Istrazivanje Podataka 1 - Belekse

Andrija Urosevic

Contents

Uvod **2**
- Sta je istrazivanje podataka? . . . 3
  - Istrazivanje podataka i otkrivanje znanja . . . 3
- Izazovi u istrazivanju podataka . . . 3
  - Skalabilnost . . . 3
  - Velika Dimenzionalnost . . . 3
  - Heterogeni i kompleksni podaci . . . 3
  - Pripadnost i distribucija podataka . . . 3
  - Netradicionalna analiza . . . 4
- Nastanak istrazivanja podataka . . . 4
- Zadatak istrazivanja podataka . . . 4

Podaci **4**
- Tipovi podataka . . . 5
  - Atributi i Mere . . . 5
  - Tipovi skupova podataka . . . 6
- Kvalitet podataka . . . 8
  - Merenje i problemi pri sakupljanju podataka . . . 8
  - Problemi pri primenama podataka . . . 10
- Predprocesiranje podataka . . . 10
  - Agregacija . . . 10
  - Uzorkovanje . . . 11
  - Redukcija dimenzija . . . 12
  - Diskretizacija i Binarizacija . . . 12
- Mere slicnosti i razlicitosti . . . 14
  - Osnove . . . 14
  - Slicnost i Razlicitost izmedju jednostavih atributa . . . 15
  - Razlicitosti izmedju objekta podataka . . . 15
  - Slicnosti izmedju objekta podataka . . . 16
  - Primeri mera blizine . . . 16
  - Problemi pri racunanju blizine . . . 18

Pretrazivanje Podataka **18**

Klasifikacija: Osnovni koncepti, drveta odlucivanja **18**
- Osnove definicije . . . 19
- Generalni pristup resavanja klasifikacionih problema . . . 19
- Drvo odlucivanja uvod . . . 20
  - Kako radi drvo odlucivanja? . . . 20
  - Kako napraviti drvo odlucivanja? . . . 21
  - Metodi za izrazavanje uslova testiranja atributa . . . 22
  - Mera za odabir najboljeng deljenja . . . 23

Algoritam indukovanja drveta odlucivanja . . . 25
Karakteristike indukovanja drveta odlucivanja . . . 25
Preprilagodjavanje modela . . . 26
Preprilagodjavanje zbog prisustva suma . . . 26
Preprilagodjavanje zbog nedostatka reprezentativnih uzoraka . . . 27
Preprilagodjavanje i procedura visestrukog poredjenja . . . 28
Procenjivanje greske generalizacije . . . 29
Preprilagodjavanje u indukovanje drveta odlucivanja . . . 30
Racunanje performanse klasifikatora . . . 31
Metod zadrzavanja . . . 31
Nasumicno uzorkovanje . . . 31
Unakrsna-Validacija . . . 31
Bootstrap . . . 32
Metodi za uporedjivanje klasifikatora . . . 32
Procenjivanje intervala pouzdanja za tacnost modela . . . 32
Racunanje performansi dva modela . . . 33
Uperedjivanje performansi dva klasifikatora . . . 33

Klasifikacije: Alternativne tehnike **34**
Klasifikator zasnovan na pravilima . . . 34
Kako klasifikator zasnovan na pravilima radi? . . . 34
Seme uredjenja-pravila . . . 35
Kako napraviti klasifikator zasnovan na pravilima? . . . 35
Direktni metodi za izdvajanje pravila . . . 35
Indirektni metodi za izdvajanje pravila . . . 38
Karakteristike klasifikatora zasnovanog na pravilima . . . 39
Klasifikator: Najblizi sused . . . 39
Algoritam . . . 39
Karakteristeke klasifikatora: najblizi sused . . . 40
Bajasov klasifikator . . . 40
Bajasova teorema . . . 40
Bajasova teorema u klasifikaciji . . . 40
Naivni Bajasov klasifikator . . . 41
Bajasova greska . . . 45
Bajasova mereza poverenja . . . 45
Vestacke neuronske mreze . . . 45
Perceptron . . . 45
Viseslojne vestacke neuronske mreze . . . 47
Karakteristike vestacke neuronske mreze . . . 48
Metod potpunih vektora (Support Vector Machine — SVM) . . . 48

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 proizvodjaci 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 Inzinjering. Prikupljaju se podati koji su kljucni za nova otkrica. Primer je NASA koja je postavila satelite oko planete Zemlje i meti kopno, okeane i atmosferu. Ali zvog kolicine podataka tradicionalni metodi nisu korisni za analizu ovakvig skupova podataka. Istrazivanje podataka moze da da odgovore na sledeca pitanja: Koja je relacija izmedju frekvencije i intenziteta vremenskih neprilika kao sto su poplave i tornadi? Kako je temperatura na kopnu u zavisnosti od temperature na povrsini okeana? Kako predvideti pocetak i kraj uzgajne sezone?

Sta je istrazivanje podataka?

Istrazivanje podataka je proces automackog otkrivanja korisnih informacija u velikim skladistenim podacima. Pronalazi nove i korisne sablone koji bi mozda ostali neotkriveni. Takodje imaju mogucnost da predvide buduca opazanja, kao sto je predvidjanje da li ce novi potrosac potrositi vise od 1000din u radnji.

Nisu sva otkrivanja informacija istrazivanje podataka. Na primer, jednostavni upit data baze ili nalazenje odredjene Web stranice preko pretrazivaca su zadaci oblasti koja se naziva *pronalazenje informacija*. Oni jesu veoma korisni ali se oslanjaju na tradicionalne algoritme i strukture podataka.

Istrazivanje podataka i otkrivanje znanja

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

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

Uloga **predprocesiranja** je da transformise sirove podatke u radne podatke koji su spremni za analizu. Ovo ukljucuje spajanje podataka sa vise izvora, ciscenje podataka od suma i duplikata, biranje karakteristika koji su relevantni za istrazivanje podataka.

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

Izazovi u istrazivanju podataka

Skalabilnost

Skupovi podataka se cuvaju u gigabajtima, terabajtima, pa cak i petabajtma. Zbog taga tehnike istrazivanja podataka moraju biti skalabilne. Mnogi algoritmi koriste specijalne strategije pretrage, pa cak i implementacije novih struktura podataka koji pristupaju slogovima efikasno.

Velika Dimenzionalnost

Sada je cesto da se nadju skupovi podataka sa stotinama ili hiljadama atributa. Za neke tradicionalne algoritme podataka, njigova kompleksnost se povecava sa povecanjem dimenzija (broja atributa). Takodje, neki uopste ne daju dobre rezultate.

Heterogeni i kompleksni podaci

Tradicionalni metodi analize podataka se primenjuju na skupove podataka koji imaju atribute istog tipa. Kako se uloga istrazivanja podataka povecava, povecava se potreba za obradjivanje heterogenih atributa. Takodje pojavljuju se i mnogi kompleksni podaci, kao sto su XML dokumenti, grafovi...

Pripadnost i distribucija podataka

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

Netradicionalna analiza

Za razliku od tradicionalnih statistickih 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 hipoteze, noviji metodi analize podataka generisu i evaluisu hiljade hipoteza, a i mnoge tehnike su napravljene tako da automatizuju ovaj proces.

Nastanak istrazivanja podataka

Istrazivanje podataka se oslanja na idejama kao st su

1. uzorkovanje, ocenjivanje, i testiranje hipoteza iz statistike
2. algoritmi pretrage, tehnike modelovanja, i teorija ucenja iz vestacke inteligencije, prepoznavanje sablona, i masinsko ucenje.

Takodje potrebne su i dodatne oblasti racunarstva kao sto su sistemi baza podataka, paralelnog izracunavanja, distributivno programiranje.

Zadatak istrazivanja podataka

Zadatak predvidjanja. Predvidja vrednost nekog atributa bazirano na vrednostima drugih atributa. Atribut koji se predvidja naziva se **target** ili **zavisna promenljiva**, dok atributi koji se koriste za predvidjanje se nazivaju **opisni** ili **nezavisne promenljive**.

Zadatak opisivanja. Izvlaci sablone koji sumiraju relacije izmedju podataka.

Model predvidjanja se odnosi na izgradnju modela za target promenljive kao funkcije koja prima ulazne promenljive. Postoje dva zadatka modela predvidjanja: **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 izmedju predvidjenje vrednosti i istinite vrednosti target promenljive.

Primer (Predvidjanje vrsta Irisa). Za dati skup podataka koji predstavlja cvet irisa, mozemo odrediti vrstu irisa na osnovu duzine i sirine latica.

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

Primer (Analiza korpe). Na osnovu podataka o kupovinama proizvoda mozemo zakljuciti da ako je potrosac kupio Pampres, onda je i kupio Mleko, pa imamo sledece pravilo `{Pampers}->{Mleko}`.

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

Primer (Klasterovanje dokumenata). Mozemo da klasterujemo artikle bazirano na njihovoj upotrebi. Na osnovu broj ponavljanja odredjene reci iz opisa artikla mozemo da zakljucimo svrhu tog artikla. Na primer, ako sadrzi reci kao sto je `medicinski, pacijent, lek, zdravlje,...` mozemo ove artikle smestiti u jedan klaster.

Otkrivanje anomalija je zadatak identifikovanje podataka cije su karakteristike znacajno drugacije od ostalih podataka. Takvi podaci se poznati kao *anomalije* ili *autlajeri*. Pri ovom procesu moramo sto preciznije odrediti anomalije, u smislu da ne smemo oznaciti normalne objekte kao anomalije, i suprotno.

Primer (Kradja kreditne kartice). Banka skuplja podatke o transakcijama korisnika kreditne kartice, zajedno sa licnim informacijama korisnika. Na osnovu toga, moze 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 sadrze neke vremenske nizove.
2. **Kvalitet podataka**. Podaci su daleko od prefektnog. Ako se poboljsa kvalitet podataka vrlo cesto se boljsa i rezultat analize. Treba otkloniti prisustvo suma, autlajere, duplikate, podatke zasnivane na sklonosti, ili druge fenomene.
3. **Koraci preprocesiranje kako bi napravili zgodnije podatke za istrazivanje podataka**. Treba modifikovati podatke tako da se uklope u odgovarajuci algoritam.
4. **Analiziranje podataka u smislu njegovih relacija**. Jedan pristu analiziranju podataka je pronalazenje relacija izmedju podataka i primenjivanje anlize nad tim relacijamo, a ne na samim objektima.

Tipovi podataka

Skup podataka je kolekcija **objekta podataka** (*slogova*, *tacaka*, *sablona*, *dogadjaja*, *slucaja*, *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	Prosecna Ocena	...
mi18083	1	9.32	...
mi17083	4	6.21	...
...	...	...	...

Atributi i Mere

Sta je atribut?

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

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

Definicija: **Merna skala** je pravilo (funkcija) koja je pridruje numericku ili simbolicku 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 pridruze zaposlenom. Ovi atributi se mogu predstaviti kao celi brojevi. Razumno je pricati o prosecnoj godini zaposlenih, ali nije razumno pricati o prosecnom IDu. Zapravo, jednino sto hocemo da znamo pomocu ID atributa je da li su isti ili razliciti, tj. jedina operacija koja moze da se pridruzi ID atributu je provera jednakost.

Primer (Duzina linijskog segmenata). Svakom linijskom segmentu mozemo da dodelimo neku vrednost koja ce oznacavati njegovu duzinu. Postoji bar dva nacina da ovo uradimo. Jedan je da ih mapiramo tako da se ocuvamo poredak duzina. Drugi nacin je da ocuvamo odnos izmedju duzina. Drugi nacin jasno opisuje i prvi nacin, pa atribut mozemo meriti na nacin na koji ne opisuje sve osobine atributa.

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

Razliciti tipovi atributa

Sledece osobine (operacije) brojeva se koriste za opisivanje atributa

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

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

Na osnovu ovih operacija mozemo definisati razlicite tipove:

Tip Atributa	Opis	Primeri	Operacije
Nominalni (Imenski)	To su samo imena na kojima se moze primeniti *razlicitost*	boja ociju, id, postansk broj	mode, entropija, pripadnostna korelacija
Ordinalni (Redni)	Informacije koje nam pruzaju i *poredak*	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 atribute nazivamo **kategoricki** ili **kvalitativni** atributi i o njima mislimo kao o simbolima, dok intervale i razmerne atribute 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 atribute 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 prelaske. 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 asimetricnih polja, gde su atributi sve moguce 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 asimetricni 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

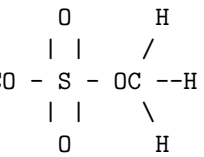
Grafovski podaci

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 molekoli, 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 indivudualnih 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 prosecna 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 atribute. 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 atribute.

Kvalitet podataka

Istrazivanje podataka se cesto primenjuje nad podacima koji su prikupljani 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 uredjaja za merenje da izmeri realni objekad precizno, ta razlika izmerene i stvarne vrednosti se naziva **greska**.

Greske pri prikupljanju podataka se odnose na greske kao sto je ne popunjavanje odredjenog polja, atributa, ili cas i celog sloga.

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

Sum i Artifakti

Sum je nasumicna komponenta nekog merenje. Sum obicno postoji u vremenskim serijama i prostornim podacima. Iako postoji mnogi merni uredjaji u sebi imaju metod za otklananje sum, algoritmi istrazivanja podataka se dizajniraju tako da mogu da se bore sa sumom.

Deterministicne greske podataka, kao sto je ogrebotina slike, nazivaju se artifakti.

Preciznost, Pristrasnost, i Tacnost

Definicija: **Preciznost** je pribliznost pri ponovljenom merenju.

Definicija: **Pristrasnost** je sistematska varijacija merenja of kolicine koja se meri.

Preciznost se obino meri standardnim odstupanjem, dok se pristrasnost meri razlikom ocekivane vrednosti sa pravom vrednoscu kvantiteta koji se meri. Na primer, ako merimo teg mase $1kg$ \$5% puta, i dobijemo sledece vrednosti $\{1.015, 0.990, 1.013, 1.001, 0.986\}$, tada je ocekvanje 1.001 pa je pristrasnost 0.001 i preciznost je 0.013 kako je to standardno odstupanje.

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

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

Autlajeri (Nepodobni)

Nepodobni podaci su ili

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

Nedostajuce vrednosti

Nedostajuce vrednosti predstavljaju polja u skupo podataka koja su prazna. Prazna polja mozemo da imamo ukoliko ta vrednosti nije prikupljena, na primer, ako osoba nije htele 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 nacin je eliminisati slogove ili atribute tamo gde imamo neku nedostajucu vrednost. Mana ovog pristupa je to sto ukoliko imamo puno nedostajucih vrednosti nije moguce dobiti dobar rezultat analize kako gubimo puno informacija. Prednosti eve metode jeste to sto ukoliko ima veoma malo nedostajucih vrednosti brisanjem nekoliko slogova ne utice na analizu, ali ovo se ipak treba raditi sa oprezom, jer cak i tada oni mogu imati kljucne informacije za analizu.

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

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

Nekoinzistentne vrednosti

Skup podataka moze da ima nekoinzistentne vrednosti. Moguce je da je doslo do zamene dve cifre pri unosu podataka, ili je pogresno seknirana rucno napisana cifra, itd. Za ovakve probleme moramo da imamo odgovarajuce metode pronalazenja i ispravljanje ovih gresaka. Neki nekoinzistentne vrednosti se lako otklanjaju, kao sto je, na primer, broj godina neke osobe ne moze biti negativan. Za lakse otkrivanje ovih gresaka dobro je znati domen svakog atributa. Za ispravljanje obicno moramo imati dodatnu informaciju o vrednostima nekog atributa.

Duplikati

Skup podataka, takodje, moze 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 slicna sloga zapravo predstavljaju dva razlicita objekta.

Problemi pri primenama podataka

Skup podataka je visokog kvaliteta ako se moze koristiti za svoje nemene. Ovakav pristu se pokaza veoma korisnim. Ali, takodje i za ovakve skupove podataka postoje problemi:

Starost. Puno podataka postaje staro cim se prikupi, kao sto je na primer, pretrazivanje weba. Ako su podaci stari, onda je bilo kakv model ili sablon prepoznat nad njima takodje star.

Relevantnost. Dostupni skupovi podataka moraju biti relevantni za svoju primernu. Ako na primer ispitujemo saobracajne nesrece, onda ukoliko nemamo informaciju o broju godina vozaca i/ili o polu vozaca, vrlo verovatno nasa analiza nece biti toliko tacna. Takodje, kao sto su atributi bitni, bitni su i slogovi, jer moze doci do **pristrasnosti pri uzorkovanju**, tj. ako pri uzorkovanju dobijamo podatke od osoba koje hoce raditi anketu.

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

Predprocesiranje podataka

Predprocesiranje podataka je siroka 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 vise objekta spajaju u jedan objekat. Razmotrimo skup podataka koje predstavlja transakcije u prodavnicama u raznim gradovima za razlicite dane u godini. Jedan nacin da se izvrsi 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 ociglednog problema, sta ce biti ostale vrednosti atributa, kao sto je cena, i proizvod. Cene mozemo sumirati, dok proizvode mozemo spojiti u novi skup koji sadrzi proizvode iz svih gradova. Kvantitivni atributi se spajaju sumiranjem ili prosekom, dok se kvalitativni atributi spajaju uniraju.

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

Prednosti agregacije su to sto ce istrazivanje podataka da se vrsi na skupu podataka koji je dosta manji, pa ce zauzimati menje memorijskog prostora i samim tim ce izracunavanje biti brze. Takodje, agregacija moze da posluzi kao menjanje oblasti koje podaci pokrivaju, sa uskog na siroko. Agregacija poboljsava stabilnost podataka. Mane agregacije su to sto mozemo izgubiti detalje koji mogu biti bitni.

Uzorkovanje

Uzorkovanje je odabir podskupa od skupa podataka nad kojim ce se vrsiti analiza. Uzorkovanje u statistici i istrazivanju podataka se razlikuje u tome sto kod statistickih analiza vremenski je ogranice sakupljanje podataka, dok je u istrazivanju podataka to iz razloga zato sto 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 priblizne vrednosti osobina kao i originalan skup podataka. Ako nam je osobina ocekivanja bitna, onda je uzorak reprezentativan ako ima priblizno ocekivanje celom skupu podataka.

Pristupi uzorkovanju

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

1. **Bez** vracanja — kada izaberemo neki objekat ne vracamo ga nazan u **populaciju**.
2. **Sa** vracanjem — objekte ne izbacujemo iz populacije, pri odabiru.

Kada populacija sadrzi objekte koji su drugacijeg tipa, i pri tome imamo veliku razliku u broju tipova, nasumicno biranje uzorka nece lepo raditi, kako moze da ne izabere objekte nekog tipa koji su znacajni za analizu, na primer, pri klasifikaciji. Zato se koristi **stratifikovano uzorkovanje**, koje uzima u obzir grupe u kojima objekti pripadaju. Najednostavnije je birati isti broj objekta iz svake grupe. Malo slozenija varijacija je biranje objekata proporcionalno velicini grupe.

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

Progresivno uzorkovanje

Odgovarajucu velicinu uzorka je tesko odrediti, pa se **adaptivno** ili **progresivno uzorkovanje** koristi. Ovaj pristup podrazumeva da se krene sa malim uzorkom, i da se velicina uzorka progresivno povecava vremenom, dok se ne dobije odgovarajuca velicina. Iako se ova tehnika cini jednostavnom, tesko je odrediti kada stati sa povecavanjem velicine. Na primer, ako imamo prediktivni model, sa povecanjem velicine 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 atribute koje nalaze nove atribute koji su:

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

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

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

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 moguce podatak (mera je disprezija).
4. Svaki sledeci atribut opisuje sto je vise moguce preostalih razlicitosti (mera je disprezija).

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

- Matrica podataka $\mathbf{D'} = \mathbf{DU}$ 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.
- Disprerzija 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 atribute (binarne atribute). Zbog toga je cesto neophodno konvertovati atribute koji su neprekidni u kategoricke (**diskretizacija**), ili neprekidne i kategoricke u binarne.

Binarizacija

Ako imamo m kategorickih aktributa, onda svakom od atributa, dodelimo jedan ceo broj iz intervala $[0, m-1]$. Sada konvertujemo tih m celih brojeva u $n = [\log_2(m)]$ 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 moze da dodje do probleme, i stvaranja veza imezju transformisanih atributa. Stavise, kod nekih analiza su nam potrebani asimetricni binarni atributi. Zbog toga kod asimetricnih 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_2
dobar	0	1	0	0
low	1	0	1	0
zao	2	0	0	1

Diskretizacija neprekidnih atributa

Transformacija neprekidnih atributa u kategoricke atribute zahteva: odredjivanje broja kategorija i odredjivanje mapiranje vrednosti neprekidnoh 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 $x_0 < x_1 < \cdots < x_n$.

Neinformisana diskretizacija. Diskretizacija u kojoj se ne koristi infromacija 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

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

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 odredjenog broja intervala.

Kategoricki atributi sa previse vrednosti

Ako su kategoricki atributi ordinalni(redni) atributi, onda mozemo koristiti tehnike slicne onim za neprekidne atribute. Ali ako imamo nominalne(imenske) atribute, 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 anomalnija. 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 indivudualnih 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 razlictost 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 atribute 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 $r \to \infty$ 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 razlicito ne zadovoljavaju jednu ili vise osobina metrike.

Primer (Ne-metricka razlicitost: Razlika skupova). Definisemo 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). Definisimo 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 sadrze samo binarne atribute se nazivaju **keoficijenti slicnosti**, i tipicno imaju vrednosti imezju 0 i 1. Neka su $\mathbf{x}$ i $\mathbf{y}$ dva objekta koja imaju n binarnih atributa. Njihovim uporedjivanjem 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 koeficijent). 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

$$\mathbf{x} \cdot \mathbf{y} = 5$$

$$\|\mathbf{x}\| = \sqrt{(3 \cdot 3 + 2 \cdot 2 + 5 \cdot 5 + 2 \cdot 2)} = 6.48$$

$$\|\mathbf{y}\| = \sqrt{(1 \cdot 1 + 1 \cdot 1 + 2 \cdot 2)} = 2.24$$

$$cos(\mathbf{x}, \mathbf{y}) = 0.31$$

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 atribute je mera linearne zavisnosti izmedju atributa objekta. **Pirsonov koeficijent korelacije** izmedju dva objekta podataka $\mathbf{x}$ i $\mathbf{y}$, je definisan kao

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

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 unutrasnji proizvod izmedju $\nabla\phi(\mathbf{y})$ i $(\mathbf{x} - \mathbf{y})$.

$D(\mathbf{x}, \mathbf{y})$ moze da se zapise kao $D(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) - L(\mathbf{x})$, gde je $L(\mathbf{x}) = \phi(\mathbf{y}) + \langle \nabla\phi(\mathbf{y}), (\mathbf{x} - \mathbf{y}))\rangle$ jednacina ravni koja je tangentna da funkciju ϕ u tacki $\mathbf{y}$. Pa je Bregmanova divergencija samo razlika izmedju 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 moze da nastane kada se meri rastojanje kada atributi nemaju isti opseg vrednosti. Na primer, jedan atribud ime 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 rastojenje**, koje se koristi kada su atributi korelisani, imaju drugacije domene, i kada je distribucija podataka priblizna normalnoj (Gausovoj).

Definicija: Mahalanobijevo rastojanje izmedju 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 atribute

Prethodne definicje slicnosti su bile bazirane na pristupe koji pretpodstavljaju 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 izmedju 0 i 1. Ovaj pristup moramo izmeniti kako bi radio i za asimetricne atribute.

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

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

3. Izracunati totalnu slicnost izmedju 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 atribute 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

Ulazn 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 sluzi 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 monstrum:

Ime	Temperatura tela	Koza	Morska stvorenja	Vazdusna stvorenja	Ima noge	Hibernira	Klasa
gila monstrum	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 klasifikacionoh 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 sluzi za pravljenje klasifikacionog modele, 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:

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

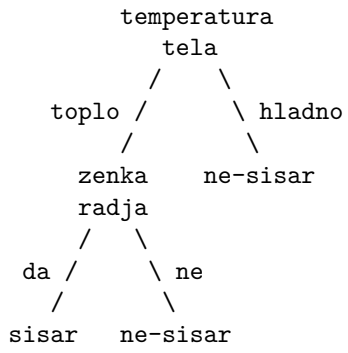
$$\text{Greska} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

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, sadrze 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 sadrzi 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 njig su tacnija od drugih, pa pronalazenje optimalnog drveta je racunski tesko. Zbog toga postoje efikasni algoritmi koji se koriste da pronalazenje, dovoljno tacnog, suboptimalnig 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.

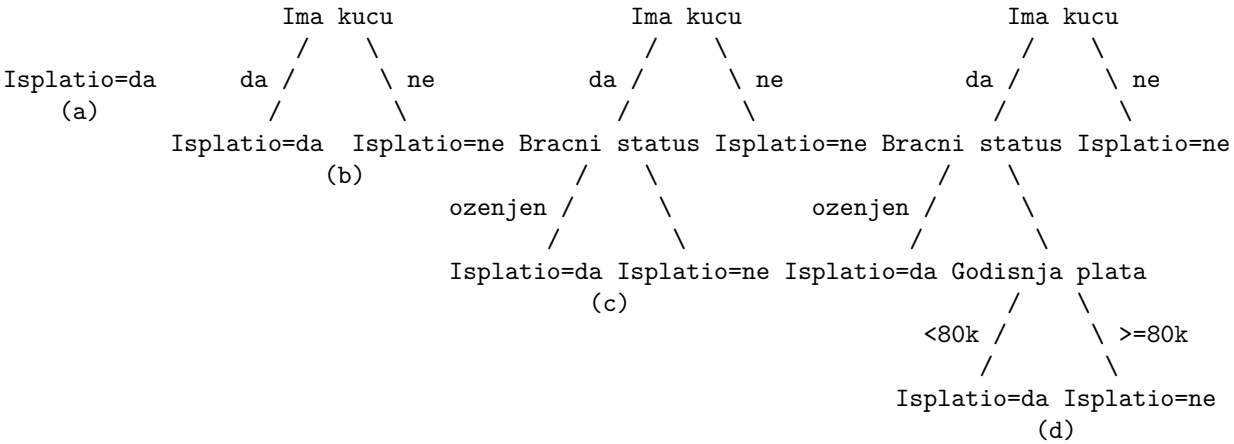
Hantov algoritam

Neka je D_t skup slogova za treniranje koji su povezani sa cvorom t i $y = \{y_1, y_2, \ldots, y_c\}$ 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 nekao osoba isplatite kredit u zavisnosi 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 moguce. Pa se dodaju sledeci uslovi:

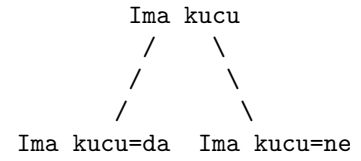
1. Moguce je da neko dete u koraku 2. bude prazno, tj. ne postoji slog koji mu odgovara. U tom slucaju se taj cvor deklarise klasnom oznakom koju ima najveci broj slogova roditeljskog cvora.
2. U koraku 2. ako svi slogovi odgovaraju D_t imaju identicne atribute (osim klasne oznake), nije moguce dalje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

Problemi pri indukovanju drveta odlucivanja

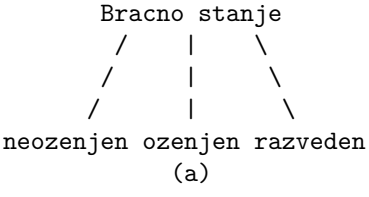
1. **Kako podeliti slogove iz skupa za treniranje?** Svaki rekuzivni korak u rastu drveta mora odabrati uslov testiranja atributa da podelu slogova u manje podskupove. Mora se implementirati algoritam za specifikaciju uslova testiranja atributa, kao i mera za racunanje koliko je svaki od uslova testiranja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

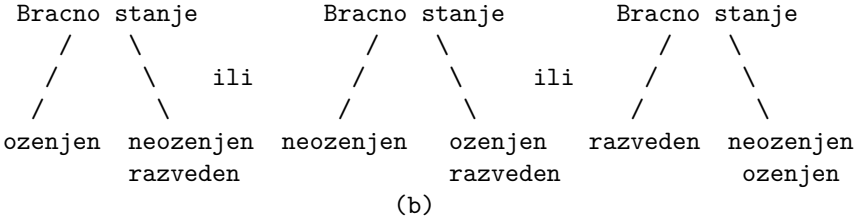
Metodi za izrazavanje uslova testiranja atributa

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

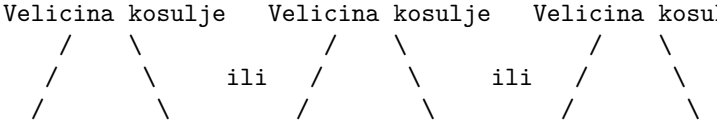


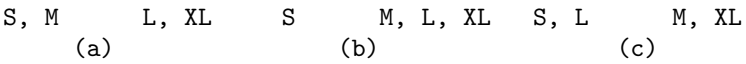
Nominalni(Imenski) atributi. (a) Visestruko razdvajanja podrazumeva da broj resenja zavisi od broja razlicitih vrednosti za odgovarajuci nominalni (imenski) atribut. (b) Binarno razdvajanje podrazumeva $2^{k-1} - 1$ nacina da se naprave binarne particije od k vrednosti atributa.



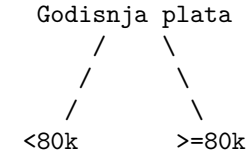


Ordinalni(Redni) atributi. Takodje, pruzaju binarno ili visestruko razdvajanje. Vrednosti se grupisu tako da cuvaju uredjenje. Razdvajanja koja civaju uredjenje su (a) i (b), a razdvajanje koje ne cuva uredjenje je (c).





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



Mera za odabir najboljeng deljenja

Mere za odabri 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

Gini $= 1 - (0/6)^2 - (6/6)^2 = 0$
Entropy $= -(0/6) \log_2(0/6) - (6/6) \log_2(6/6) = 0$
Error $= 1 - \max[0/6, 6/6] = 0$

Cvor N_1	Broj
Klasa=0	1
Klasa=1	5

Gini $= 1 - (1/6)^2 - (5/6)^2 = 0.278$
Entropy $= -(1/6) \log_2(1/6) - (5/6) \log_2(5/6) = 0.650$
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 drveta odlucivanja pokusavaju da maksimizije 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 pojedinacnih 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 atrname $\leq v$ i atrname $> v$ 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 pojedinacne atribute, 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 atribute 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 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 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 izvrsimo **potkresivanje drveta** da bi smanjili njegovu velicinu. Drveta odlucivanja koja su veoma velika su podlozna 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. Nalazenje 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 pruzaju ekspresivni reprezentaciju za ucenje diskretnih funkcija.

6. Drveta odlucivanja dobro podneso 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 nisi kornisni za klasifikaciju, onda moze doci do toga da se oni izaberu pri razdvajanju pa se time drvo nepotrebno povecava. Postoje metodi 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 paraleno 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 uskratile 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 atribute koji predstavljaju aritmeticku ili logicku kombinaciju postojanih atributa. Ovo je racunski jeftinije kako ne moramo dinamicki 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 atribute 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 ocekivana 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 dvodimenionalni 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 *:

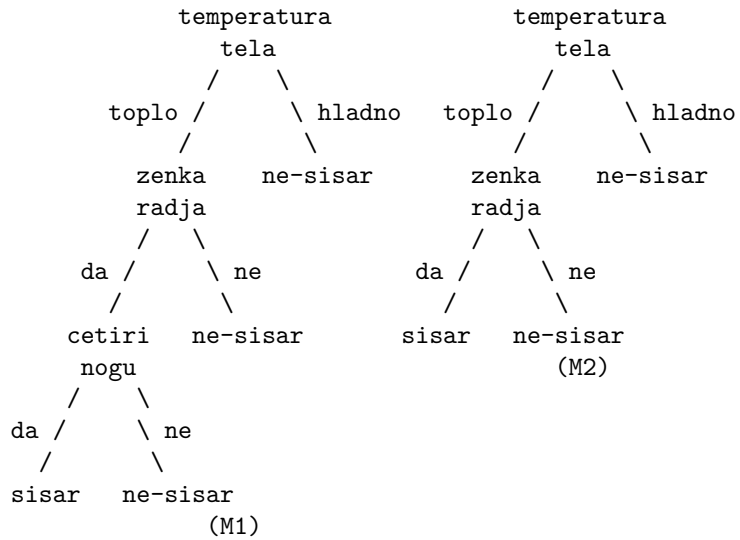
Ime	Temperatura tela	Zenka radja	Cetvoronozna	Hibernira	Klasa
bodljikavo prase	toplo-krvni	da	da	da	da
macka	toplo-krvni	da	da	ne	da
slepi mis	toplo-krvni	da	ne	da	ne*
kit	toplo-krvni	da	ne	ne	ne*
komodo zmaj	hladno-krvni	ne	da	ne	ne

Ime	Temperatura tela	Zenka radja	Cetvoronozna	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	Cetvoronozna	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` savrseno odgovara skupu podataka za treniranje, te nema gresku treniranje. Sa druge strane greska testiranja je 30%: Covek i delfin su pogresno klasifikovani kako jesu sisari ali nisu cetvoronozni, 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` preprilagodio za dati skup treniranja.

Preprilagodjavanje zbog nedostatka reprezentativnih uzoraka

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

Neka je dat sledeci skup podataka za treniranje:

Ime	Temperatura tela	Zenka radja	Cetvoronozna	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 testranja 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 $\gamma_{\max}$. Algoritam ce onda dodati $\gamma_{\max}$ i trenutni model da bi poboljsao njegove performanse. Ova procedura se nastavlja sve dok se sledece poboljsanje ne primeti. Na primer, indukovanje drveta odulicvanja koristi vise testova da odredi koji atributi ce dati najbolje razdvajanje skupa treniranja. Oni koji najbolje razdvajaju atribute se dalje biraju te prosiruju drvo sve dok se ne primeti poboljsanje koje je statisticki znacajno.

Neka je T_0 inicijalno drvo odlucivanje i T_x novo drvo nakon dodavanja unutrasnjeg covora za atribut x. x se moze dodati u drvo ako je primecena dobit, $\Delta(T_0, T_x)$, veci od nekog predefinisanog 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, $\{x_1, x_2, \ldots, x_k\}$, za particionisanje 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 preprilagodjenja.

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 slgova 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.

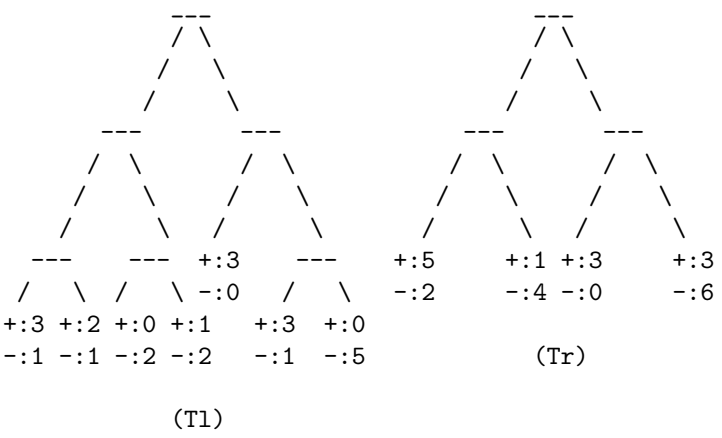
Procenjivanje greske generalizacije

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

Procenjivanje resuptitucijom

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 `Tl` je $e(T_l) = 4/24 = 0.167$, dok je greska drveta odlucivanja `Tr` je $e(T_r) = 6/24 = 0.25$. Zbog toga na osnovu procenjivanja resuptitucije, 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 drveta. 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 $e_g(T_l) = 10/24 = 0.417$. 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 **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 **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 drveta T_r prosiruje u dva detata u drvetu T_l. Pre razdvajanja greska cvora je $2/7 = 0.286$. Aproksimiranje binomne distribucije sa normalnom, gornja granica greske e je:

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

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 vece 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 drveta odlucivanja).

Preprilagodjavanje u indukovanje drveta 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 koriscenom 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 skupo 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 zavisan na strukturu skupova za treniranje i testiranje. Sto je manji skup za treniranje, veca 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 skucaj %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 nedestatak 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 $N \to \infty$, onda se ova verovatnoca ponasa kao $1 - e^{-1} = 0.632$, pa iz toga imamo da ce skup zadrzati oko 63.2 slogova iz originalnog skupa. Tacnost indukovanog modela dobijenog koriscenjem bootstrap tehnike je ϵ_i. Ova procedura moze da ima b ponavljanja.

Postoje nekoliko mogucnosti za racunanje ukupne tacnosto. 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 pouzdanja intervala tacnosti za dati model.
2. Da li je moguce objasniti razlike u tacnosti kao rezultat varijacija u skupovima za testiranje? Ovo pitanje se odnosi na problem statisticke znacajnosti skupa za testiranje

Procenjivanje intervala pouzdanja za tacnost modela

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

1. Eksperiment sadrizi 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 verovatnoca da X ima odredjenu vrednost 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 za testiranje moze se posmatrati kao binomni eksperiment. Za dati skup za testiranje koji ima N slogova, neka je X broj slogova koji je predvidjen tacno datim modelom, 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 pouzdanja kao:

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

, gde su $Z_{\alpha/2}$ i $Z_{1-\alpha/2}$ gornja i donja granica standardne normalne distribucija nivao pouzdanja $(1-\alpha)$. 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 pouzdanja za stvarnu tacnost u 95% nivou pouzdanja. Nivo pouzdanja od 95% odgovara $Z_{\alpha/2} = 1.96$ iz tabele. Iz ovoga dobijamo da je interval pouzdanja izmedju 71.1% i 86.7%. Kako broj slogova skupa za testiranja N raste tako interval pouzdanja 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 izmejdu 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 σ_d^2. Disperzija i ocekivanje od d mogu se izracunati kao:

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

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:

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

za nivo intervala pouzdanja 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 indukovan 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 ocekivanje d_t^{cv}, sto je prava razlika njihih greski, i disperziju σ^{cv}. Celokupna disperzija i ocekivanje mogu da se procene kao:

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

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

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 (Tempteratura=toplo-krvna) -> Sisar
r4: (Radja=ne) and (Leti=ne) -> Gmizavci
r5: (Pliva=semi) -> Vodozemac
```

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

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

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

$$Condition_i = (A_1 \ op \ v_1) \wedge (A_2 \ op \ v_2) \wedge \ldots \wedge (A_k \ op \ v_k)$$

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

Pravilo r *prekriva* slog x ako se preduslov od r poklapa sa atributima od x. Takodje za r se kaze da *pali* ili *pokrece* sve slogove koje pokriva. Neka su data sledeca 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 preduslod zadovoljen sokolovim atributima. Ali ovo pravilo ne prekriva drugu zivotinju, grizlija, zato sto grizli radja i ne moze da leti, pa krsi 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 odnost broja slogova iz D koja pravilo r pokrece i ukupnog broja slogova iz D. Dok je *tacnost* ili *faktor samopuzdanja* se definisa kao odnost 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 sledece 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 praviloa r_4 i r_5. Kako su posledica ovih pravila dve razlicite klase, onda

moramo razresiti ovaj problem.
- Treca zivotinja, ajkula, ne pali ni jedno pravilo. U ovom slucaju moramo ipak napraviti nekakvo predvidjanje 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 najvise jednim pravilom iz R.

Iscrpljujuca pravila. Skup pravila R ima iscrpljujuce 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 ce 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 konflik moramo resiti:

Uredjena pravila. Skup pravila uredjujemo po prioritetu. Uredjeni skup pravila se jos i naziva **lista odlucivanja**. Dati slog se klasifikuje najvecim 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 najveci broj glasova.

Seme uredjenja-pravila

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

Seme uredjenja baziranie na klasama. Pravila koja imaju istu posledicu, tj. predvidjaju 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?

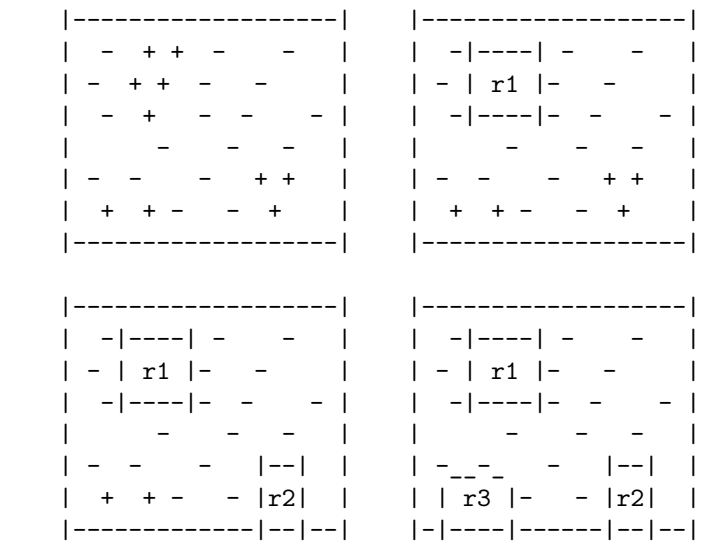
Da bi napravili klasifikator zasnovan na pravilima moramo izdvojimo skup pravila koje indentifikuju veze izmedju atributa skupa podataka i oznaka klasa. Postoje dva nacina za izdvajanje klasifikacionih pravila: (1) *Direktni metod* - izdvaja pravila direktno iz skupa podataka, i (2) *Indirektni metod*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



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 kvaliteta pravila. Ovaj proces staje kada se zadovolji neki uslov, na primer, dodata konjukcija ne poboljsava kvalitet pravila.

U slucaju generalno-u-specificno, 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
r1: poklapa 50 pozitivnih primera i 5 negativnih primera
r2: 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 jadno pokrivanje.

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

gde je k broj klasa, f_i je frevenvija 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 $e_- = 55 \times 100/160 = 34.375$. Onda 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 $e_- = 2 \times 100/160 = 1.25$. Onda 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 $3/4 = 75\%$.

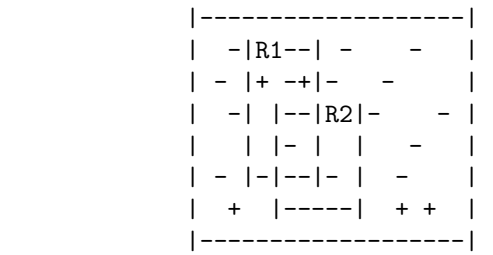
3. Evaluaciona metrika koja uzima u racu broj pozitivnih primera koje poklapa pravilo. Primer ove metrike je **FOILova informaciona dobit**. Neka pravilo $r : A \to +$ poklapa p_0 pozitivnih primera i n_0 negativnih primera. Nakon dodavanja nove konjukcije B, prosireno pravilo $r' : A \wedge B \to +$ 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.



U ovom slucaju prvo je generisan pokrican R_1, ukoliko ne eliminisemo pozitivni i negativne primere koje on poklapa moze se desiti da se u sledecoj generaciji generise pokrivac 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 *podcenimo* 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.

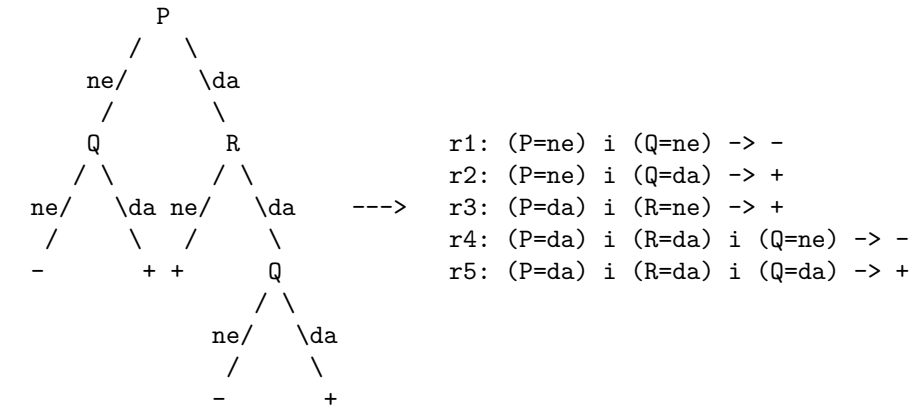
Za dvoklasne probleme, RIPPER bira klasu vecine, i uci pravila tako sto opaza klasu manjine. Za viseklasne probleme, klase se uredjuju prema njihovoj frekvenciji. Neka je $(y_1, y_2, \ldots, y_c)$ uredjene klase, gde je y_1 najmanje frekventna klasa i y_c najvise frekventna klasa. Tokom prve iteracije, 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, izdavaja 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 $ABCD \to y$, 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 pravila. Svaki uslov testiranja na top putu predstavlja jednu konjukciju pravila, dok klasna oznaka lista predstavlja posledicu pravila.



Primer: Razmotrimo pravila:
$r_2 : (P = ne) \wedge (Q = da) \to +$
$r_3 : (P = da) \wedge (R = ne) \to +$
$r_5 : (P = da) \wedge (R = da) \wedge (Q = da) \to +$
Pravila r_2 i r_5 mozemo da zamenimo sa jednim pravilom, jer kad god je $Q = da$, to klasifikujemo kao $+$. Pa dobijamo:
$r_2' : (Q = da) \to +$
$r_3 : (P = da) \wedge (R = ne) \to +$

Generisanje pravila. Klasifikaciona pravila se prosiruje za svaku putanju od korena do jednog od listova drveta odlucivanja. Za dato pravila $r : A \to y$, razmatramo pojednostavljeno pravilo $r' : A' \to y$, gde je A' dobijeno izbacivanjem nekih konjukcija iz A. Pojednostavljeno pravila sa najmanjom pesimistickom 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 najveci prioritet zato sto se ocekuje da sadrze najbolji skup pravila. Ukupna opisna duzina za klase je data sa $L_{\text{exception}} + g \times L_{\text{model}}$, gde je $L_{\text{exception}}$ 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 zasnova na uredjenju, kao sto je RIPPER, je dobra za skupove podataka koji imaju nebalansiranu distribuciju klasa.

Klasifikator: Najblizi sused

Postupak klasifikacije moze se opisati u dva koraka: (1) induktivni korak za konstruisanje modele pomocu podataka za treniranje, i (2) dekuktivni korak za primenu modela nad podacima za testiranje. Drveta odlucivanja i klasifikatori bazirani na pravilima su primeri **zeljnog uceneja**, jer uce model koji mapira ulazne atribute u neku klasnu oznaku cim 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 nece biti klasifikovani ukoliko se ne nalaze u skupu za treniranje.

Ucenje napamet moze 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 nesto 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 previse malo, onda je najblizi sused moze biti osetljiv na pretreniranje zbog suma u skupu za treniranje. Sa druge strane, ako je k previse velika, najblizi sused moze 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
```

Karakteristeke 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. Ucenje bazirano na instancama zahteva meru slicnosti za bi pronaslo slicnost ili razdaljinu izmedju instanci i klasifikacionu funkciju koja vraca predvidjenu 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 moze 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 osetljv na sum.
- Klasifikator najblizi sused moze da proizvede granicu odlucivanja bilo kog oblika, sto je vise fleksibilnije od drveta odlucivanja i klasifikatorima zasnovanim na pravilima, cije su granice odlucivanja pravougaone.
- Klasifikator najblizi sused moze 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 vise 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 verovatnca 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 podatek), 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: $\mathbf{X}$ = (`Ima kucu=ne`, `Bracni status=ozenjen`, `Godisnja plata=120k`) Da bi klasifikovali ovaj slog potrebno je izracunati zadnju verovatnocu $P(\texttt{da}|\mathbf{X})$ i $P(\texttt{ne}|\mathbf{X})$ na osnovu informacija iz skupa za treniranje. Ako je $P(\texttt{da}|\mathbf{X}) > P(\texttt{ne}|\mathbf{X})$, 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 odnost 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 $\mathbf{X} = \{X_1, X_2, \ldots, X_d\}$ 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 vazi:

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

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 atribute

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 $P(\texttt{Ima kucu=da}|\texttt{da}) = 3/7$. Slucno, uslovna verovatnoca onih koji nisu isplatili kredit, i koji su neozenjeni je $P(\texttt{Bracni status=neozenjen}|\texttt{ne}) = 2/3$.

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 strene, 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 mu_{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

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

Za dati slog iz testiranje sa godisnjom platom od $120k$, mozemo izracunati njegovu klasno-uslovnu verovatnocu kao:
$P(\texttt{Godisnja plata=120}|\texttt{da}) = \frac{1}{\sqrt{2\pi 2550}} \exp\left(-\frac{(120-110)^2}{2\cdot 2550}\right) = 0.0077$

Desni deo ove jednacine 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(Ima kucu=da|da) = 3/7
P(Ima kucu=ne|da) = 4/7

P(Ima kucu=da|ne) = 0
P(Ima kucu=ne|ne) = 3/3=1

P(Bracni status=neozenjen|da) = 2/7
P(Bracni status=ozenjen|da) = 4/7
P(Bracni status=razveden|da) = 1/7

P(Bracni status=neozenjen|ne) = 2/3
P(Bracni status=ozenjen|ne) = 0
P(Bracni status=razveden|ne) = 1/3

Godisnja plata:
da -> jednostavna sredina = 110
      jednostavna varijansa = 2550
ne -> jednostavna sredina = 90
      jednostavna varijansa = 16.6
```

Neka je $\mathbf{X} = (\texttt{Ima kucu=ne}, \texttt{Bracni status=ozenjen}, \texttt{Godisnja plata=120k})$ slog iz skupa za testiranje ciju klasu hocemo da predvidimo, tj. hocemo da izracunamo zadnje verovatnoce $P(\texttt{da}|\mathbf{X})$ i $P(\texttt{ne}|\mathbf{X})$. Znamo da je $P(\texttt{da}) = 0.7$ i $P(\texttt{ne}) = 0.3$, a klasno-uslovnu verovatnocu racunamo kao:
$P(\mathbf{X}|\texttt{da}) = P(\texttt{Ima kucu=ne}|\texttt{da})P(\texttt{Bracni status=ozenjen}|\texttt{da})P(\texttt{Godisnja plata=120k}|\texttt{da}) = 0.0018$
$P(\mathbf{X}|\texttt{ne}) = P(\texttt{Ima kucu=ne}|\texttt{ne})P(\texttt{Bracni status=ozenjen}|\texttt{ne})P(\texttt{Godisnja plata=120k}|\texttt{ne}) = 0.$

Na osnovu toga imamo za zadnju verovatnocu klase da da je $P(\texttt{da}|\mathbf{X}) = \alpha \times 0.7 \times 0.0018 = 0.00126$, gde je $\alpha = 1/P(\mathbf{X})$. Slicno, zadnja verovatnoca klase ne je $P(\texttt{ne}|\mathbf{X}) = \alpha \times 0.3 \times 0 = 0$. Kako vazi $P(\texttt{da}|\mathbf{X}) > P(\texttt{ne}|\mathbf{X})$, 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-procene* 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 parametetar 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 velicnine 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:

  $$P(A = 0|Y = 0) = 0.4, P(A = 1|Y = 0) = 0.6,$$
  $$P(A = 0|Y = 1) = 0.6, P(A = 1|Y = 1) = 0.4,$$

  gde je A binarni atribut i Y binarna klasna promenljiva. Sada predpostavimo da je drugi binarni atribut B savrseno korelisan 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:

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

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. Prosecna duzina krokodila je oko $5m$, dok je prosecna duzina aligatora oko $4m$. Predpostavljamo da duzina aligatora i krokodira ima normalnu raspodelu cije je standardno odstupanje 1:
$P(X|\texttt{Krokodil}) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(X-5\right)^2\right]$
$P(X|\texttt{Aligator}) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(X-4\right)^2\right]$

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

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

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 povrsine ispod zadnje verovatnoce za krokodile i povrsine ispod zadnje verovatnoce za aligatore:
Error $= \int_0^{\hat{x}} P(\texttt{Krokodil}|X)dX + \int_{\hat{x}}^{\infty} P(\texttt{Aligator}|X)dX$
Ukupna greska se naziva **Bajasova greska**.

Bajasova mereza poverenja

Vestacke neuronske mreze

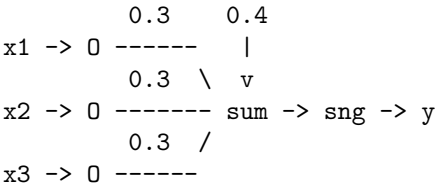
Vestacke neuronske mreze (ANN) su inspirisane simuliranjem bioloskog neuronskog sistema. Ljudski mozak sadrzi nerve koji se nazivaju **neuroni**. Neuron je povezan sa drugim neuronom preko vlakna koja se naziva **akson**. Aksoni sluze da prebace impulse jednog neurona na drugi. Neuroni su povezani preko 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 atribute, 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.



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 matematicki aparat koji racuna tezinsku sumu ulaza, oduzima pristrasnost, te primenjuje funkciju znaka. Tacnije, izlazni cvor se moze predstaviti matematicki kao:

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

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

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

gde je $w_0 = -t, x_0 = 1$ i $\mathbf{w} \cdot \mathbf{x}$ skalarni prizvod 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 $w_i^{(k)}$ tezina 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 tezina 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 povecamo 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 povecamo tezinu svih grana sa negativnim ulazom.

Tezinu u svakom azuriranju ne smemo povecati previse, kako ne bi doslo do pregazivanje prethodnih azuriranje, pa se zbog toga koristi parametar brzine ucenja λ, koji uzima vrednost izmedju 0 i 1. Ako je λ bizu nule, onda stara tezina veoma utice na novu tezinu, sa druge strane, ako je λ blizu jedinice, onda nova tezina 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 odvojiv. Ako problem nije linearno odvojiv 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 moze sadrzati nekoliko slojeva imedju ulaznog i izlaznog sloja. Takvi unutrasnji slojevi se nazivaju **skriveni slojevi** i cvorovi unutar ovih slojeva se nazivaju **skriveni cvorovi**. Takva struktura se naziva viseslojna neuronska mreza. U neuronskoj mrezi sa **propagacijom-unapred**, cvorovi jednog sloja su povezani samo sa cvorovima sledeceg sloja. Perceptron je jednoslojna neuronska mreza sa propagacijom-unapred, jer ima samo jedan sloj cvorova i izlazni cvor se ponasa kao matematicka funkcija. U **rekurentnim** neuronskim mrezama moguce je imati grane koje povezuju cvor nekog sloja, sa cvorom iz nekog prethodnog sloja.
2. Mreza moze da koristi i druge tipove aktivacionih funkcija koje nisu sinusne funkcije. Primeri drugih aktivacionih funkcija ukljucuju linearnu, sigmoid (logisticku), i hiperbolicku tangens funkciju. Ove aktivacione funkcije dozvoljavaju skrivenim i izlaznim cvorovima da imaju izlaznu vrednost koja nije linearna u odnosu na ulazne parametre.

Zbog ove kompleksnosti mozemo da resimo neodvojive probleme kao sto je XOR problem. Ako konstruisemo mrezu koja ima dva ulazna cvora, dva skrivena cvora, i jedan izlazni cvor, svaki od slojeva mozemo posmatrati kao jedan perceptron koji generise hiperravan nad prostorom. Kombinacija tih hiperravni generise granicu odlucivanja.

Da bi naucili odgovarajuce tezine za ANN model, treba nam efikasan algoritam koji konvergira u tacno resenje kada imamo dovoljno podataka u skupu za treniranje. Jedno resenje koje se namece jeste da koristimo istu tehniku kao za perceptron i da pri tome delimo mrezu na perceptrone. Ovaj metod nece raditi kako nemamo nikakvo znanje od izlazima skrivenog sloja, pa ne mozemo da izracunamo gresku $(y - \hat{y})$ za svaki od skrivenih cvorova.

Ucenje ANN modela

Ucenje tezina neuronske mreze 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 naci, koristimo gramzivu tehniku gradijentnog spusta gde ce formula azuriranja tezina biti:

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

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

Za skrivene cvorove, racunanje greske nije lako kako moramo da racunamo faktor greske $\partial E/partialw_j$, iako ne znamo koje vrednosti njihov izlaz trebe 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, tezine iz prethodne iteracije se koriste za racunanje izlazne vrednosti svakog neurona u mrezi. Racunanje izlaza neurona se propagira kroz slojeve od ulaznog, do izlaznog sloja, pa otuda ima propagacija unapred. Tokom propagacije unazad, azuriranje tezina se primenjuje u suprotnom smeru. Propagacije unazad omogucava da greske neurona na $k+1$-om sloju procenjuju greske neurona na k-tom sloju.

Problemi pri dizajniranju ANN ucanja

1. Treba odrediti broj cvorova za ulazni sloj. Svakom ulaznom cvoru 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 pomocu $[\log_2 k]$ ulaznih cvorova.
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 redudantnim osobinama zato sto su tezine automacki uce tokom treniranja. Tezine za redudantne 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 ucenje tezina za mrezu, a on vrlo cesto konvergira ka lokalnom minimumu. Jedan nacin da se izbegne loklni 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)