Gausovoj).

Definicija: Mahalanobijevo rastojanje između dva objekta \mathbf{x} i \mathbf{y} je dato sa

$$mahalanobis(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y}) \sum^{-1} (\mathbf{x} - \mathbf{y})^T$$

gde je \sum^{-1} je inverz matrice konvergencije podataka.

Spajanje slicnosti za heterogene attribute

Prethodne definicije slicnosti su bile bazirane na pristupe koji pretpostavljaju da su atributi istog tipa. Generalni pristup je potreban kada su tipovi atributi razliciti. Najjednostavniji pristup je izracunati slicnosti za svaki od atributa i onda nekako spojiti (sabrati ili uzeti prosek) taj rezultat u slicnost između 0 i 1. Ovaj pristup moramo izmeniti kako bi radio i za asimetrične attribute.

Algoritam (Slicnosti heterogenih atributa)

1. $\forall k$ izracunati slicnost k -tog atributa $s_k(\mathbf{x}, \mathbf{y})$, u intervalu $[0, 1]$
2. Definisati indikatorsku promenljivu δ_k za k -ti atribut

$$\delta_k = \begin{cases} 0 & \text{ako je } k\text{-ti atribut asimetričan i ima vrednost } 0, \text{ ili ako nedostaje vrednost } k\text{-tog atributa} \\ 1 & \text{inace} \end{cases}$$

3. Izracunati totalnu slicnost između dva objekta kao:

$$sim(\mathbf{x}, \mathbf{y}) = \frac{\sum_{k=1}^n \delta_k s_k(\mathbf{x}, \mathbf{y})}{\sum_{k=1}^n \delta_k}$$

Koriscenje Tezina

Cesto ne zelimo da nam svi atributi vrede isto pri racunanju slicnosti pa zato definisemo tezinu attribute k sa realnom vrednoscu $w_k \in [0, 1]$. Takodje moramo izmeniti totalnu slicnost i to tako da ukljucuje tezinu w_k :

$$sim(\mathbf{x}, \mathbf{y}) = \frac{\sum_{k=1}^n w_k \delta_k s_k(\mathbf{x}, \mathbf{y})}{\sum_{k=1}^n \delta_k}$$

Pretrazivanje Podataka

Klasifikacija: Osnovni koncepti, drveta odlucivanja

Klasifikacija ima za zadatak da dodeli jednu ili vise klasa nekom objektu. Neki primeri su klasifikovanje celija, galaksija, detekcija spam email poruka.

Osnove definicije

Ulazni podatak za klasifikaciju je kolekcija slogova. Svaki slog, takodje nazvan i instanca ili primer, se kategorizuje torkom (\mathbf{x}, y) , gde je \mathbf{x} skup atributa i y je specijalni atribut (kategoricki/ciljani/target atribut). Sledeca tabela sadrzi skup podataka za klasifikovanje zivotinja u sledece kategorije: sisar, gmizavac, riba, vodozemac, ptica. Skup atributa moze imati i neprekidne vrednosti, ali klasna oznaka mora da bude diskretni atribut. Ako je y neprikidni atribut, onda se ovaj postupak naziva **regresija**.

Ime	Temperatura tela	Zenka radja	Koza	Morska stvorenja	Vazdusna stvorenja	Ima noge	Hibernira	Klasa
covek	toplo-krvni	da	dlake	ne	ne	da	ne	sisar
piton	hladno-krvni	ne	krljosti	ne	ne	ne	da	gmizavac
losos	hladno-krvni	ne	krljosti	ne	da	ne	da	riba
kit	toplo-krvni	da	dlake	da	ne	ne	ne	sisar
zaba	hladno-krvni	ne	nista	semi	ne	da	da	vodozemci
komodo	hladno-krvni	ne	krljosti	ne	ne	da	ne	gmizavac
papagaj	toplo-krvni	ne	perije	ne	da	da	ne	ptica
macka	toplo-krvni	da	krzno	ne	ne	da	ne	sisar
kornjaca	hladno-krvni	ne	krljosti	semi	ne	da	ne	gmizavac
pingvin	toplo-krvni	ne	perije	semi	ne	da	ne	ptica

Definicija (Klasifikacija). Klasifikacije je zadatak ucenja **ciljne funkcije** f koja slika svaki skup atributa \mathbf{x} u jednu predefinisanu klasnu oznaku y . Funkcija f se takodje naziva i **klasifikacioni model**.

Opisno modelovanje. Klasifikacioni model moze da služi za opisivanje razlika imezju objekata drugih klasa. U primeru gore dobro je znati koje osobine ima sisar, ptica, riba, itd. . .

Model predvidjanja. Klasifikacioni model moze da se koristi za predvidjanje klase nepoznatih slogova. Na primer mozemo predvideti klasu za zivotinju gila monstnum:

Ime	Temperatura tela	Koza	Morska stvorenja	Vazdusna stvorenja	Ima noge	Hibernira	Klasa
gila monstnum	hladno-krvni	krljosti	ne	ne	da	da	?

Klasifikacione tehnike daju najbolje rezultate za predvidjanje ili opisivanje skupova podataka sa binarnim ili nominalnim(imenskim) kategorijama. Manje su efikasne za ordinalne(redne) kategorije zato sto ne razmatraju implicitan poredak izmedju kategorija.

Generalni pristup resavanja klasifikacionih problema

Klasifikaciona tehnika je sistemacki pristup pravljenja klasifikacionih modela od ulaznog skupa podataka. Neki primeri su drveta odlucivanja, neuronske mreze, pomocne vektor masine, naivni Bajesov klasifikator, klasifikator zasnovan na pravilima. Svaka tehnika pruza **algoritam ucenja** koji identifikuje model koji najbolje odgovara vezama izmedju skupa atributa i klasne oznake ulaznih podataka. Ovaj model treba da odgovara ulaznim podacima, ali takodje mora i da tacno predviti klasne oznake slogova koje jos nije video. Zbog toga je kljucni zadatak algoritma ucenja da napravi dobru model sa dobrom generalizacijom, tj. model koji tacno predvidja klasne oznake za nepoznate slogve.

Skup za treniranje sadrzi slogove cije su klasne oznake poznate. On služi za pravljenje klasifikacionog modea, koji se nakon toga primenjuje na **skup za testiranje**, koji sadrzi slogove sa nepoznatim klasnim oznakama.

Performanse klasifikacionog modele se dobijaju brojanjem test slogova koje je model predvideo tacno i netacno. Ove vrednosti se cuvaju u **matrici konfuzije**.

	class=1	class=0
class=1	f_{11}	f_{10}
class=0	f_{01}	f_{00}

Ova tabela predstavlja matricu konfuzije za binarnu klasifikaciju. Svaki element matrice f_{ij} predstavlja broj slogova iz klase i , koji su predvidjeni da budu u klasi j . Ukupan broj tacnih predvidjanja modela je $f_{11} + f_{00}$, i ukupan broj netacnih predvidjanja modela je $f_{01} + f_{10}$.

Matrica konfuzije nam daje dovoljno informacija da odreditmo performance naseg modele. **Metrika performanse** moze biti:

$$\text{Tacnost} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

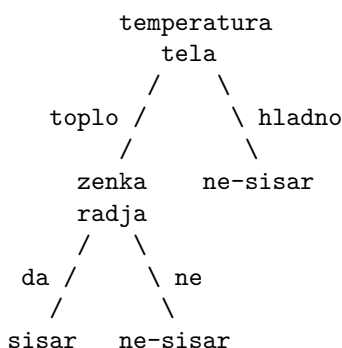
$$\text{Greska} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

Drvo odlucivanja uvod

Kako radi drvo odlucivanja?

Razmotrimo primer od malopre, samo sto cemo klasifikovati zivotinje u dve grupe: sisari i ne-sisari. Postavlja se pitanje kako odrediti da li je novo pronadjena zivotinja sisar ili nije? Jedan pristup je postavljati niz pitanja o karakteristikama te zivotinje. Prvo pitanje da li je hladno-krvna ili toplo-krvna? Ako je hladno-krvna definitivno nije sisar. Inace, je ili ptica ili sisar. Sledece pitanje moze biti da li zenke radjaju? Ako je odgovor pozitivan onda su sigurno sisari, inace vrlo verovatno nisu.

Iz primera vidimo da problem klasifikacije mozemo da resimo tako sto pazljivo postavljamo odgovarajuca pitanja o atributima sloga. Svaki put kada dobijemo odgovor, postavimo sledece pitanje, sve dok ne dodjemo do resenja. Ova pitanja i odgovori mogu se predstaviti drvetom odlucivanja, koje je hijerarhijska struktura koja sadrzi cvorove i usmerene grane.



Ovo drvo ima tri tipa cvorova:

1. **Koreni cvor** nema ulazne grane i ima nula ili vise izlaznih grana.
2. **Unutrasnji cvorovi**, imaju tacno jednu ulaznu granu i dve ili vise izlazne grane.
3. **Listovi** ili **terminali**, imaju tacno jednu ulaznu granu i nemaju izlaznih grana.

U drvetu odlucivanja, svakom listu se dodeljuje klasna oznaka. Ne-terminali, sadrže uslov atributa za odvajanje slogova koji imaju drugacije karakteristike.

Jedno kada napravimo drvo odlucivanja testiranje slogova je jednostavno. Krenemo od korenog cvora, primenimo test uslova nad atributima i pratimo granu na osnovu rezultata testiranja. Ovaj proces ponavljamo sve dok ne dodjemo do nekog terminala, koji u sebi sadrži klasnu oznaku koja nam daje resenje.

Kako napraviti drvo odlucivanja?

Postoji eksponencionalno mnogo drveta odlucivanja koja se mogu dobiti za dati skup atributa. Neka od njih su tacnija od drugih, pa pronalazenje optimalnog drveta je racunski tesko. Zbog toga postoje efikasni algoritmi koji se koriste da pronalazenje, dovoljno tacnog, suboptimalnog drveta odlucivanja u razumnom vremenu. Ovi algoritmi cesto koriste gramzivu strategiju za rast drveta odlucivanja tako sto stvaraju niz lokalno optimalnih odluka o izboru atributa za particionisanje podataka. Jedan takav algoritam je **Hantov algoritam**, koji je osnova za mnoge naprednije algoritme kao sto su ID3, C4.5, i CART.

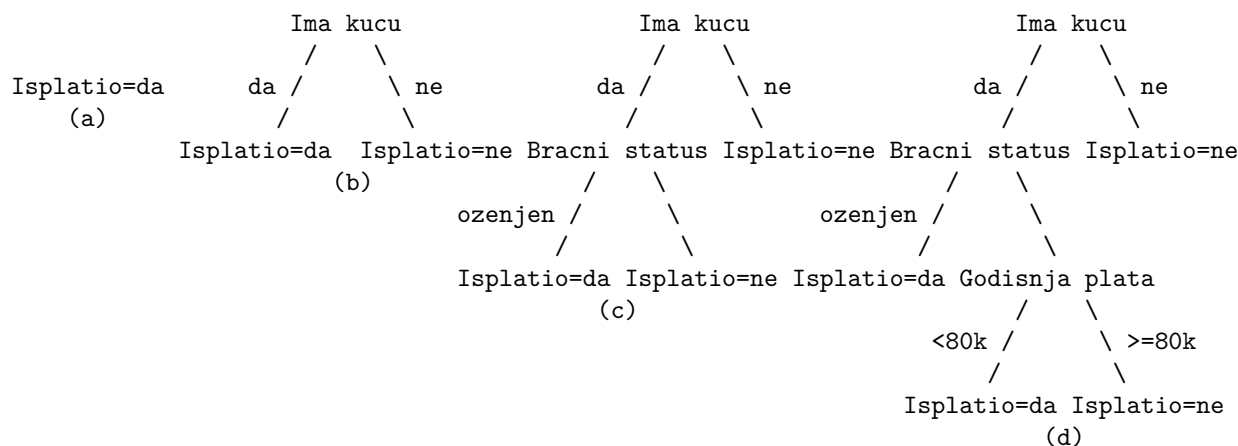
Hantov algoritam

Neka je D_t skup slogova za treniranje koji su povezani sa cvorom t i $y = \{y_1, y_2, \dots, y_c\}$ budu klasne oznake. Sledi rekurzivna definicija Hantovog algoritma.

1. Ako svi slogovi iz D_t pripadaju istoj klasi y_t , onda je t list oznacen sa y_t .
2. Ako D_t sadrzi slogove koji pripadaju vise od jedne klase, **test uslova atributa** se bira za particionisanje slogova u manje podskupove. Dete se kreira za svako resenje test uslova i slogovi iz D_t se dele deci u zavisnosti od ishoda testa. Algoritam se onda rekurzivno primenjuje na svako dete.

Primer (Primena Hantovog algoritma na predvidjanje isplate kredita). Hocemo da predvidimo da li ce neka osoba isplatiti kredit u zavisnosti od njenih osobina. Neka je skup za treniranje dat sledecom tabelom podataka:

ID	Ima kucu	Bracni status	Godisnja plata	Isplatio
1	da	neozenjen	125k	da
2	ne	ozenje	100k	da
3	ne	neozenjen	70k	da
4	da	ozenjen	120k	da
5	ne	razveden	95k	ne
6	ne	ozenjen	60k	da
7	da	razveden	220k	da
8	ne	neozenjen	85k	ne
9	ne	ozenjen	75k	da
10	ne	neozenjen	90k	ne



Inicijalno konstruisemo drvo sa jednim cvorom, koji ima klasnu oznaku **Isplatio=da** (a). Drvo moramo da rekonstruisemo kako koreni cvor ima one slogovi za koje vazi da je **Isplatio=ne**. Slogove zbog toga delimo na dve grupe pitanjem da li **Ima kucu**? Ako **Ima kucu=ne** onda znamo da sigurno **Isplatio=ne**, ali ako **Ima kucu=da** onda ne znamo da li je **Isplatio=da** (b). Onda dalje cvorove delimo sa pitanjem **Bracni status**? Ako **Bracni status=ozenjen** onda sigurno **Isplatio=da**, ali ako **Bracni status=neozenje, razveden,**

onda ne znamo da je sigurno $Isplatio=ne$ (c), pa postavljamo pitanje **Godisnja plata?** Ako je **Godisnja plata** < 80k onda vazi $Isplatio=da$, u suprotnom vazi $Isplatio=ne$ (d).

Hantov algoritam radi ako se svaka kombinacija vrednosti atributa nalazi u skupu za treniranje, sa jedinstvenom klasom oznakom. Ovo u praksi nije moguće. Pa se dodaju sledeći uslovi:

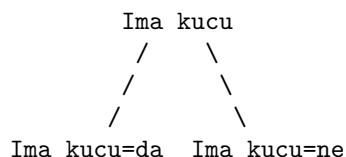
1. Moguće je da neko dete u koraku 2. bude prazno, tj. ne postoji slog koji mu odgovara. U tom slučaju se taj cvor deklarise klasnom oznakom koju ima najveći broj slogova roditeljskog cvora.
2. U koraku 2. ako svi slogovi odgovaraju D_t imaju identične atribute (osim klasne oznake), nije moguće dalje ih razdvojiti. I u ovom slučaju taj cvor se postavlja za list, a njemu odgovara klasna oznaka koju ima najveći broj slova tog cvora.

Problemi pri indukovanju drveća odlučivanja

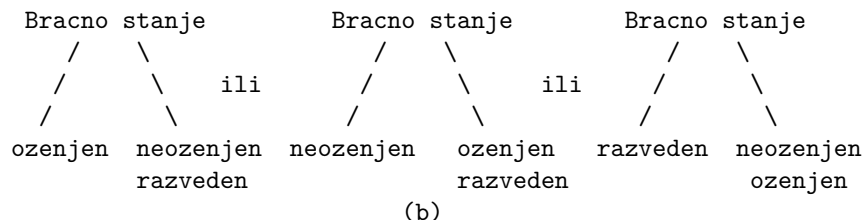
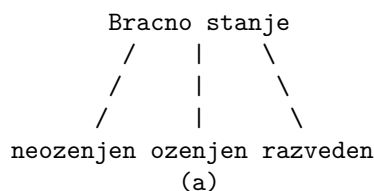
1. **Kako podeliti slogove iz skupa za treniranje?** Svaki rekuzivni korak u rastu drveća mora odabrati uslov testiranja atributa da podelu slogova u manje podskupove. Mora se implementirati algoritam za specifikaciju uslova testiranja atributa, kao i mera za računanje koliko je svaki od uslova testiranja atributa dobar.
2. **Kako zaustaviti proceduru deljenja?** Uslov zaustavljanja je potreban da bi se zaustavio rast drveća. Jedna od strategija je siriti cvor sve dok svi slogovi cvora ne pripadaju istoj klasi ili svi slogovi imaju identične vrednosti atributa. Postoji i takozvana prevremena terminacija.

Metodi za izražavanje uslova testiranja atributa

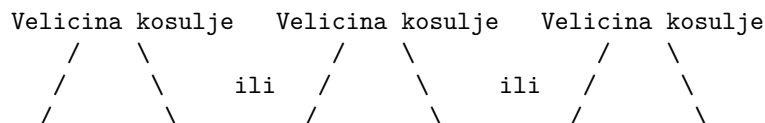
Binarni atributi. Uslov testiranja za binarne atribute generise dva potencijalna rešenja.



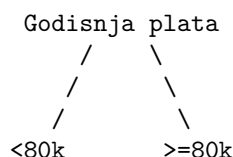
Nominalni (Imenski) atributi. (a) Višestruko razdvajanje podrazumeva da broj rešenja zavisi od broja različitih vrednosti za odgovarajući nominalni (imenski) atribut. (b) Binarno razdvajanje podrazumeva $2^{k-1} - 1$ načina da se naprave binarne particije od k vrednosti atributa.



Ordinalni (Redni) atributi. Takođe, pružaju binarno ili višestruko razdvajanje. Vrednosti se grupisu tako da čuvaju uredjenje. Razdvajanja koja čuvaju uredjenje su (a) i (b), a razdvajanje koje ne čuva uredjenje je (c).



Neprekidni atributi. Uslov testiranja moze biti kompozicija testa ($A < v$) ili ($A \geq v$) sa binarnim rezultatima, ili kompozicija testova $(v_i \leq A < v_{i+1}), i = 1, \dots, k$.



Mere za odabiri najboljeng deljenja se definisu u terminima klasne distribucije slogova pre i posle deljenja.

Neka je $p(i|t)$ je frakcija slogova koja pripada klasi i za dati cvor t . Za dvoklasne probleme, klasna distribucija bilo kog cvora je (p_0, p_1) , gde $p_1 = 1 - p_0$. Pre deljenja klasna distribucija je $(0.5, 0.5)$, pri deljenju zelimo da klasna dristribucija bude sa *nula necistoca*, tj. $(0, 1)$. Primeri mera necistoca su:

$$\text{Entropy}(t) = - \sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t)$$

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2$$

$$\text{Classification error}(t) = 1 - \max_i [p(i|t)]$$

gde je c broj klasa i $0 \log_2 0 = 0$. Maksimalne vrednosti mera necistoca se dobijaju kada je klasna distribucija oblika $(0.5, 0.5)$, dok je 0 za $p = 0$ ($p = 1$).

Cvor N_1	Broj
Klasa=0	0
Klasa=1	6

$$\text{Gini} = 1 - (0/6)^2 - (6/6)^2 = 0$$

$$\text{Entropy} = -(0/6) \log_2(0/6) - (6/6) \log_2(6/6) = 0$$

$$\text{Error} = 1 - \max[0/6, 6/6] = 0$$

Cvor N_1	Broj
Klasa=0	1
Klasa=1	5

$$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$$

$$\text{Entropy} = -(1/6) \log_2(1/6) - (5/6) \log_2(5/6) = 0.650$$

$$\text{Error} = 1 - \max[1/6, 5/6] = 0.167$$

Cvor N_1	Broj
Klasa=0	3
Klasa=1	3

$Gini = 1 - (3/6)^2 - (3/6)^2 = 0.5$
 $Entropy = -(3/6) \log_2(3/6) - (3/6) \log_2(3/6) = 1$
 $Error = 1 - \max[3/6, 3/6] = 0.5$

Da bi odredili kolike su performanse uslova testiranja, moramo da uporedimo stepen necistoce roditeljskog cvore pre rezdvajanja sa stepenom necistoce deteta nakon razdvajanja. Sto je veca njihova razlika uslov testiranja je bolji:

$$\Delta = I(\text{parent}) - \sum_{j=1}^k \frac{N(v_j)}{N} I(v_j)$$

gde je $I(\cdot)$ mera necistoce za dati cvor, N broj slogova u roditeljskom cvoru, k je broj atributa, i $N(v_j)$ broj slogova koji odgovaraju detetu, v_j . Algoritmi indukovanja drveća odlucivanja pokusavaju da maksimiziraju vrednost Δ , tj. ekvivalentno da minimizuju tezinsku sredinu mere necistoce deteta. Kada je $I = Entropy$ onda se Δ_{info} naziva **informaciona dobit**.

Razdvajanje binarnih podataka

Roditelj	
C0	6
C1	6
Gini=0.500	

A	N1	N2
C0	4	2
C1	3	3
Gini=0.486		

B	N1	N2
C0	1	5
C1	4	2
Gini=0.375		

Tezinska sredina za Gini indeks nakon deljenja atributom A je 0.486, dok je 0.375 nakon deljenja atributom B . Kako deljenjem atributom B dobijamo manji Gini indeks on se preferira u odnosu na atribut A .

Razdvajanje nominalnih (imenskih) atributa

Kod nominalnih (imenskih) atributa sa binarnih razdvajanje Gini indeks se racuna isto kao i kod binarnih atributa, dok za visestruko razdvajanje racunamo Gini indeks za svaku vrednost atributa (dete), pa je ukupni Gini indeks tezinska sredina pojeđinacnih Gini indeksa. Gini indeks je manji za visestruko razdvajanje.

Razdvajanje neprekidnih atributa

Treba odrediti mesto razdvajanja v , koje ce podeliti slogove na one za koje vazi $\text{atname} \leq v$ i $\text{atname} > v$. Jedan nacin da se odredi v jeste da se svaka vrednost atributa od N slogova razmatra kao potencijalni v , i za svaku od njih da se izracuna Gini indeks, te da se za v uzme ona vrednost sa najmanjim Gini indeksom. Slozenost ovog pristupa je $O(N^2)$. Drugi nacin da se odredi v , jeste da se prvo sortiraju vrednosti atributa slogova. To ce smanjiti slozenost racunanja Gini indeksa za pojedinačne attribute, jer se moze koristiti info o Gini indeksu prethodnog razdvajanja v . Slozenost ovog pristupa je $O(N \log N)$ za sortiranje i $O(N)$ za racunanje najmanjeg Gini indeksa, pa je ukupna slozenost $O(N \log N)$.

Odnos dobitka

Mere necistoce kao sto je entropija i Gini indeks favorizuju attribute koji imaju veliki broj razlicitih vrednosti. Na primer, **Tip automobila** se favorizuje u odnosu na **Pol**, ili jos gore **ID** se favorizuje u odnosu na **Tip automobila**. Ali **ID** je jedinstveni tako da se ne moze koristiti u predvidjanju.

Postoje dve strategije da se ovo resi. Prva strategija je restrikcija uslova testiranja na samo binarno razdvajanje. Druga strategija je modifikovanje kriterijuma za razdvajanje tako da uzme u racun broj rezultata koje uslov testiranja atributa proizvodi. Na primer, **odnos dobiti** se koristi za odredjivanje koliko je neko razdvajanje dobro:

$$\text{Gain ration} = \frac{\Delta_{\text{info}}}{\text{Split Info}}$$

Ovde je $\text{Split Info} = -\sum_{i=1}^k P(v_i) \log_2 P(v_i)$ i k je ukupan broj razdvajanja. Na primer, ako se svaka vrednost atributa pojavljuje isti broj puta u slogovima, onda $\forall i : O(v_i) = 1/k$, te je onda $\text{Split Info} = \log_2 k$. Ovaj primer pokazuje da ako atribut ima veliki broj razdvajanja, njegova informacija razdvajanja bice velika, sto smanjuje odnos dobitka.

Algoritam indukovanja drveta odlucivanja

```
def tree_growth(E, F):  
  
    if stopping_cond(E, F):  
        leaf = create_node()  
        leaf.label = classify(E)  
        return leaf  
  
    root = create_node()  
    root.test_cond = find_best_split(E, F)  
    # V sadrzi sva moguca vrednosti koja mogu  
    # biti resenja uslova testiranja  
    V = [v for v in root.test_cond.res]  
    for v in V:  
        # E_v sadrzi sve slogove ciji je rezultat  
        # uslova testiranja dati v  
        E_v = [e for e in E if root.test_cond(e) == v]  
        child = tree_growth(E_v, F)  
        root.children[v] = child  
  
    return root
```

Nakon indukovanje drveta odlucivanja, mozemo da izvorsimo **potkresivanje drveta** da bi smanjili njegovu velicinu. Drveta odlucivanja koja su veoma velika su podložna fenomenu koji se naziva **preprilagodjavanje**. Takodje potreksivanje drveta odlucivanja pomaze u generalizaciji te ce i sama klasifikacija biti bolja.

Karakteristike indukovanja drveta odlucivanja

1. Indukovanje drveta odlucivanje ne korisit ni jedan parametar za kreiranje klasifikacionog modela.
2. Nalazjenje optimalnog drveta odlucivanje je NP-kompletan problem. Zbog toga se za indukovanje drveta odlucivanja koriste neke heuristicke metode.
3. Tehnike za indukovanje drveta odlucivanja su racunski jeftine cak i na velikim skupovima za treniranje. Stavise, jednom kada se drvo odlucivanja napravi klasifikovanje sloga je ekstremno brzo, cija je slozenost $O(w)$ gde je w dubina drveta odlucivanja.
4. Mala drveta odlucivanje se lako interpretisu. Takodje drveta odlucivanje se dobro nose sa drugim tehnikama kalsifikacije.
5. Drveta odlucivanja pružaju ekspresivni reprezentaciju za učenje diskretnih funkcija.

6. Drveta odlucivanja dobro podnesu sum, pogotovo kada se koriste metodi protiv preprilagodjavanja.
7. Prisustvo jako povezanih atributa ne remeti tacnost drveta odlucivanja. Ali ako skup za treniranje sadrzi mnogo atributa koji nisu kornisni za klasifikaciju, onda moze doci do toga da se oni izaberu pri razdvajanju pa se time drvo nepotrebno povecava. Postoje metode za izbacivanje irelevantnih atributa u preprocesiranju.
8. Algoritmi drveta odlucivanja particionisu podatke, te sa dubinom drveta imamo sve manje i manje podataka. Zbog toga se gubi na generalizaciji i ovaj problem se zove **fragmentacija podataka**. Jedno od resenja jeste postavljanje odredjenje granice ispod koje podaci ne mogu biti particionisani.
9. Moguce je dobiti drvo odlucivanja koje ima ekvivalentna pod drveta, sto drvo odlucivanja cini kompleksnijim nego sto jeste.
10. Uslovi testiranja atributa se odnose samo na jedan atribut, pa zbog toga imamo granice izmedju dva komsijska regiona drugih klasa. Te granice se nazivaju **granice odluke**. Ove granice se prostiru paralelno sa kordinatnim osama pa probleme gde granice trebaju da prime neki linearni oblik drvo odlucivanja tesko resava. **Zakrivljeno drvo odlucivanja** se koristi da bi se uskratilo ove limitacije jer dopusta da se za uslov testiranja atributa koriste vise od jednog atributa. Ovaj nacin je racunski dosta skuplji od klasicnog indukovanja drveta odlucivanja. **Konstruktivna indukcija** pruza jos jedan nacin particionisanja podataka u homogene, nepravougaone regione. Ovaj pristup kreira nove attribute koji predstavljaju aritmeticku ili logicku kombinaciju postojanijh atributa. Ovo je racunski jeftinije kako ne moramo dinamički da trazimo grupu atributa koji mogu biti relevantni vec njihove kombinacije sracunamo pre samog indukovanja drveta. Mana ovog pristupa je to sto moze da kreira attribute koji su veoma povezani.
11. Izabir mere necistoce ima vrlo mali efekat na performanse drveta odlucivanja.

Preprilagodjavanje modela

Greske u klasifikacionom modelu se dele na dva tipa: **greske treniranja** i **greske generalizacije**. Greska treniranja, ili **greska resubstitucije**, ili **ocigledna greska**, je broj promaseno klasifikovanih slogova za treniranje. Greska generalizacije je ocekiwana greska modela za prethodno ne vidjene slogove.

Dobar klasifikacioni model, pored male greske treniranja, mora da ima i malu gresku generalizacije. Model koji odgovara previse skupu za treniranje moze da ima veliku gresku generalizacije, za takav model kazemo da je **preprilagodjen**.

Primer (Dvodimenzionalni podaci). Neka je dat dvodimenzionalni skup podataka, gde svaki slog pripada ili klasi o ili klasi x . Za ovakav skup podataka zelimo da napravimo klasifikacioni model koriscenjem drveta odlucivanja. Model se pravi po broju cvorova. Pokazuje se da sa brojem cvorova u modelu greska treniranja opada, dok greska testiranja opada do nekog trenutka od kog pocinje da raste. Za mali broj cvorova obe greske su velike te za model kazemo da je **podprilagodjen**. Od trenutka kada greska treniranja opada, a greska testiranja raste kazemo da je model **preprilagodjen**.

Za razumevanje ovog fenomena, primetimo da se greska treniranja smanjuje kada se povecava kompleksnost modela. Na primer, listovi drveta rastu sve dok im skup treniranje ne odgovara perfektno. Ovime dobijamo da je greska testiranja 0, ali u isto vreme kompleksnost ovog modela raste te se gubi na generalizaciji.

Preprilagodjavanje zbog prisustva suma

Neka je dat skup za treniranje i testiranje za klasifikacioni problem sisara.

Skup podataka za treniranje, koji ima dva objekta koja su pogresno klasifikovana i ona su oznacena sa *:

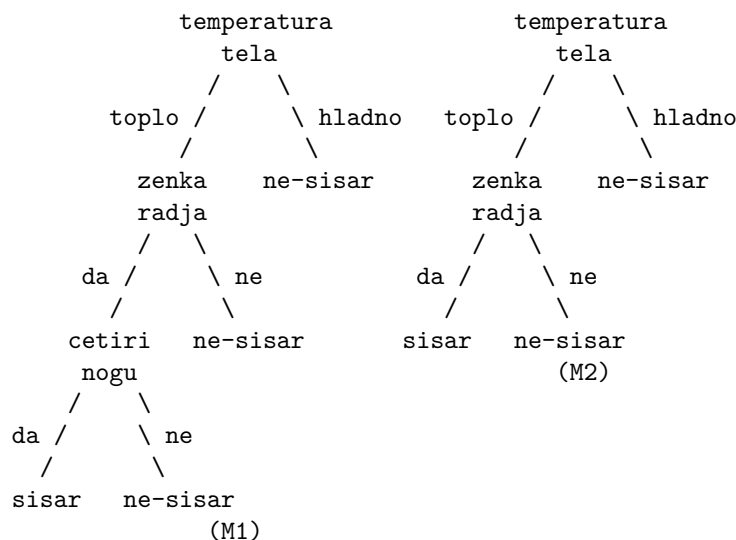
Ime	Temperatura tela	Zenka radja	Cetvoronozna	Hibernira	Klasa
bodljikavo prase	toplo-krvni	da	da	da	da
macka	toplo-krvni	da	da	ne	da
slepi mis	toplo-krvni	da	ne	da	ne*
kit	toplo-krvni	da	ne	ne	ne*
komodo zmaj	hladno-krvni	ne	da	ne	ne

Ime	Temperatura tela	Zenka radja	Cetvoronožna	Hibernira	Klasa
salamander	hladno-krvni	ne	da	da	ne
piton	hladno-krvni	ne	ne	da	ne
losos	hladno-krvni	ne	ne	ne	ne
orao	toplo-krvni	ne	ne	ne	ne
gupi	hladno-krvni	da	ne	ne	ne

Skup podataka za testiranje:

Ime	Temperatura tela	Zenka radja	Cetvoronožna	Hibernira	Klasa
covek	toplo-krvni	da	ne	ne	da
papagaj	toplo-krvni	ne	ne	ne	ne
slon	toplo-krvni	da	da	ne	da
ajkula	hladno-krvni	da	ne	ne	ne
kornjaca	hladno-krvni	ne	da	ne	ne
pingvin	hladno-krvni	ne	ne	ne	ne
jegulja	hladno-krvni	ne	ne	ne	ne
delfin	toplo-krvni	da	ne	ne	da
jez	toplo-krvni	ne	da	da	da
gila monstrum	hladno-krvni	ne	da	da	ne

Razmotrimo sledeca dva modela klasifikacije za problem klasifikacije sisara:



Model M1 savršeno odgovara skupu podataka za treniranje, te nema gresku treniranja. Sa druge strane greska testiranja je 30%: Covek i delfin su pogresno klasifikovani kako jesu sisari ali nisu cetvoronožni, dok je jez objekat koji se izuzetak u klasnoj tabeli. Greske pri izuzecima su neizbezne i one postavljaju donju granicu greske bilo kog klasifikatora.

Model M2 ima gresku treniranja 10%, dok je greska testiranja nesto veca 20%. Jasno je da je prvi model M1 prilagodjen za dati skup treniranja.

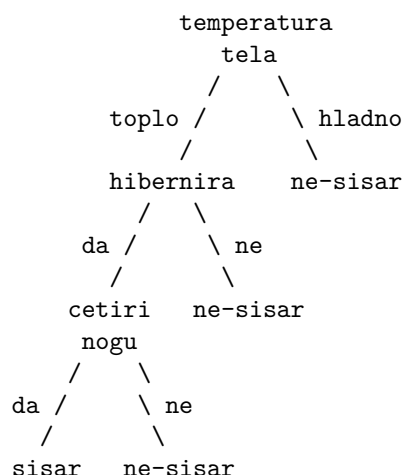
Preprilagodjavanje zbog nedostatka reprezentativnih uzoraka

Modeli klasifikacije koji se treniraju na malo broju slgova su takodje podložni preprilagodjavanju. Ovi se ne mogu generalizovati zbog nedostatka reprezentativnih uzoraka.

Neka je dat sledeci skup podataka za treniranje:

Ime	Temperatura tela	Zenka radja	Cetvoronožna	Hibernira	Klasa
salamander	hladno-krvni	ne	da	da	ne
gupi	hladno-krvni	da	ne	ne	ne
orao	toplo-krvni	ne	ne	ne	ne
golub	toplo-krvni	ne	ne	da	ne
kljunar	toplo-krvni	ne	da	da	da

Treniranjem dobijamo sledeci model:



Greska treniranja ovog modela je nula, dok ge greska testiranja 30%: Covek, slon, i delfin su pogresno klasifikovani kako ovaj model klasifikuje zivotinje koje su toplo-krvene i ne hiberniraju kao ne-sisare. Jasno je da dobijamo pogresna predvidjanja kada nemamo reprezentativne slogove.

Preprilagodjavanje i procedura visestrukog poredjenja

Preprilagodjavanje modela moze nastati u algoritmima ucenja koji koriste proceduru visestrukog poredjenja. Razmotrimo predvidjanje da li ce nekretnine na berzi rasti ili padati u sledecih 10 dana. Ako predvidjanje vrsimo nasumicno, verovatnoca da ce predvidjanje negog dana biti tacno je 0.5. Ali verovatnoca da ce predvidjanje bar 8 od 10 puta biti tacno je

$$\frac{\binom{10}{8} + \binom{10}{9} + \binom{10}{10}}{2^{10}} = 0.0547$$

Zbog toga angazujemo nekog analizatora koji ce predvideti najvise tacnih u sledecih 10 dana. Ako svi analizatori koriste nasumicno predvidjanje, verovatnoca da je bar jedan on njih imao 8 tacnih predvidjanje je

$$1 - (1 - 0.0547)^{50} = 0.9399$$

Ako znamo da ce jedan analizator tesno predvideti tacno, kada ih spojimo zajedno oni sigurno uspevaju da nadju tacno predvidjanje nasumicnim pokusavanjem.

Kako se procedura visestrukog poredjanje odnosi na preprilagodjavanje modela? Mnogi algoritmi ucenja istrazuju skup nezavisnih alternativa, $\{\gamma_i\}$, i onda biraju onu koja maksimizuje dati kriterijum γ_{\max} . Algoritam ce onda dodati γ_{\max} i trenutni model da bi poboljsao njegove performanse. Ova procedura se nastavlja sve dok se sledece poboljsanje ne primeti. Na primer, indukovanje drveta odlucivanja koristi vise testova da odredi koji atributi ce dati najbolje razdvajanje skupa treniranja. Oni koji najbolje razdvajaju attribute se dalje biraju te prosiruju drvo sve dok se ne primeti poboljsanje koje je statisticki znacajno.

Neka je T_0 inicijalno drvo odlucivanja i T_x novo drvo nakon dodavanja unutrasnjeg cvora za atribut x . x se moze dodati u drvo ako je primecena dobit, $\Delta(T_0, T_x)$, veci od nekog predefinisnog ogranicenja α . Ako postoji samo jedan uslov testiranja atributa koji moze da se primeni, onda mozemo izbeci ubacivanje cvora, tako sto odaberemo dovoljno veliko α . Ali, obicno je vise od jednog uslov testiranja atributa dostupno i algoritam indukovanja drveta odlucivanja mora da odabere najbolji atribut x_{\max} iz skupa kandidata, $\{x_1, x_2, \dots, x_k\}$, za partitionisanje podataka. U ovom trenutku algoritam koristi proceduru visestrukog poredjenja da odluci da li drvo odlucivanja treba biti prosireno. Tacnije, testira se $\Delta(T_0, T_{x_{\max}}) > \alpha$ umesto $\Delta(T_0, T_x) > \alpha$. Ako broj alternativa, k raste, tako raste i sansa da nadjemo $\Delta(T_0, T_{x_{\max}}) > \alpha$. Osim ako se funkcija Δ ili ogranicenje α ne modifikuju taok da uracunaju broj alternativa k , algoritam moze neadekvatno dodavati superiorne cvorove u model sto dovodi do prilagodjenja.

Ovaj efekat je izrazeniji kada je broj slogova za treniranje mali, jer ce disperzija $\Delta(T_0, T_{x_{\max}})$ biti velika kada ima manje slogova mogucih za treniranje. Kao rezultat verovatnoca da nadjemo $\Delta(T_0, T_{x_{\max}}) > \alpha$ raste sa manjim brojem slogova treniranja. Ovo se desava kada drvo odlucivanja raste u dubinu, sto smanjuje slogove koje pokrivaju cvorovi i povecava sansu dodavanja nepotrebnih cvorova u drvo.

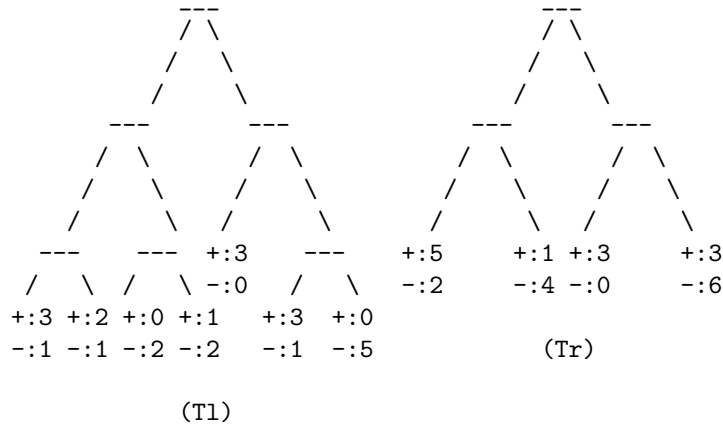
Procenjivanje greske generalizacije

Kompleksnost modela ima uticaj na prilagodjavanje modela. Postavlja se pitanje kako odrediti pravu kompleksnost modela? Idealna kompleksnost je onda koja proizvodi najmanju gresku generalizacije.

Procenjivanje resupstitucijom

Pristup resupstitucijom podrazumeva da je skup za treniranje reprezentativan. Onda greska treniranje, ili greska resupstitucije se moze istoristi za optimalno procenjivanje greske generalizacije. Ali greska treniranja je obicno losa procena greske generalizacije.

Primer: Greska drveta odlucivanja T_l je $e(T_l) = 4/24 = 0.167$, dok je greska drveta odlucivanja T_r je $e(T_r) = 6/24 = 0.25$. Zbog toga na osnovu procenjivanja resupstitucije, levo drvo je bolje od desnog drveta.



Inkorporiranje kompleksnosti modela

Definija: (Okamov brijac) Za dva modela sa istom greskom generalizacije, jednostavniji model se preferira u odnosu na kompleksniji model.

Pesimisticna procena greske Eksplicitno se racuna greska generalizacije kao suma greske treniranje i dodatnog kaznena vrednost kompleksnosti modela. Neka je $n(t)$ broj slogova treniranja klasifikacijom cvora t i $e(t)$ broj pogresno klasifikovanih slogova. Pesimisticka procena greske drveta odlucivanja T je:

$$e_g(T) = \frac{\sum_{i=1}^k [e(t_i) + \Omega(t_i)]}{\sum_{i=1}^k n(t_i)} = \frac{e(T) + \Omega(T)}{N_t}$$

gde je k broj listova, $e(T)$ greska treniranja drveta odlucivanja T , N_t broj slogova za treniranja, i $\Omega(t_i)$ kaznena vrednost za svaki cvor t_i .

Primer Za primer od malopre i $\Omega(t_i) = 0.5$ vazi sledece:

$$e_g(T_l) = \frac{4 + 7 \times 0.5}{24} = \frac{7.5}{24} = 0.3125$$

$$e_g(T_r) = \frac{6 + 7 \times 0.5}{24} = \frac{8}{24} = 0.3333$$

Pa levo drvo ima bolju pesimisticku gresku od desnog drveteta. Za binarna drveta, kaznena vrednost od 0.5 znaci da cvor treba uvek biti prosiren u dva deteta svo dok se klasifikacija poboljsava za bar po jedan slog. Za $\Omega(t_i) = 1\$$ imamo da je $e_g(T_l) = 11/24 = 0.458$, dok je $e_g(T_r) = 10/24 = 0.417$. U ovom slucaju bolju pesimisticku gresku ima levo drvo. Pa cvor se ne sme sirti u decu sem ukoliko to smanjuje pogresne klasifikacije za vise od jednog sloga.

Princip minimalno opisane duzine. Razmotrimo primer gde su A i B dati skupovi slogova sa poznatim vrednostima atributa \mathbf{x} . Dodatno, za skup A znamo tacno svaku klasnu oznaku za svaki slog, dok za skup B ne znamo ni jednu klasnu oznaku. B moze da klasifikuje svaki slog tako sto zatrazi od A da mu posalje svaku klasnu oznaku sekvencijalno. Ova poruka zahteva $\Theta(n)$ bitova informacija, gde je n ukupan broj slogova.

Alternativno, A moze da napravi klasifikacioni model veza izmedju \mathbf{x} i y . Model se moze enkodirati u kompaktnu formu pre nego sto se posalje u B. Ako je model 100 tacan, tada ce cena prebacivanja biti ekvivalentna ceni enkodiranja modela. U suprotnom, A mora prebaciti informaciju o slogu koji je klasifikovan pogresno sa tim modelom. Pa je ukupna cena prebacivanja

$$Cost(model, data) = Cost(model) + Cost(data|model)$$

Trazimo model koji minimizuje ukupni cenu.

Procenjivanje statistickih granica

Kako je greska generalizacije tipicno veca od greske treniranja, statisticka korekcija se obicno racuna kao gornja granica greske treniranja, koja uzima broj slogova treniranja koji dostignu odredjeni list.

Primer Posmatramo primer od ranije, i primetimo da se najlevlji list drveteta T_r prosiruje u dva deteta u drvetu T_l . Pre razdvajanja greska cvora je $2/7 = 0.286$. Aproksimiranje binomne distribucije sa normalnom, gornja granica greske e je:

$$e_{upper}(N, e, \alpha) = \frac{e + \frac{z_{\alpha/2}^2}{2N} + z_{\alpha/2} \sqrt{\frac{e(1-e)}{N} + \frac{z_{\alpha/2}^2}{4N^2}}}{1 + \frac{z_{\alpha/2}^2}{N}}$$

gde je α nivo samopouzdanja, $z_{\alpha/2}$ standardizovana vrednost standardne normalne distribucije, i N je ukupan broj slogova za treniranje koji se koristi da se izracuna e . Za $\alpha = 25$, imamo $e_{upper} = 0.503$ pa imamo $7 \times 0.503 = 3.521$ gresaka. Ako prosirimo cvor u decu cvorove, greske treniranja dece su $1/4 = 0.250$ i $1/3 = 0.333$, respektivno. Gornje granice ovih gresaka su $e_{upper} = 0.537$ i $e_{upper} = 0.650$, respektivno. Pa je ukupna greska deteta cvorova $4 \times 0.537 + 3 \times 0.650 = 4.098$, sto je veca nego procenjena greska odgovarajuceg cvora u T_r .

Koriscenje skupa validacija

U ovom pristupu, umesto koriscenja skupa za treniranje pri odredjivanje greske generalizacije, originalni skup za treniranje se deli u dva manja podskupa. Jedan se koristi za treniranje, dok se drugi koristi za procenu greske generalizacije. Drugi skup se jos i naziva skup validacija. Tipicno se jedna trecina skupa koristi za skup validacija. Kompleksnost modela se moze odrediti na osnovu parametara algoritama ucenja, pa tako u zavisnosti od greske skupa validacija mozemo menjati te parametre kako bi dobili najmanju gresku na tom skupu (na primer, potkresivanje drveteta odlucivanja).

Preprilagodjavanje u indukovanje drveteta odlucivanja

Predpotkresivanje (Pravilo ranog zaustavljanja)

Algoritam rasta drveta odlucivanja se zaustavlja pre nego sto drvo potpuno poraste i savrseno odgovara celokupnom skupu podataka za treniranje. U ovom pristupu se koristi stroziji uslov zaustavljanja, kao na primer, prestani da siris listove kada se primeti da rast u meri necistoce padne ispod odredjene linije. Prednosti ovog pristupa je to sto izbegavamo generisanje kompleksnog poddrveta koje moze da bude preprilagodjeno. Ali tesko je odrediti pravu granicu zaustavljanja. Prevelika granica moze razultovati u neprilagodjenom modelu, dok mala moze biti nedovoljno da prebrodi problem preprilagodjavanja. Stavise, iako trenutno sirenje mozda nema uticaja, mozda ce neko sirenje u njegovom poddrvetu imati ogroman uticaj na rezultat.

Postpotkresivanje

Inicijalno drvo odlucivanja raste u potpunosti, nakon cega sledi proces potresivanje, koje odseca drvo odozdo nagore. Odsecanjem zamenjujemo poddrvo sa: 1. Novim listom cija klasna oznaka odgovara vecini slogova poddrveta. 2. Najcesce korisrenom granom poddrveta. Ovaj postupak se zaustavlja kada nema poboljsanja. Obicno postpotkresivanje daje bolje rezultate, kako odlucuje nad potpuno izraslim drvetom, ali zbog toga je racunski skuplje.

Racunanje performanse klasifikatora

Procenjivanje greske pruza algoritmu ucenja da odradi **odabir modela**, tj. da nadje model koji je odgovarajuce kompleksnosti tako da ne bude podlozan preprilagodjavanju.

Cesto je bitno izmeriti performanse modela na skup za testiranje kako mera pruza nepristransu procenu greske generalizacije. Tacnost ili greska izracunata nad skupom za testiranje moze se uporedjivati sa relativnim performansama drugih klasifikatora istog domena. Medjutim, klasne oznake slogova za testiranje moraju biti poznata.

Metod zadrzavanja

Originalni podaci sa oznacenim klasana su podeljeni na dva disjunktna skupa, nazvana skup za treniranje i skup za testiranje. Klasifikacioni model se onda indukuje iz skupa za treniranje i njegove performanse se racunaju nad skupom za testiranje. Proporcija podataka za treniranje u odnosu na podatke za testiranje je je sklona ka podacima za treniranje, tj. na primer 50% : 50% ili 70% : 30%. Tacnost klasifikatora se moze dobiti od tacnosti indukovanog modela nad skupom za testiranje.

Ovaj metod ima nekoliko ogranicenja. (1) Imamo manje oznacenih primerza za treniranje jer neki slogovi za cuvaju za testiranje. (2) Model moze biti veoma zavisna na strukturu skupova za treniranje i testiranje. Sto je manji skup za treniranje, veka je disperzija modela. Sa druge strane, ako je skup za treniranje velik, onda je procena tacnosti izracunata od manjeg skupa za testiranje manje relevantna. Za takva procenu tacnosti se kaze da ima siroki interval samopouzdanje. (3) Skup za treniranje i testiranje nisu vise nezavisni. Klasa koja je previse zastupna u jednom skupu bice manje zastupna u drugom skupu, i obrnuto.

Nasumicno uzorkovanje

Ako ponavljamo metod zadrzavanja nekoliko puta time povoljsavamo procenu performanse klasifikatora. Ovaj pristup se zove nasumicno uzorkovanje. Neka je acc_i tacnost modela u i -toj iteraciji. Ukupna tacnost je data sa $acc = \sum_{i=1}^k acc_i / k$. Nasumicno uzorkovanje, takodje, ima svoje probleme, kao sto je ne iskoriscavanje svih podataka za treniranje. Isto tako nema ni kontrolu nad brojem koriscenja nekog sloga u testiranje ili treniranju, zbog toga neki slogovi mogu biti korisceni vise u treniranju od drugih.

Unakrsna-Validacija

Za razliku od nasumicnog uzorkovanje, unakrsna-validacija podrazumeva da je svaki slog koriscen isti broj puta za treniranje i tacno jednom za testiranje.

Pretpostavimo da su podaci podeljeni u dva jednaka podskupa. (1) Odaberemo jedan podskup za treniranje i drugi za testiranje. (2) Onda zamenimo ulogu podskupova za treniranje i testiranje. Ovaj postupak se naziva dvostruko preklapanje unakrsne-validacije. Ukupna greska se dobija sumiranje sresaka u oba slucaja. Svaki slog se koristi tacno jednom za treniranje i tacno jednom za testiranje.

k -tostruko preklapanje unakresne-validacije generalizije ovaj pristup tako sto particionise podatke u k jednakih particija. Tokom svakog pokretanja, jedna particija se bira za testiranje, dok se ostale koriste za treniranje. Ova procedura se ponavlja k puta tako da se svaka particija iskoristi tacno jednom. Ukupna greska se dobija sumiranjem gresaka za svaki od k pokretanja.

Specijalni slucaj k -tostrukog preklapanje unakrsne-validacije je za $k = N$, gde je N ukupan broj slogova. U ovom slucaju svaki skup za testiranje sadrzi samo jedan slog (**izbaci jedan**). Ovaj pristup ima za to da iskoristi sto vise podataka za bolje treniranje. Takodje, skupovi za testiranje se iskljucuju i njihova efikasnost pokriva celokupan skup podataka. Mali nedostatak ovog pristupa je to sto je racunski skupo, jer se procedura ponavlja N puta.

Bootstrap

Ovaj metod omogucava da se slogovi dupliraju tako da se nalaze i u skupu za treniranje i testiranje. Bootstrap metod uzima slogove za treniranje *sa vracanjem*. Verovatnoca da se odabere neki slog bootstrap metodom je $1 - (1 - 1/N)^N$. Kada $N \rightarrow \infty$, onda se ova verovatnoca ponasa kao $1 - e^{-1} = 0.632$, pa iz toga imamo da ce skup zadržati oko 63.2 slogova iz originalnog skupa. Tacnost indukovanih modela dobijenog koriscenjem bootstrap tehnike je ϵ_i . Ova procedura moze da ima b ponavljanja.

Postoje nekoliko mogucnosti za racunanje ukupne tacnosti. Jedna od najpopularnijih je **.632 bootstrap**, koja racuna ukupnu tacnost modela kao:

$$acc_{boot} = \frac{1}{b} \sum_{i=1}^b (0.632 \times \epsilon_i + 0.368 \times acc_s)$$

gde je acc_s tacnost izracunata iz skupa za treniranje koji sadrzi sve oznacene slogove iz originalnih podataka.

Metodi za uporedjivanje klasifikatora

Cesto je korisno uporediti performanse drugacijih klasifikatora, da bi odredilo koji klasifikator bolje odgovara datom skupu podataka.

Neka je dat par klasifikacionih modela M_A i M_B . Pretpostavimo da M_A ima 85% tacnosti kada se primeni na skup za testiranje koji sadrzi 30 slogova, dok M_B dostize 75% tacnosti na drugaciji skup za testiranje koji sadrzi 5000 slogova. Da li je onda model M_A bolji od modela M_B ?

Ovime dolazimo do dva kljucna pitanje:

1. Iako model M_A ima vecu tacnost od modela M_B , on je testiran na manjem skupu za treniranje. Koliko poverenja mozemo da imamo na tacnost modela M_A ? Ovo pitanje se odnosi na problem procene intervala poverenja tacnosti za dati model.
2. Da li je moguće objasniti razlike u tacnosti kao rezultat varijacija u skupovima za testiranje? Ovo pitanje se odnosi na problem statisticke znacajnosti skupa za testiranje.

Procenjivanje intervala poverenja za tacnost modela

Da bi odredili interval poverenja, moramo odrediti raspodelu mere tacnosti. Neka:

1. Eksperiment sadrzi N nezavisnih pokusaja, gde svaki pokusaj ima dve mogucnosti: *uspeh* ili *neuspeh*.
2. Verovatnoca uspeha je p , za svaki pokusaj.

Ako je X broj uspeha od N pokusaja, onda je X data **Binomnom raspodelom**, cije je ocekivanje Np , disperzija $Np(1 - p)$, i vazi:

$$P(X = k) = \binom{N}{k} p^k (1 - p)^{N-k}$$

Zadatak predvidjanja klasne oznake sloga iz skupa za testiranje moze se posmatrati kao binomni eksperiment. Za dati skup za testiranje koji ima N slogova, neka je X broj slogova koje model predvidja kao tacne, i neka

je p prava tacnost modela. U ovom slucaju X ima binomnu raspodelu, pa i $acc = X/N$ takodje ima binomnu raspodelu sa ocekivanjem p i disperzijom $p(1-p)/N$. Iz toga mozemo odrediti interval poverenja kao:

$$P\left(-Z_{\alpha/2} \leq \frac{acc - p}{\sqrt{(p(1-p)/N)}} \leq Z_{1-\alpha/2}\right) = 1 - \alpha,$$

gde su $Z_{\alpha/2}$ i $Z_{1-\alpha/2}$ gornja i donja granica standardne normalne distribucija nivoa poverenja $(1 - \alpha)$. Iz prethodne jednacine mozemo dobiti sledecu tabelu:

$(1 - \alpha)$	0.99	0.98	0.95	0.9	0.8	0.7	0.5
$Z_{\alpha/2}$	2.58	2.33	1.96	1.65	1.28	1.04	0.67

Primer: Neka je dat model koji ima tacnost 80% koja je izracunata nad skupom za testiranje od 100 slogova. Koji je interval poverenja za stvarnu tacnost u 95% nivou poverenja? Nivu poverenja od 95% odgovara $Z_{\alpha/2} = 1.96$ iz tabele. Iz ovoga dobijamo da je interval poverenja izmedju 71.1% i 86.7%. Kako broj slogova skupa za testiranja N raste tako interval poverenja postaje sve uzi i uzi.

Racunanje performansi dva modela

Neka je dat par modela M_1 i M_2 , koji su dobijeni od dva nezavisna skupa podataka D_1 i D_2 , respektivno. Neka je n_1 broj slogova u D_1 i n_2 broj slogova u D_2 . Takodje naka je e_1 greska modela M_1 nad D_1 , i e_2 greska modela M_2 nad D_2 . Cilj je odrediti da li je razlika izmedju e_1 i e_2 statisticki znacajna.

Neka su n_1 i n_2 dovoljno veliki, onda se greske e_1 i e_2 mogu aproksimirati normalnom raspodelom. Neka je $d = e_1 - e_2$, onda d ima normalnu raspodelu sa ocekivanjem (pravom razlikom) d_t , i disperzijom $\hat{\sigma}_d^2$. Disperzija i ocekivanje od d mogu se izracunati kao:

$$\hat{\sigma}_d^2 = \frac{e_1(1-e_1)}{n_1} + \frac{e_2(1-e_2)}{n_2}; \quad d_t = d \pm z_{\alpha/2}\hat{\sigma}_d.$$

Primer Posmatrajmo problem opisan na pocetku. Model M_A ima greksu $e_1 = 0.15$ kada se primeni na $N_1 = 30$ test slogova, dok M_B ima gresku $e_2 = 0.25$ kada se primeni na $N_2 = 5000$ slogova. Onda je $d = |0.15 - 0.25| = 0.1$. Dalje imamo da je:

$$\hat{\sigma}_d^2 = \frac{0.15(1-0.15)}{30} + \frac{0.25(1-0.25)}{5000} = 0.0043; \quad d_t = 0.1 \pm 1.96 \times 0.00655 = 0.1 \pm 0.128,$$

za nivo intervala poverenja 95%, te je onda $z_{\alpha/2} = 1.96$.

Uperedjivanje performansi dva klasifikatora

Pretpostavimo da hocemo da uporedimo performanse dva klasifikatora koja koriste k -tostruko preklapanje unakrsne-validacije. Inicijalno skup podataka D se deli u k jednakih particija. Onda svaki od klasifikatora indukuje model nad $k - 1$ particija i testira ga na neiskoriscenoj particiji. Ovaj korak se ponavlja k puta, i svaki put se koristi druga particija kao skup za testiranje.

Neka je M_{ij} model koji je indukovao klasifikacionom tehnikom L_i tokom j -te iteracije. Svaki par M_{1j} i M_{2j} je testiran na istoj particiji j . Neka su e_{1j} i e_{2j} njihove greske, respektivno. Razlika greske tokom j -te iteracije je $d_j = e_{1j} - e_{2j}$. Ako je k dovoljno veliko, onda d_j ima normalnu raspodelu sa ocekivanjem d_t^{cv} , sto je prava razlika njihovih greski, i disperziju σ^{cv} . Celokupna disperzija i ocekivanje mogu da se procene kao:

$$\hat{\sigma}_{d^{cv}}^2 = \frac{\sum_{j=1}^k (d_j - \bar{d})^2}{k(k-1)}; \quad d_t^{cv} = \bar{d} \pm t_{(1-\alpha), k-1} \hat{\sigma}_{d^{cv}}$$

gde je \bar{d} prosečna razlika, i koeficijent $t_{(1-\alpha), k-1}$ je dobijen iz tablice verovatnoce dva parametra, nivoa poverenja $(1 - \alpha)$ i broja stepena slobode $k - 1$.

Razni Algoritmi Zasnovani na Drvetima Odlucivanja

ID3

ID3 (iterative Dichotomiser 3) je algoritam zasnovan na drvetima odlucivanja koji je nastao je 1986. godine. Koristi entropiju kao meru u informacionu dobit kao kriterijum podele. Kriterijum zastavljanja se zasniva na tome da svi podaci u cvoru pripadaju istoj klasi ili kada da je najveća informaciona dobit manja ili jednaka nuli. Ne vrsi se potreksivanje drveta. Ovaj algoritam ne radi sa numerickim atributima i/ili nedostajucim vrednostima. Razmatra samo jedan atribut pri odlucivanju. Rezultat ovog algoritma je drvo odlucivanja koje je binarno.

C4.5

C4.5 predstavlja skup algoritama C4.5, C4.5 bez potkresivanja, i C4.5-pravila, koji su nastali 1993. i predstavljaju prosirenje osnovnog ID3 algoritma. Takodje koriste entropiju kao meru i informacionu dobit kao kriterijum podele. Kriterijum zaustavljanja se zasniva na tome da je broja razlicitih instanci ispod zadate granice ili da svih podaci u cvoru pripadaju istoj klasi. Koristi diskretne attribute, dok numericke attribute prebacuje u diskretne metodom diskretizacije. Za klasu datog lista bira se najbrojnija klasa metodom glasanja. Algoritam vrsi naknadno potresivanje drveta koristeći (1) smanjenje nivoa greske, (2) pesimisticku procenu greske, (3) intervale poverenja. Rezultat ovog algoritma je n-arno drvo odlucivanja, kako ovaj algoritam radi kategorickim atributima.

C4.5 Nedostajuce Vrednosti

Izbor atributa za deobu u cvoru koji sadrzi nedostajucu vrednost koristi informacinu dobit pomnozeno sa procenatom poznatih vrednosti. Ako atribut za podelu sadrzi nedostajucu vrednosti instance sa nedostajucim vrednostima dodeljujemo svakoj grani procentualno prema broju poznatih vrednosti atributa u granama. Novu instancu sa nedostajucim vrednosti klasifikujemo tako sto distribuiramo u grane prema relativnoj verovatnoci dobijenom kombinovanju rezultata simultanog pretrazivanja za sve moguće ishode testiranja. Konačna klasa je klasa sa najvećom verovatnoćom u ukupnoj distribuciji u drvetu.

C4.5 Naslednici

Naslednici su J4.8 (Java) i C5.0. C5.0 karakterise manji skup pravila sa istom preciznoscu, poboljsanje zasnovano na boosting metodama (kombinovanje razlicitih klasifikatora radi povecanja preciznosti). Radi sa tezinama atributa.

CART

CART (Classification And Regression Trees) je algoritam zasnovan na drvetima odlucivanja koji je nastao teorijski 1984. godine. Sadrzi resenja za klasifikacioni problem i za regresivni problem. Podržava različite metode za meru podele: Ginni, twoing, uredjeni twoing, simetricni Ginni, entropija, chisq... Koristi binarnu podelu, tj. instance koje se presludjuju levo imaju ispunjen uslov. Automacki uklanja nebalansirane klase mehanizmom prethodnika — verovatnoca svake ciljne kategorije u materijalu za trening koriste se kao tezine, bez uticaja na krajnju distribuciju klasa. Nedostajuce vrednosti se resavaju tako sto se posmatra atribut koji najbolje podržava atribut po kome se deli.

CART Potkresivanje drveta

Originalna verzija konstruise kompletno drvo, a tek iza toga se ide na potkresivanje. Formira se vise mogucih drveta sa potkresanim granama za svaka dva lista koja imaju zajednickog pretka. Cost complexity pruning znatno smanjuje broj drveta koji se razmatraju. Optimalno drvo je potkresano drvo sa najmanjom cenom za podatke iz skupa za treniranje. Novije verzije rade i pre-potkresivanje.

Odredjivanje vaznosti atributa dobija se kao zbir dobiti svih cvorova u kojima se atribut koristi za deobu pomnozen sa procentom traning podataka u cvoru, gde se atributi sa nedostajucim vrednostima uzimaju u obzir. Takodje koristi matrice cene, i tezine.

CHAID

CHAID (Chi-Squared Automatic Interaction Detector) je razvijen 1980. godine. Koristi samo kategoricke klase, dok ostali atributi mogu biti imenski, redni, kategoricki, neprekidni koji se transformisu u redne.

Sastoji se od tri koraka: uparivanje, podela, i zaustavljanje. Drvo klasifikacije se formira uzastopnom primenom ovih pravila na svaki cvor, pocev od korenog. Za svaki atribut A odredjuje se par vrednosti V_A sa najmanje znacajnom razlikom. Znacajnost razlike se racuna preko p -test, tako sto se racuna p -vrednost u odnosu na atribut klase. Za podelu se bira atribut sa najmanjom p -vrednoscu statistickog testa. Provera se da li je p -vrednost veca od praga α , i ako jeste par se uparuje u jednu slozenu kategoriju i tarzi se sledeci par vrednosti. Ovo se ponavlja sve dok ima znacajnih parova.

Statisticke test zavisi od tipa atributa klase. Ako je to: (1) neprekidan atribut, onda se koristi F -test, (2) nominalan atribut, onda se koristi χ^2 -test, (3) redni atribut, onda se koristi test odnosa verovatnoca.

Kriterijum zaustavljanja je ne postojanje znacajnog par, dostignuta najveća dubina drveta, distignut najmanji broj elemenata u cvoru. Dobija se n -arno drvo odlucivanja.

QUEST

QUEST (Quick, Unbiased, Efficient Statistical Trees) podrzava univarijantne i linearne kombinacije podela vrednosti u cvoru. Vezu izmedju atributa klase i atributa pri podeli je (1) ANOVA F -test za redne i neprekidne attribute i χ^2 -test za kategoricke attribute. Vrsi post-potresivanje drveta. Atribut klase moze biti samo kategoricki, dok su ostali atributi imenski, redni, kategoricki, neprekidni. Vrsi binarnu podelu cvorova.

SLIQ

SLIQ (Supervised Learning In Quest) je varijanta QUEST algoritma za klasifikaciju velikih traning podataka. Efikasno radi sa podacima koji ne mogu da stanu u memoriju racunara. Nije zasnovan na principu Hantovog algoritma. Koristi tehine rasta drveta u sirinu: tehnika presortiranja u fazi rasta drveta. Koristi vertikalni format podataka - na pocetku se svi podaci sortiraju i smestaju u listu. Podela se vrsi na osnovu Ginni indeksa. Koristi listu klasa: struktura u memoriji koja cuva oznake klasa svake instance. Velicina liste klasa je direktno proporcionalna broju ulaznih slogova. Moze da koristi i kaegoricke i numericke aribute. Potresivanje se zasniva na tehnici MDL. Implementacija za serijski i paralelno izvorsavanje

SPRINT

SPRINT (Scalable PaRallelizable INduction of decision Trees) je modifikacija SLIQ algoritma koja uklanja memorijska ogranicanje. Oznake klasa pridruzuju identifikatorima instanci. Podrzava kategoricke i numericke aribute. I implementirana je za serijsko i paralelno izvorsavanje.

Klasifikacije: Alternativne tehnike

Klasifikator zasnovan na pravilima

Klasifikator zasnovan na pravilima koristi kolekciju ‘if..then..’ pravila.

```
r1: (Radja=ne) and (Leti=da) -> Ptica
r2: (Radja=ne) and (Pliva=da) -> Riba
r3: (Radja=da) and (Temperatura=toplo-krvna) -> Sisar
r4: (Radja=ne) and (Leti=ne) -> Gmizavci
r5: (Pliva=semi) -> Vodozemac
```

Pravila se predstavljaju u disjunktnoj normalnoj formi $R = (r_1 \vee r_2 \vee \dots \vee r_k)$, gde se R naziva **skup pravila** i r_i se nazivaju klasifikaciona pravila ili disjunktne. Svako klasifikaciono pravilo moze da se napise kao

$$r_i : (Condition_i) \rightarrow y_i$$

Leva strana pravila se naziva **preduslov**, i oblika je

$$Condition_i = (A_1 \text{ op } v_1) \wedge (A_2 \text{ op } v_2) \wedge \dots \wedge (A_k \text{ op } v_k)$$

gde je (A_j, v_j) par atribut-vrednost i gde je *op* odgovarajuća logička operacija iz skupa $\{=, \neq, <, >, \leq, \geq\}$. Desna strana pravila se naziva **posledica pravila**, koja sadrži klasu y_i .

Pravilo r *prekriva* slog x ako se preduslov od r poklapa sa atributima od x . Takođe za r se kaže da *pali* ili *pokreće* sve slogove koje pokriva. Neka su data sledeća dva sloga, za sokola i grizlija:

Ime	Temperatura tela	Zenka radja	Koza	Morska stvorenja	Vazdusna stvorenja	Ima noge	Hibernira
sokol	toplo-krvna	ne	perije	ne	da	da	ne
grizli	toplo-krvna	da	krzno	ne	ne	da	da

Kazemo da r_1 prekriva prvu zivotinju, sokola, zato sto je preduslov zadovoljen sokolovim atributima. Ali ovo pravilo ne prekriva drugu zivotinju, grizlija, zato sto grizli radja i ne moze da leti, pa krši pravilo r_1 .

Kvalitet klasifikacionog pravila moze se izracunati merama kao sto su prekrivenost i tacnost. Za dati skup podataka D i klasifikaciono pravilo $r : A \rightarrow y$, *prekrivenost pravila* se definise kao odnos broja slogova iz D koja pravilo r pokrece i ukupnog broja slogova iz D . Dok je *tacnost* ili *faktor samopuzdanja* se definisa kao odnos broja slogova koje pokrece r i cija je klasna oznaka jednaka y , i ukupnog broja slogova iz D :

$$\text{Coverage}(r) = \frac{|A|}{|D|}; \text{Accuracy}(r) = \frac{|A \cap y|}{|D|}$$

Kako klasifikator zasnovan na pravilima radi?

Razmotrimo sledeće slogove:

Ime	Temperatura tela	Zenka radja	Koza	Morska stvorenja	Vazdusna stvorenja	Ima noge	Hibernira
lemur	toplo-krvna	da	krzno	ne	ne	da	da
kornjaca	hladno-krvna	ne	krljosti	semi	ne	da	ne
ajkula	hladno-krvna	da	krljosti	da	ne	ne	ne

- Prva zivotinja, lemur, je toplo-krvna i radja mlade. Ovo pali pravilo r_3 , pa se klasifikuje kao sisar.
- Druga zivotinja, kornjaca, pali pravila r_4 i r_5 . Kako su posledica ovih pravila dve razlicite klase, onda moramo razresiti ovaj problem.
- Treća zivotinja, ajkula, ne pali ni jedno pravilo. U ovom slucaju moramo ipak napraviti nekakvo predviđanje klase za slogove koji nisu pokriveni pravilom.

Medjusobno iskljuciva pravila. Pravila u skupu pravila R su medjusobno iskljuciva ako se ni koja dva pravila iz R ne pokrecu istim slogom. Ovime se omogucava da svaki slog bude pokriven najviše jednim pravilom iz R .

Iscrpljujuća pravila. Skup pravila R ima iscrpljujuće pokrivanje ako postoji pravilo za svaku kombinaciju vrednosti atributa. Ovo omogucava da svaki slog bude pokriven bar jednim pravilom iz R .

Primer skupa pravila R za koji vazi da je medjusobno iskljuciv i iscrpljujuc:

```

r1: (Temperatura=hladno-krvna) -> ne-sisar
r2: (Temperatura=toplo-krvna) and (Radja=da) -> sisar
r3: (Temperatura=toplo-krvna) and (Radja=ne) -> ne-sisar

```

Ukoliko skup nije iscrpljiv moramo dodati defaultno pravilo $r_d : () \rightarrow y_d$, koje će pokriti sve ostale slucaje.

Ovde y_d nazivamo defaultnom klasom.

Ako skup pravila nije medjusobno iskljuciv, onda neki slog moze da trigeruju vise pravila, koji za posledicu pravila imaju razlicitu klasnu oznaku. Ovaj konflikt moramo resiti:

Uredjena pravila. Skup pravila uredjujemo po prioritetu. Uredjeni skup pravila se jos i naziva **lista odlucivanja**. Dati slog se klasifikuje najvećim pravilom koje ga prekriva.

Neuredjena pravila. Ovaj pristup dopusta da se vise klasifikacionih pravila pokrenu i razmatra svako od njih kao tezinski glas neke klase. Slog se onda klasifikuje klasnom oznakom koja ima najveći broj glasova.

Seme uredjenja-pravila

Seme uredjenja baziranje na pravilima. Ovaj pristup uredjuje pravila po nekoj meri kvaliteta pravila. Potencijalna opasnost ovog pristupa je interpretacija pravila koja su rangirana nisko. Njih posmatramo kao negaciju svih pravila iznad njega i njega samog.

Seme uredjenja baziranje na klasama. Pravila koja imaju istu posledicu, tj. predviđaju istu klasu, se grupisu jedna do drugog. Relativno uredjenje unutar jedne grupe nije bitno. Ovima je interpretacija pravila jednostavnija.

Kako napraviti klasifikator zasnovan na pravilima?

Da bi napravili klasifikator zasnovan na pravilima moramo izdvojimo skup pravila koje indentifikuju veze izmedju atributa skupa podataka i oznaka klasa. Postoje dva nacina za izdvajanje klasifikacionih pravila: (1) *Direktni metod* - izdvaja pravila direktno iz skupa podataka, i (2) *Indirektni metod* - izdvaja pravila iz nekih drugih klasifikatora.

Direktni metodi za izdvajanje pravila

Algoritam **sekvencijalnog pokrivanja** se koristi za izdvajanje pravila direktno iz skupa podataka. Algoritam je zasnovan na gramzivoj tehnici, i izdvaja pravila jedne po jedne klase za skup podatak koji sadrzi vise od dve klase. Kriterium odlucivanja redosleda klasa zavisi od broja slogova koji pripadaju odredjenoj klasi, i cene slogova koji su pogresno klasifikovani za datu klasu.

```
def sequential_covering(training_records, attribute_value, classes, default_class):
    rule_list = []
    for y in ordered(classes):
        while not stop_cond():
            rule = learn_one_rule(training_records, attribute_value, y)
            training_records.remove_records_covered_by(rule)
            rule_list.append(rule)
    default_rule = make_default_rule(default_class)
    rule_list.append(default_rule)
    return rule_list
```

-----	-----
- + + - -	- ---- - -
- + + - -	- r1 - -
- + - - -	- ---- - - -
- - - -	- - - -
- - - - + +	- - - - + +
+ + - - +	+ + - - +
-----	-----

-----	-----
- ---- - -	- ---- - -
- r1 - -	- r1 - -

	-		----		-	-	-			-		----		-	-	-	
	-	-	-	-	-	-	-			-	-	-	-	-	-	-	
	-	-	-	-		--				-	-	-	-		--		
	+	+	-	-		r2				r3		-	-		r2		
	-----	-----	-----	-----		--				----	----	----	----		--		

Funkcija `learn_one_rule`

Glavni cilj `learn_one_rule` funkcije je da izdvoji klasifikaciono pravilo koje poklapa mnoge pozitivne primere, a ne poklapa negativne primere iz skupa za treniranje. Ali nalazenje optimalnog pravila je racunski skupo. Prvo se generise inicijalno pravilo r koje se obradjuje dok uslov zaustavljanja nije zadovoljen. To pravilo se onda potkresuje za poboljsanje greske generalizacije.

Strategija rasta pravila. Postoje dva nacina za rast klasifikacionog pravila: generalno-u-specificno ili specificno-u-generalno.

U slucaju generalno-u-specificno, inicijalno pravilo je $r : \{\} \rightarrow y$. Nova konjukcija se dodaje u preduslovu da bi poboljsala kvalitet. Svaka naredna konjukcija se ispituje i gramzivo se bira ona koja najbolje poboljsava kvalitet pravila. Ovaj proces staje kada se zadovolji neki uslov, na primer, dodata konjukcija ne poboljsava kvalitet pravila.

U slucaju specificno-u-generalno, inicijlano se nasumicno biraja jedno pravilo. U toku poboljsavanje ovog pravila, izbacuje se jedna po jedna konjukcija takoda pravilo pokriva vise pozitivnih slogova. Ovaj proces staje kada se zadovolji neki uslov, na primer, kada pravilo pocne da pokriva negativne slogove.

Ova dva postupka ne daju optimalno pravilo, jer koriste gramzicu tehniku. Zbog toga algoritam moze odrzavati k najboljih kandidata. Svaki kandidat za pravilo raste posebno tako sto mu se dodaju ili izbacuju konjukcije. Kvalitet kandidata se racuna i k najboljih se bira za sledecu iteraciju.

Evaluacija pravila. Evaluaciona metrika se koristi za odredjivanje konjukcije koju treba dodati ili izbaciti tokom rasta pravila. Jedna od mera koja moze da se koristi jeste tacnost pravila. Ali tacnost ne uzima u obzir broj primera koje pokriva pravilo.

Skup: 60 pozitivnih primera i 100 negativnih primera
 r_1 : poklapa 50 pozitivnih primera i 5 negativnih primera
 r_2 : poklapa 2 pozitivna primera i 0 negativnih primera

Tacnosti za r_1 i r_2 su 90.9% i 100%, respektivno. Ali r_1 je bolje pravilo bez obzira sto ima manju tacnost. Ovo mozemo resiti na vise nacina:

1. Mozemo koristiti statisticki test za potkresivanje pravila koje ima jedno pokrivanje.

$$R = 2 \sum_{i=1}^k f_i \log(f_i/e_i)$$

gde je k broj klasa, f_i je frekvencija pojave primera klase i koji su pokriveni pravilom, i e_i ocekivana frekvencija pravila koje pravi nasumicno predvidjanje. Za pravilo r_1 imamo da je: $e_+ = 55 \times 60/160 = 20.625$, i $e_- = 55 \times 100/160 = 34.375$. Ona je

$$R(r_1) = 2 \times [50 \times \log_2(50/20.625) + 5 \times \log_2(5/34.375)] = 99.9.$$

Dok je za pravilo r_2 : $e_+ = 2 \times 60/160 = 0.75$, i $e_- = 2 \times 100/160 = 1.25$. Ona je

$$R(r_2) = 2 \times [2 \times \log_2(2/0.75) + 0 \times \log_2(0/1.25)] = 5.66.$$

Pa je jasno pravilo r_1 bolje od pravila r_2 .

2. Evaluacione metrike koje uzimaju u racun pokrivenost pravila.

$$\text{Laplace} = \frac{f_+ + 1}{n + k}; \text{ m-estimate} = \frac{f_+ + kp_+}{n + k}$$

gde je n broj primera pokrivenih pravilom, f_+ broj pozitivnih primera pokrevenih pravilom, k ukupan broj klasa, p_+ prvobitna verovatnoca pozitivne klase. Za r_1 imamo da je Laplasova mera $51/57 = 89.47\%$, dok je Laplasova mera za r_2 jednaka $3/4 = 75\%$.

3. Evaluaciona metrika koja uzima u racu broj pozitivnih primera koje poklapa pravilo. Primer ove metrike je **FOILOva informaciona dobit**. Neka pravilo $r : A \rightarrow +$ poklapa p_0 pozitivnih primera i n_0 negativnih primera. Nakon dodavanja nove konjukcije B , prosireno pravilo $r' : A \wedge B \rightarrow +$ pokriva p_1 pozitivnih primera i n_1 negativnih primera. Onda je FOILOva informaciona dobit prosirenog pravila definisan kao

$$\text{FOIL's information gain} = p_1 \times (\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0})$$

Potkresivanje pravila. Pravilo koje dobijamo funkcijom `learn_one_rule` mozemo da potkresemo kako bi dobili bolju gresku generalizacije.

Uklanjanje instanci

Nakon sto je pravilo izdvojene, algoritam sekvencionalnog poklapanja mora eliminisati sve pozitivni i negativne primere koje poklapa pravilo.

```
|-----|
| -|R1--| -   - |
| - |+ -+|-   - |
| -| |---|R2|-  - |
|   | |-  |   -  |
| - |-|-|-|-  -  |
| +  |-----| + + |
|-----|
```

U ovom slucaju prvo je generisan pokrivan R_1 , ukoliko ne eliminisemo pozitivni i negativne primere koje on poklapa moze se desiti da se u sledecoj generaciji generise pokrivan R_2 , koji takodje pokriva neka pravila koja je pokrivao R_1 . Ako se pozitivni primeri poklopljeni sa R_1 ne eliminisu, onda mozemo da *procenimo* tacnost nekog sledeceg R_k , slicno ako ne ukonimo negativne primere poklopljene sa R_2 , onda mozemo da *procenimo* tacnost nekog sledeceng R_k .

RIPPER Algorithm

RIPPER je direktan metod izdvajanja pravila. Ovaj algoritam se skalira linearno sa brojem primera za treniranje i dobar je za podatke koji imaju nebalansirane distribucije, takodje, radi dobro za skupove podataka koji imaju sum.

Za dvoklasne probleme, RIPPER bira klasu vecine, i uci pravila tako sto opaza klasu manjine. Za viseklasne probleme, klase se uredjuju prema njihovoj frekvenciji. Neka je (y_1, y_2, \dots, y_c) uredjene klase, gde je y_1 najmanje frekventna klasa i y_c najvise frekventna klasa. Tokom prve iteracije, instance koje pripadaju u y_1 se oznace kao pozitivni primere, dok se sve druge oznace kao negativni primeri. Algoritam sekvencionalnog pokrivanja se onda koristi za generisanje pravila koja prekrivaju pozitivne i negativne primere. RIPPER, dalje, izdavaa pravila koja razdvajaju y_2 od ostalih klasa. Ovo ponavljamo svo dok ne dodjemo do y_c , koja je defaultna klasa.

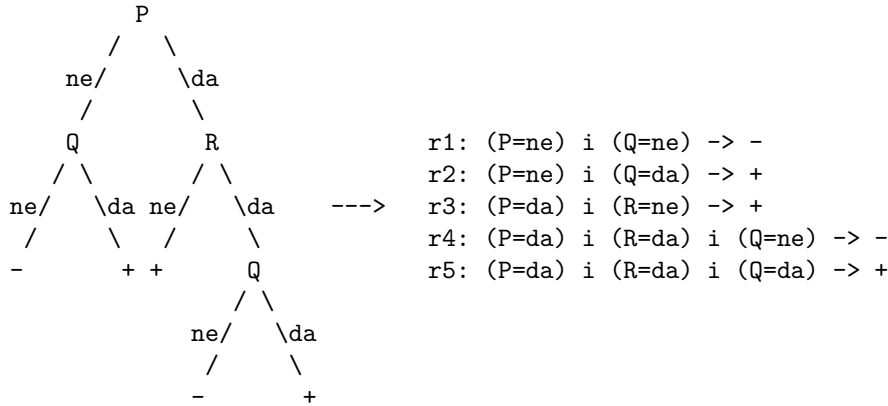
Rast pravila. RIPPER koristi strategiju generalno-u-specificno za rast pravila i FOILOva informaciona dobit za meru naboljeg konjukta kojeg dodajemo u pravilo. Prestajemo da dodajemo konjukte kada pravilo pocne da prekriva negativne primere. Pravilo se onda potkrese na osnovu njegove performanse nad skupom za validaciju. Sledeca metrisa se koristi za odredjivanje da li potkresivanje potrebno: $(p - n)/(p + n)$, gde je $p(n)$ broj pozitivnih(negativnih) primera u skupu za validaciju koje pokriva pravila. Potkresivanje se zapocinje od zadnje dodatog pravila. Na primer, za pravilo $ABCD \rightarrow y$, RIPPER proverava da li D treba potresati, nakon toga proverava CD , pa BCD , itd. Potkresano pravilo, onda, moze da pokriva i neka negativna pravila.

Gradjenje skupa pravila. Nakon generisanja pravila, svi pozitivni i negativni primeri koji su pokriveni pravilom su uklonjeni. Pravilo se dodaje u skup ravila sve dok ne krsi uslov zaustavljanja, koji se bazira na principut minimalno opisane duzine. Ako novo pravilo povecava ukupno opisanu duzinu skupa pravila za bar

d bita, onda RIPPER prestaje da dodaje novo pravilo u skup pravila ($d = 64bits$). Drugi uslov zaustavljanja koji koristi RIPPER jeste da odnos greske pravila nad validacionim skupom ne premasuje 50%.

Indirektni metodi za izdvajanje pravila

Svaka putanje od korena do lista u drvetu odlucivanja moze da se predstavi kao klasifikaciono pravilo. Svaki uslov testiranja na top putu predstavlja jednu konjukciju pravila, dok klasna oznaka lista predstavlja posledicu pravila.



Primer: Razmotrimo pravila:

$$r_2 : (P = ne) \wedge (Q = da) \rightarrow +$$

$$r_3 : (P = da) \wedge (R = ne) \rightarrow +$$

$$r_5 : (P = da) \wedge (R = da) \wedge (Q = da) \rightarrow +$$

Pravila r_2 i r_5 mozemo da zamenimo sa jednim pravilom, jer kad god je $Q = da$, to klasifikujemo kao $+$. Pa dobijamo:

$$r'_2 : (Q = da) \rightarrow +$$

$$r_3 : (P = da) \wedge (R = ne) \rightarrow +$$

Generisanje pravila. Klasifikaciona pravila se prosičuje za svaku putanju od korena do jednog od listova drveta odlucivanja. Za dato pravila $r : A \rightarrow y$, razmatramo pojednostavljeno pravilo $r' : A' \rightarrow y$, gde je A' dobijeno izbacivanjem nekih konjukcija iz A . Pojednostavljeno pravila sa najmanjom pesimisticom greskom se zadrzava, sa tim da je odnos greske manji od originalnog pravila. Nakon potkresivanja mozemo dobiti identicna pravila, pa duplikate izbacujemo iz skupa pravila.

Uredjivanje pravila. Nakon generisanja skupa pravila, koristimo semu uredjivanje baziranu na klasama za uredjivanje izdvojenih pravila. Pravila koja predvidjaju iste klase se grupisu zajedno u neki podskup. Ukupna opisna duzina za svaki podskup se racuna, i klase se uredjuju u rastucem poretku po ukupnoj opisnoj duzini. Klase sa najmanjom opisnom duzinom imaju najveći prioritet zato sto se ocekuje da sadrže najbolji skup pravila. Ukupna opisna duzina za klase je data sa $L_{exception} + g \times L_{model}$, gde je $L_{exception}$ broj bitova potrebnih za enkodiranje pogresno klasifikovanih primera, L_{model} je broj bitova potrebnih za enkodiranje modela, i g je parametar sa default vrednoscu od 0.5.

Karakteristike klasifikatora zasnovanog na pravilima

- Izrazitost skupa pravila je skoro ekvivalentna onoj od drveta odlucivanja, zato sto se svako drvo odlucivanja moze predstaviti kao skup uzajamno iskljucivih i iscrpljujucih pravila. Klasifikacije bazirane na pravilima i drvetima odlucivanja kreiraju pravougaonastu particiju prostora atributa i dodeljuju klasu svakoj particiji.
- Klasifikatori bazirani na pravilima se generalno koristi za modele koji su opisne prirode, ali daje slicne performanse klasifikatorima baziranim na drvetima odlucivanja.
- Klasifikacija zasnovana na uredjenju, kao sto je RIPPER, je dobra za skupove podataka koji imaju nebalansiranu distribuciju klasa.

Klasifikator: Najblizi sused

Postupak klasifikacije može se opisati u dva koraka: (1) induktivni korak za konstruisanje modele pomoću podataka za treniranje, i (2) dekuktivni korak za primenu modela nad podacima za testiranje. Drveta odlucivanja i klasifikatori bazirani na pravilima su primeri **zeljnog ucenja**, jer uče model koji mapira ulazne atribute u neku klasnu oznaku čim su im podaci iz skupa za treniranje dostupni. Druga strategija bi bila da odlazemo proces modelovanja podataka za treniranje sve dok nam nije potrebno da klasifikujemo podatke za testiranje (**lenjo ucenje**). Jedan primer lenjog ucenja je **ucenje napamet**, ova strategija memorise sve instance iz skupa za treniranje i klasifikuje samo ako se atributi instance iz skupa za testiranje poklope sa nekom instancom iz skupa za treniranje. Ovom metodom mnogi slogovi neće biti klasifikovani ukoliko se ne nalaze u skupu za treniranje.

Učenje napamet može da se relaksira, u smislu da se pronalaze instance iz skupa za treniranje koje su relativno bliske instanci iz skupa za testiranje. Ovoj pristup je poznat kao **najblizi sused**. Ideja ovog pristupa dolazi iz poslovice: “*Ako nešto hoda kao patka, kvace kao patka, i izgleda kao patka, onda je to verovatno patka.*”

Klasifikator najblizeg suseda predstavlja svaki objekat kao d -dimenzionu tacku u prostoru, gde je d broj atributa. Za datu instancu iz skupa za testiranje, racunamo meru slicnosti sa svim ostalim instancama iz skupa za treniranje. k -najblizih suseda date instance z referise na k tacaka koje su najslicnije instanci z . Instanca z se klasifikuje kao vecinom klasnih oznaka podataka koje pripadaju u tih k tacaka. U slucaju neresenih glasova, mozemo nasumicno izabrati jednu od te dve klasne oznake.

Bitno je izabrati odgovarajucu vrednost za k . Ako je k previše malo, onda je najblizi sused može biti osetljiv na pretreniranje zbog suma u skupu za treniranje. Sa druge strane, ako je k previše velika, najblizi sused može pogresno klasifikovati podatke, jer u k najblizih mogu se naci o oni koji nisu toliko slicni instanci z .

Algoritam

```
def k_nearest_neighbor(trening_set, test_set, k):
    for z in test_set:
        dist_z = {y: d(z, y) for y in trening_set}
        k_nearest = find_k_nearest(trening_set, dist_z)
        y_winner = most_frequent(k_nearest)
        z.y = y_winner
```

Karakteristike klasifikatora: najblizi sused

- Najblizi sused spada u tehnike poznate kao ucenje bazirano na instancama, koje koristi instance skupa za treniranje da bi napravilo predikciju bez pravljenja abstraktnog modela. Učenje bazirano na instancama zahteva meru slicnosti za bi pronaslo slicnost ili razdaljinu izmedju instanci i klasifikacionu funkciju koja vraća predviđenu klasu instance iz skupa za testiranje, baziranoj na slicnosti drugih instanci.
- Lenjo ucenje ne zahteva pravljenje modele. Ali zbog toga klasifikovanje instanci iz skupa za testiranje može biti veoma skupa operacija, kako zahteva racunanje mere slicnosti sa svakom instancom iz skupa za treniranje. Sa druge strane zeljno ucenje zahteva puno vremena za pravljenje modele, ali jednom kada se napravi model klasifikacija instanci iz skupa za testiranje je veoma brza.
- Najblizi sused klasifikuje tako sto koristi lokalne informacije, dok drveta odlucivanja i klasifikatori zasnivani na pravilima pokusavaju da pronadju globalni model koji se uklapa u celokupni ulazni prostor. Zbog lokalnosti najblizi sused je osetljiv na sum.
- Klasifikator najblizi sused može da proizvede granicu odlucivanja bilo kog oblika, sto je više fleksibilnije od drveta odlucivanja i klasifikatorima zasnovanim na pravilima, cije su granice odlucivanja pravougaoe.
- Klasifikator najblizi sused može da pogresno klasifikuje podatke ukoliko se izabere neodgovarajuca mera slicnosti i ukoliko se ne izvrsi predprocesiranje podataka na odgovarajuci nacin. Na primer, ako klasifikujemo ljude po tezini i visini, domen za tezinu je od $50kg$ do $150kg$, dok je domen za visinu od $1.5m$, do $2m$. Ako se ne racunaju skale u meri slicnosti, razlika u kilogramima imace mnogo više uticaja od razlike u visini.

Bajasov klasifikator

U mnogim slucajevima veza imezju skupa atributa i klasne promenljive nije deterministicka. Drugim recima, klasa oznaka sloga iz skupa za testiranje ne moze se predvideti sa sigurnoscu iako je njegov skup atributa identican nekom slogu iz skupa za treniranje. Na primer, razmotrimo zadatak predvidjanja: Da li osoba ima rizik od dobijanja srcanog napada na osnovu njene dijete i ucestalosti treniranja? Iako mnogi ljudi koji jednu zdravo i treniraju regularno imaju manje sanse da dobiju srcane probleme, postoje i ostali faktori kao sto je genetika, pusenje, alkohol. Takodje ispitivanje da li osoba ima zdravu ishranu i da li trenira regularno je takodje osetljiva tema, sto takodje moze da proizvede neodredjenost.

Bajasova teorema

Neka su X i Y slucajne promenljive. Verovatnoca $P(X = x, Y = y)$ je verovatnoca da promenljiva X uzme vrednost x , i promenljiva y uzme vrednost y . Uslovna verovatnoca $P(Y = y|X = x)$ je verovatnoca da ce slucajna promenljiva Y uzete vrednost y , pod uslovom da znamo da je slucajna promenljiva X uzela vrednost x . Veza izmedju ovih verovatnoca je data formulom mnozenja:

$$P(X, Y) = P(X)P(Y|X) = P(Y)P(X|Y)$$

Iz formule mnozenja sledi Bajasova teorema:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Bajasova teorema u klasifikaciji

Neka je \mathbf{X} skup atributa i Y klasna promenljiva. Ako klasna promenljiva nema deterministicku vezu sa atributima, onda su \mathbf{X} i Y slucajne promenljive, i mozemo njihovu vezu predstaviti verovatnocom $P(Y|\mathbf{X})$. Ova uslovna verovatnoca $P(Y|\mathbf{X})$ se zove **zadnja verovatnoca** za Y (verovatnoca dodeljivanja opazanja grupama za date podatke), dok je njoj suprotna $P(Y)$ poznata kao **prednja verovatnoca** (verovatnoca da ce nego zapazanje upasti u grupu pre nego sto se prikupe podaci).

Tokom ucenja moramo nauciti zadnju verovatnocu $P(Y|\mathbf{X})$ za svaku kombinaciju \mathbf{X} , i Y na osnovu informacije sakupnjene u skupu za treniranje. Ako znamo ovu verovatnocu, slog iz skupa za testiranje \mathbf{X}' se moze klasifikovati nalazenjem klase Y' koja maksimizuje zadnju verovatnocu $P(Y'|\mathbf{X}')$.

Posmatrajmo sledeci skup za treniranje sa atributima: Ima kucu, Bracni Status, Godisnja plata, i klasnom oznakom: Isplatio.

ID	Ima kucu	Bracni status	Godisnja plata	Isplatio
1	da	neozenjen	125k	da
2	ne	ozenje	100k	da
3	ne	neozenjen	70k	da
4	da	ozenjen	120k	da
5	ne	razveden	95k	ne
6	ne	ozenjen	60k	da
7	da	razveden	220k	da
8	ne	neozenjen	85k	ne
9	ne	ozenjen	75k	da
10	ne	neozenjen	90k	ne

Predpostavimo da je dat slog u skupu za testiranje sa sledecim atributima: $\mathbf{X} = (\text{Ima kucu}=\text{ne}, \text{Bracni status}=\text{ozenjen}, \text{Godisnja plata}=120\text{k})$ Da bi klasifikovali ovaj slog potrebno je izracunati zadnju verovatnocu $P(\text{da}|\mathbf{X})$ i $P(\text{ne}|\mathbf{X})$ na osnovu informacija iz skupa za treniranje. Ako je $P(\text{da}|\mathbf{X}) > P(\text{ne}|\mathbf{X})$, onda se slog klasifikuje kao **da**, inace se klasifikuje kao **ne**.

Kako je tacno procenjivanje zadnje verovatnoce veoma tezak problem, mozemo iskoristiti Bajesovu teoremu, ako se zadnja verovatnoca moze predstaviti u terminima prednje verovatnoce $P(Y)$, i **klasno-uslovne** verovatnoce $P(\mathbf{X}|Y)$, i dokaz, $P(\mathbf{X})$:

$$P(Y|\mathbf{X}) = \frac{P(\mathbf{X}|Y)P(Y)}{P(\mathbf{X})}$$

Kada se racuna zadnja verovatnoca za razlicite vrednosti Y , verovatnoca $P(\mathbf{X})$ je uvek ista i konstanta, te se moze ignorisati. Prednja verovatnoca $P(Y)$ se moze lako proceniti pomocu skupa za treniranje, kao odnos slugova koji pripadaju klasi Y i ukupan broj slogova. Za procenjivanje klasno-uslovne verovatnoce $P(\mathbf{X}|Y)$ imamo dve implementacije naivni Bajasov klasifikator i Bajasova mreza poverenja.

Naivni Bajasov klasifikator

Naivni Bajasov klasifikator procenjuje klasno-uslovnu verovatnocu tako sto predpostavlja da su atributi uslovne nezavisni, za datu klasnu oznaku y . Atributni su uslovno nezavisni, za datu klasnu oznaku y , akko:

$$P(\mathbf{X}|Y) = \prod_{i=1}^d P(X_i|Y = y),$$

gde svaki skup atributa $\mathbf{X} = \{X_1, X_2, \dots, X_d\}$ sadrzi d atributa.

Uslovna nezavisnost

Neka su \mathbf{X} , \mathbf{Y} , i \mathbf{Z} tri skupa slucajne promenljive. Promenljive u \mathbf{X} su uslovno nezavisne od \mathbf{Y} , za dato \mathbf{Z} , ako vazii:

$$P(\mathbf{X}|\mathbf{Y}, \mathbf{Z}) = P(\mathbf{X}|\mathbf{Z}).$$

Primer uslovne nezavisnosti je veza izmedju duzine ruke i vestine citanja. Mozemo primetiti da ljudi sa duzim rukama citaju bolje. Ova veza se objasnjava tako da ljudi koji imaju kratke ruke spadaju u decu ili bebe, pa losije citaju od odraslih ljudi koji imaju duze ruke. Ako je godina ljudi fiksirana, onda ova veza nestaje. Pa su duzina ruke i vestina citanja uslovno nezavisne kada su godine fiksirane.

Uslovna nezavisnost izmedju \mathbf{X} i \mathbf{Y} moze se zapisati kao:

$$\begin{aligned} P(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) &= \frac{P(\mathbf{X}, \mathbf{Y}, \mathbf{Z})}{P(\mathbf{Z})} \\ &= \frac{P(\mathbf{X}, \mathbf{Y}, \mathbf{Z})}{P(\mathbf{Y}, \mathbf{Z})} \frac{P(\mathbf{Y}, \mathbf{Z})}{P(\mathbf{Z})} \\ &= P(\mathbf{X}|\mathbf{Y}, \mathbf{Z})P(\mathbf{Y}|\mathbf{Z}) \\ &= P(\mathbf{X}|\mathbf{Z})P(\mathbf{Y}|\mathbf{Z}) \end{aligned}$$

Kako radi naivni Bajasov klasifikator?

Zbog uslovne nezavisnosti, umesto da racunamo klasno-uslovnu verovatnocu za svaku kombinaciju od \mathbf{X} , procenjujemo uslovni verovatnocu svakog X_i , za dati Y .

Za klasifikovanje test slogova, naivni Bajasov klasifikator racuna zadnju verovatnocu za svaku klasu Y kao:

$$P(Y|\mathbf{X}) = \frac{P(Y) \prod_{i=1}^d P(X_i|Y)}{P(\mathbf{X})}$$

Kako je $P(\mathbf{X})$ fiksno za svaki Y , dovoljno je odabrati klasu koja maksimizuje brojilac $P(Y) \prod_{i=1}^d P(X_i|Y)$.

Procenjivanje uslovne verovatnoce za kategoricke attribute

Za kategoricke atribute X_i , uslovna verovatnoca $P(X_i = x_i|Y = y)$ se procenjuje kao odnos instanci iz skupa za treniranje u klasi y koji uzimaju odredjene vrednosti atribute x_i sa ukupnim brojem instanci iz skupa za treniranje. Na primer, u skupu za treniranje gore, tri od sedam ljudi koji su isplatili kredit, takodje poseduju i kucu. Kao rezultat, uslovna verovatnoca je $P(\text{Ima kucu}=\text{da}|\text{da}) = 3/7$. Slucno, uslovna verovatnoca onih koji nisu isplatili kredit, i koji su neozenjeni je $P(\text{Bracni status}=\text{neozenjen}|\text{ne}) = 2/3$.

Procenjivanje uslovne verovatnoce za neprekidne atribute

Postoje dva pristupa kada se procenjuje klasno-uslovna verovatnoca za neprekidne atribute:

1. Mozemo zameniti neprekidne atribute sa odgovarajucim ordinalne metodom diskretizacije. Onda je uslovna verovatnoca $P(X_i|Y = y)$ procenjena racunanjem odnosa slogova treniranje koji pripadaju klasi y koja upada u odgovarajuci interval za X_i . Procenjena greska zavisi od strategije diskretizacije, kao i od broja diskretnih intervala. Ako je broj diskretnih intervala previse velik, onda imamo manje slogova za treniranje u svakom od intervala pa dobijamo pogresku procenu za $P(X_i|Y)$. Sa druge strane, ako je broj intervala previse mali, onda neki intervali mogu agregirati slogovi iz drugih klasa i mozemo promasiti tacne granice odlucivanja.
2. Mozemo koristiti odredjenu distribuciju za neprekidne atribute i proceniti parametre distribucije na osnovu skupa za treniranje. Gausova distribucija se obicno koristi za predstavljanje klasno-uslovne verovatnoce za neprekidne atribute. Distribuciju karakterisu dva parametra, ocekivanje μ , i disperzija σ^2 . Za svaku klasu y_j , klasno-uslovna verovatnoca atributa X_i je

$$P(X_i = x_i|Y = y_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} \exp\left(-\frac{(x_i - \mu_{ij})^2}{2\sigma_{ij}^2}\right)$$

Parametri μ_{ij} i σ_{ij}^2 se mogu proceniti kao ocekivanje i disperzija od atributi X_i svih slogova treniranja koji pripadaju klasi y_j . Na primer jednostavno ocekivanje i disperzija za atribute sa klasom **da** je

$$\begin{aligned}\bar{x} &= \frac{125 + 100 + 70 + \dots + 75}{7} = 110 \\ \sigma^2 &= \frac{(125 - 110)^2 + (100 - 110)^2 + \dots + (75 - 110)^2}{7} = 2550 \\ \sigma &= 50.5\end{aligned}$$

Za dati slog iz testiranje sa godisnjom platom od 120k, mozemo izracunati njegovu klasno-uslovnu verovatnocu kao:

$$P(\text{Godisnja plata}=120|\text{da}) = \frac{1}{\sqrt{2\pi 2550}} \exp\left(-\frac{(120-110)^2}{2 \cdot 2550}\right) = 0.0077$$

Desni deo ove jednacline se naziva **funkcija gustine verovatnoce** $f(X_i; \mu_{ij}, \sigma_{ij}^2)$. Kako je ova funkcija neprekidna, verovatnoca da slucajna promenljiva X_i uzme odredjenu vrednost je nula. Zbog toga racunamo uslovnu verovatnocu da X_i upada u neki interval izmedju x_i i $x_i + \epsilon$, gde je ϵ neka mala konstanta, pa je:

$$\begin{aligned}P(x_i \leq X_i \leq x_i + \epsilon|Y = y_j) &= \int_{x_i}^{x_i + \epsilon} f(X_i; \mu_{ij}, \sigma_{ij}^2) dX_i \\ &= f(x_i; \mu_{ij}, \sigma_{ij}^2) \epsilon\end{aligned}$$

Kako je ϵ konstantan multiplikativni faktor za svaku klasnu oznaku, on se skracuje kada normalizujemo zadnju verovatnocu za $P(Y|\mathbf{X})$. Zbog toga, aproksimiramo klasno-uslovnu verovatnocu $P(X_i|Y)$, kao funkciju gustine u datoj tacki.

Primer naivnog Bajasovog klasifikatora

Neka je data sledece tabela za koju racunamo klasno-uslovne verovatnoce za svaki kategoricki atribut, zajedno sa jednostavnom sredinom i varijansom za neprekidne atribute metodom iz prethodnog poglavlja.

ID	Ima kucu	Bracni status	Godisnja plata	Isplatio
1	da	neozenjen	125k	da
2	ne	ozenje	100k	da
3	ne	neozenjen	70k	da
4	da	ozenjen	120k	da
5	ne	razveden	95k	ne
6	ne	ozenjen	60k	da
7	da	razveden	220k	da
8	ne	neozenjen	85k	ne
9	ne	ozenjen	75k	da
10	ne	neozenjen	90k	ne

$P(\text{Ima kucu}=\text{da}|\text{da}) = 3/7$

$P(\text{Ima kucu}=\text{ne}|\text{da}) = 4/7$

$P(\text{Ima kucu}=\text{da}|\text{ne}) = 0$

$P(\text{Ima kucu}=\text{ne}|\text{ne}) = 3/3=1$

$P(\text{Bracni status}=\text{neozenjen}|\text{da}) = 2/7$

$P(\text{Bracni status}=\text{ozenjen}|\text{da}) = 4/7$

$P(\text{Bracni status}=\text{razveden}|\text{da}) = 1/7$

$P(\text{Bracni status}=\text{neozenjen}|\text{ne}) = 2/3$

$P(\text{Bracni status}=\text{ozenjen}|\text{ne}) = 0$

$P(\text{Bracni status}=\text{razveden}|\text{ne}) = 1/3$

Godisnja plata:

da -> jednostavna sredina = 110

jednostavna varijansa = 2550

ne -> jednostavna sredina = 90

jednostavna varijansa = 16.6

Neka je $\mathbf{X} = (\text{Ima kucu}=\text{ne}, \text{Bracni status}=\text{ozenjen}, \text{Godisnja plata}=120\text{k})$ slog iz skupa za testiranje ciju klasu hocemo da predvidimo, tj. hocemo da izracunamo zadnje verovatnoce $P(\text{da}|\mathbf{X})$ i $P(\text{ne}|\mathbf{X})$. Znamo da je $P(\text{da}) = 0.7$ i $P(\text{ne}) = 0.3$, a klasno-uslovnu verovatnocu racunamo kao:

$P(\mathbf{X}|\text{da}) = P(\text{Ima kucu}=\text{ne}|\text{da})P(\text{Bracni status}=\text{ozenjen}|\text{da})P(\text{Godisnja plata}=120\text{k}|\text{da}) = 0.0018$

$P(\mathbf{X}|\text{ne}) = P(\text{Ima kucu}=\text{ne}|\text{ne})P(\text{Bracni status}=\text{ozenjen}|\text{ne})P(\text{Godisnja plata}=120\text{k}|\text{ne}) = 0.$

Na osnovu toga imamo za zadnju verovatnocu klase da da je $P(\text{da}|\mathbf{X}) = \alpha \times 0.7 \times 0.0018 = 0.00126$, gde je $\alpha = 1/P(\mathbf{X})$. Slicno, zadnja verovatnoca klase ne je $P(\text{ne}|\mathbf{X}) = \alpha \times 0.3 \times 0 = 0$. Kako vazi $P(\text{da}|\mathbf{X}) > P(\text{ne}|\mathbf{X})$, onda slog \mathbf{X} klasifikujemo kao da.

M-procene uslovne verovatnoce

Prethodni primer ilustruje potencijalni problem procenjivanja zadnje verovatnoce odnosom pomocu skupa za treniranje. Ako je klasno-uslovna verovatnoca nula onda je zadnja verovatnoca takodje nula. Ovo dobijamo ukoliko je skup podataka za treniranje velik, i pri tome ima malo podataka koji pripadaju trazenoj klasi, pa je odnos izmedju njih blizak nuli.

Ekstrimniji problem je slog za koji dobijemo da su obe klasno-uslovne verovatnoce nula. Onda naivni Bajasov klasifikator nece moci da klasifikuje dati slog. Ovaj problem mozemo resiti pomocu *m-procena* uslovne verovatnoce:

$$P(x_i|y_j) = \frac{n_c + mp}{n + m},$$

gde je n ukupan broj instanci klase y_j , n_c je broj instanci treniranje sa klasom y_j koji uzima vrednost x_i , m je parametar koji se naziva ekvivalent uzoracke velicine, i p je korisnicki definisan parametar. Ako je $n = 0$, tj. nema skupa za treniranje onda $P(x_i|y_j) = p$, pa je zbog toga p zadnja verovatnoca atributa x_i unutar slogova klase y_j . Ekvivalent uzoracke velicine odredjuje razmenu izmedju zadnje verovatnoce p i primecene verovatnoce n_c/n .

Karakteristike naivnog Bajasovog klasifikatora

- Naivni Bajasov klasifikator je dobar kada imamo podatke sa puno suma, ili nedostajucih vrednosti, jer se oni tokom pravljenja modela pripoje proseku, ili eliminisu.
- Ako \mathbf{X} sadrzi neki atribut X_i koji je irelevantan za klasifikaciju, onda $P(X_i|Y)$ ima uniformnu distribuciju. Zbog toga klasno-uslovna verovatnoca za X_i nema uticaj na ukupnu vrednost zadnje verovatnoce.
- Korelisani atributi smanjuju performanse naivnog Bajasovog klasifikatora, kako uslovna nezavisnost onda ne vazi za takve atribute. Razmotrimo sledeci primer:

$$\begin{aligned} P(A = 0|Y = 0) &= 0.4, P(A = 1|Y = 0) = 0.6, \\ P(A = 0|Y = 1) &= 0.6, P(A = 1|Y = 1) = 0.4, \end{aligned}$$

gde je A binarni atribut i Y binarna klasna promenljiva. Sada pretpostavimo da je drugi binarni atribut B savrseno korelisani sa atributom A kada je $Y = 0$, ali je nezavistan od A kada je $Y = 1$. Takodje neka su klasno-uslovne verovatnoce za B iste kao i za A . Neka je dat slog sa atributom $A = 0, B = 0$. Za njega mozemo izracunati zadnju verovatnocu kao:

$$\begin{aligned} P(Y = 0|A = 0, B = 0) &= \frac{P(A = 0|Y = 0)P(B = 0|Y = 0)P(Y = 0)}{P(A = 0, B = 0)} \\ &= \frac{0.16P(Y = 0)}{P(A = 0, B = 0)} \\ P(Y = 1|A = 0, B = 0) &= \frac{P(A = 0|Y = 1)P(B = 0|Y = 1)P(Y = 1)}{P(A = 0, B = 0)} \\ &= \frac{0.36P(Y = 1)}{P(A = 0, B = 0)} \end{aligned}$$

Pa naivni Bajasov klasifikator slogu dodeljuje klasu 1. Medjutim, kako su A i B savrseno korelisane kada je $Y = 0$, pa je:

$$P(A = 0, B = 0|Y = 0) = P(A = 0|Y = 0) = 0.4$$

Pa kao rezultat imamo da je

$$P(Y = 0|A = 0, B = 0) = \frac{0.4P(Y = 0)}{P(A = 0, B = 0)}$$

sto znaci da slogu treba dodeliti klasu 0.

Bajasova greska

Primer (Klasifikovanje aligatora i krokodila). Hocemo da klasifikujemo aligatore i krokodile na osnovu njihove duzine. Prosečna duzina krokodila je oko $5m$, dok je prosečna duzina aligatora oko $4m$. Predpostavljamo da duzina aligatora i krokodira ima normalnu raspodelu cije je standardno odstupanje 1:

$$\begin{aligned} P(X|\text{Krokodil}) &= \frac{1}{\sqrt{2\pi}} \exp \left[-\frac{1}{2} (X - 5)^2 \right] \\ P(X|\text{Aligator}) &= \frac{1}{\sqrt{2\pi}} \exp \left[-\frac{1}{2} (X - 4)^2 \right] \end{aligned}$$

Ako pretpostavimo da su njihove verovatnoce iste, onda je idealna gradica odlucivanja neka duzina \hat{x} takva da je $P(X = \hat{x}|\text{Krokodil}) = P(X = \hat{x}|\text{Aligator})$, tj.

$$(\hat{x} - 5)^2 = (\hat{x} - 4)^2$$

Kada resimo ovu kvadratnu jednacinu dobijamo da je $x = 4.5$. U ovam slucaju to je sredina izmedju dve sredine.

Kako je idealna granica odlucivanja duzina \hat{x} za koju su manje vrednosti klasifikuju kao aligatori, a vece kao krokodili, mozemo izracunati gresku koja nastaje prilikom klasifikovanja. Greska klasifikatore je data sumom površine ispod zadnje verovatnoce za krokodile i površine ispod zadnje verovatnoce za aligatore:

$$\text{Error} = \int_0^{\hat{x}} P(\text{Krokodil}|X)dX + \int_{\hat{x}}^{\infty} P(\text{Aligator}|X)dX$$

Ukupna greska se naziva **Bajasova greska**.

Bajasova mreza poverenja

Vestacke neuronske mreze

Vestacke neuronske mreze (ANN) su inspirisane simuliranjem bioloskog neuronskog sistema. Ljudski mozak sadrzi nerve koji se nazivaju **neuroni**. Neuron je povezan sa drugim neuronom preko vlakna koja se naziva **akson**. Aksoni sluze da prebace impulse jednog neurona na drugi. Neuroni su povezani preko aksona jednog neurona, i **dendrita**, drugog neurona. Mesto na kome se dendriti spaja sa aksonom se nazivaju **sinapse**. Ljudski mozak uci tako sto menja jacinu sinapsnih veza izmedju neurona.

Perceptron

Neka je dat sledeci skup podataka, gde su ulazne promenljive (x_1, x_2, x_3) bulove promenljive, i izlazna promenljiva y uzima vrednost -1 ako su bar dve od tri ulazne promenljive nula, inace uzima vrednost $+1$.

x1	x2	x3	y
1	0	0	-1
1	0	1	+1
1	1	0	+1
1	1	1	+1
0	0	1	-1
0	1	0	-1
0	1	1	+1
0	0	0	-1

Perceptron predstavlja najjednostavniju arhitekturu vestacke neuronske mreze. Perceptron sadrzi dva tipa cvora: ulazne cvorove, koji predstavljaju ulazne attribute, i izlazni cvor, koji predstavlja izlazni atribut. Cvorovi u mrezi su poznati kao neuroni ili jedinice. U perceptronu, svaki ulazni cvor je povezan preko tezinske grane sa izlaznim cvorom. Tezinska grane predstavljaju jacinu sinapsa izmedju neurona. Treniranje perceptrona zasniva se na adaptiranju tezina grana sve dok ne odgovaraju relaciji ulaza i izlaza.

```

      0.3    0.4
x1 -> 0 ----- |
      0.3    \  v
x2 -> 0 ----- sum -> sng -> y
      0.3    /
x3 -> 0 -----
```

Perceptron racuna svoju izlaznu vrednost, \hat{y} , tako sto primeni tezinsko sumiranje svoji ulaza i oduzimanjem faktora pristrasnosti b od sume, te posmatranjem znaka rezultata:

$$\hat{y} = \begin{cases} +1 & \text{ako } 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.4 > 0 \\ -1 & \text{ako } 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.4 < 0 \end{cases}$$

Primetimo razliku izmedju ulaznih i izlaznih cvorova perceptrona. Ulazni cvor jednostavno prosledjuje

vrednosti koje primi na ilaznu granu bez primene bilo kakve transformacije. Izlazni cvor, sa druge strane, je matematički aparat koji racuna tezinu sumu ulaza, oduzima pristrasnost, te primenjuje funkciju znaka. Tacnije, izlazni cvor se moze predstaviti matematički kao:

$$\hat{y} = \text{sign}(w_d x_d + w_{d-1} x_{d-1} + \dots + w_2 x_2 + w_1 x_1 - b),$$

gde su w_1, w_2, \dots, w_d tezine ulaznih grana i x_1, x_2, \dots, x_d ulazne vrednosti atributa. Funkcija znaka deluje kao **aktivaciona funkcija** za izlazne neurone. Model perceptrona mozemo zapisati i krace kao:

$$\hat{y} = \text{sign}[w_d x_d + w_{d-1} x_{d-1} + \dots + w_2 x_2 + w_1 x_1 + w_0 x_0] = \text{sign}(\mathbf{w} \cdot \mathbf{x}),$$

gde je $w_0 = -t$, $x_0 = 1$ i $\mathbf{w} \cdot \mathbf{x}$ skalarni proizvod izmedju vektora tezina w i ulaznih atributa \mathbf{x} .

Ucenje modela perceptrona

```
def learn_perceptron(training_set, lambda):
    w = [random(-1,1) for _ in range(len(training_set))]
    while stop_condition(w):
        for x_i, y_i in D:
            y_i_hat = scalar_product(w, x)
            for w_j in w:
                w_j += lambda * (y_i - y_i_hat) * x_i[j]
```

Kljucan deo ovog algoritma je azuriranje tezina koje se radi po formuli:

$$w_j^{(k+1)} = w_j^{(k)} + \lambda(y_i - \hat{y}_i^{(k)})x_{ij},$$

gde je $w_i^{(k)}$ teza u i -te grane nakon k -te iteracije, λ je parametar koji se zove **brzina ucenja**, i x_{ij} je vrednost j -tog atributa za instancu treniranje \mathbf{x}_i . Do ove formule se dolazi intuitivno, tj. zavisi vecinom od greske predvidjanja ($y - \hat{y}$). Ako je predvidjanje korektno, onda teza ostaje ista. Inace, menja na sledeci nacin:

- Ako je $y = +1$ i $\hat{y} = -1$, onda je greska predvidjanja ($y - \hat{y}$) = 2. Da bi ublazili gresku moramo povecati vrednost predvidjenog izlaza, tako sto povecemo tezinu svih grana sa pozitivnim ulazom i smanjimo tezinu svih grana sa negativnim ulazom.
- Ako je $y = -1$ i $\hat{y} = +1$, onda je greska predvidjanja ($y - \hat{y}$) = -2. Da bi ublazili gresku moramo smanjiti vrednost predvidjenog izlaza, tako sto smanjimo tezinu svih grana sa pozitivnim ulazom i povecemo tezinu svih grana sa negativnim ulazom.

Tezinu u svakom azuriranju ne smemo povecati previse, kako ne bi doslo do pregazivanja prethodnih azuriranja, pa se zbog toga koristi parametar brzine ucenja λ , koji uzima vrednost izmedju 0 i 1. Ako je λ bizu nule, onda stara teza veoma utice na novu tezinu, sa druge strane, ako je λ blizu jedinice, onda nova teza puno zavisi od greske predvidjanja. Moze se koristiti i adaptivna vrednost parametra λ , na primer, na pocetku moze biti umereno velika, a da se svakom sledecom iteracijom smanjuje.

Kako je model perceptrona iz gornjeg primera bio linearan po atributima \mathbf{x} i tezinama \mathbf{w} , granica odlucivanja je linearna hiperravan koja dele podatke na dve klase -1 i $+1$. Perceptron zbog toga daje optimalna resenje ako je problem klasifikacije linearno razdvojiv. Ako problem nije linearno razdvojiv onda algoritam ne konvergira. Jedan takav primer je XOR funkcija.

Viseslojne vestacke neuronske mreze

Viseslojne vestacke neuronske mreze imaju kompikovaniju strukturu od modela perceptrona:

1. Mreza može sadržati nekoliko slojeva između ulaznog i izlaznog sloja. Takvi unutrasnji slojevi se nazivaju **skriveni slojevi** i čvorovi unutar ovih slojeva se nazivaju **skriveni čvorovi**. Takva struktura se naziva višeslojna neuronska mreža. U neuronskoj mrezi sa **propagacijom-unapred**, čvorovi jednog sloja su povezani samo sa čvorovima sledećeg sloja. Perceptron je jednoslojna neuronska mreža sa propagacijom-unapred, jer ima samo jedan sloj čvorova i izlazni čvor se ponasa kao matematička funkcija. U **rekurentnim** neuronskim mrežama moguće je imati grane koje povezuju čvor nekog sloja, sa čvorom iz nekog prethodnog sloja.
2. Mreza može da koristi i druge tipove aktivacionih funkcija koje nisu sinusne funkcije. Primeri drugih aktivacionih funkcija uključuju linearnu, sigmoid (logisticku), i hiperboličku tangens funkciju. Ove aktivacione funkcije dozvoljavaju skrivenim i izlaznim čvorovima da imaju izlaznu vrednost koja nije linearna u odnosu na ulazne parametre.

Zbog ove kompleksnosti možemo da rešimo nerazdvojive probleme kao što je XOR problem. Ako konstruisemo mrežu koja ima dva ulazna čvora, dva skrivena čvora, i jedan izlazni čvor, svaki od slojeva možemo posmatrati kao jedan perceptron koji generise hiperravan nad prostorom. Kombinacija tih hiperravni generise granicu odlučivanja.

Da bi naučili odgovarajuće težine za ANN model, treba nam efikasan algoritam koji konvergira u tačno rešenje kada imamo dovoljno podataka u skupu za treniranje. Jedno rešenje koje se nameće jeste da koristimo istu tehniku kao za perceptron i da pri tome delimo mrežu na perceptrone. Ovaj metod neće raditi kako nemamo nikakvo znanje od izlazima skrivenog sloja, pa ne možemo da izračunamo gresku $(y - \hat{y})$ za svaki od skrivenih čvorova.

Učenje ANN modela

Učenje težina neuronske mreže se zasniva na gradijentnom spustu, tj. cilj nam je da minimizujemo ukupnu kvadriranu gresku:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

Primetimo da je \hat{y}_i zavisi od $\mathbf{w} \cdot \mathbf{x}$.

Kako je minimum za nelinearnu funkciju $E(\mathbf{w})$ tesko naći, koristimo gramzivnu tehniku gradijentnog spusta gde će formula azuriranja težina biti:

$$w_j \leftarrow w_j - \lambda \frac{\partial E(\mathbf{w})}{\partial w_j}$$

Ova formula znači da se u svakoj iteraciji pomeramo malo u smeru tak da smanjujemo celokupnu gresku. Moguće je zaglaviti se u lokalnom minimumu.

Za skrivene čvorove, računanje greske nije lako kako moramo da računamo faktor greske $\partial E / \partial w_j$, iako ne znamo koje vrednosti njihov izlaz treba da bude. Zbog toga se koristi tehnika koja se zove **propagacija unazad**. Postoje dve faze u svakoj iteraciji algoritma: propagacije unapred i propagacije unazad. Tokom propagacije unapred, težine iz prethodne iteracije se koriste za računanje izlazne vrednosti svakog neurona u mreži. Računanje izlaza neurona se propagira kroz slojeve od ulaznog, do izlaznog sloja, pa otuda ima propagacija unapred. Tokom propagacije unazad, azuriranje težina se primenjuje u suprotnom smeru. Propagacije unazad omogućava da greske neurona na $k + 1$ -om sloju procenjuju greske neurona na k -tom sloju.

Problemi pri dizajniranju ANN učenja

1. Treba odrediti broj čvorova za ulazni sloj. Svakom ulaznom čvoru dodeljujemo numericku ili binarnu ulaznu promenljivu. Ako je ulazna promenljiva kategoricka, onda ili imamo onoliko ulaznih promenljivih koliko ima kategorija ili enkodiramo k -arnu promenljivu pomoću $\lceil \log_2 k \rceil$ ulaznih čvorova.

2. Broj cvorova izlaznog sloja treba odrediti. Za dvoklasne probleme, dovoljno je koristiti jedan izlazni cvor. Za k -klasni problem, imamo k izlaznih cvorova.
3. Treba odrediti topologiju mreze, tj. treba odrediti broj skrivenih slojeva, broj cvorova u svakom sloju, i nacin povezivanja cvorova izmedju slojeva.
4. Tezine i pristrasnosti treba inicijalizovati. Nasumicno dodeljivanje vrednosti je prihvatljivo.
5. Instance iz skupa za treniranje sa nedostajucim vrednostima treba obrisati ili zameniti sa najcescom vrednoscu.

Karakteristike vestacke neuronske mreze

1. Viseslojne vestacke neuronske mreze sa bar jednim skrivenim slojem su **univerzalni aproksimatori**, tj. mogu se koristiti da aproksimisu bilo koju ciljnu funkciju. Kako neuronska mreza ima veoma ekspresivan hipoteticki prostor, vazno je odrediti odgovarajucu topologiju mreze za dati problem kako bi izbegli pretreniranje modela.
2. Vestacke neuronske mreze se mogu nositi sa redundantnim osobinama zato sto su tezine automacki uce tokom treniranja. Tezine za redundantne osobine su veoma male, pa tako ne uticu.
3. Neuronske mreze su veoma senzitivne na prisustvo suma u skupu za treniranje. Jedan pristup je validacija skupa za odredjivanje generalne greske modela. Dok je drugi pristup smanjivanje tezina sa nekim faktorom u svakoj iteraciji.
4. Gradijentni spust se koristi za učenje tezina za mrezu, a on vrlo cesto konvergira ka lokalnom minimumu. Jedan nacin da se izbegne lokni minimum jeste da se doda momentum za azuriranje tezina formula.
5. Treniranje vestacke neuronske mreze je vremenski zahtevan proces, posebno kada je broj skrivenih cvorova velik.

Metod potpunih vektora (Support Vector Machine — SVM)

SVM ima mnoge primene, a potice iz teorije statistickog učenja. Radi dobro za podatke za puno dimenzija. Predstavlja granicu odluke pomocu podskupova instanci za treniranje, koji se zovu **potpuni vektor**.

Hiperravni maksimalne margine

U slucaju da imamo linearno razdvojiv skup podataka, kroz njega mozemo provuci beskonacno mnogo hiperravni koje razdvajaju podatke. Za svaki od tih hiperravni vazi da ce greska modela nad trening skupu biti jednaka nuli, ali nemamo nikakvu garanciju da ce ta hiperravan dobro generalizovati, tj. da ce greska nad test skupom biti mala.

Razmotrimo sada dve granice odluke, B_1 i B_2 . Svako od granica odluke je dodeljen par hiperravni b_{i1} i b_{i2} . Ove hiperravni dobijamo tako sto se paralelno razdvajamo b_{i1} i b_{i2} , sve dok one ne dodiruju najblizu instancu treniranja. Distanca izmedju te dve hiperravni b_{i1} i b_{i2} naziva se **margin** klasifikatora.

Rasudjivanje o maksimalnoj margini

Prirodno je ocekivati da ce granice odluke koje imaju vecu marginu bolje generalizovati: Ako je margina mala, onda male promene u granici odluke mogu da imaju veliki uticaj na klasifikaciju.

Formalnije, generalizacionu gresku posmatramo kao princip statistickog učenja: **strukturni rizik minimizacije** (SRM). Ovaj princip pruza gornju granicu generalizacione greske klasifikatora (R) u terminima greske treniranja (R_e), broja instanci treniranje (N), i kompleksnosti modela (**kapacitet**) (h):

$$R \leq R_e + \varphi\left(\frac{h}{N}, \frac{\log(\eta)}{N}\right),$$

gde je φ monotono raste kada raste kompleksnost modela h . Zbog toga SRM pokazuje da sa porastom kompleksnosti modela greska treniranja opada, i obrnuto. Kao i do sada treba naci optimalnu sredinu.

Kompleksnost linearnog modela je obrnuto proporcionalna od njegove margine. Modeli sa malom marginom imaju veliki kapacitet jer su vise fleksibilni i mogu se prilagoditi mnogim skupovima za treniranje, za razliku

od modela sa velikom marginom. Ali SRM princip pokazuje da sa porastom kompleksnosti, generalizacija opada. Zbog toga, zelimo da napravimo linearni model koji maksimizuje marginu da bi minimizovali gresku generalizacije.

Linearna SVM: Razdvojitiv slucaj

Linearna granica odluke

Razmotrimo problem binarnu klasifikacije koji ima N instanci za treniranje. Svaka instanca je oblika (\mathbf{x}_i, y_i) , gde je $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{id})^T$, odgovara atributima i -te instance. Neka je $y_i \in \{-1, +1\}$ labela klase. Granica odluke linearnog klasifikatora je:

$$\mathbf{w} \cdot \mathbf{x} + b = 0,$$

gde su \mathbf{w} i b parametri modela.

Za bilo koje dve tacke \mathbf{x}_a i \mathbf{x}_b na granici odluke vazi:

$$\mathbf{w} \cdot \mathbf{x}_a + b = 0,$$

$$\mathbf{w} \cdot \mathbf{x}_b + b = 0.$$

Odatle imamo da je:

$$\mathbf{w} \cdot (\mathbf{x}_a - \mathbf{x}_b) = 0$$

Vektor $(\mathbf{x}_a - \mathbf{x}_b)$ je paralelan granici odluke, pa kako je skalarni proizvod jednak nuli, onda mora biti da je \mathbf{w} normalan na granicu odluke.

Za bilo koju instancu \mathbf{z} iznad granice odluke vazi:

$$\mathbf{w} \cdot \mathbf{z} + b > 0.$$

Dok za bilo koju instacu \mathbf{z} ispod granice odluke vazi:

$$\mathbf{w} \cdot \mathbf{z} + b < 0.$$

Zbog toga ako oznacimo sve instance iznad granice odluke klasom $+1$ i sve instance ispod granice odluke klasom -1 , onda vazi:

$$y = \begin{cases} +1 & , \text{ ako } \mathbf{w} \cdot \mathbf{z} + b > 0; \\ -1 & , \text{ ako } \mathbf{w} \cdot \mathbf{z} + b < 0. \end{cases}$$

Margina Linearnog Klasifikatora

Razmotrimo par najblizih instanci obe klase u odnosu na granicu odluke. Mozemo skalirati parametre \mathbf{w} i b granice odlucivanja tako da za dve paralelne vrednosti hiperravni b_{i1} i b_{i2} vazi:

$$b_{i1} : \mathbf{w} \cdot \mathbf{x} + b = +1,$$

$$b_{i2} : \mathbf{w} \cdot \mathbf{x} + b = -1.$$

Margina granice odluke je distanca izmedju dve hiperravni. Neka je \mathbf{x}_1 tacka na b_{i1} i \mathbf{x}_2 tacka na b_{i2} , onda dobijamo marginu d kao:

$$\begin{aligned} \mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2) &= 2 \\ \|\mathbf{w}\| \times d &= 2 \\ d &= \frac{2}{\|\mathbf{w}\|} \end{aligned}$$

Ucenje Linearnog SVM Modela

Učenje linearnog SVM modela se zasniva na procenjivanje parametara \mathbf{w} i b tako da su sledeća dva ograničenja zadovoljena za svaku od instanci za treniranje (\mathbf{x}_i, y_i) :

$$\begin{aligned}\mathbf{w} \cdot \mathbf{x}_i + b &\geq +1 \text{ ako } y_i = +1, \\ \mathbf{w} \cdot \mathbf{x}_i + b &\leq -1 \text{ ako } y_i = -1.\end{aligned}$$

Ova ograničenja obezbeđuju da će sve instance skupa za treniranje sa klasom $y = +1$ biti iznad hiperravni $\mathbf{w} \cdot \mathbf{x} + b = +1$, dok će sve instance skupa za treniranje sa klasom $k = -1$ biti ispod hiperravni $\mathbf{w} \cdot \mathbf{x} + b = -1$. Oba ova ograničenja možemo zapisati kao:

$$y_i(\mathbf{w} \cdot \mathbf{x} + b) \geq 1, \quad i = 1, 2, \dots, N.$$

SVM dodaje dodatna ograničenja koji obezbeđuju da će margina granice odluke biti maksimalna. To se svodi na minimizaciju sledeće funkcije:

$$f(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2}.$$

Definicija (Linearni SVM: Razdvojitiv slučaj). Zadatak učenja u SVM može se formulirati kao sledeći optimizacioni problem:

$$\min_w \frac{\|w\|^2}{2},$$

pod ograničenjima da je $\forall i: y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$.

Kako je funkcija za minimizaciju kvadratna i ograničenja su linearna po parametrima \mathbf{w} i b , ovaj optimizacioni problem se naziva **konveksan** optimizacioni problem, koji može da se reši metodom **Lagranzovih multiplikatora**.

Prvo, definisemo novu funkciju za optimizacioni problem koja se naziva **Lagranzova funkcija** kao:

$$L_P = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^N \lambda_i (y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1)$$

gde se parametri λ_i nazivaju **Lagranzovi mnozioci**. Ciljna funkcija se mora modifikovati zato što bi minimizacijom funkcije $\frac{\|\mathbf{w}\|^2}{2}$ dobili da je $\mathbf{w} = \mathbf{0}$, a to ne zadovoljava ograničenja $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$ za bilo koju vrednost parametra b .

Da bi minimizovali Lagranzovu funkciju L_P , prvo nalazimo parcijalne izvode od L_P u odnosu na \mathbf{w} i b i izjednačavamo ih sa nulom:

$$\begin{aligned}\frac{\partial L_P}{\partial \mathbf{w}} = 0 &\implies \mathbf{w} = \sum_{i=1}^N \lambda_i y_i \mathbf{x}_i; \\ \frac{\partial L_P}{\partial b} = 0 &\implies \sum_{i=1}^N \lambda_i y_i = 0.\end{aligned}$$

Kako su Lagranzovi mnozioci nepoznati, ne možemo da nadujemo \mathbf{w} i b . U slučaju ograničenja koji su jednakosti, jednostavno ih možemo dodati u sistem jednačina, ili ekvivalentno možemo dodati svaku od $\frac{\partial L_P}{\partial \lambda_i} = 0$ jednačina u dobijeni sistem. Tako možemo rešiti sistem od $N + 2$ jednačina sa $N + 2$ promenljive (i dobiti \mathbf{w} , b , $\lambda_1, \dots, \lambda_N$).

Jedan način rešavanja ovog problema sa ograničenja koji su nejednakosti je njihovo transformisanje u jednakosti. Ovo je moguće samo pod dodatnim ograničenjem da su Lagranzovi mnozioci nenegativni. Ograničanja postaju (Karush-Kuhn-Tucker —KKT):

$$\begin{aligned}\lambda_i &\geq 0, \\ \lambda_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] &= 0.\end{aligned}$$

Cini se da Lagranzovih mnozioca ima koliko i instanica za treniranje. Ali ispostavlja se da mnogi Lagranzovi mnozioci postaju nula, iz drugog ogranicenja, tj. vazi $\lambda_i = 0$ akko $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \neq 1$. Sve one instance, za koje vazi $\lambda_i > 0$, leze na hiperravnima b_{i1} ili b_{i2} i nazivaju se pomocni vektori. Sve ostale instance, za koje vazi $\lambda_i = 0$, ne leze na hiperravnima.

Resavanje ovih problema je dosta tesko je ukljucuju veliki broj parametara i jednacina, tacnije $N + 2$ njih (\mathbf{w} , b , $\lambda_1, \dots, \lambda_N$). Zbog toga ovaj problem stavimo na njegov dualni problem, koji ukljucuju samo Lagranzove mnoziocce. Zamenicemo $\mathbf{w} = \sum_i \lambda_i y_i \mathbf{x}_i$ i $\sum_i \lambda_i y_i = 0$ u L_P :

$$L_D = \sum_i \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

Dve kljucne razlike izmedju dva dualna problema L_P i L_D :

1. Dualna Lagranzova funkcija L_D sadrzi samo Lagranzove mnoziocce λ_i i instance skupa za treniranje (\mathbf{x}_i, y_i) , dok primarna Lagranzova funkcija L_P sadrzi Lagranzove mnoziocce λ_i , instance skupa za treniranje (\mathbf{x}_i, y_i) , kao i parametre za granizu odluke \mathbf{w} i b . Resenje oba optimizaciona problema su ekvivalentna.
2. Kvadratni clan u dualnoj Lagranzovoj funkciji L_D ima negativan znak ispred, sto znaci da je originalni problem minimizacije koji ukljucuje primarnu Lagranzovu funkciju L_P , postaje problem maksimizacije koji ukljucuje dualnu Lagranzovu funkciju L_D .

Dualni problem optimizacije moze se resiti numerickim tehnikama kao sto je kvadratno programiranje, ili nekim drugim tehnikama. Kada se nadje λ_i , dobijamo i parametare

$$\begin{aligned} \mathbf{w} &= \sum_i \lambda_i y_i \mathbf{x}_i, \\ b_i &= 1/y_i - \mathbf{w} \cdot \mathbf{x}_i, \\ b &= \overline{b_N} \end{aligned}$$

Linearni SVM: Nerazdvojni slucaj

Ako je skup za treniranje linearno razdvojni ukoliko se zanemare instance iz skupa koje dovode do nerazdvojjivih podataka, moguće je dobiti da je granica odluke B_1 sa vecom marginom, ima gresku nad trening skupom, dok granica odluke B_2 sa manjom marginom, nema gresku. Zbog toga B_1 se nikada nece dobiti prethodnom procedurom, te model nece lepo generalizovati. Da bi algoritam prepoznao B_1 kao potencionalnu ogranicu odluke, koristi se metod **mekih margina**. Sta vise ovaj metod dobro radi i za podatke koji nisu linearno razdvojni. Ovim pristupom dobijamo fleksibilnost izmedju broja pogresno klasifikovanih instanci iz skupa za treniranje i sirine margine.

Zbog toga moramo relaksirati ogranicenja kako bi se uklopila u linearno nerazdvojive podatke tako sto uvodimo dodatnu vrednost: **popustivu promenljivu** ($\xi > 0$). Ogranicenja optimizacionog problema, onda postaju:

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x}_i + b &\geq +1 - \xi_i \text{ ako } y_i = +1, \\ \mathbf{w} \cdot \mathbf{x}_i + b &\leq -1 + \xi_i \text{ ako } y_i = -1. \end{aligned}$$

Neka instanca \mathbf{x}_a krši ogranicenje $\mathbf{w} \cdot \mathbf{x}_a + b \leq -1$, onda ce $\mathbf{w} \cdot \mathbf{x} + b = -1 + \xi$ biti linija koja je paralelna sa granicom odlucivanja i prolazi kroz tacku \mathbf{x}_a . Odatle se dobija duzina izmedju te linije i hiperravni $\mathbf{w} \cdot \mathbf{x} + b = -1$:

$$\begin{aligned} \mathbf{w} \cdot (\mathbf{x} - \mathbf{x}_a) &= \xi, \\ \|\mathbf{w}\| \times d &= \xi, \\ d &= \frac{\xi}{\|\mathbf{w}\|} \end{aligned}$$

Zbog toga ξ pruza dobaru procenu greske granice odluke nad instancom \mathbf{x}_a .

Kako bi se izbegao ogranicen broj gresaka ciljna funkcija se defnise kao:

$$f(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2} + C \left(\sum_{i=1}^N \xi_i \right)^k,$$

gde su C i k korisnici definisani parametri koji predstavljaju kaznu pogresno klasifikovanih instanci iz trening skupa. Pretpostavimo da je $k = 1$, a da se parametar C moze dobiti na osnovu performansa modela na validacionom skupu.

Dobijamo Lagranzovu funkciju kao:

$$L_P = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \lambda_i (y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i) - \sum_{i=1}^N \mu_i \xi_i,$$

sa sledecim KKT ogranicenjima:

$$\begin{aligned} \xi_i &\geq 0, \quad \lambda_i \geq 0, \quad \mu_i \geq 0, \\ \lambda_i (y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i) &= 0, \\ \mu_i \xi_i &= 0. \end{aligned}$$

Odatle nalazimo prvi parcijalni izvod od L_P u odnosu na \mathbf{w} , b , i ξ_i .

$$\begin{aligned} \frac{\partial L_P}{\partial w_j} &= w_j - \sum_i \lambda_i y_i x_{ij} = 0 \implies w_j = \sum_i \lambda_i y_i x_{ij}, \\ \frac{\partial L_P}{\partial b} &= - \sum_i \lambda_i y_i = 0 \implies \sum_i \lambda_i y_i = 0 \\ \frac{\partial L_P}{\partial \xi_i} &= C - \lambda_i - \mu_i = 0 \implies \lambda_i + \mu_i = C. \end{aligned}$$

Odatle problem optimizacije primarne Lagranzove funkcije mozemo da svedemo na problem optimizacije dualne Lagranzove funkcije:

$$\begin{aligned} L_D &= \frac{1}{2} \sum_{ij} \lambda_i \lambda_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\ &\quad + C \sum_i \xi_i \\ &\quad - \sum_i \lambda_i (y_i (\sum_j \lambda_j y_j (\mathbf{x}_i \cdot \mathbf{x}_j) + b) - 1 + \xi_i) \\ &\quad - \sum_i (C - \lambda_i) \xi_i \\ &= \sum_i \lambda_i - \frac{1}{2} \sum_{ij} \lambda_i \lambda_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j). \end{aligned}$$

Nelinearni SVM

Do sada smo posmatrali samo skupove podataka kod kojih postoji linearna granica odluke. Kod skupova sa nelinearnom granicom odluke koristimo sledeci trik: Transformisemo podatke iz originalnog koordinatnog prostora \mathbf{x} u novi prostor $\Phi(\mathbf{x})$ za koji postoji linearna granica odluke.

Transformacija Atributa

Da bi pokazali kako se transformacijom atributa dobijaju linearno razdvojivi podaci, pretpostavimo da imamo binarni skup podataka za treniranje, gde su sve instance sa labelom $y = +1$ van kruga \mathcal{K} , dok su sve instance sa labelom $y = -1$ unutar kruga \mathcal{K} . Krug $\mathcal{K} : (x_1 - 0.5)^2 + (x_2 - 0.5)^2 = 0.2^2$.

Mozemo napraviti model sledecom jednačinom:

$$y(x_1, x_2) = \begin{cases} +1 & , (x_1 - 0.5)^2 + (x_2 - 0.5)^2 > 0.2^2, \\ -1 & , (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.2^2, \end{cases}$$

a granica odluke ce u ovom slucaju biti krug \mathcal{K} , tj.

$$x_1^2 - x_1 + x_2^2 - x_2 = -0.46.$$

Nelinearna transformacija Φ treba da preslikava iz originalnog prostora atributa u novi prostor gde ce granica odluke biti linearna. Jedna takva transformacija je:

$$\Phi : (x_1, x_2) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1).$$

U transformisanom prostoru mozemo naci parametar $\mathbf{w} = (w_0, w_1, w_2, w_3, w_4)$ tako da granica odluke postaje:

$$w_4x_1^2 + w_3x_2^2 + w_2\sqrt{2}x_1 + w_1\sqrt{2}x_2 + w_0 = 0.$$

Jedan potencijalni problem ovog pristupa je to sto se dimenzionalnost povecava drasticno sa porastom dimenzija originalnog prostora. Kernel trik resava ovaj problem.

Ucenje Nelinearnog SVM Modela

Tehnika transformisanja atributa ima nekoliko problema. Prvi je to sto je veoma tesko odrediti optimalnu transformaciju Φ . Drugi je to sto resavanje optimizacionog problema sa velikom dimenzijom je izracunljivo tezak zadatak.

Pretpostavimo da postoji odgovarajuca funkcija Φ . Nakon transformisanja takodje treba odrediti linearnu granicu odluke, koja je sledeceg oblika: $\mathbf{w} \cdot \Phi(\mathbf{x}) + b = 0$.

Definicija (Nelinearni SVM). Zadatak ucenja nelinearnog SVM je optimizacioni problem:

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2}$$

pod ogranicenjem da je $\forall i : y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b) \geq 1$.

Slicnim postupkom, kao kod linearnog SVM, dobija se dualna Lagranzova funkcija:

$$L_D = \sum_i \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)).$$

Kada pronadjemo λ_i nekom tehnikom, parametre \mathbf{w} i b mozemo da izvedemo iz:

$$\begin{aligned} \mathbf{w} &= \sum_i \lambda_i y_i \Phi(\mathbf{x}_i) \\ \lambda_i \left[y_i \left(\sum_i \lambda_i y_i (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_i)) + b \right) - 1 \right] &= 0 \end{aligned}$$

Konacno za klasifikaciju instance testiranja \mathbf{z} koristimo:

$$\begin{aligned} f(\mathbf{z}) &= \text{sgn}(\mathbf{w} \cdot \Phi(\mathbf{z}) + b) \\ &= \text{sgn} \left(\sum_i \lambda_i y_i (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{z})) + b \right). \end{aligned}$$

Kernel trik

Skalarni proizvod $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ mozemo posmatrati kao meru slicnosti izmedju dve instance, \mathbf{x}_i i \mathbf{x}_j u transformisanom prostoru. **Kernel trik** je moted koji se koristi za meru slucnosti izmedju dve instance u transformisanom prostoru koristeći skup atributa. Ovim postupkom resavamo problem prokletstva dimenzionalnosti.

Razmotrimo funkciju Φ koju smo gore koristili:

$$\begin{aligned}\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) &= (u_1^2, u_2^2, \sqrt{2}u_1, \sqrt{2}u_2, 1) \cdot (v_1^2, v_2^2, \sqrt{2}v_1, \sqrt{2}v_2, 1) \\ &= u_1^2 v_1^2 + u_2^2 v_2^2 + 2u_1 v_1 + 2u_2 v_2 + 1 \\ &= (\mathbf{u} \cdot \mathbf{v} + 1)^2.\end{aligned}$$

Ova formula pokazuje da se skalarni proizvod u transformisanom prostoru moze prikazati u terminima skalarnog proizvoda u originalnom prostoru. Ova funkcija naziva se **kernel funkcija** i definisana je kao:

$$K(\mathbf{u}, \mathbf{v}) = \Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^2$$

Ovime dobijamo da nije potrebno znati preciznu definiciju preslikavanja Φ , jer kernel funkcija mora da zadovoljava **Marcerovu teoremu**. Ona kaze da se kernel funkcija moze izraziti preko skalarnog proizvoda izmedju dva ulazna vektora u nekom vise-dimenzionom prostoru. Taj transformisani prostor se naziva **reproduktivni kernel Hilbert prostor**.

Instanca iz skupa za testiranje \mathbf{z} se klasifikuje kao:

$$\begin{aligned}f(\mathbf{z}) &= \text{sgn}\left(\sum_i \lambda_i y_i (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{z})) + b\right) \\ &= \text{sgn}\left(\sum_i \lambda_i y_i K(\mathbf{x}_i, \mathbf{z}) + b\right) \\ &= \text{sgn}\left(\sum_i \lambda_i y_i (\mathbf{x}_i \cdot \mathbf{z} + 1)^2 + b\right).\end{aligned}$$

Teorema (Mercerova Teorema). *Kernel funkcija K se moze izraziti kao*

$$K(\mathbf{u}, \mathbf{v}) = \Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}).$$

akko, za bilo koju funkciju $g(\mathbf{x})$ takvu da je $\int g(\mathbf{x})^2 d\mathbf{x}$ konacno, onda

$$\int K(\mathbf{x}, \mathbf{y}) g(\mathbf{x}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0.$$

Neke kernel funkcije koje zadovoljavaju Marcerovu teoremu:

$$\begin{aligned}K(\mathbf{x}, \mathbf{y}) &= (\mathbf{x} \cdot \mathbf{y} + 1)^p \\ K(\mathbf{x}, \mathbf{y}) &= e^{-\|\mathbf{x} - \mathbf{y}\|^2 / (2\sigma^2)} \\ K(\mathbf{x}, \mathbf{y}) &= \tanh(k\mathbf{x} \cdot \mathbf{y} - \delta)\end{aligned}$$

Karakteristike SVM

1. Problem ucenja SVM modela moze da se formulise kao konveksan optimizacioni problem, u kome su dostupni efikasni algoritmi za nalazenje globalnog minimuma ciljne funkcije. Dok drugi klasifikacioni modeli kao sto su klasifikatori zasnovani na pravilima i neuronske mreze, koriste gramzicu strategiju za pretragu prostora, pa se vrlo cesto zaglavljaju u loklanim minimumima.
2. SVM regulise kapacitet maksimizovanjem margini granice odluke. Ostale parametre kao sto je tip kernel funkcije i funkcija cene C definise korisnik.
3. SVM se moze koristiti nad kategorickim podacima tako sto se prave glupe promeljive za svaku kategoriju.
4. Postoje neke metode za prosirivanje SVM na viseklasne probleme.

Metod Ansambla

Problem: Nebalansiranih Klasnih Instanci

Skupovi podataka sa nebalansiranom distribucijom klasa su veoma cesti u mnogim primenama. Na primer, broj neispravnih masina tokom neke produkcije je znacajno manji od ispravnih masina. Isto tako, broj neovlasenog koriscenja kreditne kartice je veoma manji od broja ovlasenih koriscenja. Bez obzira na njihovo retko pojavljivanje, korektno klasifikovanje ovih instanci je veoma vazno. Medjutim, kako ima vise ovih koji nisu od znacaja dobijamo da je ocena dosta velika.

Na primer, ako ima 1% neovlasenog koriscenja kreditne kartice, a nas model predvidja sve kao ovlaseno koriscenje kreditne kartice, ima da je preciznost modela 99%, je je veoma visoka ocena, ali model je neupotrebljiv.

Alternativne Metrike

Kod osnovnih metrika, sve klase su jednake, tj. imaju jednake tezine. Za binarnu klasifikaciju, retka klasa se obicno oznacava kao pozitivna klasa +, dok se gusta klasa oznacava kao negativna klasa -. Na osnovu modela, dobijamo sledecu matricu konfuzije:

	+	-
+	f_{++} (TP)	f_{+-} (FN)
-	f_{-+} (FP)	f_{--} (TN)

- TP(True Positive) f_{++} ima vrednost koji odgovara broju pozitivnih instanci koje je model predvideo da su pozitivne.
- FN(False Negative) f_{+-} ima vrednost koja odgovara broju pozitivnih instanci koje je model predvideo da su negativne.
- FP(False Positive) f_{-+} ima vrednost koja odgovara broju negativnih instanci koje je model predvideo da su pozitivne
- TN(True Negative) f_{--} ima vrednost koja odgovara broju negativnih instanci koje je model predvideo da su negativne.

Definicija: True Positive Rate ili **senzibilitet** je definisan kao odnos izmedju broja korektno klasifikovanih pozitivnih instanci, od ukupnog broja pozitivnih instanci, tj.

$$TPR = \frac{TP}{TP + FN}.$$

Definicija: True Negative Rate ili **specificitet** je definisan kao odnos izmedju broja korektno klasifikovanih negativnih instanci, od ukupnog broja negativnih instanci, tj.

$$TNR = \frac{TN}{TN + FP}.$$

Definicija: False Positive Rate je definisan kao odnos izmedju broja pogresno klasifikovanih pozitivnih instanci, od ukupnog broja negativnih instanci, tj.

$$FPR = \frac{FP}{TN + FP}.$$

Definicija: False Negative Rate je definisan kao odnos izmedju broja pogresno klasifikovanih negativnih instanci, od ukupnog broja pozitivnih instanci, tj.

$$FNR = \frac{FN}{TP + FN}.$$

Definicija: Recall i Preciznost.

$$p = \frac{TP}{TP + FP}$$

$$r = \frac{TP}{TP + FN}$$

Preciznost odredjuje odnos izmedju korektno pozitivno klasifikovanih instanci i ukupnog broja pozitivno klasifikovanih instanci. Klasifikatori sa malom preciznoscu rezultuju u mnogo pogresno klasifikovanih negativnih instanci. Recall je odnos izmedju korektno pozitivno klasifikovanih instanci i ukupnog broja pozitivnih instanci. Klasifikatori sa velikim recall imaju malo pogresno klasifikovanih pozitivnih.

Kljucna stvar u pravljenju modela je maksimizovanje i preciznosti i recolla, jer povecanjem jednog smanjuje se drugo. Zbog toga ove dve metrike mozemo spojiti u F_1 metriku koja sadrzi obe:

$$F_1 = \frac{2}{\frac{1}{r} + \frac{1}{p}} = \frac{2rp}{r + p}$$

Receiver Operating Characteristic Krive

Receiver Operating Characteristic (ROC) krive su graficki prikaz izmedju true positive rate i false positive rate nekog klasifikatora. Svaka tacka na krivi odgovara jednom modelu koji je indukovao klasifikatorima.

Postoji nekoliko interpretacija:

- (TPR=0, FPR=0): Model predvidja svaku instancu kao negativne.
- (TPR=1, FPR=1): Model predvidja svaku instancu kao pozitivne.
- (TPR=1, FPR=0): Idealan model.

Dobar klasifikacioni model treba biti sto blizi gornjem desnom uglu, dok je model sa nasumicnim predvidjanjem blizak diagonalu. Model nasumicnog predvidjanja klasifikuje instancu kao pozitivnu klasu sa nekom fiksiranom verovatnocom p . Na primer, razmotrimo skup podataka koji sadrzi n_+ pozitivnih instanci i n_- negativnih instanci. Zbog toga, imamo $TPR = (pn_+)/n_+$, dok je $FPR = (pn_-)/p = n_-$. Kako su TPR i FPR identicni, ROC kriva za nasumicni klasifikator lezi na glavnoj diagonalu.

Dobra mera kvaliteta izmedju dve ROC krive je površina ispod ROC krive (AUC). Idealan model ima površinu 1, dok nasumicni klasifikator ima površinu 0.5.

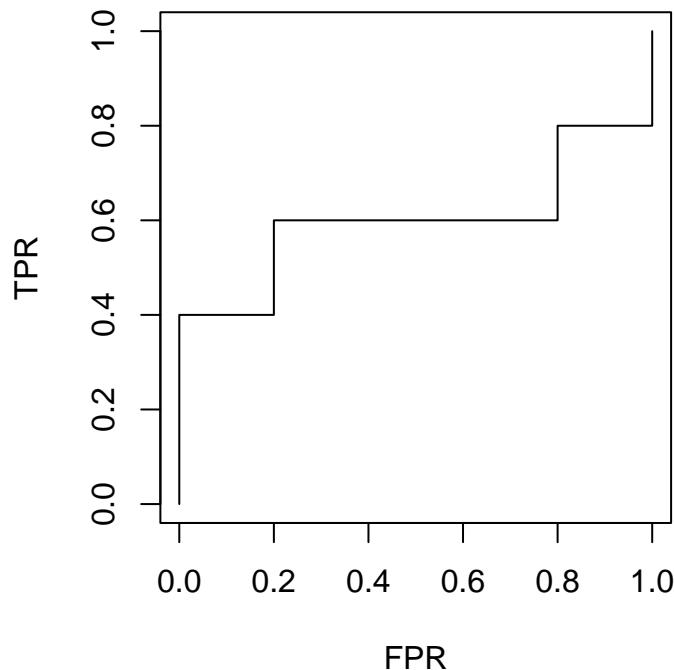
Generisanje ROC krive

1. Predpostavimo da su neprekidne-vrednosti definisani za instance pozitivne klase, sortiramo test instance u rastucem poretku po njihovim izlazima.
2. Biramo najnize rangiranu test instancu. Dodeljujemo je svim instancama koje su iznad nje kao pozitivne. Ovo je ekvivalentno sa tim da smo klasifikovali sve test instance kao pozitivne klase. Zato sto su sve pozitivne klase klasifikovane tacno, a sve negativne klase klasifikovane netacno, imamo da je $TPR = FPR = 1$.
3. Izaberi sledecu test instancu iz sortirane liste. Klasifikujemo izabranu instancu i sve iznad kao pozitivni, dok sve ispod kao negativne. Azuriramo broj TP i FP . Ako je prethodno odabrana instanca pozitivna, onda dekrementiramo TP i FP ostaje isti kao pre, dok ako je prethodno odabrana instanca negativna, onda dekrementiramo FP i TP ostaje isti.
4. Ponavljamo Korak 3 i azuriramo TP i FP sve do poslednje test instance.
5. Plotujemo rezultat klasifikatora.

Primer:

Klasa	+	-	+	-	-	-	+	-	+	+	
	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0

Klasa	+	-	+	-	-	-	+	-	+	+	
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0



Učenje: Cena-Senzibilitet

Matrica cena kaznjava klasifikovanje instanci jedne klase više nego druge. Neka je $C(i, j)$ cena predviđanja instance iz klase i kao klase j . Sa ovim notacijom, $C(+, -)$ je cena pravljenja false negative greske, dok je $C(-, +)$ cena generisanja pogresne uzbune. Za kolekciju od N instanci, ukupna cena modela M je

$$C_t(M) = TP \times C(+, +) + FP \times C(-, +) + FN \times C(+, -) + TN \times C(-, -)$$

Za jedinicnu matricu cene, tj. $C(+, +) = C(-, -) = 0$ i $C(+, -) = C(-, +) = 1$ imamo da je ukupna cena modela ekvivalentna broju gresaka:

$$C_t(M) = 0 \times (TP + TN) + 1 \times (FP + FN) = N \times Err,$$

gde je Err stopa greske klasifikatora.

Kod drveta odlucivanja infomaciju o cenama mozemo koristiti na vise nacina:

1. Biranje najboljeg atributa za razdvajanje podataka
2. Odluka o potresivanju drveta
3. Svaka instanca ima odredjenu cenu, pa algoritam konvergira u drvo odlucivanja sa najmanjom ukupnom cenom
4. Modifikacija pravila odlucivanja svakog lista.

Objasnimo nacin (4): Neka je $p(i|t)$ deo trening instanci iz klase i za klasifikacioni problem koji dodeljuje

pozitivnu klasu cvoru t ako vazi sledece:

$$\begin{aligned} p(+|t) &> p(-|t) \\ \implies p(+|t) &> 1 - p(+|t) \\ \implies 2p(+|t) &> 1 \\ \implies p(+|t) &> 0.5 \end{aligned}$$

Ovo sugerise na to da ce postupak odlucivanja biti na osnovu glasa vecine, tj. listu dodeljujemo klasu na osnovu vecine instanci koje se u njemu nalaze.

Kod cena-senzibilitet algoritma klasna oznaka i se dodeljuje cvoru t ako minimizuje sledeci izraz:

$$C(i|t) = \sum_j p(j|t)C(j, i).$$

Ako je $C(+, +) = C(-, -) = 0$, onda se listu t dodeljuje pozitivna klasa ako:

$$\begin{aligned} p(+|t)C(+, -) &> p(-|t)C(-, +) \\ \implies p(+|t)C(+, -) &> (1 - p(+|t))C(-, +) \\ \implies p(+|t)C(+, -) &> C(-, +) - p(+|t)C(-, +) \\ \implies p(+|t)(C(+, -) + C(-, +)) &> C(-, +) \\ \implies p(+|t) &> \frac{C(-, +)}{C(+, -) + C(-, +)} \end{aligned}$$

Ovo sugerise na to da ce postupak odlucivanja zavisiti od $C(-, +)/(C(+, -) + C(-, +))$, tj. ako je $C(-, +) < C(+, -)$, onda ce granica biti manja od 0.5.

Postupak Uzorkovanja

Ideja uzorkovanja je modifikacija distribucije instanci tako da retke klase dobro reprezentuju skup za treniranje. Neko od tehnika su poduzorkovanje, naduzorkovanje, i hibrid oba pristupa. Neka skup podataka ima 100 pozitivnih i 1000 negativnih instanci.

Kod poduzorkovanja, bira se nasumicni uzorak od 100 negativnih instanci i svih 100 pozitivnih instanci. Jedan potencionalni problem ovog uzorkovanja je to da 100 odabranih instanci ne reprezentuju dobru celu populaciju pozitivnih instanci. Jedan nacin da se ovo resi je pravljenje vise modela sa razlicitim uzorcima negativnih instanci, slicno kao kod metoda ansambla. Takodje mogu se koristiti dodatna znanja o negativnim instancama za fokusirano poduzorkovanje.

Naduzorkovanje pravi nove pozitivne instance sve dok ih nema jednako mnogo kao i negativnih instanci. Jedan potencionalni problem ovog uzorkovanje je to da, kod podataka sa sumom, moze doci do preteniranja.

Hibridan pristup koristi kombinaciju poduzorkovanja za mnogobrojne klase i naduzorkovanja za retke klase. Ovim pristupom dobijamo uniformnu distribuciju klasa. Takodje, i kod ovog pristupa moze doci do preteniranja.

Problem: Viseklasnih Skupova Podataka

Prosirimo binarne klasifikatore na klasifikatore koji mogu da se nose sa viseklasnim skupovima podataka. Neka je $Y = y_1, y_2, \dots, y_K$ skup klasa ulaznih podataka.

Prvi pristup dekomponovanja viseklasnih problema u K binarnih problema. Za svaku klasu $y_i \in Y$, pravimo binarni problem gde su sve instance koje pripadaju klasi y_i pozitivne instance, dok su ostale instance negativne. Onda pravimo binarni klasifikator za odvajanje instanca klase y_i od ostalih klasa. Ovaj pristup je poznat kao *jedan-protiv-svih*.

Drugi pristup, poznat kao *jedan-protiv-jednog*, konstruise $K(K - 1)/2$ binarnih klasifikatora, gde se svaki klasifikator koristi za odavajanje izmedju para klasa, (y_i, y_j) . Instance koje na pripadaju ni jednoj od y_i i y_j se ignorisu tokom pravljenja klasifikatora za (y_i, y_j) .

U oba pristupa test instance se klasifikuju kombinacijom predviđanja binarnih klasifikatora. Koristi se shema glasanja za kombinovanje predviđanja, u kome klasa koja ima najveći broj predviđanja povedjuje u glasanju. U pristupu jedan-protiv-svih, ako se instanca klasifikuje kao negativna, onda sve klase sem pozitivne klase dobijaju glas. Ovaj pristup može da dovede do izjednačenih glasova. Drugi pristup je transformisanje izlaza binarnih klasifikatora u verovatnoće na osnovu kojih se test instanci dodeljuje klasa sa najvećom verovatnoćom.

Kodiranje Izlaza Pomocu Korekcije Gresaka

Potencionalni problem prethodna dva pristupa je senzitivnost greske binarnih klasifikatora. Kodiranje izlaza pomocu korekcije gresaka je metod koji pruza robustniji nacin za resavanja viseklasnih problema. Ovaj metod je inspirisan informacionim pristupom slanja poruke preko kanala sa velikim sumom. Zasniva se na dodavanju redundantnosti za otkrivanje greske tokom prenosjenja poruke, pa cak i njeno obnavljanje.

Za viseklasno ucenje, svaka klasa y_i se predstavlja kao jedinstveni string bitova duzine n i naziva se kodirana rec. Onda treniramo n binarnih klasifikatora da predvide svaki bit kodirane reci. Za predviđanje klase test instanca koristimo onu klasu cije je Hamingovo rastojanje kodirane reci najmanje u odnosu na predikciju. (Hamingovo rastojanje je broj parova bitova koji se razlikuju)

Klaster Analiza: Osnovni Koncepti i Algoritmi

Klaster analiza deli podatke u grupe (klaster) koji su znacajni, korisni, ili oba. Znacajni klasteri hvataju prirodnu strukturu podataka. Korisni klasteri se koriste za neke druge svrhe.

Klasterovanje radi razumevanja

Klase, ili konceptualno znacajne grupe objekata koje dele zajednicke karakteristike, imaju vaznu ulogu u razumevanju i analizi sveta. Ljudi imaju vestinu da podele objekte u grupe (klasterovanje) i da im pridruzuju odredjene grupe (klasifikacija).

- **Biologija.** Biologici provode mnogo vremena na taksonomiji (hijerarhijsko klasterovanje) svih zivih bica. Zbog toga se i razvila matematicka taksonomija koja može automacki pronaci takve klasifikacione strukture. Isto tako, klaster analiza se koristi u pronalazenju grupa gena koje imaju slicne funkcije.
- **Pronalazenje informacija.** Web sadrzi bilione stranice, i rezultati svakog upita mogu vratiti hiljade stranica. Klasterovanje se onda može koristiti za grupisanje rezultata u mali broj klastera, od kojih svaki sadrzi odredjeni aspekt upita.
- **Klima.** Razumevanje klime podrazumeva nalazenje sablona u atmosferi i okeanima. Klaster analiza se koristi za pronalazenje tih sablona.
- **Psihologija i medicina.** Neka bolest može imati brojne varijacije, pa se klaster analiza koristi za pronalazenje tih razlicitih podkategorija.
- **Biznis.** Biznisi prikupljaju veliku kolicinu informacija od trenutnih i potencijalnih potrosaca. Klaster analiza se koristi za grupisanje potrosaca za dalju analizu.

Klasterovanje radi korisnosti

Klasterovanje predstavlja abstrakciju individualne instance podataka kao klaster kome ta instanca pripada. Takodje, klaster analiza se koristi i za pronalazenje reprezentativnog klaster prototipa.

- **Sumarizacija.** Mnoge tehnike za analizu podataka, kao sto je regresija ili PCA, imaju kvadratnu vremensku ili prostornu kompleksnost, pa nisu prikladni za velike skupove podataka. Medjutim, umesto primenjivanja algoritma na celokupan skup podataka, mozemo ga primeniti na redukovani skup podataka koji sadrzi samo klaster prototipe.

- **Kompresija.** Klaster prototipi se mogu koristiti za kompresiju podataka. Pravi se tabela koja sadrži samo prototipe za svaki klaster, tj. svakom prototipu se dodeljuje celobrojna vrednost koja predstavlja indeks u tabeli. Svaki objekat je predstavljen kao indeks prototipa kojem je pridružen klaster. Ovaj tip kompresije se zove **vektorska kvantizacija** i obično se koristi za slike, zvuk, video, gde su (1) mnogi objekti podataka slični jedni drugima, (2) gubitak informacija je prihvatljiv, i (3) smanjenje veličine podataka je poželjno.
- **Efikanso pronalazenje najblizeg suseda.** Umesto da računamo udaljenost između svakake dve tačke, koristimo klaster analizu za pronalazenje njihovih prototipa. Ako su dva klaster prototipa jako udaljena jedan od drugog, nema potreba računati udaljenost objekata unutar odgovarajućih klastera kako oni sigurno neće biti najbliži susedi.

Pregled

Sta je Klaster Analiza?

Klaster analiza grupise objekte na osnovu informacija pronađenim u podacima koji opisuju objekte i njihove relacije. Cilj klaster analize je da su objekti unutar jedne grupe slični (ili povezani) jedan drugome i različiti (ili nepovezani) objektima drugih grupa.

Pojam klaster nije dobro definisan. Za bolje razumevanje šta predstavlja jedan klaster razmotrimo primer u kome možemo grupisati podatke na 2, 4, i 6 grupa. Zbog toga definicija klastera zavisi od samog skupa podataka.

Klasifikacija spada u **nadgledano učenje**, tj. novim, neklasifikovanim objektima dodeljujemo klasne promenljive, na osnovu objekata čije klase znamo. Iz tog razloga, klaster analiza spada u **nenadgledano učenje**.

Termini kao što su **segmentacija** i **particionisanje** se nekada koriste kao sinonimi za klasterovanje, ali su van tradicionalnih tehnika klaster analize.

Razni Tipovi Klasterovanja

Parcijalno klasterovanje podrazumeva podelu skupa podataka na disjunktne podskupove (klastera) tako da svaki objekat skupa podataka pripada tačno jednom podskupu.

Hijerarhijski klasterovanje je skup ugnjezdenih klastera koji mogu da se urede kao drvo. Svaki cvor (klaster) u drvetu (osim listova) je unija svoje dece (podklastera), i cvor drveta je klaster koji sadrži sve objekte.

Ekskluzivno klasterovanje podrazumeva da se svakom objektu podataka dodeljuje jedan klaster.

Neekskluzivno klasterovanje se koristi kada neki objekat može *istovremeno* da pripada u više od jedne grupe (klase).

Rasplanato klasterovanje svakom objektu dodeljuje svaki klaster sa tezinom pripadnosti između 0 (apsolutno ne pripada) i 1 (apsolutno pripada). Drugim rečima, klasteri se tretiraju kao rasplanati skupovi.

Kompletno klasterovanje dodeljuje svakom objektu klaster, dok **parcijalno klasterovanje** ne. Motivacija za parcijalno klasterovanje leži u tome da neki objekti nemaju dobro definisanu grupu kojoj pripadaju. Mnogo puta takvi objekti predstavljaju sum, ili outlajere.

Razni Tipovi Klastera

Dobro-Razdvojeni

Klaster je skup objekata, gde su svi objekti blizu (ili sličniji) svim drugim objektima unutar tog klastera nego bilo kog drugog. Nekada se koristi granica koja ograničava sve objekte unutar jedne klase. Ovako idealna definicija klastera zadovoljiva je samo kada podaci imaju prirodne klastera koji su daleko jedni od drugih.

Klasteri Zasnovani na Prototipu

Klaster je skup objekata u kome je svaki objekta blizi (ili slicniji) prototipu koji jedinise taj klaster nego prototipu bili kog drugog klastera. Za podatke sa neprekidnim atributima, prototip klastera je obicno centroid, tj. sredina svih tacaka u klasteru. Za podatke sa kategorickim atributima, prototip je obicno medoid, tj. najvisje zastupljena tacka klastera. Cesto se za ovakve klaster kaze jos i da su **zasnovani na centru**.

Klasteri Zasnovani na Grafu

Ako su podaci predstavljeni kao graf, gde su cvorovi objekti, a grane predstavljaju veze imezju njih, onda klaster definisemo kao **komponentu povezanosti**, tj. grupa objekata koji su medjusobno povezani, ali nemaju granu sa objektima izvan grupe. Jedan vazan primer klastera zasnovanog na grafu je **klaster zasnovan na susedstvu**, gde su dva objekta povezana samo ako su unutar odredjene udaljenosti jedni od drugih. Ovo znaci da je svaki objekat unutar klastera blizi svim objektima unutar klastera, nego bilo kom objektu van klastera.

Klasteri Zasnovani na Gustini

Klaster je gusti region objekata koji je okruzen sa regionim sa malom gustom.

Zajednicke-Osobine (Konceptualni Klasteri)

Klaster je skup objekata koji dele neku osobinu.

K-sredina

Klasterovanje zasnovana na prototipu pravi jedno particionisanje objekata podataka. Tehnike za resavanje ovog problema su K-sredina i K-medoida. K-sredina definise prototip u terminima centroida, sto je obicno sredina grupe tacaka, i tipicno se primenjuje na objekte u neprekidnom n -dimenzionalnom prostoru. K-medoid definise prototip u terminima medoida, sto je najreprezentativnija tacka unutar neke grupe tacaka, i moze se primeniti na oblasti koje zahtevaju samo aproksimacionu meru parova objekata. Centroid ne mora biti ni jedna tacka podataka, dok medoid po definiciji mora.

Osnovni Algoritam K-sredina

```
def k_means(X, K):
    centroids = sample(X, K) # biramo K inicijalni centroida
    new_centroids = []
    while new_centroids != centroids:
        clusters = form_clusters(centroids, X)
        new_centroids = recompute_centroids(clusters)
    return centroids
```

Za neku kombinaciju meru blizina i tipova centroida, K-sredina uvek konvergira, tj. K-sredina dostize stanje u kome se ni jedna tacka ne prebacuje iz jednog klastera u drugi, samim tim centroidi ostaju nepromenjeni.

Dodeljivanje Tacaka Najblizem Centroidu

Da bi dodeli tacku najblizem centroidu, potrebna nam je mera blizine koja moze da kvantifikuje 'najblizem centroidu'. Euklidsko (L_2) rastojanje se obicno koristi za podatke u Euklidskom prostoru, dok se kosinusna slicnost koristi za dokumente. Mozemo koristiti i Manhetn (L_1) rastojanje za Euklidske podatke, i Jakardovu meru za dokumente.

Podaci u Euklidskom Prostoru

Razmotrimo podatke cija je mera blizine Euklidsko rastojanje. Za meru kvaliteta klasterovanja mozemo koristiti **sumu kvadratnih gresaka (SSE)**:

$$SSE = \sum_{i=1}^K \sum_{\mathbf{x} \in C_i} dist(\mathbf{c}_i, \mathbf{x})^2$$

Pod ovim pretpostavkama, može se pokazati da je centroid koji minimizuje SSE sredina klastera. Pa se centroid i -tog klastera definiše kao:

$$\mathbf{c}_i = \frac{1}{m_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

Na primer, centroid za tačke $(1, 1)$, $(2, 3)$, i $(6, 2)$, je $((1 + 2 + 6)/3, (1 + 3 + 2)/3) = (3, 2)$.

Dokumenta

Razmotrimo dokumenta sa kosinusnom merom sličnosti. Hoćemo da maksimizujemo sličnosti dokumenata unutar jednog klastera sa centroidom tog klastera. Ovo se naziva **kohezija** klastera.

$$\text{Totalna Kohezija} = \sum_{i=1}^K \sum_{\mathbf{x} \in C_i} \cos(\mathbf{x}, \mathbf{c}_i)$$

Globalni slučaj

Funkcija Blizine	Centroid	Ciljna funkcija
Manhetn (L_1)	medijana	minimizovati sumu L_1 rastojanja od objekta do njegovog klaster centroida
Kvadratni Euklid (L_2^2)	sredina	minimizovati sumu L_2 rastojanja od objekta do njegovog klaster centroida
Cosinus	sredina	maksimizovati sumu kosinusne sličnosti objekta sa njegovim klaster centroidom
Bergmanova divergencija	sredina	minimizovati sumu Bergmanove divergencije objekta sa njegovim klaster centroidom

Biranje inicijalnih centroida

Kada nasumično izaberemo centroide, svako pokretanje $k_sredina$ tipično pravi različite SSE e. Zbog toga biranje inicijalnih centroida je ključan korak u osnovnoj proceduri K -sredina.

Primer (Losi inicijalni centroidi). Ako odaberemo loše inicijalne centroide zaglavljujemo se lako u nekom od lokalnog minimuma ciljne funkcije SSE .

Primer (Granice nasumične inicijalizacije). Jedna tehnika koja se često koristi za problem inicijalnih centroida je da se procedura K -sredina ponovi nekoliko puta sa različitim inicijalnim centroidima. Na osnovu dobijenih vrednosti za svaki od multiskupa centroida biramo onaj skup centroida čiji klasteri proizvode najmanje SSE . Ovaj pristup iako zvuči dobro u praksi se ne pokazuje dobro sa svaku strukturu skupa podataka.

Jos jedan pristup inicijalizacije centroida je uzorkovanje tacaka koje klasterujemo pomoću neke hijerarhijske tehnike klasterovanja. K klastera izdvajamo iz hijerarhijskog klasterovanja, i centroidi tih klastera koristimo kao inicijalne centroide. Ovaj pristup radi samo ako vazi (1) uzorak je relativno mali, i (2) K je relativno malo u odnosu na veličinu uzorka.

Jos jedan pristup inicijalizacije centroida je korak-po-korak, tj. inicijalizujemo početni centroid nasumično ili kao centroid svih tacaka. Onda svaki sledeći centroid biramo kao njegovu najudaljeniju tacku. Ovaj metod imam mana ako podaci imaju autlajere, koje ne želimo da predstavljamo kao poseban klaster.

Vremenska i Prostorna Složenost

Prostorna kompleksnost nije toliko zahtevna zato što čuvamo samo objekte i centroide. Tacnije, potreban prostor nije veći od $O((m + K) * n)$, gde je m broj objekata, K broj klastera i n broj atributa. Vreme potrebno za K -sredina je linearno po broju objekata. Tacnije, vreme potrebno je $O(I * K * m * n)$, gde je I broj iteracija potrebnih za konvergenciju (vrednost I je obično malo i može se sigurno ograničiti).

K-sredina: Dodatni Problemi

Obradjivanje Praznih Klastera

Jedan problem sa osnovnim K-sredina algoritmom je dobijanje praznih klastera. Zbog toga je potrebna strategija za zamenu centroida, kako ce SSE biti veca nego sto je potrebno. Jedan pristup da se uzme najdalja tacka od svih trenutnih centroida. Dok je drugi pristup uzimanje centroida sa najvećim SSE , sto ce ga podeliti na dva dela. U oba slucaja garantujemo smanjenje SSE .

Autlajeri

Autjaleri ne pripadaju ni jednom klasteru pa ce biti udaljeniji od svih klastera sto proizvodi vekili SSE . Zbog ovoga, korisno je otkriti i eliminisati sve autlajere pre pozivanja procedure K-sredina.

Smanjenje SSE sa Postprocesiranjem

Jedno od ociglednih resenja za smanjivanje SSE je pronalazenje vise klastera, tj. koristimo vecu vrednost hiperparametra K . Medjutim, u mnogim slucajevima, zelimo da unapredimo SSE , ali ne zelimo da povecemo broj klastera.

Postoje dve strategije za smanjenje ukupnog SSE povecavanjem broja klastera:

- **Razdvajanje klastera:** Klaster sa najvećim SSE je obicno izabran za razdvajanje, ali moze biti i neki klaster sa najvećim standardnim odstupanjem.
- **Uvodjenje novog klastera centroida:** Obicno tacka koja je najdalja od svih klastera je izabrana za novi centroid. Mozemo je lako odrediti kako ona utice najvise na SSE . Drugi pristup je odabrati nasumicnu tacku.

Postoje dve strategije za smanjenje broja klastera, koje istovremeno pokusavaju da minimizuju povecanje ukupnog SSE :

- **Odpustanje klastera:** Biramo centroid koji odpusta dati klaster, dok sve tacke koje su pripadale tom klasteru dodeljujemo odgovarajucim klasterima. Idealno, klaster koji odpustamo treba najmanje da doprinosi ukupno SSE .
- **Spajanje dva klastera:** Spajamo klastere ciju su centroidi najblizi, ili ona dva klastera koja spajanjem najmanje doprinose ukupnom SSE .

Azuriranje Centroida Inkrementalno:

Umesto da azuriramo klaster centroida nakon sto svim tackama dodelimo klaster, centroidi se mogu azurirati inkrementalno, nakon svake dodele tacke klasteru. Inkrementalna strategija garantuje da necemo dobiti prazni klaster kako svi kasteri pocinju sa jednom tackom, i ako svaki klaster ima samo jednu tacku, onda ce tacka uvek biti ponovo dodeljena istom klasteru.

Kod inkrementalnog azuriranja, relativne tezine tacaka se mogu menjati, tj. tezina tacke se obicno smanju pri odvijanju klasterovanja. Ovo moze dovesti do bolje tacnosti, ali je tesko odabrati dobre relativne tezine.

Takodje, ako koristimo proizvoljnu ciljnu funkciju, koja nije 'minimizuj SSE ', mozemo dobiti bolje rezultate. U svakoj inkrementaciji mozemo izbrati onaj korak koji najbolje optimizuje ciljnu funkciju.

Negativna strana koriscenja inkrementalne tehnike je vaznost reda kojim inkrementalno azuriramo klastere.

Bisekcija K-sredina

Bisekcija K-sredina se bazira na jednoj jednostavnoj ideji: Da bi dobili K klastera, podelio skup svih tacaka na dva klastera, odaberima jedan od njih za dalje deljenje. Ponavljamo proces sve dok ne dobijemo K klastera.

```
def bisecting_k_means(X, K, num_trials):
    clusters = [X]
    while len(clusters) < K:
        current_cluster = cluster.drop(1)
```

```

cluster1, cluster2 = None

for i in range(num_trials):
    [c1, c2] = k_means(current_cluster, 2)
    if SSE(c1) + SSE(c2) < SSE(cluster1) + SSE(cluster2):
        cluster1, cluster2 = c1, c2

clusters.append(cluster1)
clusters.append(cluster2)

return clusters

```

Nakon primene ove procedure dobijene centroide koristimo kao inicijalne centroide za osnovni algoritam K-sredina. Razlog toga lezi u tome da Bisekcija K-sredina koristi lokalno algoritam K-sredina pa zbog toga ne pronalazi lokalni minimum funkcije *SSE*. Ponovno pogretanje K-sredina sa inicijalnim centroidima procedure Bisekcija K-sredina garantuje pronalazenje lokalnog minimuma funkcije *SSE*.

K-sredina i Drugaciji Tipovi Klastera

K-sredina ima mnoge probleme pri nalazenju drugacijih tipova klastera. K-sredina ima poteskoce da pronadje ‘prirodne’ klastera, tj. kada klasteri imaju drugacije oblike od sfernih ili drugacije su velicine ili gustine.

Poteskoca u ovim slucajevima lezi u tome da se ciljna funkcija ne poklapa sa tim tipovima klastera. Ovo mozemo resite ako povecemo broj klastera. Tako dobijamo prirodno klasterovanje koje kasnije mozemo spojiti u postprocesiranju.

Prednosti i Mane

K-sredina je jedan jednostavan i efikasan algoritam cak i kada se primenjuje vise puta, a jos je efikasniji ako se koristi Bisekcija K-sredina, koja eliminise inicijalne probleme.

K-sredina ne moze da se nosi sa svim tipovima klastera. Takodje, lose rezultate daje kada podaci sadrze autlajere (tada moramo koristiti neke tehnike detekcije i otklanjanja autlajera). K-sredina je ogranicen na podatke za koje postoji smisao centra (centroid).

K-sredina kao Optimizacioni Problem

Za datu ciljnu funkciju kao sto je minimizacija *SSE*, klasterovanje moze da se smatra kao optimizacioni problem. Jedan dancin da se ovaj problem resi (nadje globalni optimum) je da se probaju svi moguci nacini tacaka i klastera, i da se od njih izabere najbolji, tj. onaj koji minimizuje ukupnu *SSE*. Ovaj problem je u praksi nemoguće resiti na ovaj nacin, pa se zbog toga koriste neke sufisticiranije metode. Jedna takva tehnika je **gradijentni spust**, koji na osnovu inicijalnog resenja pokusava da azurira resenje tako da u svakom koraku ima bolje resenje od prethodnog sve dok je to moguće.

Za dalji tekst pretpostavimo da su podaci jednodimenzioni, tj. da vazi $dist(x, y) = (x - y)^2$.

Izvodjenje K-sredina kao Algoritma za Minimizaciju SSE

Pokazimo kako centroid za K-sredina mozemo matematicki izvesti kada je funkcija blizine Euklidsko rastojanje, a ciljna funkcija minimizacija *SSE*. Tacnije, istrazujemo kako najbolje azurirati centroid klastera tako da klaster *SSE* bude minimalni. Matematicki hocemo da minimizujemo sledecu funkciju:

$$SSE = \sum_{i=1}^K \sum_{x \in C_i} (c_i - x)^2,$$

gde je C_i i -ti klaster, x tacka u C_i , c_i sredina i -tog klastera.

Nadjimo k -ti centroid c_k , koji minimizuje funkciju SSE , tako sto nalazimo parcijalni izvod od SSE i izjednacavamo ga sa nulom:

$$\begin{aligned}\frac{\partial}{\partial c_k} SSE &= \frac{\partial}{\partial c_k} \sum_{i=1}^k \sum_{x \in C_i} (c_i - x)^2 \\ &= \sum_{x \in C_k} \frac{\partial}{\partial c_k} (c_k - x)^2 \\ &= \sum_{x \in C_k} 2 * (c_k - x) = 0 \\ \implies m_k c_k &= \sum_{x \in C_k} x \\ \implies c_k &= \frac{1}{m_k} \sum_{x \in C_k} x\end{aligned}$$

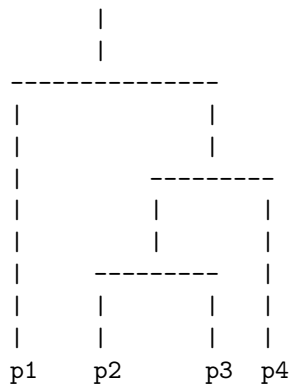
Kao sto smo prethodno pretpostavili najbolji centroid koji minimizuje SSE klastera je sredina svih tacaka tog klastera.

Hijerarhijsko Klasterovanje

Postoje dva pristupa hijerarhijskog klasterovanja:

- **Sakupljajuće Klasterovanje:** Pocinje sa tackama kao individualnim klasterima, na svakom koraku, spaja dva najbliza para klastera. Potrebno je definisati blizinu klastera.
- **Razdvajajuće Klasterovanje:** Pocinje sa jednim klasterom koji obuhvata sve tacke, na svakom koraku, deli se klaster sve dok ne dobijemo pojedinačne klastera. U ovom slucaju, treba odluciti koji klaster razdvojiti u kom trenutku i kako izvršiti razdvajanje.

Hijerarhijsko klasterovanje se predstavlja grafikci drvolikim dijagramom koji se zove **dendrogram**, koji prikazuje relaciju klaster-podklaster, i redosled u kome se klasteri spajaju ili razdvajaju. Za dvodimenzione tacke, hijerarhijsko klasterovanje se moze predstaviti udnjezdenim klaster dijagramima.



Osnovni Algoritam Sakupljajućeg Klasterovanja

```
def agglomerative_hierarchical_clustering(X):

    # pocetne klastere inicijalizujemo tako da
    # svaki sadrzi tacno jednu tacku
    clusters = X
    proximity_matrix = proximity(clusters)
```

```
while len(clusters) != 1:
    clusters = merge_best(clusters, proximity_matrix)
    proximity_matrix = update_proximity(clusters)
```

Definisanje Blizine izmedju Klastera

Definisanje blizine izmedju klastera zavise od određenog tipa klastera. **MIN** definiše blizinu izmedju dva klastera kao blizinu izmedju dva para najbližih tacaka. Alternativno, **MAX** uzima za blizinu izmedju dva klastera blizinu izmedju dva para najudaljenijih tacaka. Za grafovske klastere ili klastere čiji objekti predstavljaju tacke u Euklidskom prostoru MIN i MAX su intuitivna imena. Za sličnost kao meru blizine, gde veće vrednosti pokazuju na bliže tacke, imena se čine obrnuto (iz tog razloga koristimo termine **pojedinačna veza**, i **kompletna veza**). **Grupna sredina** definiše blizinu klastera kao prosek blizine parova tacaka.

Ako koristimo klastere bazirane na prototipu, definisanje blizine klastera je prirodno. Ako su to centriodi, blizina klastera se definiše kao blizina izmedju centriodi klastera. Jedna alternativna tehnika, **Vardova metoda**, definiše meru blizine izmedju dva klastera u terminima povećanja *SSE* koje nastaje spajanjem ta dva klastera.

Vremenska i Prostorna Složenost

Osnovni algoritam sakupljajućeg klasterovanja prati matricu blizina i klastere. Matrica blizina zahteva $\frac{1}{2}m^2$ (vrednost) blizina pod pretpostavkom da je matrica blizina simetrična, gde je m broj tacaka. Za praćenje klastera dovoljno je imati broj klastera koji je najviše $m - 1$. Ukupna vremenska složenost je onda $O(m^2)$.

Vreme za računanje inicijalne matrice blizina je $O(m^2)$. Nakon toga imamo $m - 1$ iteracija. Tokom i -te iteracija potrebno je $O((m - i + 1)^2)$ za spajanje dva najbolja klastera. Dok je potrebno samo $O(m - i + 1)$ za azuriranje matrice blizine, nakon spajanja dva klastera. Ovakav pristup daje ukupnu vremensku složenost $O(m^3)$. Moguće je čuvati udaljenost za svaki par klastera u sortiranoj listi (ili hipu), pa je moguće smanjiti složenost pronalazenja dva najbolja na $O(m - i + 1)$. Međutim dobijamo na kompleksnosti održavanja sortirane liste (ili hipa), pa je ukupna složenost algoritma $O(m^2 \log m)$.

Specifčne Tehnike

Jednostavni Podaci

Tacke	x	y
p1	0.40	0.53
p2	0.22	0.38
p3	0.35	0.32
p4	0.26	0.19
p5	0.08	0.41
p6	0.45	0.30

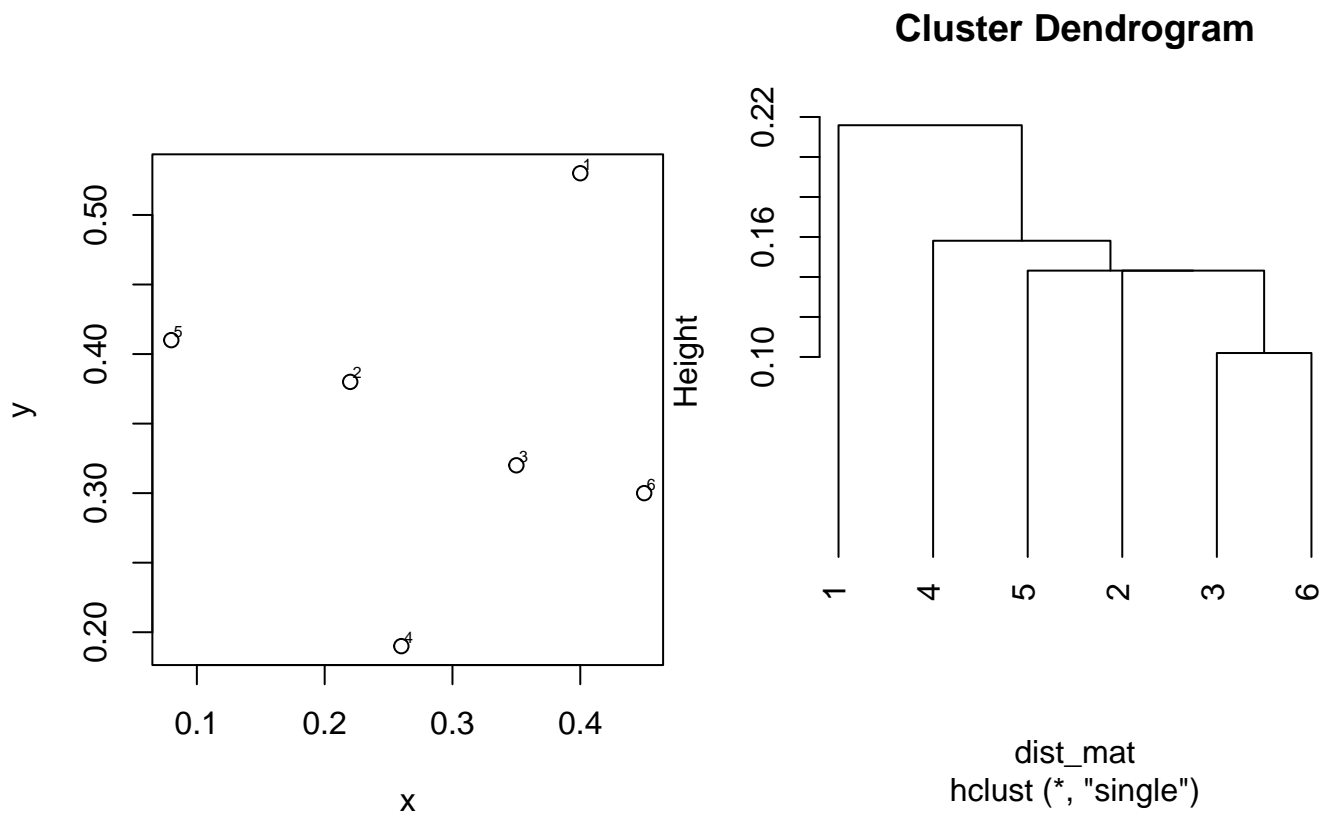
Dist	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Pojedinačna Veza ili MIN

Blizinu izmedju dva klastera definišemo kao minimalnu udaljenost (maksimalnu sličnosti) izmedju bili koje

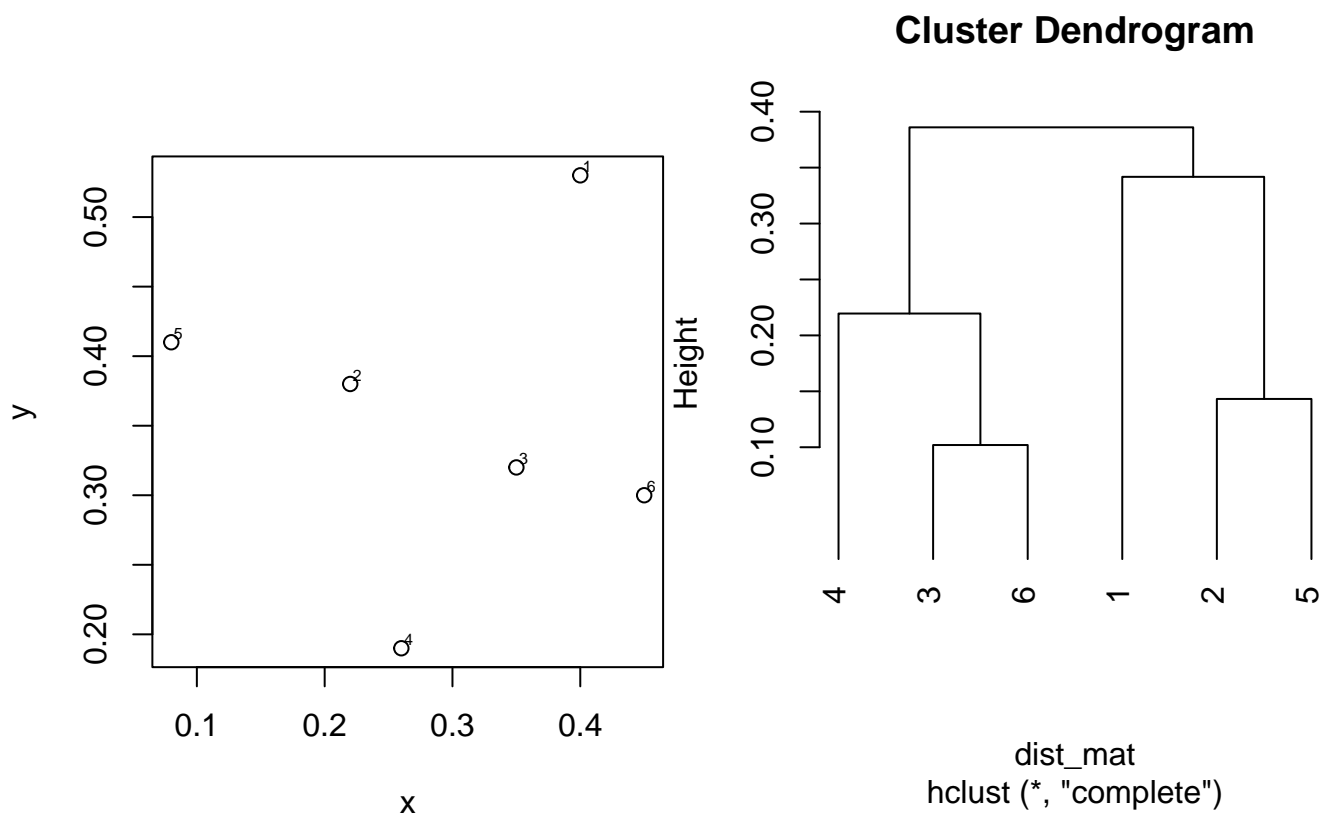
dve tacke iz dva razlicita klastera.

```
## Loading required package: MASS
```



Komplentna Veza ili MAX

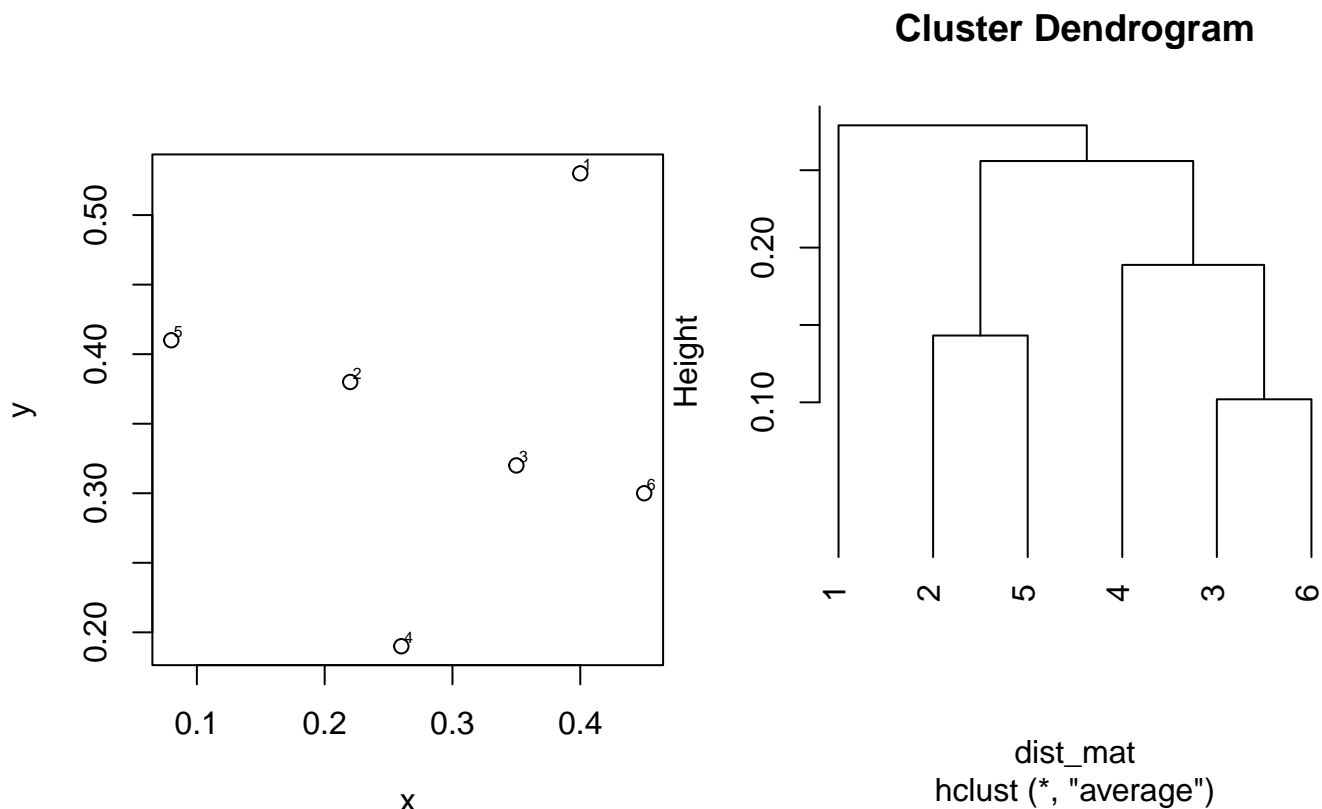
Blizinu izmedju dva klastera definisemo kao maksimalnu udaljenost (minimalnu slicnost) izmedju bili koje dve tacke iz dva razlicit klastera.



Prosek Grupa

Blizinu između dva klastera definišemo kao prosečnu blizinu svih parova tačaka iz ta dva klastera, tj. važi sledeće

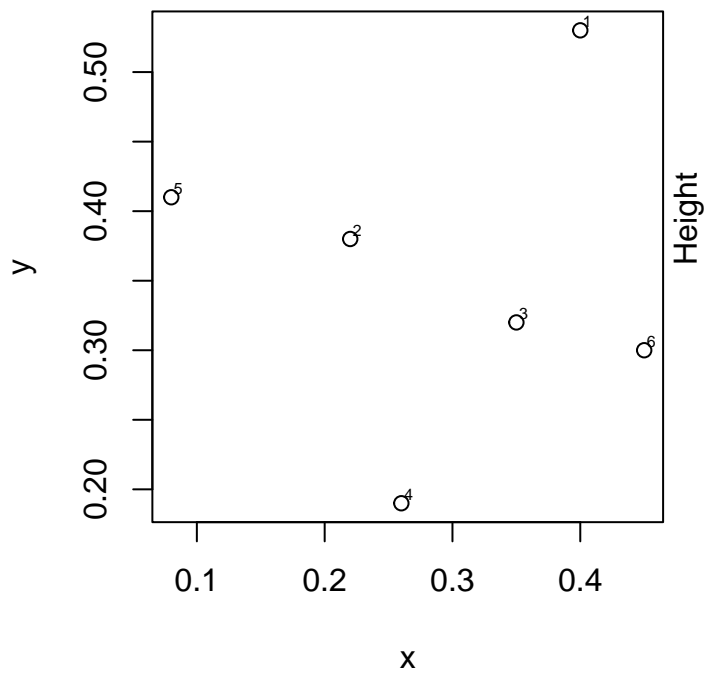
$$blizina(C_i, C_j) = \frac{\sum_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} blizina(\mathbf{x}, \mathbf{y})}{m_i m_j}$$



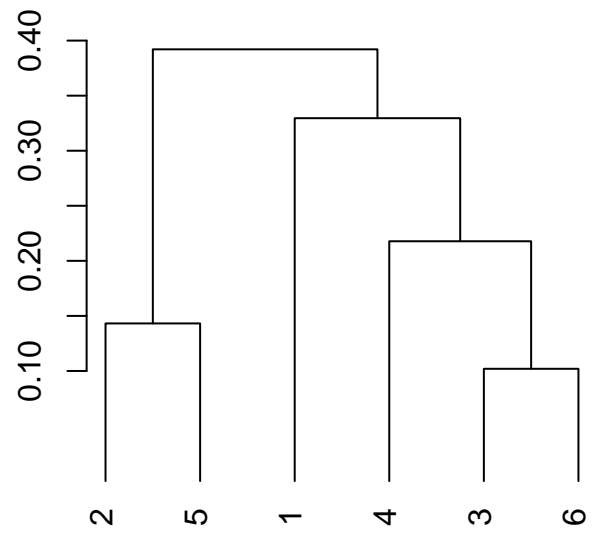
Vardov Metod i Metod Centroida

Za Vardov metod blizina izmedju dva klastera se definise kao povecanje u SSE koje nastaje kada se ta dva klastera spoje. Ovaj metod koristi istu ciljne funkciju kao K-sredina.

Centroid metod racuna blizinu izmedju dva klastera kao udaljenost izmedju cetroida ta dva klastera. Metod centroida ima jednu losu karakteristiku: mogucnost **inverzija**. Dva klastera koja su spojena mogu biti vise slicnija (manje rezlicita) od para klastera koji su spojeni u prethodnom koraku. Za druge metode udaljenost izmedju klastera monotono opada (ili u najgorem slucaju ostaje nepromenjena).

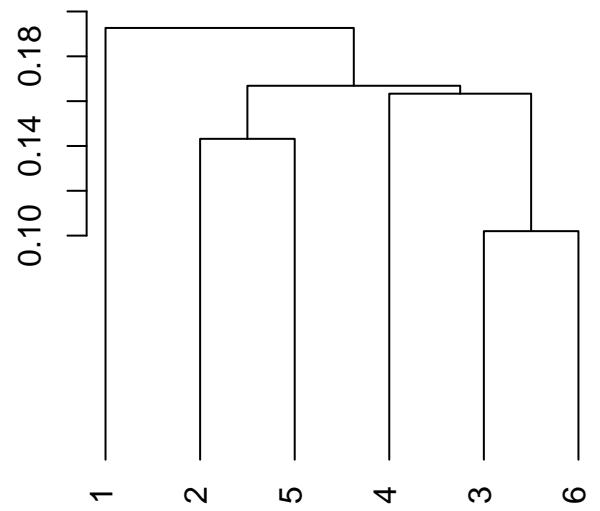
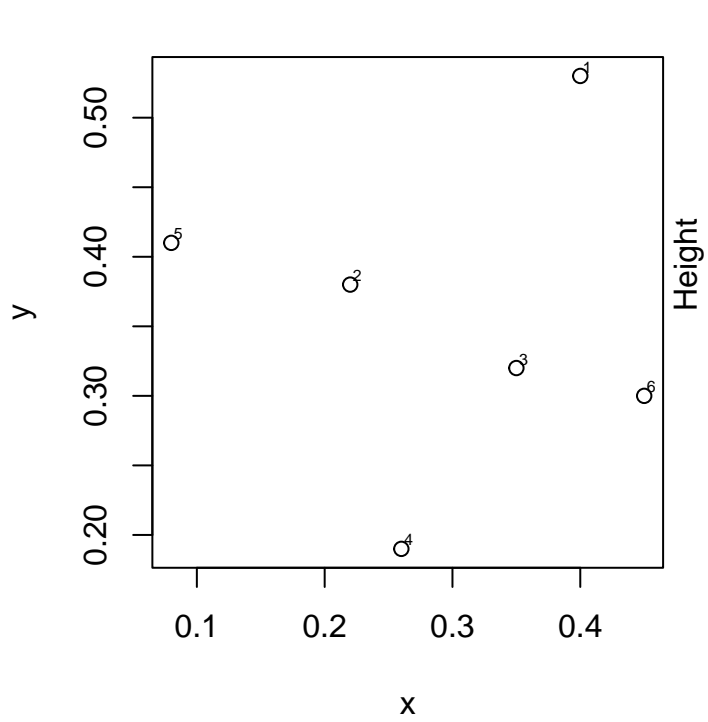


Cluster Dendrogram



dist_mat
hclust (*, "ward.D")

Cluster Dendrogram



dist_mat
hclust (*, "centroid")

Lens-Vilijams Formula za Klaster Blizine

Svaka od klaster blizina može se uzeti kao parametar (u Lens-Vilijams formuli) za blizinu između dva klastera Q i R , gdje se R formira spajanjem klastera A i B .

$$p(R, Q) = \alpha_A p(A, Q) + \alpha_B p(B, Q) + \beta p(A, B) + \gamma |P(A, Q) - p(B, Q)|,$$

gdje je p funkcija blizine, m_A , m_B , i m_Q su brojevi tacaka u klasterima A , B , i Q

Metod Klasterovanja	α_A	α_B	β	γ
Pojedinacna veza	$1/2$	$1/2$	0	$-1/2$
Komplenta veza	$1/2$	$1/2$	0	$1/2$
Prosek grupa	$\frac{m_A}{m_A + m_B}$	$\frac{m_B}{m_A + m_B}$	0	0
Centroid	$\frac{m_A}{m_A + m_B}$	$\frac{m_B}{m_A + m_B}$	$\frac{-m_A m_B}{(m_A + m_B)^2}$	0
Centroid	$\frac{m_A + m_Q}{m_A + m_B + m_Q}$	$\frac{m_B + m_Q}{m_A + m_B + m_Q}$	$\frac{-m_Q}{m_A + m_B + m_Q}$	0

Ključni Problemi u Hijerarhijskom Klasterovanju

Nedostatak Globalne Ciljne Funkcije

Sakuplja juce hijerarhijsko klasterovanje odlucuje lokalno, tj. u svakom koraku odlucuju koje klastere treba spojiti (ili razdvojiti za razdvojni pristup).

Tretiranje Velicina Klastera

Postoje dva pristupa: (1) **tezinsko**, tretira sve klastere kao jednake, (2) **netezinsko**, uzima u obzir broj tacaka u svakom klasteru. Ovi termini se odnose na tacke, a ne na klastere, tj. kod nejednakih velicina klastera drugacije tezine dodeljuju se tacaka u zavisnosti od klastera kome pripadaju, dok ako se uzima velicina klastera tacke imaju istu tezinu bez obzira u kom se klasteru nalaze.

Odluke o Spajanju su Konacne

Jednom napravljena odluka o spajanju dva klastera se ne može promeniti u buducim iteracijama.

DBSCAN

Klasterovanje bazirano na gustine otkriva regione sa velikom gustinom koji su odvojeni jedni od drugih regionima sa malom gustinom. DBSCAN je jednostavan i efektivan algoritam za klasterovanje bazirano na gustini.

Tradicionalne Tehnike: Pristup Baziran na Centru

Kod pristupa baziranog na centru, gustina za neku tacku se odredjuje kao broj tacaka unutar nekog radiusa, Eps , te tacke. Brajanje ukljucuje i samu tacku od koje se povlaci radius. Gustina veoma zavisi od velicine radiusa. Na primer, za veoma veliki radius gustina svake tacke ce biti m broj primeraka u skupu podataka, dok ce za veoma mali radius gustina svake tacke biti 1.

Klasifikacija Tacaka Prime Gustini Baziranoj na Centru

Tacku možemo klasifikovati na tri nacina: (1) unutrasnjost gustog regiona (tacke jezgra), (2) na granici gustog regiona (granicna tacka), (3) unutar nekog drugog regiona (sum ili pozadinska tacka).

Algoritam: DBSCAN

Ideja: Bilo koje dve tacke jezgra koje su unutar radiusa daleko jedna od druge mogu se staviti u isti klaster. Bilo koja granicna tacka koja je dovoljno blizu tacke jezgra može se staviti u isti klaster kao i tacka jezgra.

1. Oznaci sve tacke kao tacke jezgra, granice, ili suma.
2. Eliminise sve tacke suma.
3. Postavi granicu izmedju svih tacaka koje su unutar radiusa
4. jedne izmedju drugih.
5. Neka svaka grupa povezanih tacaka bude jedan klaster.
6. Dodeli sve granicne tacke jednom od odgovarajucih klastera.

Vremenska i Prostorna Slozenost

Vremenska slozenost DBSCAN algoritma je $O(m \times t)$, gde je t vreme nalazenja tacaka u Eps -okolini, i m broj tacaka. U najgorem slucaju kompleksnost je $O(m^2)$. Ali u praksi je to $O(m \log m)$. Prostorna slozenost DBSCAN je $O(m)$, kako se jedino cuva mali podataka za svaku tacku, tj. oznake klastera i identitet svake tacke (jezgra, granica, ili sum).

Odabir Parametara za DBSCAN

Kako odrediti parametre Eps i $MinPts$? Osnovni pristup je posmatrati ponasanje udaljenosti tacke do njenih k najblizih suseda, koje nazivamo k -udaljenost. Za tacke koje pripadaju istom klasteru, vrednost k -udaljenost ce biti mala ako k nije vece od velicine klastera. Medjutim, za tacke koje nisu u klasteru, kao sto su tacke suma, k -udaljenost ce biti relativno velika. Ako bismo racunali k -udaljenost za sve tacke podataka za neko k , i sortirali ih u rastucem poretku, nakon njihovog plotovanja dobili bi smo nagli porast (lakat). Ako uzmemo k -udaljenost za Eps parametar i ako uzmemo vrednost k kao $MinPts$ parametar, onda ce tacke za koje je k -udaljenost manja od Eps biti oznacene kao tacke jezgra, dok ce ostale biti oznacene kao sum ili granicne tacke.

Originalan DBSCAN algoritam koristi vrednost $k = 4$, sto je dosta razumna vrednost za mnoge dvodimenzionalne podatke.

Klasteri Promenljive Gustine

DBSCAN ima problem sa gustinama ako se gustine klastera medjusobno drasticno razlikuju.

Prednosti i Mane

DBSCAN je otporan na sum i moze se nositi sa klasterima proizvoljnog oblika i velicine. Ali ima problem sa medjusobno razlicitim gustinama klastera. Takodje sa visedimenzione podatke gustinu je teze definisati.

Evaluacija Klastera

Pravila Pridruzivanja: Osnovni Koncepti i Algoritmi

Skup podataka kao sto su **transakcije** ili **potrosacka korpa** su veoma ceste pri prikupljanju podataka potrosaca i njihove kupovine. Svaki red u tabeli odgovara jednoj transakciji, koja ima jedinstvenu identifikacioni broj TID i skup proizvoda koje je kupio potrosac.

TID	Proizvodi
1	{Hleb, Mleko}
2	{Hleb, Pelene, Pivo, Jaja}
3	{Mleko, Pelene, Pivo, Kola}
4	{Hleb, Mleko, Pelene, Pivo}
5	{Hleb, Mleko, Pelene, Kola}

Analiza pridruzivanja se korisna pri otkrivanju interesantnih skrivenih veza unutar veliking skupova podataka. Neotkrivene veze se mogu predstaviti kao **pravila pridruzivanja** ili skupovi frekventnih proizvoda. Na prime, sledece pravilo se moze izvesti iz gornje tabele:

$$\{Pelene\} \rightarrow \{Pivo\}.$$

Ovo sugerise na veliku relaciju koja postoji izmedju prodaje Pelena i Piva u smislu da potrosaci koji kupju Pelene takodje kupuju i Pivo.

Analiza pridruzivanja se koristi i u drugim oblastima kao sto su bioinformatika, medicina, pretraga veba, naucna istrazivanja.

Dva kljucna problema pri analizi potrosacke korpe: (1) Pronalazenje sablona iz velikog skupa transakcija moze biti racunski skupo, (2) Neki od pronadjениh sablona su potencijalno nevazeci kako se mogu stvariti slucajno.

Problem Definicije

Binarna Reprezentacija: Potrosacka korpa se moze predstaviti u binarnom formatu, gde svaki red odgovara jednoj transakciji i svaka kolona odgovara jednoj stavki. Stavka se tretira kao binarna promenljiva cija je vrednos 1 ako je *ima* u transakciji ili 0 ako je *nema* u transakciji. Bitnije je da se stavka nalazi u transakciji pa je stavka **asimetrična** binarna promenljiva.

<i>TID</i>	Hleb	Mleko	Pelene	Pivo	Jaja	Kola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

Skup Stavki i Pomocni Brojac: Neka je $I = \{i_1, i_2, \dots, i_d\}$ skup svih stavki i $T = \{t_1, t_2, \dots, t_N\}$ skup svih transakcija. Svaka transakcija t_i sadrzi podskup skupa svih stavki I . Kolekcija nula ili vise stavki se naziva skup stavki. Ako skup stavki ima k proizvoda, onda se naziva k -skup stavki. *Sirina transakcij* se definise kao broj stavki unutar jedne transakcije. Transakcija t_j *sadrzi* skup stavki X ako je X podskup od t_j . *Pomocni brojac*, $\sigma(X)$, za skup stavki X definise se kao broj transakcija koje sadrže X , tj. vazi:

$$\sigma(X) = |\{t_i | X \subseteq t_i, t_i \in T\}|$$

Pravilo Pridruzivanja: *Pravilo predruzivanja* je implicitan izraz $X \rightarrow Y$, gde su X i Y diskunktni skupovi stavki. Jacina pravila pridruzivanja se moze meriti u terminima **cvrstine** (koliko se puta pravilo pojavljuje u skupu podataka) i **pouzdanosti** (koliko se frekventno stavku u Y pojavljuju u transakcijama koje sadrže X):

$$\begin{aligned} \text{Cvrstina: } s(X \leftarrow Y) &= \frac{\sigma(X \cup Y)}{N}; \\ \text{Pouzdanost: } c(X \leftarrow Y) &= \frac{\sigma(X \cup Y)}{\sigma(X)}. \end{aligned}$$

Primer: Razmotrimo pravilo $\{\text{Mleko, Pelene}\} \rightarrow \{\text{Pivo}\}$. Kako je $\sigma(\{\text{Mleko, Pelene, Pivo}\}) = 2$, $N = 5$, i $\sigma(\{\text{Mleko, Pelene}\}) = 3$, imamo da je cvrstina $s(\{\text{Mleko, Pelene}\} \rightarrow \{\text{Pivo}\}) = 2/5 = 0.4$, i pouzdanost $c(\{\text{Mleko, Pelene}\} \rightarrow \{\text{Pivo}\}) = 2/3 = 0.67$

Zasto Koristiti Cvrstinu i Pouzdanost? Cvrstina je bitna jer pravilo koje ima malu cvrstinu moze nastati slucajno. Takodje mala cvrstina ne ostvaruje veliku dobit jer je pravilo neinteresantno. Zbog ovih razloga, podrška se koristi za eliminisanje neinteresantnih pravila. Pouzdanost meri visku uzracnost pravila. Za dato pravilo $X \leftarrow Y$, sa velikom pouzdanosti, vazi da je vise verovatno da se Y javi u transakcijama koje sadrže X . Pruza proske ulovne verovatnoce od Y za dato X .

Definicja (Odredjivanje Pravila Pridruzivanja). Za dati skup transakcija T , nadjі sva pravila pridruzivanja $X \rightarrow Y$ za koja vazi $s(X \rightarrow Y) \geq \text{minsup}$ i $c(X \rightarrow Y) \geq \text{minconf}$, gde su minsup i minconf najmanje granice za cvrstinu i pouzdanost respektivno.

Iscrpna pretraka za odredjivanje pravila pridruzivanja se zasniva na racunanju cvrstine i pouzdanosti za svako moguće pravilo pridruzivanja. Ukupan broj mogucih pravila, izdvojenih iz skupa podataka koji sadrzi d stavki je

$$R = 3^d - 2^{d+1} + 1.$$

Cak i za male vrednost parametra d imamo veliki broj pravila: $3^6 - 2^7 + 1 = 602$. Vise od 80% pravila ce biti odbaceno nakon primene $minsup = 20\%$ i $minconf = 50\%$, pa nema smisla generisati sva moguca pravila pridruzivanja. Zbog toga je od kljucnog znacaja koristiti tehnike odsecanja pretrage.

Primetimo da pravilo pridruzivanja $X \rightarrow Y$ zavisi samo od cvrstine njegovog odgovarajuceg skupa stavki, $X \cup Y$. Na primer, razmotrimo sledeći skup stavki {Pivo, Pelene, Mleko} koji moze da generise sledeca pravila pridruzivanja:

$$\begin{aligned} \{Pivo, Pelene\} &\rightarrow \{Mleko\}; \\ \{Pivo, Mleko\} &\rightarrow \{Pelene\}; \\ \{Pelene, Mleko\} &\rightarrow \{Pivo\}; \\ \{Pivo\} &\rightarrow \{Pelene, Mleko\}; \\ \{Mleko\} &\rightarrow \{Pelene, Pivo\}; \\ \{Pelene\} &\rightarrow \{Mleko, Pivo\}. \end{aligned}$$

Ako je skup pravila {Pivo, Pelene, Mleko} nefrekventan, tj. njegova cvrstina nije velika, onda svih sest kandidata za pravila mozemo odseci u pretrazi.

Mnogo algoritmi istrazivanja pravila pridruzivanja se mogu dekomponovati u dva velika podzadatka:

1. **Generisanje frekventnih skupova stavki**, tj. generisanje svih onih skupova stavki za koje vazi da cvrstiji od donje granice $minsup$. Oni se nazivaju frekventnim skupovima stavki.
2. **Generisanje pravili**, tj. generisati sva pravila od frekventnih skupova stavki sa velikom pouzdanoscju. Ova pravila se nazivaju jaka pravila.

Generisanje Frekventnih Skupova Stavki

Struktura resetke se moze koristiti za prebrojavanja svih mogucih skupova stavki. Skup podataka od k stavki moze generisati i do $2^k - 1$ frekventnih skupova stavki. Zbog toga dobijamo eksponencijalnu slozenost prostora pretrage.

Gruba sila nalazenje frekventnih skupova stavki podrazumeva racnanje cvrstine za svaki **kandidat skup stavki** u strukturi resetke. Uporedjujemo svaki kandidat skup stavki sa svim transakcijama. Ako se kandidat nalazi u transakciji, onda povecavamo brojac cvrstine. Ovakav pristup je veoma skup i zahteva $O(NMw)$ uporedjivanja, gde je N broj transakcija, $M = 2^k - 1$, i w maksimalna duzina transakcije.

Postoje nekoliko nacina za smanjivanje slozenosti nalazenje fekvntnih skupova stavki:

1. **Smanjenje broja kandidata skupa stavki (M)**. **Apriori** princip, je efektivna nacin da se eliminiše kandidati bez racunanja njihove cvrstine.
2. **Smanjenje broja poredjenja**. Umesto da se poredi svaki kandidat sa svakom transakcije, mozemo smanjiti broj poredjenja tako sto koristimo slozeniju strukturu podataka, za cuvanje kandidata ili kompresovanje transakcija.

Apriori Princip

Teorema (Apriori Princip). *Ako je skup stavki frekventan, onda su svi njegovi podskupovi moraju biti frekvntni.*

Neka je data resetka pomocu sledećih stavki $I = \{a, b, c, d, e\}$. Pretpostavimo da je $\{c, d, e\}$ frekventni skup stavki. Jasno je da svaka transakcija koja sadrzi $\{c, d, e\}$ mora takodje da sadrzi $\{c, d\}$, $\{c, e\}$, $\{d, e\}$, $\{c\}$, $\{d\}$, i $\{e\}$. Kao rezultat, ako je $\{c, d, e\}$ frekventno, onda svi njegovi podskupovi takodje moraju biti frekventni.

Obratno, ako je skup stavki kao sto je $\{a, b\}$ nefrekventan, onda njegovi nadskupi moraju takodje biti nefrekventni. Zbog toga kada pronadjemo da je $\{a, b\}$ nefrekventno, celokupni podgraf, ciji cvorovi sadrže

$\{a, b\}$ može biti odsečen. Ova strategija se naziva **odsecanje na osnovu cvrstine**. Ovo je moguće zbog osobine mere cvrstine, tačnije, cvrstina skupa stavki nikada ne premasuje cvrstinu njegovih podskupova (**anti-monotonost**).

Definicija (Osobina Monotonosti). Neka je I skup svih stavki, i $J = 2^I$ partitivni skup od I . Mera f je monotona ako

$$\forall X, Y \in J; (X \subseteq Y) \rightarrow f(X) \leq f(Y),$$

sto znaci da ako je X podskup od Y , onda $f(X)$ ne sme premasiti $f(Y)$. Sa druge strane, f je anti-monotona ako

$$\forall X, Y \in J; (X \subseteq Y) \rightarrow f(Y) \leq f(X),$$

sto znaci da ako je X podskup od Y , onda $f(Y)$ ne sme premasiti $f(X)$.

Generisanje Frekventnih Skupova Stavki i Apriori Algoritam

Apriori je prvi algoritam otkrivanja pravila predruzivanja koji koristi strategiju odsecanja baziranu na cvrstini.

Stavka	Brojac	2-Skup stavki	Brojac	3-Skup stavki	Brojac
Pivo	3	{Pivo, Hleb}	2	{Pivo, Pelene, Mleko}	3
Hleb	4	{Pivo, Pelene}	3		
Kola	2	{Pivo, Mleko}	2		
Pelene	4	{Hleb, Pelene}	3		
Mleko	4	{Hleb, Mleko}	3		
Jaja	1	{Pelene, Mleko}	3		

Svaka stavka se smatra kao 1-skup stavki. Nakon sto izbrojimo njegovu cvrstinu, kandidate {Kola} i {Jaja} uklanjamo zato sto se pojavljuju u manje od tri transakcije. U sledecoj iteraciji, kandidate 2-skup stavke generisemo tako sto koristimo frekvente 1-skup stavke, jer *apriori* princip garantuje da nefrekventne 1-skup stavke generisu nefrekventne 2-skup stavke. Nastavljamo postupak dok je moguće.

Efektivnost *Apriori* strategije odsecanja se može pokazati brojanje kandidata koji se generisu. Kod iscrpne pretrage imamo:

$$\binom{6}{1} + \binom{6}{2} \binom{6,3}{=} 6 + 15 + 20 = 41,$$

dok sa *Apriori* principom taj broj smanjujemo na:

$$\binom{6}{1} + \binom{4}{2} + 1 = 6 + 6 + 1 = 13.$$

```
def apriori_algorithm(I, T, sigma, minsup, minconf):
    k = 1
    # Nalazimo sve frekventne 1-skup stavke
    F[k] = [i for i in I if sigma[i] >= N * minsup]
    while not F[k].empty():
        k = k + 1
        # Generise sledece kandidate
        C[k] = apriori_gen(F[k - 1])
        for t in T:
            # Identifikuj sve kandidate koji pripadaju
            # transakciji t
            C[t] = subset(C[k], t)
            for c in C[t]:
                # Inkrementiraj brojac
                sigma[c] = sigma[c] + 1
    # Izdvoji sve frekventne k-skup stavke
```

```

F[k] = [c for c in C[k] if sigma[c] >= N * minsup]

return F

```

Generisanje Kandidata i Odsecanje

`apriori_gen` funkcija generise kandidate u dva koraka:

1. **Generisanje kandidata.** Ova operacija generise nove kandidate k -skup stavke od frekventnih $(k-1)$ -skupa stavki pronadjene u prethodnoj iteraciji.
2. **Odsecanje kandidata.** Ova operacija eliminise neke kandidate k -skup stavke koriscenjem strategije odsecanja bazirane na cvrstini.

Razmotrimo k -skup stavki kandidate, $X = \{i_1, i_2, \dots, i_k\}$. Algoritam mora odrediti da li su svi pravi podskupovi $X - \{i_j\}$, frekventni. Ako jedan od njih nije frekventan, onda se X odmah odseca.

Metodi za generisanje kandidata:

- **Iscrpna pretraga.** Razmatra svaki k -skup stavki kao potencijalnog kandidata i onda primenjuje potresivanje za eliminisanje nepotrebnih kandidata. Slozenost ovog pristupa je $O(\sum_{k=1}^d k \times \binom{d}{k}) = O(d \times 2^{d-1})$.
- $F_{k-1} \times F_1$ **Metod.** Ovaj metod podrazumeva siranje svakog od $(k-1)$ -skupa stavki sa drugim frekventnim stavkama. Slozenost ovog algoritma je $O(\sum_k k \times |F_{k-1}| \times |F_1|)$. Mana ovog metoda je to sto se moze desiti da generise vise puta iste kandidate, pa se njihova kasnija eliminacija mora uzeti u obzir.
- $F_{k-1} \times F_{k-1}$ **Metod.** Ova procedura spaja dva para $(k-1)$ -skupa stavki samo ako su njihove prve $k-2$ stavke identicne. Neka su $A = \{a_1, a_2, \dots, a_{k-1}\}$ i $B = \{b_1, b_2, \dots, b_{k-1}\}$ parovi $(k-1)$ -skupova stavki. A i B se spajaju ako zadovoljavaju sledeci uslov:

$$a_i = b_i, \text{ za } i = 1, 2, \dots, k-2, \text{ i } a_{k-1} \neq b_{k-1}.$$

Pri generisanju novih kandidata dobro je cuvati $(k-1)$ -skupove stavki u leksikografskom poretku kako bi izbegli generisanje duplikata.

Brojanje Cvrstine

Jedan pristup za brojanja cvrstina je uporedjivanje svake transakcije sa svakim kandidatom i inkrementiranje broja cvrstine ako se kandidat sadrzi u transakciji. Ovaj pristup je racunski zahtevan, pogotovo kada su broj transakcija i broj kandidata veliki.

Brojanje Cvrstine Koriscenjem Hes Drveta

Apriori algoritam particionise kandidate korpe i smesta ih u hes drvo. Tokom brojanja skupovi stavki koji se sadrze u svim transakcijama se takodje hesiraju u njihove odgovarajuce korpe. Tako je dovoljno uparedjivati svaku transakciju sa korpom.

Svi unutasnji cvorovi drveta imaju sledecu hes funkciju $h(p) = p \bmod 3$ za odredjivanje sledece grane.

Generisanje Pravila

Svake frekventni k -skup stavki, Y , moze proizvesti $2^k - 2$ pravila pridruzivanje. Pravilo pridruzivanja mozemo dobiti particionisanjem skupa stavki Y u dva ne prazna podskupa, X i $Y - X$, tako da $X \rightarrow Y - X$ zadovoljava granicu pouzdanja.

Primer: Neka je $X = \{1, 2, 3\}$ frekventni skup stavki. Postoji sest kandidata za pravilo pridruzivanja koja se mogu generisati od X : $\{1, 2\} \rightarrow \{3\}$, $\{1, 3\} \rightarrow \{2\}$, $\{2, 3\} \rightarrow \{1\}$, $\{1\} \rightarrow \{2, 3\}$, $\{2\} \rightarrow \{1, 3\}$, $\{3\} \rightarrow \{1, 2\}$.

Ponovno prolazenje kroz transakcije da bi izracunali pouzdanost pravila pridruzivanje nije potrebno. Posmatramo pravilo $\{1, 2\} \rightarrow \{3\}$, koje je generisano od frekventnih skupova podataka $X = \{1, 2, 3\}$. Pouzdanost

za ovo pravilo je $\sigma(\{1, 2, 3\})/\sigma(\{1, 2\})$. Zato što je $\{1, 2, 3\}$ frekventno, osobina anit-monotonosti tvrdi da $\{1, 2\}$ mora biti takodje frekventno. Obe vrednosti su vec nadjene pa nije potrebno ponovno prolaziti kroz skup podataka.

Potkresivanje na Osnovu Pouzdanosti

Teorema. *Ako pravilo $X \rightarrow Y - X$ ne zadovoljava granicu pouzdanosti onda bilo koje pravilo $X' \rightarrow Y - X'$, gde je $X' \subset X$, sigurno ne zadovoljava granicu pouzdanosti.*

Generisanje Pravila u Apriori Algoritmu