

Mladen Nikolić

Andželka Zečević

MAŠINSKO UČENJE

Beograd
2019.

Sadržaj

Sadržaj	2
1 Uvod	7
I Nadgledano učenje	11
2 Teorijske osnove nadgledanog učenja	13
2.1 Postavka problema nadgledanog učenja	14
2.2 Princip minimizacije empirijskog rizika	15
2.3 Preprilagođavanje	20
2.4 Regularizacija	23
2.5 Nagodba između sistematskog odstupanja i varijanse	27
2.6 Teorijske garancije kvaliteta generalizacije	29
2.7 Veza statističke teorije učenja sa filozofijom nauke	37
2.8 Vrste modela	39
2.9 Dimenzije dizajna algoritama nadgledanog učenja	41
3 Probabilistički modeli	43
3.1 Linearna regresija	43
3.2 Logistička regresija	50
3.3 Multinomijalna logistička regresija	53
3.4 Uopšteni linearni modeli	55
3.5 Naivni Bajesov algoritam	56
4 Modeli zasnovani na širokom pojasu	61
4.1 Metod potpornih vektora za klasifikaciju	61
4.2 Metod potpornih vektora za regresiju	68
4.3 Algoritam k najbližih suseda zasnovan na širokom pojasu	70
5 Modeli zasnovani na instancama	73
5.1 Osnove neparametarske ocene gustine raspodele	73
5.2 Metodi zasnovani na kernelima	77
5.3 Metodi zasnovani na najbližim susedima	87

6 Ansambli	93
6.1 Prosta agregacija	93
6.2 Pojačavanje	98
7 Evaluacija i izbor modela	103
7.1 Mere kvaliteta modela	103
7.2 Tehnike evaluacije i izbora modela	108
7.3 Napomene vezane za pretprocesiranje	113
8 Regularizacija	115
8.1 Proređeni modeli	115
8.2 Modeli složenije strukture i uključivanje domenskog znanja	119
8.3 Učenje više poslova odjednom	121
9 Optimizacija	125
9.1 Gradijentni spust	125
9.2 Metod inercije	129
9.3 Nesterovljev ubrzani gradijentni spust	129
9.4 Adam	130
9.5 Stohastički gradijentni spust	131
10 Neuronske mreže i duboko učenje	135
10.1 Potpuno povezane neuronske mreže	136
10.2 Konvolutivne neuronske mreže	144
10.3 Rekurentne neuronske mreže	154
10.4 Praktične tehnike i napredni koncepti	163
11 Šta ako ne radi?	171
11.1 Preprilagođavanje i potprilagođavanje	171
11.2 Problemi podataka	172
11.3 Greške u pretprocesiranju	173
11.4 Neadekvatnost algoritma	173
11.5 Neadekvatnost optimizacije	174
11.6 Greške u evaluaciji	176
11.7 Greške u interpretaciji modela	177
11.8 Greške u implementaciji	177
II Učenje potkrepljivanjem	179
12 Markovljevi procesi odlučivanja i njihovo rešavanje	181
12.1 Osnovni pojmovi	181
12.2 Rešavanje MDP-a dinamičkim programiranjem	187
13 Učenje u nepoznatom okruženju	189
13.1 Osnovni algoritmi	189

13.2 Funkcionalna aproksimacija	191
III Nenadgledano učenje	197
14 Klasterovanje	199
14.1 K sredina	200
14.2 Mešavina normalnih raspodela i EM algoritam	203
15 Učenje reprezentacije	209
15.1 Metod glavnih komponenti	209
15.2 Autoenkoderi	211
16 Generativni modeli	217
16.1 Generativne suparničke mreže	217
IV Dodatak	223
17 Matematičko predznanje	225
17.1 Sopstvene vrednosti i sopstveni vektori	225
17.2 Definitnost	226
17.3 Norma i skalarni proizvod	227
17.4 Izvod, parcijalni izvod i gradijent	228
17.5 Konveksnost	229
17.6 Lokalni optimumi	230
17.7 Integral	231
17.8 Verovatnoća	232
17.9 Sredina i rasipanje slučajne promenljive	234
17.10 Statističke ocene i njihova svojstva	236
17.11 Statistički modeli	237
17.12 Metod maksimalne verodostojnosti	237

Na korisnim sugestijama zahvaljujemo se kolegama Milošu Jovanoviću i Jovani Kovačević i studentima Milošu Stankoviću, Blagoju Ivanoviću, Nemanji Mićoviću, Marijani Milenković, Nikoli Dimitrijeviću, Andrijani Marjanović, Rastku Đordjeviću, Slobodanu Jenku...

Glava 1

Uvod

Od početka dvehiljaditih, razvoj veštačke inteligencije je dobio nov zamah. Niz izrazito važnih problema, od kojih se za neke pretpostavljalo da će još dugo ostati van domaćaja, biva rešen. U nekim domenima, u kojima računari do tada po uspešnosti nisu mogli da se porede sa ljudima, postižu se rezultati superiorni u odnosu na rezultate ljudskih eksperata. U srcu ovog novog zamaha, nalazi se *mašinsko učenje*. Iako u prvi plan izbija upravo dvehiljaditih, ova oblast ima dugu istoriju razvoja. Zamišljena u radovima Alena Tjuringa, četrdesetih godina prošlog veka, aktivno se razvija od pedesetih kada je konstruisan *perceptron*, prvi sistem koji uči jednostavne zakonitosti i predstavlja dalekog preteču modernih *neuronskih mreža* koje se uz uspone i padove razvojuju do devedestih, kada primat uzimaju *metod potpornih vektora* i drugi metodi zasnovani na *kernelima*. Ipak, za skorašnji uspon mašinskog učenja, zaslужna je baš renesansa neuronskih mreža koja je dovela do toga da se danas veštačka inteligencija i mašinsko učenje u opštoj percepciji neretko poistovećuju sa njima. Treba imati u vidu da su ove oblasti neuporedivo šire.

Dugi razvoj mašinskog učenja motivisan je s jedne strane željom da se bolje razume ljudski i životinjski potencijal za učenje, koji se nalazi u srcu onoga što nazivamo inteligencijom, a s druge, željom da se takav proces oponaša u praktične svrhe. Druga motivacija bila je dominantna u toku razvoja mašinskog učenja. Delom zbog toga što je taj put bio dostupniji jer razumevanje bioloških mehanizama učenja još nije dovoljno uznapredovalo, a delom jer njihovo razumevanje možda i nije neophodno za rešavanje praktičnih problema. Razvoj mašinskog učenja ima dva žarišta – akademske institucije u kojima je poničlo i u kojima je dovedeno do određenog nivoa upotrebljivosti i privredu koja je u njemu prepoznala potencijal za praktične primene i daje veliki doprinos njegovom razvoju u toku dvehiljaditih godina.

Precizno definisanje naučnih disciplina nezahvalan je i neizgledan, a možda i nepotreban poduhvat. Zato mu i nećemo posvetiti mnogo pažnje. U opštoj percepciji mašinsko učenje predstavlja disciplinu koja se bavi izvođenjem algoritama iz podataka, bez eksplicitnog programiranja. Ovaj pogled naglašava njegovu praktičnu stranu. Ipak, mašinsko učenje ima i svoju fundamentalnu di-

menziju. Kao što se logika bavi proučavanjem *dedukcije* objašnjavajući šta čini neki zaključak potpuno opravdanim i time formalizuje jedan važan vid ljudskog zaključivanja, mašinsko učenje se bavi proučavanjem *indukcije*, odnosno *generalizacije* i time formalizuje drugi vid ljudskog zaključivanja – uopštavanje od ograničenog broja uzoraka ka univerzalnim zaključcima. Ovaj drugi problem se može smatrati i težim. Osnove dedukcije razumeo je (uprkos nekim propustima) Aristotel pre više od dve hiljade godina. Indukcija se ozbiljnije izučava tek od strane Frensisa Bejkona na prelazu sa šesnaestog na sedamnaesti vek. Deduktivno zaključivanje se kroz formalnu logiku proučava od devetnaestog veka i početkom dvadesetog veka već je na čvrstim nogama. Induktivno zaključivanje se, kroz statističku teoriju učenja, u nekoj meri formalizuje tek krajem dvadesetog veka.

Kako i deduktivno i induktivno zaključivanje imaju važnu ulogu u prirodoj inteligenciji, odgovarajuće discipline koje ih automatizuju – automatsko rasudivanje i mašinsko učenje, imaju važne uloge u veštačkoj inteligenciji. Postavlja se pitanje, kada su metode koje od ovih oblasti pogodniji izbor za rešavanje konkretnog problema. Metode zasnovane na logici, koje se razvijaju u okviru automatskog rasudivanja, pogodne su u slučajevima u kojima je problem moguće precizno matematički definisati. Obično se radi o problemima koje čovek može relativno lako da formuliše, ali ih vrlo teško rešava (najčešće zbog kombinatorne eksplozije pri pretrazi prostora mogućih rešenja) i u kojima nisu prihvatljiva pogrešna rešenja. S druge strane, mašinsko učenje je posebno pogodno upravo za suprotnu vrstu problema – probleme koje čovek ne može lako ni da definiše, iako neke od njih čak vrlo lako rešava (neke, s druge strane, ne) i u kojima je prihvatljiva povremena greška. Jedan primer takvog problema je prepoznavanje lica. Osim u slučajevima specifičnih neuroloških poremećaja, svi ljudi su vrlo dobri u rešavanju ovog problema. Čak je neobično nazvati ga problemom i govoriti o njegovom rešavanju. Ipak, ukoliko pokušamo da taj problem precizno definиšemo, naletimo na mnoštvo poteškoća. Prvi naivni pokušaj definisanja bi se verovatno sastojao u nekom opisu poput toga da je lice nešto što se sastoji od nosa, očiju, usta čela, jagodica i obrva. Ovo ne samo što vodi daljem pitanju definisanja tih pojmova, već postavlja i pitanje definisanja njihovih relativnih pozicija i slično i uprkos trudu, bićemo prinuđeni da odustanemo. Stoga se ovakvim problemima ne pristupa metodama automatskog rasudivanja. S druge strane, kao i ljudi, metode mašinskog učenja mogu vrlo uspešno da se nose sa ovim problemom.

Neki od problema na koje je mašinsko učenje uspešno primenjeno su prepoznavanje lica na slikama, prepoznavanje različitih objekata na slikama i videu, prepoznavanje tumora na medicinskim snimcima, autonomna vožnja automobila, autonomno letenje, igranje igara na tabli poput šaha i igre go, ali i računarskih igara kao što je Super Mario ili Doom, klasifikacija teksta, mašinsko prevođenje, automatsko opisivanje sadržaja slike, analiza osećanja izraženih u tekstu, predviđanje razvoja bolesti kod pacijenata i preporučivanje terapije, analiza društvenih mreža, prepoznavanje i sinteza govora i tako dalje. U mnogim od ovih primena, mašinsko učenje je već prevazišlo nivo efikasnosti ljudskih

eksperata. Sve nabrojane primene predstavljaju očigledno primere važnih praktičnih problema, ali iza svih stoji i ozbiljna teorija. Možda je upravo ovaj spoj ključ uspeha mašinskog učenja.

Nabrojani problemi su vrlo raznorodni kako po svojoj prirodi, tako i po metodama mašinskog učenja koje se koriste za njihovo rešavanje. Te metode se mogu razvrstati na mnogo načina, ali osnovna podela bi bila prema prirodi problema učenja. Obično se identifikuju tri grupe problema mašinskog učenja, a to su problemi *nadgledanog učenja* (eng. *supervised learning*), problemi *nenađegledanog učenja* (eng. *unsupervised learning*) i problemi *učenja potkrepljivanjem* (eng. *reinforcement learning*).¹

Nadgledano učenje je verovatno najznačajniji vid mašinskog učenja. Osnovna karakteristika mu je da se podaci sastoje iz parova opisa onoga na osnovu čega se uči i onoga što je iz toga potrebno naučiti. Naziv je motivisan analogijom sa procesom učenja pri kojem učitelj učeniku zadaje zadatke, ali mu nakon njegovih odgovora, radi poređenja, daje i tačna rešenja. Problem prepoznavanja lica na slici bi zajedno sa slikom uključivao i informaciju da li se na slici nalazi lice ili ne. Problem prepoznavanja objekata bi uz sliku uključivao i spisak objekata koji se na njoj nalaze, eventualno i sa njihovim pozicijama. Prilikom mašinskog prevođenja, zajedno sa rečenicom u polaznom jeziku, bila bi data i rečenica na ciljnem jeziku.

Nenadgledano učenje se karakteriše upravo nedostatkom informacije o tome šta je potrebno naučiti. Na prvi pogled, ovakve metode mogu zvučati ili nemoguće ili neuporedivo moćnije od metoda nadgledanog učenja. Kakva je to metoda koja je u stanju da nauči šta treba, a da joj nije rečeno šta treba? Naravno, to je metoda kojoj nedostaje opštost. Metoda pomoću koje se ne može učiti bilo šta, već isključivo nešto za šta je metoda eksplizitno dizajnirana. Zapravo, ovaj vid učenja se obično bavi pronalaženjem neke vrste strukture u podacima, a metode koje na taj način uče su obično napravljene polazeći od konkretne vrste strukture koja se traži. Problem *klasterovanja* je jedan problem nalaženja strukture u podacima – strukture grupa. U mnogim primenama potrebno je identifikovati grupe podataka. Primera radi, grupisati se mogu slični tekstovi, slične slike, akcije čije se cene slično kreću na berzi i tako dalje. U poslednjem primeru, razlog za grupisanje bi recimo mogao biti taj što je za jednu akciju dostupna relativno kratka istorija cena, koja ne omogućava obučavanje modela nadgledanog učenja koji će predviđati buduće cene. Međutim, ako se veliki broj akcija čije se cene slično ponašaju grupiše, podataka može biti dovoljno. Drugi vid pronalaženja strukture u podacima bi bilo učenje reprezentacije podataka. Na primer, podaci koji su visoko dimenzionalni, često zapravo leže u linearnom prostoru ili na nekoj površi značajno manje dimenzije. Izražavanjem polaznih podataka u terminima koordinata u takvim prostorima, smanjuje se njihova dimenzionalnost, što često dovodi do dosta boljih performansi algoritama nadgledanog učenja koji se primenjuju na tim podacima. U oba slučaja,

¹Inspiracija za poslednji prevod uzeta je iz psihologije u kojoj se tako zove odgovarajući način učenja kod životinja i ljudi.

pomenuto je da se nakon primene metode nenadgledanog učenja, na te podatke primenjuju metode nadgledanog učenja. To nije uvek slučaj, ali neretko metode nenadgledanog učenja mogu biti vid pretprecesiranja podataka kako bi se na njima primenile metode nadgledanog učenja.

Učenje potkrepljivanjem je, neformalno rečeno, između prethodna dva posmenuta pristupa. Koristi se u situacijama u kojima je potrebno rešiti neki problem preduzimajući niz akcija, čijim se zajedničkim dejstvom dolazi do rešenja problema. Pretpostavlja se da postoji *agent* (odnosno, neko ko dela) koji opaža tekuće *stanje okruženja*, u mogućnosti je da preduzima *akcije* usled kojih dobija *nagrade* predstavljene numeričkom vrednošću. Ishod učenja je *optimalna politika*, odnosno preslikavanje stanja u akcije koje vodi maksimalnoj (ili, u praksi, dovoljno visokoj) ukupnoj nagradi. Pritom, ključna je pretpostavka da nije poznato koja od preduzetih akcija je bila prava u datom kontekstu, a koja nije. U suprotnom, radilo bi se o problemu nadgledanog učenja. Primera radi, razmotrimo problem autonomne vožnje. Agent je sistem koji vozi automobil i koji je u stanju da opaža pozicije drugih automobila, pešaka, saobraćajne zname, svetla semafora i slično (a što čini okruženje), a koji je u stanju da menjaj smer kretanja i da povećava i smanjuje brzinu kretanja automobila (što su akcije). Agent pritom dobija nagrade koje su recimo 1 za svaki kilometar pređenog puta, 100 za stizanje na cilj, -100 u slučaju sudara i -1000 u slučaju smrtnog ishoda u saobraćaju. Optimalnu politiku nije lako opisati na osnovu ličnog znanja (što je tipičan razlog za upotrebu mašinskog učenja!), ali ona bi verovatno uključivala kočenje na žutom i crvenom svetlu ili pred pešacima, skretanje tamo gde je znakom naznačeno da je obavezno skrenuti itd.

U daljem tekstu, najviše pažnje biće posvećeno nadgledanom učenju, kako zbog njegove najšire primene, tako i zbog razvijenosti fundamentalne teorije.

Deo I

Nadgledano učenje

Glava 2

Teorijske osnove nadgledanog učenja

Kao što je rečeno, nadgledano učenje se karakteriše time da su uz vrednosti ulaza, date i vrednosti izlaza koje im odgovaraju. Potrebno je ustanoviti odnos koji važi između ulaza i izlaza. Na osnovu ovog odnosa se najčešće za neke buduće ulaze vrši predviđanje izlaza. Ulaz i izlaz se najčešće predstavljaju u vektorskom obliku i označavaju sa x i y , pri čemu je x tipično vektor vrednosti nekih promenljivih koje se nazivaju *atributima* (eng. *features*), dok je y tipično jedna promenljiva koja se naziva *ciljnom promenljivom* (eng. *target variable*). Mogući su i mnogo opštiji scenariji. Na primer oni u kojima je y takođe višedimenzionalno, ali i oni u kojima ni x ni y nisu predstavljeni numeričkim vrednostima, već mogu predstavljati sekvene, grafove i slično.

Problemu otkrivanja veze između nekih promenljivih na osnovu opažanja može se pristupiti na različite načine. Na primer, već hiljadama godina naučnici na osnovu opažanja postavljaju hipoteze o odnosima nekih veličina, a onda predviđanja dobijena na osnovu tih hipoteza testiraju u praksi i na osnovu toga odlučuju o verodostojnosti tih hipoteza. Jedan primer takvog odnosa je formula $F = ma$ koja uspostavlja vezu između sile, mase i ubrzanja. Do ove formule se došlo ljudskim uvidom, a na osnovu opažanja iz iskustva. Ovakav pristup je moguć i uobičajen pod pretpostavkom da fenomen i interakcije među razmatranim veličinama nisu previše komplikovani za ljudsko poimanje. Ipak, u vreme kada je uobičajeno da raspolaćemo gigabajtima, pa i terabajtima podataka koji predstavljaju milione ili milijarde opažanja sa desetinama hiljada promenljivih, potrebne su metode koje su u stanju da uočavaju takve veze automatski.

Ovakve metode obično nalaze funkcije koje na neki način izražavaju vezu između vrednosti atributa i vrednosti ciljne promenljive. Ove funkcije i njihova svojstva su od centralnog značaja u mašinskom učenju i nazivaju se *modelima*. Modela može biti (beskonačno) mnogo, ali ne možemo od svih, pa čak ni od jednog, očekivati da savršeno opisuje zavisnosti koje važe među promenljivim. Ono što se od modela očekuje je da dobro *generalizuje*, odnosno da prilikom

predviđanja vrednosti ciljne promenljive na osnovu vrednosti atributa, retko pravi velike greške. Idealan slučaj u kojem grešaka nema, nije realističan i ne dešava se u praksi. Pojam generalizacije je centralni pojam mašinskog učenja i biće mu posvećena posebna pažnja.

Dve osnovne vrste problema nadgledanog učenja su *regresija* i *klasifikacija*. Regresija je problem predviđanja neprekidne ciljne promenljive. Na primer, moguće je predviđati cenu deonica na berzi na osnovu njihovih cena u prethodnih nekoliko dana i globalnih kvantitativnih pokazatelja tržišta ili količinu teških metala u zemljištu na osnovu udaljenosti od zagadivača, udaljenosti od vodenih tokova, vrste zemljišnog pokrivača i slično. Klasifikacija je problem predviđanja kategoričke ciljne promenljive. Kategoričkim se smatraju promenljive koje uzimaju konačan broj vrednosti među kojima nema uređenja. Na primer, prepoznavanje jedne iz skupa poznatih osoba čije se licne nalazi na slici je problem klasifikacije. Prepoznavanje da li se novinski članak tiče ekonomije, sporta ili politike je takođe problem klasifikacije.

2.1 Postavka problema nadgledanog učenja

O vrednostima atributa se može razmišljati kao o okolnostima u kojima nastaje neki ishod koji je predstavljen vrednošću ciljne promenljive. Na primer, ukoliko je dan letnji u ukoliko nema oblačnosti, očekuje se da je temperatura visoka. Ipak, dva merenja temperature po sunčanom letnjem danu se mogu značajno razlikovati. Na primer, zbog razlike u geografskoj širini, udaljenosti od vodenih površina, ili kretanja hladnjih vazdušnih masa. Može se očekivati da uključivanjem većeg broja promenljivih u atribute može biti uočena jača veza sa ciljnom promenljivom. Ipak, u praksi nije realistično da se mogu identifikovati i adekvatno kvantifikovati svi faktori koji igraju ulogu u nekom fenomenu, pa čak i ako fenomen deluje jednostavno. Zbog toga, možemo očekivati da u podacima za iste vrednosti atributa vidimo različite vrednosti ciljne promenljive. Očito, opis veze između atributa i ciljne promenljive se može utemeljiti na pojmovima verovatnoće.

U najopštijem slučaju, pretpostavlja se da je odnos između atributa i ciljne promenljive zadat zajedničkom raspodelom verovatnoće $p(x, y)$. Najčešće, pa i sada, ćemo pod ovim podrazumevati gustinu raspodele. Poznavanje ovog verovatnosnog zakona među promenljivim od interesa predstavlja potpuno znanje o njihovim odnosima. Kako takva raspodela nije dostupna, pristupa se određivanju modela $f(x)$ koji vrednostima atributa pridružuje vrednost ciljne promenljive. Takvih modela može biti puno, ali od značaja je u nekom smislu najbolji takav model. Ipak, pojam kvaliteta je potrebno definisati. Poželjno je da važi $y \approx f(x)$, odnosno da je razlika između pravih vrednosti ciljne funkcije i njihove aproksimacije modelom mala. Otud je prvo potrebno definisati *funkciju greške* (eng. *loss*) koja meri odstupanje predviđenih i stvarnih vrednosti ciljne promenljive. Ovu funkciju ćemo označavati sa L . Ipak $L(y, f(x))$, predstavlja razliku jedne prave vrednosti i jednog predviđanja. Bilo kog, ali pojedinačno.

Ni jedno konkretno x i y nisu dovoljni da opišu kvalitet modela. Umesto toga, potrebno je izraziti kvalitet modela po svim parovima x i y . Greška $L(y, f(x))$ nastaje kada je za neke vrednosti atributa x prava vrednost ciljne promenljive y . Ipak, neke kombinacije x i y su vrlo verovatne, a neke nisu, a ne moraju biti ni moguće. Stoga, postavlja se pitanje da li su baš sve kombinacije vrednosti atributa i ciljne promenljive podjednako važne. Naravno, nisu. Utoliko je važnija greška koja se dešava na verovatnjim kombinacijama. Otud se gustina raspodele ovih kombinacija $p(x, y)$ prirodno koristi za uprosečavanje grešaka, čime se definiše *funkcional rizika*¹ ili *stvarni rizik* ili samo *rizik*:

$$R(f) = \mathbb{E}[L(y, f(x))] = \int L(y, f(x))p(x, y)dxdy$$

Smisao rizika je da predstavlja grešku uprosečenu po svim mogućim podacima.

Kada je pojmom rizika definisan (ne)kvalitet modela, prirodno je tražiti funkciju koja minimizuje rizik, odnosno rešavati problem

$$\min_f R(f)$$

Ovo je *problem nadgledanog učenja* u svom teorijskom obliku. Ipak, ovaj problem nije moguće direktno rešiti u datom obliku iz dva razloga. Prvi je što je gustina raspodele $p(x, y)$ nedostupna. Taj problem je fundamentalan i ne može se ignorisati ni u teorijskim razmatranjima. Drugi problem se tiče skupa funkcija kojem pripada funkcija f . Teorijski, može se razmatrati skup svih mogućih funkcija iz \mathbb{R}^n u \mathbb{R} , gde je n broj atributa (sada i zauvek u ovoj knjizi!), ali takav scenario nije od praktičnog značaja, pošto je pitanje kako bi se rešenje tako formulisanog problema uopšte moglo izraziti na upotrebljiv način.

2.2 Princip minimizacije empirijskog rizika

Svaki metod mašinskog učenja počiva na nekim prepostavkama o skupu potencijalnih modela. Nekad su ove prepostavke implicitne u konstrukciji metoda, a nekad su eksplicitne. Nekad i korisnik ima mogućnost ograničenog izbora tih prepostavki. U nastavku prepostavljamo da postoji određena *repräsentacija modela* kojom je definisan skup svih modela. Ona će najčešće biti parametarska, odnosno model je funkcija $f_w(x)$ koja zavisi od parametara w . Izborom ovih parametara dobijaju se modeli različitih kvaliteta. Otud se nadalje minimizacija po skupu svih funkcija svodi na minimizaciju po skupu svih vrednosti parametara i problem postaje:

$$\min_w R(f_w(x))$$

Predstavljajući rizik kao funkciju parametara, to možemo pisati i kao

$$\min_w R(w)$$

¹Funkcional je preslikavanje vektorskog prostora u skup njegovih skalara. Primera radi, integral integrabilnim funkcijama pridružuje vrednosti iz njihovog kodomena.

Primer reprezentacije modela je recimo *linearna reprezentacija*, koja prepostavlja linearnost modela *po parametrima*. Tipičan linearni model se može zapisati na sledeći način:

$$f_w(x) = w_0 + \sum_{i=1}^n w_i x_i$$

Predstavljanje slobodnog člana w_0 se često izbegava tako što se prepostavi da je vrednost prvog atributa uvek 1.²

Drugi pomenuti problem, problem nedostupnosti gustine raspodele $p(x, y)$ se rešava aproksimacijom na osnovu slučajnog uzorka

$$\mathcal{D} = \{(x_i, y_i) | i = 1, \dots, N\}$$

iz date raspodele. Prepostavlja se nezavisno uzorkovanje. Time se rizik zamenjuje *empirijskim rizikom*:

$$E(w, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^N L(y_i, f_w(x_i))$$

koji ćemo ubuduće nazivati *prosečnom greškom* ili samo *greškom*, osim u situacijama u kojima je potrebno naglasiti razliku u odnosu na stvarni rizik. Kad god nije naveden, skup podataka \mathcal{D} se podrazumeva.

Sada je moguće formulisati *princip minimizacije empirijskog rizika* (eng. *empirical risk minimization principle*) na kojem se tipično zasnivaju algoritmi nadgledanog mašinskog učenja – *funkcija koja minimizuje prosečnu grešku* $E(w, \mathcal{D})$ *se uzima za aproksimaciju funkcije koja minimizuje rizik* $R(w)$.

Treba imati u vidu da ovaj princip ne sledi logičkom nužnošću iz osnovne postavke problema nadgledanog učenja. Stoga je potrebno zapitati se kakva su svojstva ovog principa, odnosno da li vodi dobroj aproksimaciji funkcije koja minimizuje stvarni rizik. Pre svega, da li sa povećanjem slučajnog uzorka dolazi do konvergencije rešenja koje ovaj princip nudi ka optimalnom teorijskom rešenju. Ako to ne važi u opštem slučaju, bitno je znati u kojim slučajevima važi. Kako je rizik definisan kao očekivanje, a empirijski rizik kao prosek i kako znamo da proseci teže očekivanjima, kad veličina slučajno izabranog uzorka raste, u iskušenju smo da pomislimo da je odgovor lak i potvrđan. Ipak, to što pomenuto svojstvo važi, ne znači da minimum aproksimacije funkcionala mora uvek da teži minimumu funkcionala. Odnosno činjenica da važi

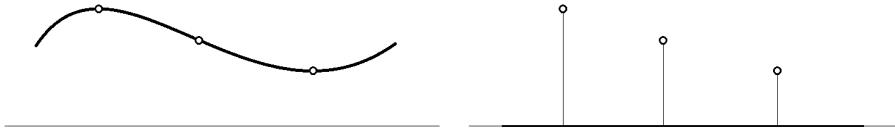
$$E(w, \mathcal{D}) \rightarrow R(w) \text{ kad } (|\mathcal{D}| \rightarrow \infty)$$

ne znači da važi

$$\operatorname{argmin}_w E(w, \mathcal{D}) \rightarrow \operatorname{argmin}_w R(w) \text{ kad } (|\mathcal{D}| \rightarrow \infty)$$

jer se prvo svojstvo odnosi na bilo koju, ali fiksiranu vrednost parametara w , istu i za E i za R , dok se drugo odnosi na potencijalno različite vrednosti

²U praksi se vektori podataka često eksplisitno proširuju jedinicom.



Slika 2.1: Levi model ima i stvarni i empirijski rizik jednak nula za bilo koju veličinu uzorka. Modeli poput desnog, mogu se konstruisati za bilo koju veličinu uzorka i svi imaju empirijski rizik nula i veliki stvarni rizik jer se od optimalnog modela razlikuje skoro svuda za veliku vrednost. Otud niz takvih modela ne konvergira stvarnom modelu.

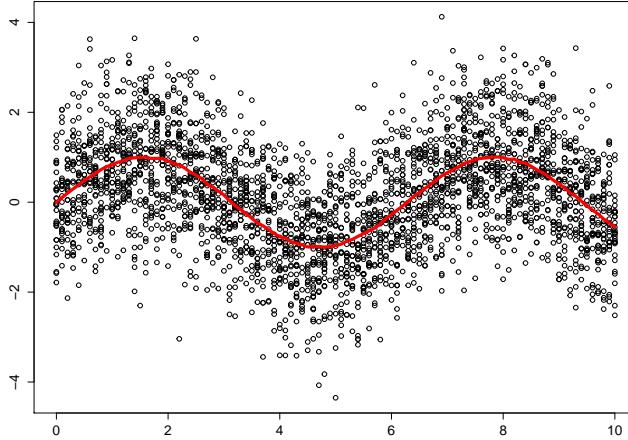
parametara w jer E i R ne moraju dostizati minimum u istoj tački. Ipak, poželjno je demonstrirati da drugo svojstvo u nekom slučaju zaista ne važi. Takav primer je dat na slici 2.1. Prepostavlja se da je skup svih modela skup svih mogućih funkcija f takvih da važi $f : \mathbb{R}^n \rightarrow \mathbb{R}$. Leva slika prikazuje optimalnu funkciju f^* u odnosu na stvarni rizik, pri čemu se prepostavlja da je $R(f^*) = 0$ (prema grafiku važi $y_i = f(x_i)$ za svaku vrednost $i = 1, \dots, N$), dok desna prikazuje funkciju \hat{f} koja minimizuje empirijski rizik. Ona svakoj tački x_i pridružuje baš vrednost y_i , ali je u svim ostalim tačkama jednaka 0. Očito važi $E(\hat{f}) = 0$. Za svaki skup podataka \mathcal{D} funkcije f^* i \hat{f} se poklapaju na skupu podataka i samo na njemu, pri čemu je svaki skup podataka mere nula, što znači da se razlikuju skoro svuda, odnosno da konvergencija ne mora da važi.

Razmatrano pitanje zavisi od svojstava skupa funkcija po kojem radimo minimizaciju. Odgovor na ovo pitanje je netrivijalan. Proučavanjem ovakvih problema se bavi *statistička teorija učenja*. Na kraju ovog dela će biti skiciran odgovor na ovo pitanje, ali je detaljna razrada van okvira ove knjige.

Pored prethodnih teorijskih pitanja, potrebno je dati odgovor i na jedno krajnje praktično – kako se rešava problem minimizacije prosečne greške modela? Odgovor u najvećem broju slučajeva nude metode matematičke optimizacije, o kojima će biti reči kasnije. Za sad je dovoljno prepostaviti da postoji metod za rešavanje problema minimizacije. Rešavanje takvog problema se naziva *obučavanjem modela* na datom skupu primera za obučavanje.

2.2.1 Minimizacija empirijskog rizika u slučaju regresije

Kao što je već rečeno, jednoj vrednosti atributa, ne mora odgovarati tačno jedna vrednost ciljne promenljive, već ima smisla govoriti o raspodeli vrednosti ciljne promenljive pri datim vrednostima atributa. Stoga se postavlja pitanje koju vrednost ciljne promenljive izabrati prilikom predviđanja. Jedan intuitivan odgovor je za date vrednosti atributa izabrati srednju vrednost ciljne promenljive od svih koje im odgovaraju. Sredina se često formalizuje pojmom



Slika 2.2: Ilustracija regresione funkcije.

očekivanja. Željena funkcija se onda može definisati kao

$$r(x) = \mathbb{E}[y|x] = \int y p(y|x) dy$$

i naziva se *regresionom funkcijom*. Regresiona funkcija je ilustrovana na slici 2.2.

Definisanju regresione funkcije može se pristupiti i sa druge strane. Potrebno je da vrednosti koje daje model što bolje odgovaraju pravim vrednostima, odnosno da je razlika $y - f_w(x)$ što manja. U tom slučaju i kvadrat razlike je mali. Primetimo da se kvadrat razlike može uzeti za funkciju greške i da se rizik onda definiše kao

$$\mathbb{E}[(y - f_w(x))^2]$$

Važi

$$\min_w \mathbb{E}[(y - f_w(x))^2] =$$

$$\min_w \left\{ \int (y - f_w(x))^2 p(x, y) dx dy \right\} =$$

$$\min_w \left\{ \int (y - r(x) + r(x) - f_w(x))^2 p(x, y) dx dy \right\} =$$

$$\min_w \left\{ \int [(y - r(x))^2 + 2(y - r(x))(r(x) - f_w(x)) + (r(x) - f_w(x))^2] p(x, y) dx dy \right\} =$$

$$\min_w \left\{ \int (y - r(x))^2 p(x, y) dx dy + \int [2(y - r(x))(r(x) - f_w(x)) + (r(x) - f_w(x))^2] p(x, y) dx dy \right\} =$$

$$\begin{aligned}
& \min_w \left\{ \int [2(y - r(x))(r(x) - f_w(x)) + (r(x) - f_w(x))^2] p(x, y) dx dy \right\} = \\
& \min_w \left\{ 2 \int (y - r(x))(r(x) - f_w(x)) p(x, y) dx dy + \int (r(x) - f_w(x))^2 p(x, y) dx dy \right\} = \\
& \min_w \left\{ 2 \int (y - r(x))(r(x) - f_w(x)) p(y|x) p(x) dx dy + \int (r(x) - f_w(x))^2 p(x, y) dx dy \right\} = \\
& \min_w \left\{ 2 \int (r(x) - f_w(x)) \left(\int (y - r(x)) p(y|x) dy \right) p(x) dx + \int (r(x) - f_w(x))^2 p(x, y) dx dy \right\} = \\
& \min_w \left\{ 2 \int (r(x) - f_w(x)) \left(\int y p(y|x) dy - r(x) \int p(y|x) dy \right) p(x) dx + \int (r(x) - f_w(x))^2 p(x, y) dx dy \right\} = \\
& \min_w \left\{ 2 \int (r(x) - f_w(x))(r(x) - r(x)) p(x) dx + \int (r(x) - f_w(x))^2 p(x, y) dx dy \right\} = \\
& \min_w \left\{ \int (r(x) - f_w(x))^2 p(x, y) dx dy \right\} = \\
& \min_w \left\{ \int (r(x) - f_w(x))^2 p(y|x) p(x) dx dy \right\} = \\
& \min_w \left\{ \int (r(x) - f_w(x))^2 \left(\int p(y|x) dy \right) p(x) dx \right\} = \\
& \min_w \left\{ \int (r(x) - f_w(x))^2 p(x) dx \right\} = \\
& \min_w \mathbb{E}[(r(x) - f_w(x))^2]
\end{aligned}$$

Očigledno, ako regresiona funkcija pripada skupu modela, u njoj se postiže minimum datog rizika. Ukoliko ne pripada, kako poslednji integral predstavlja metriku, jasno je da je najbolja funkcija ona koja joj je u smislu te metrike najbliža.

Kako je rizik dat izrazom $\mathbb{E}[(y - f_w(x))^2]$, prirodna formulacija principa minimizacije empirijskog rizika za regresiju je

$$\min_w \frac{1}{N} \sum_{i=1}^N (y_i - f_w(x_i))^2$$

Ovaj pristup je sveprisutan u problemima regresije. Naglasimo da je funkcija greške data izrazom

$$L(u, v) = (u - v)^2$$

i da se često naziva *kvadratnom greškom* (eng. squared loss), a odgovarajuća srednja greška *srednjekvadratnom greškom*. Zamislite su i drugačije funkcije greške. Na primer $L(u, v) = |u - v|$. Tada optimalno rešenje problema minimizacije rizika nije više uslovno očekivanje $\mathbb{E}[y|x]$, već uslovna medijana $m[y|x]$.

Pored toga, funkcija greške ne mora biti ni simetrična. Na primer, prilikom predviđanja cena akcija na berzi, važnije je predvideti da li će cene akcija porasti ili opasti, nego koliko. Stoga, ukoliko cena raste, greška naniže, koja može rezultovati negativnim predviđanjem (padom) je opasnija od greške naviše usled koje će zarada od kupovine takvih akcija biti manja nego što je očekivano, ali će kupac i dalje biti na dobitku.

2.2.2 Minimizacija empirijskog rizika u slučaju klasifikacije

U slučaju klasifikacije, formulacija principa empirijskog rizika je još jedostavnija. Model je utoliko bolji ukoliko pravi manje grešaka pri klasifikaciji. Neka je F tvrđenje. *Indikatorska funkcija* se definiše tako da važi

$$I(F) = \begin{cases} 1 & \text{ako } F \\ 0 & \text{ako } \neg F \end{cases}$$

Onda se princip minimizacije empirijskog rizika za klasifikaciju može definisati kao rešavanje problema

$$\min_w \frac{1}{N} \sum_{i=1}^N I(y_i \neq f_w(x_i))$$

Funkcija greške je

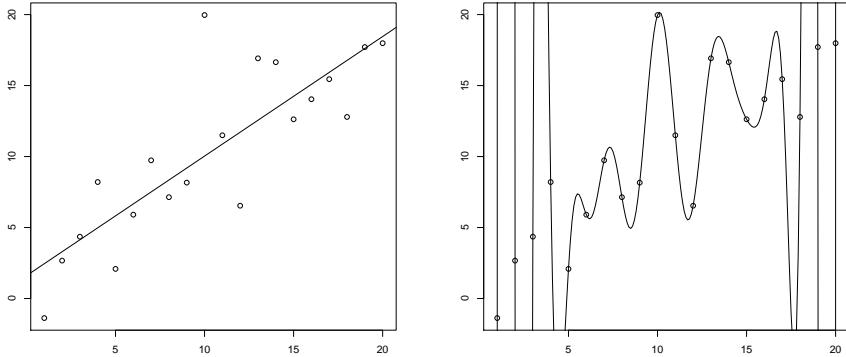
$$L(u, v) = I(u \neq v)$$

i naziva se prosto *greška klasifikacije*. Moguć je, a često i poželjan drugačiji izbor funkcije greške. Na primer, ne moraju sve greške biti jednakovarne. Nekе klase se mogu smatrati srodnijim od drugih, pa je pogrešnu klasifikaciju između tih klasa lakše tolerisati nego pogrešnu klasifikaciju između klasa koje nisu srodne. Ovo je primer *klasifikacije osetljive na cenu greške* (eng. *cost sensitive classification*). Takođe, funkcija greške ne mora biti ni simetrična. Nekada je opasnije instancu jedne klase klasifikovati kao instancu druge nego obrnuto. Na primer, prilikom klasifikacije medicinskih snimaka, pogrešna detekcija kancerogenog oboljenja može biti potresna za pacijenta, ali će se daljim testovima ustanoviti da je dijagnoza bila pogrešna. S druge strane, ukoliko se ustanovi da pacijent nije bolstan u slučaju kada jeste, greška može biti fatalna. Stoga u ovom kontekstu i funkcija greške treba da pridruži različite vrednosti različitim vrstama grešaka.

2.3 Preprilagođavanje

Minimizacija srednje greške izborom parametara modela predstavlja prilagođavanje modela podacima. Pojam prilagođavanja je jednostavno i vizuelno približiti. Neka je dat skup tačaka u dve dimenzije. Dimenzija x predstavlja vrednost atributa, a dimenzija y vrednost ciljne promenljive. Razmotrimo dve vrste modela. Prvi predstavlja skup svih modela oblika

$$f_w(x) = w_0 + w_1 x$$



Slika 2.3: Najbolja aproksimacija tačaka pravom i polinomom visokog stepena.

a drugi, skup svih modela oblika

$$g_w(x) = \sum_{i=0}^n w_i x^i$$

za proizvoljno $n \geq 0$. Prvi skup očito sadrži prave, a drugi polinome proizvoljnog stepena. Drugi skup je bogatiji u smislu da su sve prave istovremeno i polinomijalne krive. Otud se može očekivati da će se među funkcijama drugog skupa pre naći model sa manjom srednjom greškom bez obzira kakvi su podaci dati. I zaista, dok god za istu vrednost koordinate x u skupu tačaka ne postoje dve tačke sa različitim vrednostima koordinate y , postoji neki interpolacioni polinom koji prolazi kroz sve tačke skupa. Na slici 2.3 prikazani su prava i polinom sa najmanjim srednjekvadratnim greškama na datom skupu tačaka. Postavlja se pitanje koji je model bolji. U smislu srednjekvadratne greške, očito to je polinomijalni model. Ipak, dok se u slučaju linearног modela jasno uočava da, uprkos odstupanjima, prati opšti trend rasta koji je vidljiv u tačkama, u slučaju polinomijalnog modela, to se ne može jasno reći zbog drastičnih oscilacija, posebno u krajevima. Dok su za linearni model vrednosti predviđene u tačkama između datih bliske vrednostima u datim tačkama, polinomijalni model u takvim tačkama daje vrednosti koje su nekoliko redova veličine veće od okolnih. Otud bi svako ko bi u nekom relevantnom problemu za predviđanje koristio ovakav model bio na šteti.

Ovo zapažanje govori da minimizacija srednje greške nije nužno najbolji kriterijum, odnosno da postoje situacije u kojima funkcija koja minimizuje srednju grešku na dostupnim podacima pravi drastične greške na ostalim podacima iz iste raspodele, odnosno da vrlo slabo aproksimira funkciju koja minimizuje stvarni rizik, a koja bi davala relativno malu grešku i na podacima van datog uzorka. Drugim rečima model koji minimizuje srednju grešku na

podacima ne mora nužno dobro generalizovati. Ovaj problem, da se minimizacijom srednje greške model prilagođava podacima u toj meri da izgubi moć generalizacije, naziva se *problemom preprilagođavanja modela podacima* ili samo *preprilagođavanjem*. Ovaj problem predstavlja jedan od teorijski i praktično najznačajnijih problema mašinskog učenja. Iako bi početnik lako pomislio da je glavni izazov naći model koji se što bolje prilagođava podacima, to zapravo nije teško – poznat je veliki broj vrlo prilagodljivih modela. Suštinski problem je ne prilagoditi model podacima previše. Iz proučavanja ovog problema potekli su najvredniji teorijski uvidi mašinskog učenja, o čemu će biti reči kasnije.

Često se navodi da gubitak generalizacije nastaje iz preteranog prilagođavanja modela šumu. Iako to može biti deo problema, dato zapažanje nije suštinsko, jer se isti problem javlja i bez prisustva šuma. Naime, moguće je da u konačnom (nekad i malom) skupu za obučavanje svi podaci pripadaju nekom specifičnom delu prostora atributa. Preprilagođavanjem model uči da su i neke nesuštinske specifičnosti datih podataka od presudnog značaja za odnos sa ciljnom promenljivom koji se u podacima vidi, a kad ta svojstva nisu prisutna, greši.

Kao još jedna ilustracija problema preprilagođavanja, u cilju njegovog približavanja intuiciji, biće dat jedan pomalo veštački, ali ilustrativan primer. Pretpostavimo da profesor na ispitu daje zadatke iz obimne zbirke koja je dostupna studentima. Jedan pristup polaganju ispita, koji podržavamo, bi mogao biti da student razume suštinske principe materije, koji se često mogu izložiti u drastično manjem obimu od obima pomenute zbirke. Primenom tih principa student je u stanju da u velikom broju slučajeva reši zadatak. Možda ne u svakom slučaju, ali dovoljno da položi ispit. Drugi pristup polaganju ispita, koji ne podržavamo, bi mogao biti da student napamet nauči baš sve zadatke i njihova rešenja iz pomenute zbirke, uprkos njenom obimu. U tom slučaju student sa savršenim uspehom rešava sve zadatke na ispitu. Ipak, ukoliko bi pomenuti profesor u nezgodnom trenutku otišao na odmor i prepustio ispit drugom profesoru koji bi dao zadatke van pomenute zbirke, student koji bi učio po prvom principu bi kod drugog profesora prošao podjednako dobro, dok bi student koji je učio po drugom principu kod drugog zasigurno pao ispit. Sličnost sa primerom linearnih i polinomijalnih modela je upečatljiva. Linearni model sa manjim brojem parametara odgovara uočavanju zavisnosti koje se mogu predstaviti u obimu mnogo manjem od zbirke. Polinomijalni model koji se savršeno prilagođava podacima ima onoliko parametara koliko ima tačaka kojima se prilagođava, baš kao što je student u drugom slučaju morao imati onoliko memorije koliko u zbirci ima zadataka.

Deluje da je za uspešnu generalizaciju potrebno birati modele iz relativno siromašnog skupa, a koji relativno dobro odgovaraju podacima, pre nego modele iz vrlo bogatog skupa koji savršeno odgovaraju podacima. Ipak, vrlo je teško unapred odrediti koliko bogat skup treba da bude. Stoga se problemu pristupa drugačije – moguće je dozvoliti vrlo fleksibilnu reprezentaciju modela (koja odgovara bogatom skupu modela), ali kontrolisati fleksibilnost te reprezentacije u vreme obučavanja *regularizacijom*.

2.4 Regularizacija

Pristup učenju izborom modela iz siromašnog skupa funkcija pati od očiglednog nedostatka. Ukoliko se u datom skupu ne nalazi nijedan model sa relativno malom srednjom greškom, ne može se očekivati dobra generalizacija, jer model nije ništa naučio. S druge strane u bogatom skupu se može očekivati da se nalazi model koji dobro generalizuje, ali ga je teško naći. Jedna tehniku koja često omogućava izbor modela koji dobro generalizuje iz bogatog skupa funkcija, smanjujući fleksibilnost reprezentacije u vreme obučavanja, naziva se *regularizacijom*. Ona predstavlja modifikaciju minimizacionog problema, koja se sastoji u dodavanju takozvanog *regularizacionog izraza* $\Omega(w)$ koji otežava prilagođavanje modela podacima i tako predstavlja kontratežu preprilagođavanju. Naravno, ako se prilagođavanje potpuno onemogući, obučavanje je nemoguće. Stoga je potrebno kontrolisati dve pomenute suprotstavljene težnje. Tome služi nenegativan *regularizacioni hiperparametar* λ kojim se kontroliše koliko se težine pridaje minimizaciji srednje greške, a koliko regularizacionom izrazu. *Regularizovani problem* je sledeći

$$\min_w \frac{1}{N} \sum_{i=1}^N L(y_i, f_w(x_i)) + \lambda \Omega(w)$$

Za regularizacioni izraz se često uzima kvadrat ℓ_2 norme vektora koeficijenata, pa minimizacioni problem postaje sledeći

$$\min_w \frac{1}{N} \sum_{i=1}^N L(y_i, f_w(x_i)) + \lambda \|w\|_2^2$$

pri čemu se podrazumeva da se minimizuje ceo zbir, a ne da se regularizacioni izraz dodaje na minimalnu vrednost srednje greške.

Smisao regularizacije se može lakše razumeti posmatrajući ekstreme. Prvi, u slučaju kada važi $\lambda = 0$ znači da regularizacije nema i ukoliko je reprezentacija modela dovoljno fleksibilna, lako dolazi do preprilagođavanja. Drugi, u slučaju kada je vrednost parametra λ velika (neformalno, razmišljajmo kao da je beskonačna), minimizovanjem srednje greške se ništa značajno ne dobija. Jedino je važno minimizovati regularizacioni izraz, koji svoj minimum dostiže kada važi $w = 0$, što je jedan jedini model, od kojeg se ne može očekivati nikakva prilagodljivost. Stoga, prvi sabirak zahteva prilagođavanje modela podacima, dok drugi to otežava, a regularizacionim parametrom se vrši vaganje između tih suprotstavljenih tendencija.

Još jedna zanimljiva interpretacija³ je sledeća. Nauka uopšte, bavi se uspostavljanjem veza između različitih pojava. Prepostavimo da ih možemo predstaviti numerički atributima i cilnjom promenljivom. Uobičajena prepostavka

³Na kojoj se zahvaljujemo kolegi Milošu Jovanoviću.

u nauci, koja sledi iz principa Okamove oštice⁴ je da veza između posmatranih pojava nema. Ona se u statistici naziva nultom hipotezom. Onaj ko želi da pokaže da neka veza postoji mora prezentovati argumente u korist te hipoteze koja je alternativa nultoj. Regularizacioni izraz predstavlja pretpostavku da veze nema jer, zaista, ukoliko mu se u potpunosti udovoljji, ishod minimizacije je model $w = 0$, a ukoliko su parametri modela 0, to znači da promene u vrednostima atributa nikako ne utiču na promenu u vrednosti ciljne promenljive. Srednja greška, koja uključuje informaciju o podacima, predstavlja težinu argumentacije da veze ima. Potrebna je jaka argumentacija, odnosno visoka vrednost srednje greške na opaženim podacima da bi nas uverila da je naša nulta hipoteza pogrešna i da treba prihvati činjenicu da neka veza postoji, odnosno da koficijenti modela ne treba da budu nula.

Treba imati u vidu da regularizacija nije nužno uvek od koristiti. Posebno je značajna u slučaju da je količina podataka za obučavanje mala u odnosu na prilagodljivost modela (što nije lako kvantitativno izraziti). Kada je količina podataka velika, regularizacija nije neophodna. Često se kaže da je povećanje količine podataka najbolja regularizacija, ali velika količina podataka nije uvek dostupna.

U opštijem smislu, regularizacijom se često naziva bilo kakva modifikacija problema koja ograničava prilagodljivost modela i čini ga manje podložnim preprilagođavanju. Čak i odredene pretpostavke o reprezentaciji modela koje je ograničavaju se nekad u literaturi tako nazivaju. U najopštijem smislu, van konteksta mašinskog učenja, regularizacijom se naziva modifikacija bilo kog matematičkog problema koja ga čini bolje uslovjenim, odnosno čini da za male promene ulaznih parametara problema i promene rešenja problema budu male.

Za kraj, razmotrimo primer dejstva regularizacije na klasifikacione modele. Neka su u ravni date tačke koje pripadaju dvema klasama, kao na slici 2.4. Kao jedan prirodan izbor modela klasifikacije, nameće se prava, odnosno linearni model oblika:

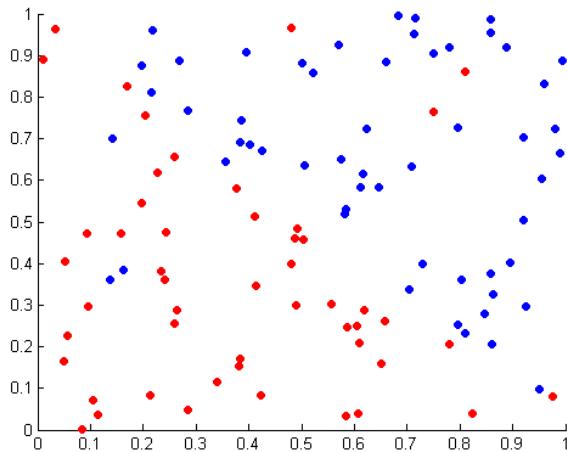
$$f_w(x) = w_0 + w_1 x_1 + w_2 x_2$$

Alternativno, moguće je koristiti i linearni model koji predstavlja polinom dve promenljive (linearnost je po parametrima, ne po promenljivim!):

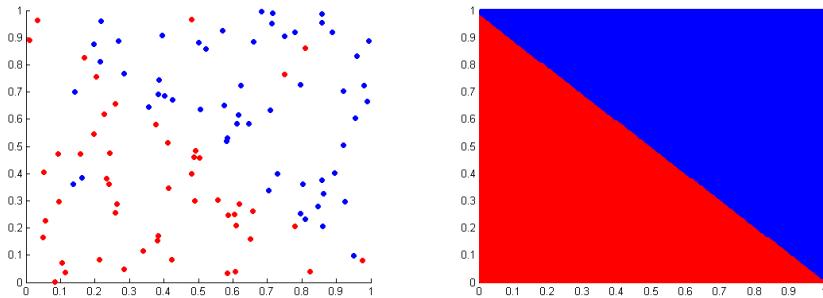
$$g_w(x) = \sum_{i=0}^n \sum_{j=0}^i w_{ij} x_1^j x_2^{i-j}$$

U oba slučaja indikatorom klase se smatra znak funkcije. Na slici 2.5 prikazano je koje klase svim tačkama u datom segmentu ravni pridružuje najbolja prava. Globalna zakonitost deluje ispravno ustanovljena, iako sa obe strane ima pogrešno klasifikovanih tačaka. Na slici 2.6 prikazana je klasifikacija polinomijal-

⁴Okamova oštica je jedan od osnovnih principa naučnog zaključivanja i kaže da entitete kojima se nešto objašnjava ne treba umnožavati preko potrebe ili, drugim rečima, najjednostavnije objašnjenje saglasno sa podacima je najbolje. Pitanje jednostavnosti objašnjenja je netrivijalno i potpada u domen filozofije nauke. O tome će biti reči kasnije.

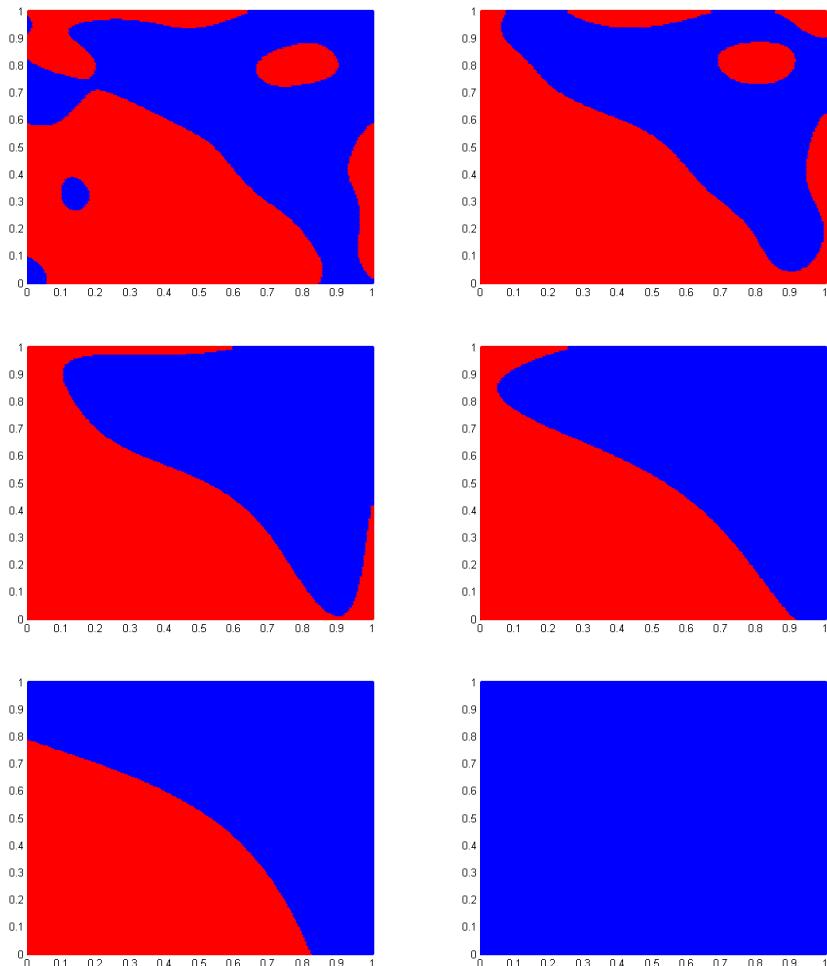


Slika 2.4: Tačke u ravni koje pripadaju dvema klasama.



Slika 2.5: Klasifikacija tačaka linearnim modelom.

nim modelom za različite, rastuće, vrednosti regularizacionog parametra. Može se uočiti da granica između dve klase postaje sve jednostavnija sa povećanjem regularizacionog parametra, dok u poslednjem slučaju ekstremno visoke vrednosti regularizacionog parametra, model ne postane konstantan. Jedno pitanje koje vredi razmotriti je zašto konstantan model predviđa baš plavu klasu, a ne crvenu. Razlog je taj što na slici postoji jedna plava tačka više. Ipak, ako je vrednost regularizacionog parametra velika (a jeste) i ako u tom slučaju model ne bi trebalo da ima moć prilagođavanja, kako se model prilagodio odnosu broja tačaka? Odgovor se krije u čestoj praksi, koja je primenjena i ovde, da se slobodni član modela, koji ne množi nijedan atribut, izuzme iz regularizacije.



Slika 2.6: Klasifikacija tačaka polinomijalnim modelom za rastuće vrednosti regularizacionog parametra.

2.5 Nagodba između sistematskog odstupanja i varijanse

Kako se prilikom regularizacije menja optimizacioni problem, menja se i njegovo rešenje, to jest model koji se dobija optimizacijom. Na osnovu prethodne diskusije, ovo je u slučaju nedovoljne količine podataka upravo željeni efekat. U slučaju velike količine podataka, regularizacija bi tipično onemogućila model da im se se adekvatno prilagodi. Ovo ponašanje se da osvetliti sa još jedne strane nagodbom između sistematskog odstupanja i varijanse. Razmotrimo slučaj regresije. Kada je analiziran princip minimizacije rizika za regresiju, pokazano je da važi

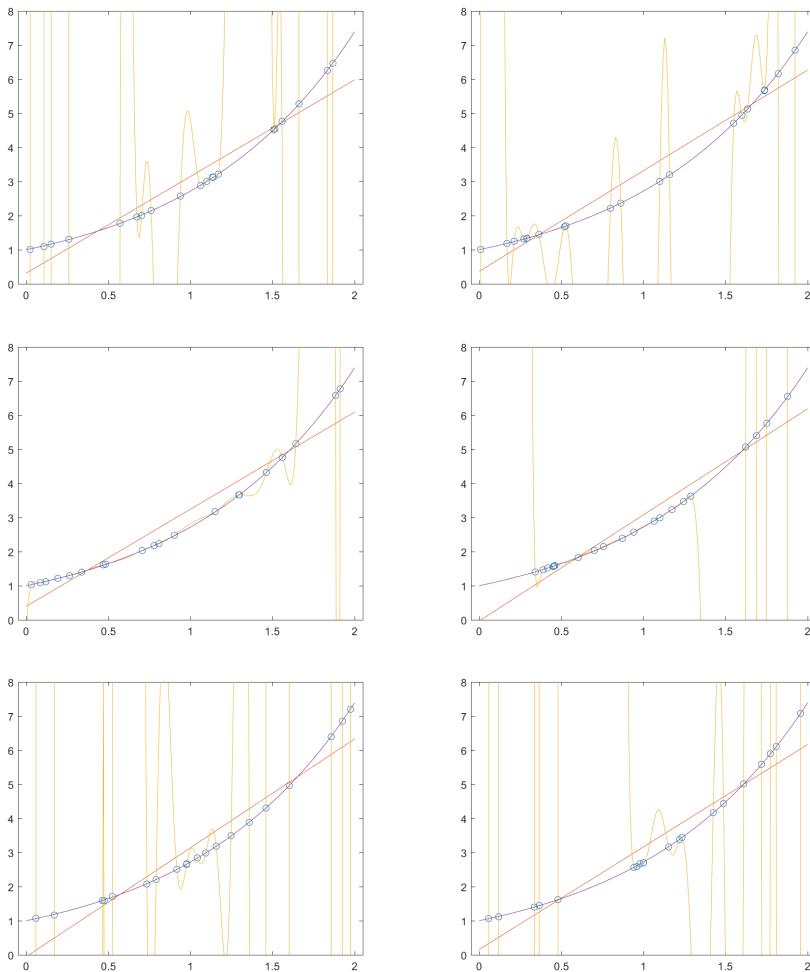
$$\min_w \mathbb{E}[(y - f_w(x))^2] = \min_w \mathbb{E}[(r(x) - f_w(x))^2]$$

Odnosno, optimalna funkcija koju je potrebno naučiti je regresiona funkcija $r(x)$. Razmotrimo sada sledeći misaoni eksperiment. Neka se iz raspodele podataka puno puta generiše skup nezavisnih opažanja \mathcal{D} i neka se na njemu obučava model $f_{\mathcal{D}}(x)$. U ovom konkretnom slučaju umesto parametara nagašavamo skup instanci na osnovu kojih je model dobijen. U svakoj tački x , različiti modeli, dobijeni na osnovu različitih skupova podataka, imaće različita odstupanja od idealne regresione funkcije. Performanse konkretnog algoritma učenja se u svakoj tački mogu proceniti kao prosek velikog broja takvih odstupanja. Ova procedura zapravo u svakoj tački ocenjuje очекivanje funkcije greške po svim skupovima podataka $\mathbb{E}_{\mathcal{D}}[(r(x) - f_{\mathcal{D}}(x))^2]$. Razmotrimo prvo vizuelno šta se dešava za različite skupove podataka na sledećem jednostavnom primeru.

Neka se više puta generiše skup podataka tako što se nasumično biraju tačke x u intervalu $[0, 2]$, na svaku tačku se primeni eksponencijalna funkcija i doda mali šum, čime se dobija vrednost y . Svi generisani skupovi predstavljaju isti eksponencijalni zakon koji povezuje promenljive x i y , odnosno, može se reći da su promene u podacima male. Na slici 2.7 je prikazano pet takvih skupova podataka i za svaki od njih prava i polinom sa najmanjom srednjekvadratnom greškom. Oba modela bi trebalo da ocenjuju eksponencijalnu funkciju. Ukoliko čitalac obrati pažnju na bilo koju vrednost na x osi i pogleda kako se od slike do slike menjaju vrednosti oba modela u toj tački, primetiće da vrednosti koje predviđa prava malo variraju. Obično su sve manje ili sve veće od vrednosti eksponencijalne funkcije i zbog toga se pri uprosećavanju ne poništavaju. Zaključujemo da u različitim tačkama, prava tipično sistematski odstupa od regresione funkcije, ali malo varira. S druge strane, vrednosti polinoma u istoj tački izuzetno variraju. Nekada su veće od vrednosti eksponencijalne funkcije, a nekad manje i stoga se pri uprosećavanju poništavaju.

Prethodna zapažanja se mogu formalizovati kroz sledeću analizu очекivanja $\mathbb{E}_{\mathcal{D}}[(r(x) - f_{\mathcal{D}}(x))^2]$. Oznaka skupa podataka će biti izostavljena zbog čitljivosti, ali se podrazumeva. Važi:

$$\begin{aligned}\mathbb{E}[(r(x) - f(x))^2] &= \mathbb{E}[(r(x) - \mathbb{E}[f(x)] + \mathbb{E}[f(x)] - f(x))^2] = \\ \mathbb{E}[(r(x) - \mathbb{E}[f(x)])^2] &+ 2\mathbb{E}[(r(x) - \mathbb{E}[f(x)])(\mathbb{E}[f(x)] - f(x))] + \mathbb{E}[(\mathbb{E}[f(x)] - f(x))^2]\end{aligned}$$



Slika 2.7: Prava i polinom najmanje srednjekvadratne greške za različite podatke iz istog eksponencijalnog zakona.

Kako regresiona funkcija ne zavisi od skupa podataka, ovaj izraz je jednak izrazu

$$\begin{aligned}(r(x) - \mathbb{E}[f(x)])^2 + 2(r(x) - \mathbb{E}[f(x)])\mathbb{E}[\mathbb{E}[f(x)] - f(x)] + \mathbb{E}[(\mathbb{E}[f(x)] - f(x))^2] = \\ (r(x) - \mathbb{E}[f(x)])^2 + 2(r(x) - \mathbb{E}[f(x)])(\mathbb{E}[f(x)] - \mathbb{E}[f(x)]) + \mathbb{E}[(\mathbb{E}[f(x)] - f(x))^2] = \\ (r(x) - \mathbb{E}[f(x)])^2 + \mathbb{E}[(\mathbb{E}[f(x)] - f(x))^2]\end{aligned}$$

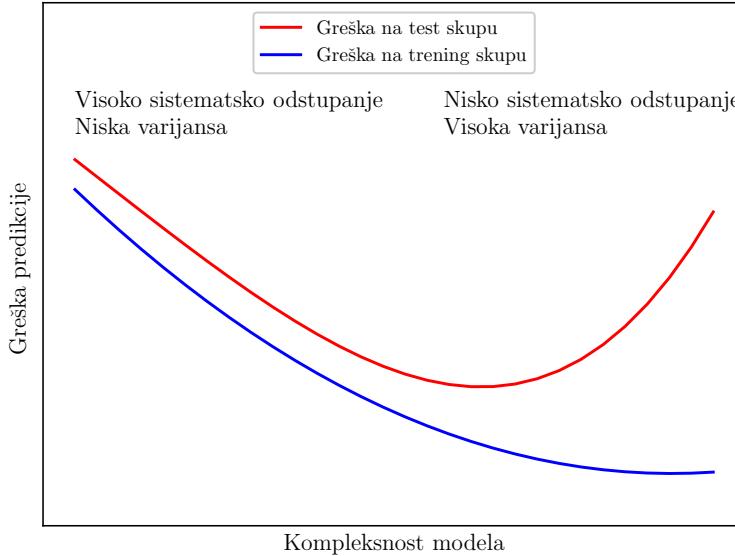
Prvi izraz je kvadrat sistematskog odstupanja modela od regresione funkcije u tački x – vrednost regresione funkcije u tački x je ono što model ocenjuje, ali kako za različite podatke dobijamo različite modele, možemo razmatrati očekivanje vrednosti koje ti modeli daju u tački x . A razlika između onoga što treba oceniti i očekivanja onoga čime se ocena vrši je po definiciji sistematsko odstupanje. Drugi izraz je očigledno varijansa modela. Pojasnimo o kakvoj varijansi modela je reč. Ponovo, kako se za različite podatke dobijaju različiti modeli koji u nekoj tački nude različita predviđanja, u svakoj tački postoji određena varijansa tih predviđanja. Ta varijansa se naziva varijansom modela u toj tački.

Izvedena jednakost se naziva *dekompozicijom greške na sistematsko odstupanje i varijansu*. Vrlo jednostavnii modeli poput pravih, zbog svoje rigidnosti obično ne mogu dobro da se prilagode podacima, zbog čega njihova predviđanja sistematski dosta odstupaju od regresione funkcije bez obzira na konkretni izbor skupa za obučavanje. S druge strane, vrlo fleksibilni modeli se tipično preprilagođavaju specifičnim podacima i već za male promene u podacima daju značajno različita predviđanja. Time prave veliku grešku koja je odraz njihove visoke varijanse. Ovo je shematski prikazano na slici 2.8. Regularizacija omogućava kontrolu fleksibilnosti modela. Izborom prave vrednosti regularizacionog hiperparametra moguće je značajno smanjiti varijansu, a po cenu umerenog rasta sistematskog odstupanja. Ovo se naziva *nagodbom između sistematskog odstupanja i varijanse*. Time se objašnjava uspešnost regularizacije u obučavanju modela mašinskog učenja. Kako se bira prava vrednost regularizacionog hiperparametra biće objašnjeno kasnije.

2.6 Teorijske garancije kvaliteta generalizacije

Kao što je već više puta naglašeno, moći generalizacije nekog algoritma mašinskog učenja koji minimizuje empirijski rizik (u ovom delu koristimo ovaj izraz umesto izraza srednja greška), zavisi od bogatstva skupa iz kojeg se model bira, odnosno od fleksibilnosti reprezentacije modela. Takozvana *statistička teorija učenja*, bavi se proučavanjem moći generalizacije na osnovu različitih skupova modela. Njeni rezultati se obično prikazuju u vidu granica za vrednost rizika. Osnovna pitanja koja postavlja su:

- Ukoliko je poznat empirijski rizik modela f , koliki može biti njegov stvarni rizik? Odgovor se svodi na pronalaženje gornje granice stvarnog rizika



Slika 2.8: Ponašanje greške na podacima za obučavanje i na odvojenim podacima za evaluaciju iz iste raspodele u slučaju modela različite fleksibilnosti.

koja je izražena u terminima empirijskog rizika $E(f)$, veličine skupa podataka N i skupa svih mogućih modela \mathcal{F} iz kojih se bira najbolji. Rezultat treba da bude nejednakost oblika $R(f) \leq E(f) + c(N, \mathcal{F})$.

- Koliko je stvarni rizik izabranog modela f viši od stvarnog rizika najboljeg modela $f^* \in \mathcal{F}$? Rezultat treba da bude oblika $R(f) \leq R(f^*) + c(N, \mathcal{F})$
- Koliko je stvarni rizik izabranog modela f viši od stvarnog rizika najboljeg mogućeg modela (koji ne mora stvarno biti u \mathcal{F})? Taj rizik označavamo sa R^* i ne bavimo se time kom modelu odgovara. Rezultat treba da bude oblika $R(f) \leq R^* + c(N, \mathcal{F})$.

U nastavku se bavimo prvim pitanjem kao verovatno najvažnijim. Njegov značaj je u tome što daje odgovor na to koliko prosečna greška na svim zamslivim podacima (stvarni rizik) može odstupiti od greške dobijene na skupu za obučavanje (empirijski rizik). Ukoliko je greška na skupu za obučavanje mala, zanima nas možemo li i pod kojim uslovima očekivati da je i greška na ostalim podacima mala. Kako bismo to znali, potrebno je da izvedemo veličinu c koja se pominje u prvom pitanju, a koju možemo smatrati širinom intervala poverenja za stvarni rizik, kada nam je dat empirijski rizik. Fundamentalni rezultat je da

se pod određenim uslovima sa povećanjem veličine uzorka, uniformno za sve funkcije, širina intervala poverenja smanjuje, odnosno da se empirijskoj oceni može verovati.

U nastavku će biti razmatran samo slučaj binarne klasifikacije sa funkcijom greške $L(u, v) = I(u \neq v)$, a rezultati se mogu uopštiti i na višeklasnu klasifikaciju, na druge funkcije greške i na regresiju.

Za fiksiranu funkciju f , potrebno je okarakterisati odstupanje veličine $R(f)$ od $E(f)$, odnosno razliku:

$$R(f) - E(f) = \mathbb{E}[L(y, f(x))] - \frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i))$$

Prema zakonu velikih brojeva važi

$$P\left(\lim_{N \rightarrow \infty} \frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i)) - \mathbb{E}[y, f(x)] = 0\right) = 1$$

Brzina ove konvergencije se čak može preciznije kvantifikovati pomoću Hefdingove (eng. Hoeffding) nejednakosti:

Teorema 1 (Hefdingova nejednakost) Neka su X_1, \dots, X_N nezavisne i jednako raspodeljene slučajne promenljive takve da važi $X_i \in [a, b]$ za $1 \leq i \leq N$. Tada za svaku $\varepsilon > 0$ važi

$$P\left(\frac{1}{N} \sum_{i=1}^N (\mathbb{E}[X_i] - X_i) > \varepsilon\right) \leq \exp\left(-\frac{2N\varepsilon^2}{(b-a)^2}\right)$$

Kako je funkcija greške $L(u, v) = I(u \neq v)$ ograničena i uzima vrednosti u intervalu $[0, 1]$ iz prethodne teoreme sledi

$$P\left(\frac{1}{N} \sum_{i=1}^N (\mathbb{E}[L(y, f(x))] - L(y_i, f(x_i))) > \varepsilon\right) \leq \exp(-2N\varepsilon^2)$$

$$P\left(\mathbb{E}[L(y, f(x))] - \frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i)) > \varepsilon\right) \leq \exp(-2N\varepsilon^2)$$

$$P(R(f) - E(f) > \varepsilon) \leq \exp(-2N\varepsilon^2)$$

Poželjno je što desna strana opada eksponencijalno sa povećavanjem broja instanci, ali prisustvo ε^2 usporava opadanje ove granice. Označimo desnu stranu sa δ i izrazimo ε kako bismo dobili željenu širinu c intervala poverenja. Iz $\delta = \exp(-2N\varepsilon^2)$ sledi

$$\varepsilon = \sqrt{\frac{\log \delta}{2N}}$$

Zamenjujući u prethodnu nejednakost dobijamo

$$P\left(R(f) - E(f) > \sqrt{\frac{\log \frac{1}{\delta}}{2N}}\right) \leq \delta$$

odnosno

$$P\left(R(f) - E(f) \leq \sqrt{\frac{\log \frac{1}{\delta}}{2N}}\right) \geq 1 - \delta$$

Drugačije, za svako $\delta > 0$ sa verovatnoćom od bar $1 - \delta$ važi

$$R(f) \leq E(f) + \sqrt{\frac{\log \frac{1}{\delta}}{2N}}$$

Drugim rečima za svaku funkciju f postoji skup C_f mere δ uzoraka koji krše gornju nejednakost, odnosno kod kojih je stvarni rizik značajno veći od empirijskog. Algoritam učenja uvek dobija neki uzorak \mathcal{D} na kojem uči i obično minimizuje regularizovani empirijski rizik birajući neku od funkcija iz skupa \mathcal{F} . Za izabranu funkciju f može se desiti da je bas \mathcal{D} nepovoljan uzorak za koji je gornja nejednakost prekršena, odnosno $\mathcal{D} \in C_f$. Razlog za ovo je što je pri izvođenju gornje nejednakosti analizirana proizvoljna funkcija bez osvrta na to da algoritam učenja može da bira bilo koju iz šireg skupa funkcija \mathcal{F} . Kako prethodna granica važi za fiksiranu funkciju f , nazivamo je *pojedinačnom granicom*. Kako za svaku funkciju f može postojati različit skup C_f , skup svih uzoraka u kojima granica ne važi bar za neku funkciju

$$C = \bigcup_{f \in \mathcal{F}} C_f$$

može biti mere veće od δ , pa se postavlja pitanje kolika treba da bude granica da bi se moglo garantovati da je skup C koji sadrži sve uzorce na kojima bilo koja funkcija pravi grešku veću od ε mere ne veće od δ . Ovakva granica se naziva *uniformnom granicom* jer važi za sve funkcije odjednom i, naravno, šira je od pojedinačne granice.

Kako bismo izveli uniformnu granicu, pretpostavimo prvo da je skup \mathcal{F} konačan. Potom, umesto ograničavanja razlike $R(f) - E(f)$, potrebno je ograničiti

$$\sup_{f \in \mathcal{F}} (R(f) - E(f))$$

Takva granica mora važiti za sve funkcije iz \mathcal{F} . Važi

$$P(\sup_{f \in \mathcal{F}} (R(f) - E(f)) > \varepsilon) =$$

$$P(\{\mathcal{D} \mid \sup_{f \in \mathcal{F}} (R(f) - E(f, \mathcal{D})) > \varepsilon\}) =$$

$$\begin{aligned}
P \left(\bigcup_{f \in \mathcal{F}} \{\mathcal{D} \mid R(f) - E(f, \mathcal{D}) > \varepsilon\} \right) &\leq \\
\sum_{f \in |\mathcal{F}|} P(\{\mathcal{D} \mid R(f) - E(f, \mathcal{D}) > \varepsilon\}) &= \\
\sum_{f \in |\mathcal{F}|} P(R(f) - E(f, \mathcal{D}) > \varepsilon) &\leq \\
\sum_{f \in |\mathcal{F}|} \exp(-2N\varepsilon^2) &= |\mathcal{F}| \exp(-2N\varepsilon^2)
\end{aligned}$$

Uvođenjem oznake δ za desnu stranu, dobija se

$$\varepsilon = \sqrt{\frac{\log |\mathcal{F}| + \log \frac{1}{\delta}}{2N}}$$

odnosno, za svako $\delta > 0$ sa verovatnoćom od bar $1 - \delta$, za svaku funkciju $f \in \mathcal{F}$ važi

$$R(f) \leq E(f) + \sqrt{\frac{\log |\mathcal{F}| + \log \frac{1}{\delta}}{2N}}$$

Neformalno, za većinu skupova podataka, za sve funkcije $f \in \mathcal{F}$ važi

$$R(f) - E(f) \in O\left(\sqrt{\frac{\log |\mathcal{F}|}{N}}\right)$$

Osnovni problem sa poslednjim rezultatom je što se oslanja na pretpostavku konačnosti skupa modela. Uopštenje ovog rezultata na prebrojivo beskonačan skup modela nije teško i preporučuje se čitaocu za vežbu. Pravi izazov dolazi sa neprebrojivošću skupa modela. Osnovna ideja je da u mašinskom učenju uvek radimo sa konačnim skupovima podataka i da stoga nije zaista bitno koliko ima različitih funkcija u skupu \mathcal{F} , već koliko ima funkcija koje se ponašaju različito na datom skupu podataka. Uvedimo oznake $z_1 = (x_1, y_1), \dots, z_N = (x_N, y_N)$ i

$$\mathcal{L}_{z_1, \dots, z_N} = \{(L(y_1, f(x_1)), \dots, L(y_N, f(x_N)) \mid f \in \mathcal{F}\}$$

Veličina ovog skupa je broj načina na koje podaci mogu biti klasifikovani (tačno ili netačno) i očito je konačna čak i kad skup \mathcal{F} to nije.

Definicija 1 (Funkcija rasta) *Funkcija rasta je maksimalan broj načina na koji N tačaka mogu biti klasifikovane pomoću funkcija iz skupa \mathcal{F} :*

$$S_{\mathcal{F}}(N) = \sup_{z_1, \dots, z_N} |\mathcal{L}_{z_1, \dots, z_N}|$$

Ispostavlja se da se funkcija rasta može iskoristiti kao mera „veličine“ skupa \mathcal{F} .

Teorema 2 (Vapnik-Červonenkis) Za svako $\delta > 0$, sa verovatnoćom bar $1 - \delta$ za svako $f \in \mathcal{F}$,

$$R(f) \leq E(f) + 2\sqrt{2\frac{\log S_{\mathcal{F}}(2N) + \log \frac{2}{\delta}}{N}}$$

U konačnom slučaju uvek važi $S_{\mathcal{F}}(N) \leq 2^N$, pa je ova teorema korisna i tada. Ipak, bitno je razumeti zašto važi u opštem slučaju. Glavni oslonac u dokazu ove teoreme daje takozvana lema o simetrizaciji. Njena osnovna ideja je da se izražavanje u terminima stvarnog rizika zameni izražavanjem u terminima empirijskog rizika na nekom odvojenom skupu za evaluaciju. Ovo zvuči razumno, tim pre što se u praksi tako i procenjuju performanse modela na podacima koji nisu viđeni u vreme obučavanja.

Teorema 3 (Lema o simetrizaciji) Neka su data dva uzorka z_1, \dots, z_N i z'_1, \dots, z'_N i neka je E' empirijski rizik na drugom uzorku. Za svako $\varepsilon > 0$, tako da važi $N\varepsilon^2 \geq 2$, važi

$$P(\sup_{f \in \mathcal{F}}(R(f) - E(f)) > \varepsilon) \leq 2P(\sup_{f \in \mathcal{F}}(E'_n(f) - E_n(f)) \geq \varepsilon/2)$$

Dokaz. Neka je funkcija f^* funkcija za koju se dostiže supremum na levoj strani nejednakosti. Tada važi:

$$\begin{aligned} I(R(f^*) - E(f^*) > \varepsilon)I(R(f^*) - E'(f^*) < \varepsilon/2) = \\ I(R(f^*) - E(f^*) > \varepsilon \wedge E'(f^*) - R(f^*) \geq -\varepsilon/2) \leq \\ I(E'(f^*) - E(f^*) > \varepsilon/2) \end{aligned}$$

Uzimajući očekivanje po drugom uzorku dobija se:

$$I(R(f^*) - E(f^*) > \varepsilon)P(R(f^*) - E'(f^*) < \varepsilon/2) \leq P(E'(f^*) - E(f^*) > \varepsilon/2)$$

Prema Čebišovljevoj nejednakosti, važi

$$P(R(f^*) - E'(f^*) > \varepsilon/2) \leq \frac{4 \operatorname{var}[f^*]}{N\varepsilon^2} \leq \frac{1}{N\varepsilon^2}$$

Poslednja nejednakost važi jer nijedna promenljiva sa vrednostima na intervalu ne može imati veću varijansu od $1/4$. Odavde sledi

$$I(R(f^*) - E(f^*) > \varepsilon) \left(1 - \frac{1}{N\varepsilon^2}\right) \leq P(E'(f^*) - E(f^*) > \varepsilon/2)$$

Prelaskom na očekivanje po prvom uzorku, dobija se:

$$P(R(f^*) - E(f^*) > \varepsilon) \left(1 - \frac{1}{N\varepsilon^2}\right) \leq P(E'(f^*) - E(f^*) > \varepsilon/2)$$

Na osnovu uslova da važi $N\varepsilon^2 \geq 2$ i polaznog izbora funkcije f^* kao funkcije za koju se postiže supremum, dobija se traženi rezultat. \square

Sada je lako dokazati prethodnu teoremu:

$$\begin{aligned}
& P(\sup_{f \in \mathcal{F}}(R(f) - E(f)) > \varepsilon) \leq \\
& 2P(\sup_{f \in \mathcal{F}}(E'(f) - E(f)) > \varepsilon/2) = \\
& 2P\left(\sup_{l \in \mathcal{L}_{z_1, \dots, z_N, z'_1, \dots, z'_N}} \left(\frac{1}{N} \sum_{i=N+1}^{2N} l_i - \frac{1}{N} \sum_{i=1}^N l_i\right) > \varepsilon/2\right) \leq \\
& 2 \sum_{l \in \mathcal{L}_{z_1, \dots, z_N, z'_1, \dots, z'_N}} P\left(\frac{1}{N} \sum_{i=N+1}^{2N} l_i - \frac{1}{N} \sum_{i=1}^N l_i > \varepsilon/2\right) = \\
& 2 \sum_{l \in \mathcal{L}_{z_1, \dots, z_N, z'_1, \dots, z'_N}} P\left(\frac{1}{N} \sum_{i=N+1}^{2N} l_i - R(f) + R(f) - \frac{1}{N} \sum_{i=1}^N l_i > \varepsilon/2\right) \leq \\
& 2 \sum_{l \in \mathcal{L}_{z_1, \dots, z_N, z'_1, \dots, z'_N}} \left(P\left(\frac{1}{N} \sum_{i=N+1}^{2N} l_i - R(f) > \varepsilon/2\right) + P\left(R(f) - \frac{1}{N} \sum_{i=1}^N l_i > \varepsilon/2\right) \right) \leq \\
& 2 \sum_{l \in \mathcal{L}_{z_1, \dots, z_N, z'_1, \dots, z'_N}} (\exp(-N\varepsilon^2/2) + \exp(-N\varepsilon^2/2)) = \\
& 4 \sum_{l \in \mathcal{L}_{z_1, \dots, z_N, z'_1, \dots, z'_N}} \exp(-N\varepsilon^2/2) = \\
& 4S_{\mathcal{F}}(2N) \exp(-N\varepsilon^2/2)
\end{aligned}$$

Odnosno

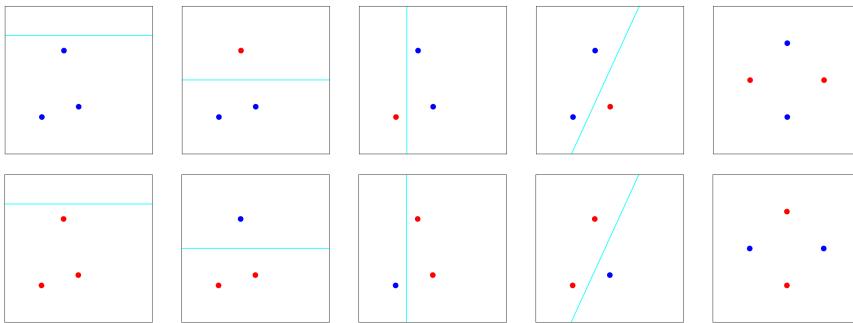
$$P(\sup_{f \in \mathcal{F}}(R(f) - E(f)) > \varepsilon) \leq 4S_{\mathcal{F}}(2N) \exp(-N\varepsilon^2/2)$$

Zamenjujući desnu stranu sa δ , teorema je dokazana. Ipak, postavlja se pitanje kako se funkcija rasta izračunava.

Definicija 2 (VC dimenzija) Vapnik-Červonenekisova (VC) dimenzija skupa funkcija \mathcal{F} je najveći broj N , takav da važi

$$S_{\mathcal{F}}(N) = 2^N$$

Ukoliko takav broj ne postoji, VC dimenzija je beskonačna.



Slika 2.9: Sve klasifikacije skupa od tri tačke pomoću pravih u ravni i problematičan primer četiri tačke u opštem rasporedu.

Drugim rečima, to je najveći broj N za koji postoji skup tačaka koji se funkcijama iz skupa \mathcal{F} mogu klasifikovati na sve moguće načine. Pritom, dovoljno je da postoji jedan takav skup. Nije potrebno da se svaki skup tačaka može klasifikovati na sve moguće načine. Primer za slučaj dve dimenzije dat je na slici 2.9. Neka skup pravih predstavlja skup modela. Svakoj pravoj odgovaraju dva modela, koji suprotno označavaju poluravni koje prava indukuje. U prostoru \mathbb{R}^2 postoje tri nekolinearne tačke koje se svim pravima mogu razdvojiti na sve moguće načine, ali ne postoje 4 takve tačke. Otud je skup svih pravih u ravni VC dimenzije 3. Analogno, skup svih hiperravnih u prostoru \mathbb{R}^N je VC dimenzije $N + 1$. To je upravo broj parametara koji ima hiperravan u ovom prostoru. Ispostavlja se da broj parametara i VC dimenzija, iako se poklapaju za linearne modele, ne moraju uvek biti isti. Štaviše, ne moraju biti ni u kakvoj relaciji. Na primer, skup funkcija sa jednim parametrom

$$\{\operatorname{sgn}(\sin(wx)) \mid w \in \mathbb{R}\}$$

ima beskonačnu VC dimenziju.

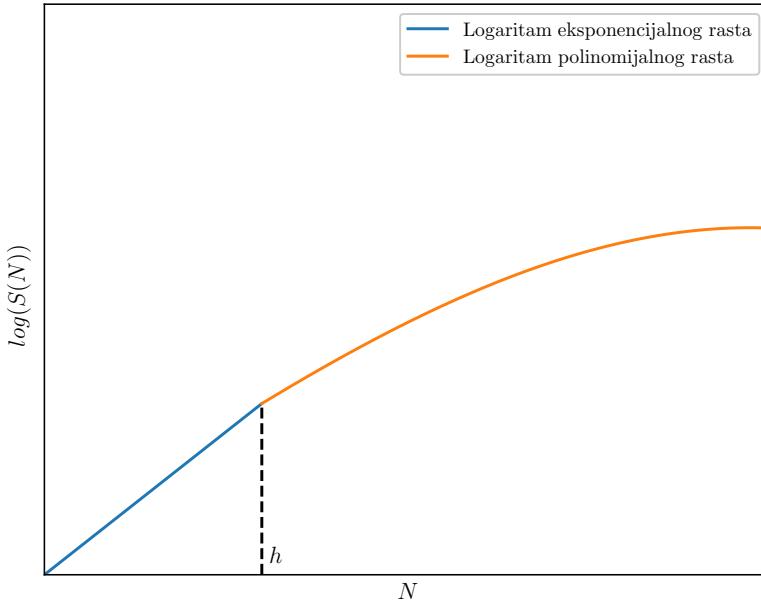
Prema definiciji VC dimenzije, ako je h VC dimenzija skupa \mathcal{F} , za $N \leq h$ važi $S_{\mathcal{F}}(N) = 2^N$, za $N > h$ važi $S_{\mathcal{F}}(N) < 2^N$. Zapravo, ispostavlja se i jače – do vrednosti h , funkcija rasta je eksponencijalna, a nakon nje polinomijalna. Slika 2.10 ilustruje ovo svojstvo.

Lema 1 (Sauerova lema) *Neka je \mathcal{F} skup funkcija konačne VC dimenzije h . Tada za sve $N \in \mathbb{N}$ važi*

$$S_{\mathcal{F}}(h) \leq \sum_{i=0}^h \binom{N}{i}$$

a za $N \geq h$

$$S_{\mathcal{F}}(h) \leq \left(\frac{en}{h}\right)^h$$



Slika 2.10: Logaritam funkcije rasta.

Ovu lemu ne dokazujemo. Iz Vapnik-Červonenkisove teoreme i Sauerove leme direktno se dobija da za skup funkcija \mathcal{F} VC dimenzije h , za svako $\delta > 0$ sa verovatnoćom bar $1 - \delta$ za svaku funkciju $f \in \mathcal{F}$ važi

$$R(f) \leq E(f) + 2\sqrt{2 \frac{h \log \frac{2eN}{h} + \log \frac{2}{\delta}}{N}}$$

ili ugrubo

$$R(f) - E(f) \in O\left(\sqrt{\frac{h \log N}{N}}\right)$$

Imajući sve dosadašnje rezultate u vidu, zaključuje se da u slučaju skupa modela konačne VC dimenzije, dovoljno velik broj podataka garantuje povereće u empirijsku ocenu stvarnog rizika.

2.7 Veza statističke teorije učenja sa filozofijom nauke

Već je pomenuto jedno od osnovnih načela naučnog zaključivanja – Okamo-va oštrica i rečeno je da kaže da entitete kojima se nešto objašnjava ne treba

umnožavati preko potrebe, odnosno da je najjednostavnije objašnjenje najbolje. Postavlja se pitanje šta znači da je objašnjenje najjednostavnije. Da li to znači da se može izraziti pomoću najmanje reči? Ili da se uklapa u ustaljena viđenja? Ili nešto drugo? Prethodno pomenute interpretacije nekome mogu biti jednostavne, ali je pitanje da li su relevantne, a svakako su subjektivne i kulturno zavisne. Jedan način razumevanja složenosti objašnjenja je vezan za to koliko toga objašnjenje može da objasni. Stoga je najjednostavnije objašnjenje ono koje najmanje toga može da objasni, a najbolje objašnjenje neke pojave u smislu Okamove oštice je ono koje objašnjava tu pojavu i što manje toga pored nje.

Primetimo da modele mašinskog učenja ima smisla posmatrati kao objašnjenja veza između promenljivih koje podaci predstavljaju, a koje predstavljaju neke fenomene iz stvarnog sveta. Atributi i ciljna promenljiva variraju – menjaju vrednosti. Model objašnjava promenu ciljne promenljive tako što je povezuje sa promenom vrednosti atributa. Slično rade i fizički zakoni. Na primer zakon $F = ma$ objašnjava promenu u veličini F promenom u veličinama m i a . U terminima statističke teorije učenja Okamova oštice ima spremnu interpretaciju, pošto je pojam jednostavnosti već formalizovan VC dimenzijom. Skup modela je utoliko jednostavniji što mu je VC dimenzija niža. Otud, najbolji model je model koji odgovara podacima, a koji je izabran iz skupa niske VC dimenzije.

Treba primetiti da statistička teorija učenja formalizuje pojam indukcije – govori pod kojim uslovima je verovatno da će greška induktivnog zaključivanja (predviđanje modela) biti mala.

Pored Okamove oštice, dubok epistemološki uvid predstavlja Popperova filozofija nauke, koja se pre svega fokusira na razlučivanje između empirijske (naučne) teorije i metafizičke teorije. Prema Popisu, teorija je empirijska ukoliko je *poreciva* (eng. *falsifiable*). Pod porecivošću se podrazumeva zamislivost eksperimenta koji bi tu teoriju oborio. Na prvi pogled, mogućnost obaranja teorije je njena slabost, a nemogućnost obaranja zvuči preferirano. Međutim to je pogrešno. Moguće je konstruisati najrazličitije neporecive teorije bez ikakvog uporišta u empirijskim opažanjima. Čuveni primer Raselovog čajnika postulira da postoji čajnik koji orbitira oko Sunca negde između orbita Marsa i Jupitera, ali je napravljen od takvog materijala i takve je veličine da se nikako ne može detektovati. Očigledno, nemogućnost detektovanja čajnika čini da uprkos mnoštvu eksperimenata kojima pokušavamo da ustanovimo njegovo postojanje, ne možemo da razlikujemo njegovo posedovanje pomenutih svojstava od njegovog nepostojanja. Nešto bliže terminima mašinskog učenja, kakvi god da su podaci, teza o Raselovom čajniku je sa njima saglasna. To je upravo ono što važi u slučaju beskonačne VC dimenzije – kakvi god da su podaci, postoji model koji ih objašnjava.

Primer Raselovog čajnika nekome može delovati nezanimljiv, pošto u njega niko ni u jednom trenutku nije poverovao. To sa naučne tačke gledišta nije ni najmanje relevantan kriterijum, a puno puta se ispostavilo da važe stvari koje niko nije očekivao (npr. opšta relativnost, Gedelove teoreme, teorija evolucije,

i slično). Ipak, postoje i teorije koje su bile naučno prihvaćene, ali su neporecive. Primer je Ptolemejeva geocentrična teorija, koja prepostavlja da se sva nebeska tela kreću oko Zemlje ili u krugovima ili u krugovima čiji su centri na prethodno uvedenim krugovima i tako dalje. Ova teorija je bila u stanju da uvođenjem dovoljnog broja krugova na krugovima objasni proizvoljno kretanje na nebu. Štaviše, pokazano je da se ovakvim funkcijama može aproksimirati bilo koje zamislivo kretanje (ovo znači beskonačnu VC dimenziju sistema funkcija). To znači da ne postoji baš nikakvo opaženo kretanje koje može pobiti ovu teoriju, odnosno da je ona neporeciva. S druge strane, heliocentrična teorija prepostavlja da se nebeska tela kreću oko drugih po elipsama, pri čemu je telo oko kojeg se drugo kreće u žiži elipse i pri čemu radijus vektor između dva tela u istim vremenskim intervalima prelazi iste površine. Ukoliko se desi da Merkur malo odstupi od predviđene putanje oko Sunca, heliocentrična teorija u obliku u kojem je znamo će biti oboren.⁵ Upravo je održavanje teorije, uprkos testovima u kojima je mogla biti oborenna osnov poverenja u tu teoriju.

2.8 Vrste modela

Modeli mašinskog učenja se mogu podeliti na tri vrste, prema tome koliku količinu informacije pokušavaju da modeluju – na *generativne modele* i *diskriminativne modele*.

2.8.1 Generativni modeli

Generativni modeli, modeluju zajedničku raspodelu najčešće datu svojom gustinom $p(x)$, koja opisuje ne samo zavisnosti između atributa i ciljne promenljive, već i zavisnosti među svim promenljivim, pa otud nema razloga da se jedna od njih izdvaja. Pomoću takve raspodele bilo bi moguće vršiti predviđanja za bilo koji podskup promenljivih ukoliko su poznate vrednosti nekih od promenljivih. Na primer, neka je $x = (x_1, x_2, x_3, x_4, x_5)$. Tada, ukoliko su poznate vrednosti promenljivih x_2 i x_5 , moguće je predvideti ostale na sledeći način

$$(x_1^*, x_3^*, x_4^*) = \underset{x_1, x_3, x_4}{\operatorname{argmax}} p(x_1, x_3, x_4 | x_2, x_5)$$

Pri čemu se gustina raspodele $p(x_1, x_3, x_4 | x_2, x_5)$ dobija na sledeći način

$$p(x_1, x_3, x_4 | x_2, x_5) = \frac{p(x_1, x_2, x_3, x_4, x_5)}{p(x_2, x_5)} = \frac{p(x_1, x_2, x_3, x_4, x_5)}{\int \int \int p(x_1, x_2, x_3, x_4, x_5) dx_1 dx_3 dx_4}$$

Pored samog predviđanja, ako je poznata raspodela verovatnoće, moguće je i izračunati pouzdanost predviđanja pomoću intervala poverenja. Ukoliko je poznata zajednička raspodela podataka, moguće je i generisati nove podatke iz

⁵Zapravo, relativistički efekti dovode do malih razlika, pa heliocentrična teorija i nije precizna u svojoj izvornoj formulaciji.

te raspodele, koji po svojim statističkim svojstvima liče na dostupne podatke, što je nekada takođe korisno.

S druge strane, modelovanje zajedničke raspodele zahteva značajnu količinu podataka, a samim tim i vreme za obučavanje. Naime, ukoliko je potrebno uočiti zavisnosti između velikog broja promenljivih, potrebno je u skupu za obučavanje imati veliki broj kombinacija vrednosti tih promenljivih. Zapravo, kako se dimenzionalnost podataka povećava, da bi se održala gustina podataka u nekom jediničnom intervalu, potrebno je da broj podataka eksponencijalno raste! A gustinu raspodele nije moguće dobro oceniti ako gustina podataka nije zadovoljavajuća. Ovo je primer ozloglašenog problema u mašinskom učenju, poznatog pod nazivom *prokletstvo dimenzionalnosti* (eng. *curse of dimensionality*). Ovo nije sveprisutan problem u mašinskom učenju, ali modelovanje zajedničke raspodele je tipičan kontekst u kojem se javlja.

U praktičnim kontekstima, predviđanje najčešće funkcioniše od atributa ka ciljnoj promenljivoj i kako su vrednosti atributa date, modelovanje zajedničke raspodele obično nije neophodno, pa nema osnova ni zahtevati tako velike količine podataka za modelovanje odnosa koji nisu važni. Stoga, ukoliko dati problem to ne zahteva, generativni model nije prvi model koji bi trebalo primeniti.

2.8.2 Diskriminativni modeli

Diskriminativni modeli, modeluju uslovnu raspodelu $p(y|x)$, koja opisuje samo zavisnost ciljne promenljive od atributa. Ne i zavisnosti među atributima. Kako su vrednosti atributa tipično date i samo vrednost ciljne promenljive nedostaje, to je dovoljno u većini praktičnih konteksta. Povrh predviđanja, koje se vrši na sledeći način

$$y^* = \operatorname{argmax}_y p(y|x)$$

i ovi modeli zahvaljujući poznavanju raspodele pružaju i procenu pouzdanosti predviđanja. Pomoću njih nije moguće generisati podatke, ali to najčešće nije ni potrebno. Pošto ne modeluju odnose među atributima, ukoliko se modeluje raspodela samo jedne ciljne promenljive, ne pate od prokletstva dimenzionalnosti.

U ovom kontekstu je važna jedna napomena. Ako bi se za date vrednosti atributa x modelovao veliki broj ciljnih promenljivih odjednom, uzimajući u obzir njihove međuzavisnosti, uprkos uslovnoj formi raspodele, taj model bi ponovo trebalo smatrati generativnim u odnosu na ciljne promenljive. Problemi poput prokletstva dimenzionalnosti su utoliko blaži što nije potrebno modelovati odnose među atributima, ali broj potrebnih podataka ponovo raste eksponencijalno sa brojem ciljnih promenljivih koje se modeluju.

Diskriminativni modeli mogu modelovati i samo funkciju $y = f(x)$ bez informacije o raspodeli bilo koje od promenljivih.

2.9 Dimenzije dizajna algoritama nadgledanog učenja

Algoritmi nadgledanog mašinskog učenja se mogu značajno razlikovati po mnogim svojstvima, konstrukciji i nameni. Ipak, kod mnogih od njih je moguće primetiti zajedničku strukturu, odnosno uočiti da predstavljaju instance jedne opštije sheme dizajna. Poznavanje ove sheme je korisno kako prilikom dizajna algoritama, tako i prilikom razumevanja postojećih algoritama zato što se ispostavlja da različiti aspekti ponašanja algoritama učenja zavise od konkretnih odluka donetih prilikom dizajna i da ih je moguće povezati sa različitim dimenzijama pomenute sheme. Pod dimenzijama neformalno podrazumevamo različite elemente koji se mogu birati kako bi se konstruisao algoritam. Dimenzije dizajna algoritma nadgledanog učenja, sa nekim primerima mogućih izbora, su ugrubo sledeće:

- Vrsta modela – generativni ili diskriminativni.
- Forma modela – linearni, zasnovan na instancama, neuronska mreža itd.
- Funkcija greške – srednjekvadratna, prosečna apsolutna, unakrsna entropija itd.
- Regularizacija – ℓ_1 , ℓ_2 , grupna itd.
- Optimizacioni algoritam – gradijentni spust, Nesterovljev algoritam, Adam, itd.

Izbori po različitim dimenzijama se nekad mogu doneti nezavisno, ali nije retkost da neki od izbora po jednoj dimenziji nije kompatibilan sa nekim izborima po drugim dimenzijama. Na primer, gradijentni spust ne funkcioniše dobro sa ℓ_1 regularizacijom, koja koristi $\|\cdot\|_1$ normu (sumu apsolutnih vrednosti razlika koordinata dva vektora), zbog njene nediferencijabilnosti, odnosno nemogućnosti računanja gradijenata u svim tačkama. Takvi problemi se prevazilaze ili primenom drugih postojećih algoritama ili razvojem novih koji imaju odgovarajuća svojstva.

Glava 3

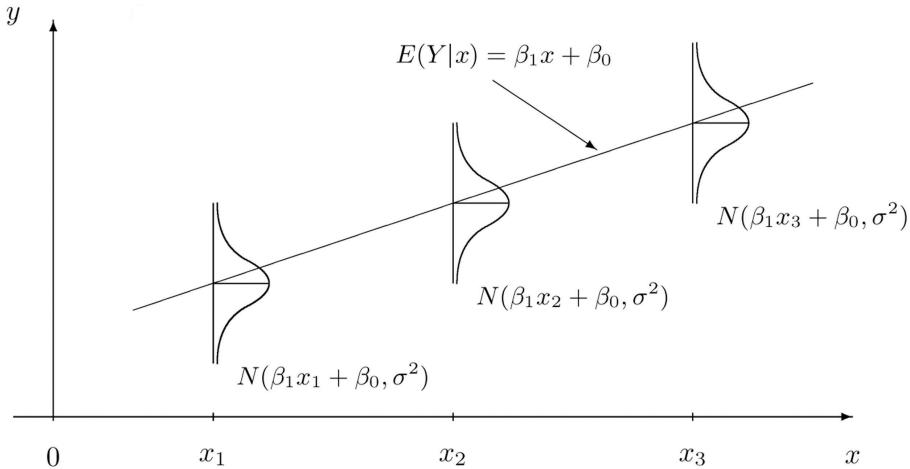
Probabilistički modeli

Verovatnoća predstavlja prirodan okvir za modelovanje neizvesnosti koja je sveprisutna u induktivnom zaključivanju. Otud se veliki broj algoritama mašinskog učenja u nekoj meri oslanja na probabilističke koncepte. Neki u potpunosti. Dizajn algoritama probabilističkih modela mašinskog učenja se zasniva na definisanju raspodele verovatnoće koju je potrebno oceniti iz podataka. Kako bi ocena, a kasnije i zaključivanje bili računski izvodljivi, obično se uvođe neke pretpostavke vezane za to kako promenljive zavise jedne od drugih i kakva je forma raspodele. Forma raspodele, na primer, može biti normalna, dok struktura zavisnosti može biti izabrana tako da su vrednosti ciljnih promenljivih na različitim instancama nezavisne ili da je zavisnost definisana nekim grafom ili nekako drugačije. U oba slučaja, izbor pogrešnih pretpostavki može značajno uticati na performanse algoritma. Ako je pretpostavljena neka forma raspodele koja ne odgovara raspodeli podataka, vrlo je verovatno da će predviđanja biti lošija, a još verovatnije je da će informacija o pouzdanosti koju model pruža biti nekorektna. Slično važi i ukoliko se pretpostavi nezavisnosti promenljivih koje su zapravo zavisne. Jednostavnosti radi, u nastavku ćemo se baviti samo modelima kod kojih se pretpostavlja međusobna nezavisnost vrednosti ciljne promenljive pri datim vrednostima atributa, ali naravno modeluje se zavisnost ciljne promenljive od atributa.

3.1 Linearna regresija

Linearna regresija predstavlja jedan od najjednostavnijih i najčešće korišćenih modela mašinskog učenja. Postoje različiti načini njenog uvođenja. Jedan je probabilistički i verovatno otkriva više o ovom metodu od ostalih. Sa probabilističke tačke gledišta, osnovna pretpostavka je pretpostavka normalne raspodele ciljne promenljive y , pri datim vrednostima atributa x . Odnosno, važi

$$p(y|x) = \mathcal{N}(f(x), \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - f(x))^2}{2\sigma^2}\right)$$



Slika 3.1: Prikaz normalne raspodele konstantne standardne devijacije sa linearnim modelom proseka.

gde je f funkcija koja uspostavlja vezu između atributa i očekivanja ciljne promenljive. U zavisnosti od forme ove funkcije, moguće je dobiti vrlo različite modele. Tehnički najjednostavnija forma modela je linearна:

$$f_w(x) = w \cdot x$$

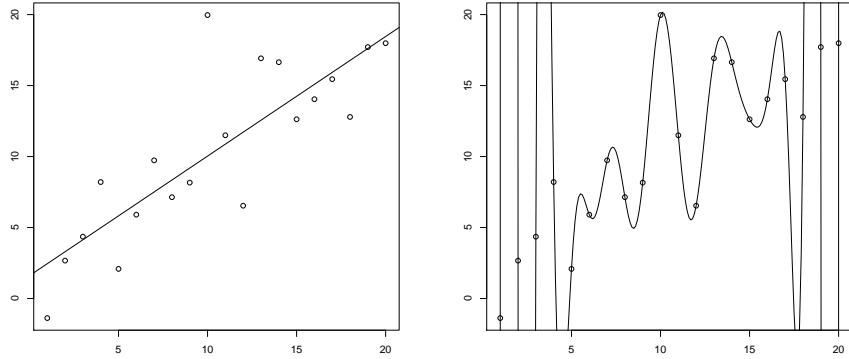
i ta pretpostavka je sastavni deo modela linearne regresije. Slika 3.1 prikazuje ovakvu raspodelu sa prosekom koji zavisi od neke promenljive. Najčešće se podrazumeva da je skup atributa proširen atributom koji je konstantne vrednosti 1, kako bi model sadržao slobodni koeficijent. Ukupni model je opisan na sledeći način:

$$p_w(y|x) = \mathcal{N}(w \cdot x, \sigma^2)$$

Kako modeluje uslovnu raspodelu, ovaj model je očigledno probabilistički diskriminativni.

Na slici 3.2, prikazan je primer dva poznata modela linearne regresije na istom skupu podataka. Kako linearost označava linearost po parametrima, ne treba da bude iznenađenje da linearni model može da predstavlja polinom. To je takođe linearna funkcija, ali nad bazom $(1, x, \dots, x^n)$.

Treba primetiti i da model pretpostavlja konstantnost varijanse normalne raspodele. Odnosno, očekuje se da je odstupanje predviđanja od ciljne vrednosti podjednako veliko i u slučaju velikih vrednosti ciljne promenljive i u slučaju malih. Ovo nije uvek realistično, a često nije ni poželjno. Recimo, ukoliko se model koristi za predviđanje dobiti u nekom poslu, greška od 10 hiljada dinara je zanemarljiva ukoliko je očekivana dobit oko 10 miliona dinara. Ipak, ako je očekivana dobit oko 10 hiljada dinara, onda je predviđanje sa takvom greškom potpuno beskorisno.



Slika 3.2: Dva modela dobijena linearnom regresijom. Jedan nad bazom $(1, x)$ i drugi nad bazom $(1, x, \dots, x^n)$.

Prilikom ocene parametara probabilističkih modela, tipično se koristi metod maksimalne verodostojnosti – potrebno je odrediti parametre za koje su dostupni podaci najverovatniji. Funkcija verodostojnosti je sledeća

$$\mathcal{L}(w) = p_w(y_1, \dots, y_N | x_1, \dots, x_N)$$

Uz prepostavku nezavisnosti instanci, dolazi se do jednostavnije forme ove funkcije:

$$\mathcal{L}(w) = \prod_{i=1}^N p_w(y_i | x_i)$$

Potrebno je rešiti problem

$$\max_w \mathcal{L}(w)$$

Funkcija verodostojnosti predstavlja proizvod gustina normalne raspodele izračunatih u različitim tačkama. Reprezentacija u vidu proizvoda se smatra nepoželjnom iz dva razloga. Prvi se tiče mogućnosti prekoračenja ili potkoračenja kada se množe veliki ili mali brojevi. Drugi se tiče praktične komplikovanosti izračunavanja parcijalnih izvoda proizvoda, koji su najčešće potrebni optimizacionim metodama. Stoga se umesto funkcije verodostojnosti češće koristi njen logaritam, kojim se proizvod prevodi u sumu. Kako je logaritam monotono rastuća funkcija, viša, pa i maksimalna, vrednost logaritma verodostojnosti nužno znači i višu vrednost same verodostojnosti. Takođe, kako se u praksi češće govorи о minimizaciji, nego о maksimizaciji, umesto logaritma verodostojnosti, češće se koristi njegova negativna vrednost (eng. *negative log likelihood*) ili skraćeno *NLL*, pa je problem koji se rešava sledeći:

$$\min_w -\log \mathcal{L}(w)$$

pri čemu važi

$$-\log \mathcal{L}(w) = -\sum_{i=1}^N \log p_w(y_i|x_i)$$

Primetimo da se funkcija $-\log p(y|x)$ može uzeti za funkciju greške. Ukoliko je verovatnoća vrednosti y za dato x velika, na primer, bliska 1, $-\log p(y|x)$ je blizu 0, a ako je mala, bliska nuli, ovaj izraz postaje ogroman. Nastavimo sa izvođenjem:

$$-\log \mathcal{L}(w) = -\sum_{i=1}^N \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - f_w(x_i))^2}{2\sigma^2}\right)$$

odnosno

$$-\log \mathcal{L}(w) = \frac{1}{2} \sum_{i=1}^N \log(2\pi\sigma^2) + \sum_{i=1}^N \frac{(y_i - f_w(x_i))^2}{2\sigma^2}$$

Kako prva suma i standardna devijacija ne zavise od parametara w , minimizacioni problem se svodi na

$$\min_w \sum_{i=1}^N (y_i - w \cdot x_i)^2$$

ili matrično

$$\min_w \|y - Xw\|^2$$

gde je X matrica redova x_1, \dots, x_N , a y kolona vektor (y_1, \dots, y_N) :

Lako je uveriti se da je kvadratna greška konveksna po parametrima w – svi sabirci su konveksne funkcije ovih parametara pošto se kvadrat koji je konveksna funkcija primenjuje na afinu transformaciju ovih parametara (pogledati dodatak 17). Ukoliko je moguće rešiti sistem $\nabla E(w) = 0$, problem je rešen.

$$\frac{\partial E(w)}{\partial w_j} = -2 \sum_{i=1}^N x_{ij} (y_i - w \cdot x_i) = 0$$

ili matrično:

$$X^T(y - Xw) = 0$$

$$X^T X w = X^T y$$

$$w = (X^T X)^{-1} X^T y$$

Izraz $(X^T X)^{-1} X^T$ naziva se *Mur-Penrouzovim pseudoinverzom* (eng. *Moore-Penrose pseudoinverse*) matrice X . Evo razloga za takav naziv. Norma $\|y - Xw\|$ bi mogla biti najmanje 0 ukoliko bi matrica X bila kvadratna i invertibilna. Tada bi rešenje bilo lako $w = X^{-1}y$. Kako ovaj uslov uopšte nije realističan, rešenje je bilo potrebno izvesti na prikazani način (mada ima i drugih). Ali Mur-Penrouzov pseudoinverz se ponaša prilično slično inverzu:

$$(X^T X)^{-1} X^T X = I$$

Množenjem matrice X njenim pseudoinverzom sleva, dobija se jedinična matrica, baš kao da je kvadratna i invertibilna matrica množena svojim inverzom.

Kako bi ovaj model dao interval poverenja, potrebno je oceniti i parametar σ^2 . I ovo se može raditi ocenom maksimalne verodostojnosti. Ispostavlja se da je nepristrasna ocena

$$\frac{1}{N-n} \sum_{i=1}^N (y_i - f(x_i))^2$$

Takođe, i možda još bolje, varijansa bi se mogla oceniti na odvojenom skupu podataka koji nije korišćen za obučavanje.

Algebarska forma rešenja problema linearne regresije jasno ukazuje na potencijalni problem. Matrica $X^T X$ ne mora biti invertibilna! Ovo se može desiti u slučaju linearnih zavisnosti među njenim kolonama. Da li je to realistično? Ukoliko se među podacima nalaze nabavna cena proizvoda i porez plaćen na taj proizvod, već te dve kolone su međusobno linearno zavisne, pa time i kolone matrice ukupno. Čak i ako matrica jeste invertibilna, može biti loše uslovljena zato što su kolone iako ne linearne zavisne, ipak visoko korelisane. Loša uslovjenost znači da je za male promene elemenata matrice, moguće dobiti drastično različite inverze ili rešenja odgovarajućeg sistema jednačina. Ovo je vrlo nepoželjno svojstvo u praksi, jer podatke nikada ne znamo sa savršenom tačnošću, odnosno male promene u odnosu na realne podatke su opšte mesto praktične primene. Postoji više načina rešavanja ovog problema, ali jedan, koji nam je već poznat, je regularizacija, odnosno rešavanje problema

$$\min_w \|y - Xw\|^2 + \frac{\lambda}{2} \|w\|_2^2$$

gde je polovina tu zbog lepseg izgleda rešenja:

$$w = (X^T X + \lambda I)^{-1} X^T y$$

Povećavanjem vrednosti parametra λ , invertovana matrica postaje sve bolje uslovljena, ali se naravno regularizovani problem udaljava od polaznog problema, kao što je sa regularizacijom uvek slučaj.

Još jedan praktični izazov i u polaznom i u regularizovanom problemu je potencijalno velika dimenzionalnost matrice $X^T X$. Ukoliko matrica X ima značajno više kolona nego vrsta, regularizacija i dalje omogućava invertibilnost matrice, ali njene dimenzije mogu postati prevelike za praktičnu upotrebu. Stoga se od izvedenog rešenja često odustaje u korist gradijentnih tehniki optimizacije koje će biti objašnjene kasnije.

Pored već navedenih, mogući su i drugi problemi. U podacima za obučavanje, neretko se dešava da se nađe mali broj podataka koji značajno odstupaju od zakonitosti koja važi za ostale podatke. Takvi podaci nazivaju se *odudarajućim* (eng. *outliers*). Kako takvi podaci utiču na model linearne regresije? Jedan način razmišljanja je probabilistički. Pristup obučavanju polazi od principa maksimalne verodostojnosti – potrebno naći parametre modela za koje su podaci najverovatniji. Kako verovatnoća nekog podatka eksponencijalno opada

sa udaljavanjem od proseka normalne raspodele, da bi taj podatak, a time i proizvod verovatnoća svih podataka, bio iole verovatan, neophodno je približiti mu prosek. Na taj način prisustvo odudarajućih podataka može značajno uticati na dobijeni model i nekada učiniti da on loše modeluje trend u podacima koji bi u suprotnom mogao modelovati. Zbog toga se ovakvi podaci nekada odstranjuju iz skupa za obučavanje. Ipak, to ne treba raditi po automatizmu, već treba proučiti prirodu konkretnih podataka i uveriti se da je isključivanje takvih podataka opravdano. Alternativa izbacivanju je modelovanje podataka nekom drugom raspodelom umesto normalne, a koja sporije opada, pa time ne daje drastično male vrednosti odudarajućim podacima i time im daje manji uticaj na model, ali je onda potrebno izvesti i nov algoritam koji odgovara toj raspodeli.

Iz čisto algebarske perspektive, prethodna osetljivost linearne regresije na odudarajuće podatke se mogla uočiti iz korišćenja kvadratne greške. Naime, kvadrat će velike razlike koje odgovaraju odudarajućim podacima učiniti još većim i time će se proces optimizacije neproporcionalno fokusirati na smanjivanje grešaka modela na tim podacima.

Jedna važna praktična prednost linearnih modela je njihova *interpretabilnost*, odnosno mogućnost analize i interpretacije, kojom se saznaje nešto o vezama koje važe između ciljne promenljive i atributa. Naime, linearni model

$$y = 5x_1 + 0x_2 - 0.5x_3$$

pokazuje da promena vrednosti promenljive x_1 proizvodi proporcionalnu promenu vrednosti ciljne promenljive, gde je koeficijent proporcionalnosti 5. Kako je ovaj koeficijent po apsolutnoj vrednosti veći od ostalih koeficijenata, zaključuje se da je promenljiva pojedinačno x_1 najvažnija za predviđanje vrednosti ciljne promenljive. Kako uz promenljivu x_2 стоји 0, jasno je da ta promenljiva uopšte nije korisna za predviđanje ciljne promenljive. Kada se njeni vrednosti menjaju, to se nikako ne odražava na ciljnu promenljivu. Promenljiva x_3 ima osetno manji uticaj od promenljive x_1 , ali dodatno primećujemo da je taj uticaj negativan. Često se kaže da je promenljiva x_1 pozitivno, a promenljiva x_3 negativno korelirana sa ciljnom promenljivom. Ovakvom analizom moguće je uočiti šta i koliko nam vrednosti promenljivih govore o vrednostima ciljne promenljive.

Prethodna diskusija važi samo ukoliko su zadovoljeni neki uslovi. Prvo, potrebno je da model ima malu grešku. Ukoliko model vrlo neuspšno predviđa vrednost ciljne promenljive, ovakva analiza nije od značaja. Drugo, promenljive se moraju meriti na istoj skali. Ukoliko se promenljiva x_1 meri u metrima, a promenljiva x_3 u milimetrima, ispostavlja se da mnogo drastičniju promenu ciljne promenljive uzrokuje promena promenljive x_3 za jedan metar, nego promenljive x_1 , iako koeficijenti sugerisu drugačije. Stoga je tipično da se pre primene linearne regresije izvrši *preprocesiranje* podataka kojim se sve promenljive svede na istu skalu. Često korišćen vid takve transformacije je *standardizacija* koja se sastoji u tome da se od svake vrednosti nekog atributa oduzme prosek svih

vrednosti tog atributa, pa da se potom svaka vrednost tog atributa podeli standardnom devijacijom svih vrednosti tog atributa. Time se obezbeđuje da svaki atribut ima prosek 0 i standardnu devijaciju 1.

Transformacije poput standardizacije se ne koriste samo zbog interpretabilnosti, već i zbog boljih računskih svojstava, poput brže konvergencije metoda. Ipak, postoji još jedan kontekst u kojem je suštinski važno voditi računa o redovima veličine u kojima se promenljive izražavaju. Ukoliko promenljive nisu standardizovane, neki koeficijenti mogu biti veliki samo zbog skale na kojoj se vrednost promenljive meri. Ukoliko model uključuje regularizaciju, poput ℓ_2 regularizacije, taj koeficijent će biti više umanjen nego drugi koeficijenti, iako razlog za njegovu veličinu nije suštinski vezan za odnose među promenljivim. Otud je pravilo da se standardizacija ili neka slična transformacija vrši uvek.

Još jedno praktično razmatranje odnosi se na upotrebu kategoričkih promenljivih. Njima se mogu pridružiti numeričke oznake kako bi se predstavile u računaru, ali te numeričke oznake se ne mogu koristiti kao numerički atributi. Naime, ako su klase novinskih članaka *ekonomija*, *sport* i *politika* označene brojevima 0, 1 i 2 i u modelu im odgovara koeficijent w , pojava vrednosti *sport* utiče na ciljnu vrednost kroz sabirak w , a pojava vrednosti *politika*, kao sabirak $2w$, dok pojava vrednosti *ekonomija* ne doprinosi zbiru. Ovakva aritmetika sa kategoričkim atributima nema smisla, a dodatno se postavlja pitanje šta bi bilo kada bismo drugačije označili različite kategorije. Otud se ovakav pristup *nikad* ne koristi. Umesto njega, koristi se *binarno kodiranje* (eng. *dummy coding*) tako što se uvode nove promenljive kojima se predstavljaju kategoričke promenljive. Ukoliko kategorička promenljiva x ima C vrednosti, uvodi se $C - 1$ novih binarnih promenljivih x_1, \dots, x_{C-1} takve da se i -ta kategorija za $1 \leq i \leq C - 1$ predstavlja vrednostima $(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_{C-1}) = (0, \dots, 0, 1, 0, \dots, 0)$, dok se kategorija C predstavlja vrednosima $(x_1, \dots, x_C) = (0, \dots, 0)$. Zapravo, pridruživanje kategorija datim kombinacijama bitova je proizvoljno. Bilo koje je prihvatljivo dok god se sva razlikuju. Šta bi bilo ako bismo umesto datog kodiranja koristili kodiranje koje ne definiše kategoriju C na specijalan način, već primenjuje dato kodiranje za $0 \leq i \leq C$? U tom slučaju kolone koje odgovaraju novim promenljivim bi se uvek sumirale na 1. Kako se slobodni član može videti baš kao parametar koji uvek množi jedinicu, i njemu u matrici podataka odgovara jedinica, što znači da su te kolone i kolona jedinica međusobno linearne zavisne, a kao što je već naglašeno, to vodi neinvertibilnosti matriće $X^T X$. Ipak, ovakvo kodiranje se redovno koristi kod nekih drugih modela poput neuronskih mreža.

Jedna slabost linearnih modela je nezavisnost dejstva atributa na ciljnu promenljivu. Naime, jedinična promena svakog atributa doprinosi promeni ciljne promenljive tačno proporcionalno koeficijentu koji odgovara tom atributu, nezavisno od vrednosti drugih atributa. Ovo je jaka prepostavka i neadekvatna u mnogim problemima. Primera radi, u genetici, ispoljavanje nekih gena može zavisiti od varijacije nekog drugog gena. Jedan način da se linearni modeli značajno ojačaju predstavlja uključivanje *interakcija* – proizvoda atributa, kao novih atributa u model. U slučaju linearne regresije, na koju ova tehnika nije

ograničena, model sa interakcijama se može izraziti na sledeći način:

$$f_w(x) = w_0 + \sum_{i=1}^n w_i x_i + \sum_{1 \leq i \leq j \leq n} w_{ij} x_i x_j$$

Primetimo da u prisustvu interakcija, vrednost jedne promenljive kontroliše ispoljavanje druge promenljive. Konkretno, ukoliko važi $x_i = 0$, promena promenljive x_j se ne odražava na vrednost proizvoda $x_i x_j$. Visoka vrednost promenljive x_i će omogućiti da male promene promenljive x_j imaju veliki efekat na vrednost ciljne promenljive. Interakcije značajno povećavaju izražajnost linearnih modela i vrlo se često koriste u njihovim praktičnim primenama. Treba primetiti da su ovako prošireni modeli i dalje linearni po parametrima i da se za njihovo obučavanje koriste standardni algoritmi.

3.2 Logistička regresija

Dok linearna regresija predstavlja regresioni model, logistička regresija, uprkos svom nazivu, predstavlja model binarne klasifikacije. Osnovna pretpostavka sa probabilističke tačke gledišta je pretpostavka Bernulijeve raspodele ciljne promenljive y , pri datim vrednostima atributa x . Drugim rečima za date vrednosti atributa x , postoji parametar $\mu \in [0, 1]$ tako da važi

$$p(y|x) = \begin{cases} \mu, & y = 1 \\ 1 - \mu, & y = 0 \end{cases}$$

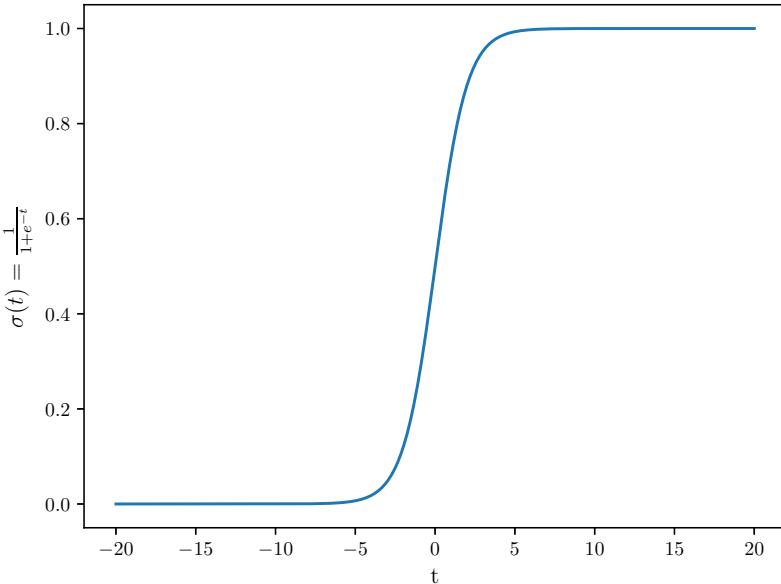
pri čemu je $p(y|x)$ diskretna funkcija raspodele. Umesto ovog izraza, češće se piše samo $p(y = 1|x) = \mu$, dok se vrednost za $p(y = 0|x)$ odatle jednoznačno izračunava. Ovaj model nije kompletan, jer nije ustanovljena zavisnost parametra μ od vrednosti atributa x . Kako taj parametar mora biti u intervalu $[0, 1]$ da bi verovatnoća bila ispravno definisana, do sada korišćeni linearni model nije prihvatljiv. Ipak, ukoliko bi se vrednost linearног modela, koja može biti u intervalu $[-\infty, \infty]$, transformisala nekom monotonom (i po mogućству neprekidnom i diferencijabilnom) funkcijom u interval $[0, 1]$, takav model bi bio prihvatljiv. Jedna takva funkcija, koja se često koristi u mašinskom učenju, je *sigmoidna funkcija*:

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$

Njen grafik je prikazan na slici 3.3. Postoje i druge funkcije koje zadovoljavaju navedene kriterijume. Glavni razlog korišćenja sigmoidne funkcije vezan je za njeno prirodno pojavljivanje u uopštenim linearnim modelima o kojima će biti reči kasnije.

Ukoliko se vrednost linearног modela transformiše sigmoidnom funkcijom, model logističke regresije će biti određen sledećom relacijom:

$$p_w(y = 1|x) = \sigma(w \cdot x)$$



Slika 3.3: Sigmoidna funkcija.

Time je definisana zavisnost parametra Bernulijeve raspodele od vrednosti atributa x . Verovatnoća $p_w(y=1|x)$ pripadanja klasi 1 je utoliko veća, što je tačka x dalje od hiperravnih definisane relacijom $w \cdot x = 0$ u njenom pozitivnom poluprostoru. Verovatnoća pripadanja drugoj klasi je utoliko veća što je tačka dublje u negativnom poluprostoru.

Primetimo da se izrazom $\sigma(w \cdot x)$ ne definiše raspodela, već verovatnoća da je $y = 1$. Puna specifikacija se može zapisati u obliku

$$p_w(y|x) = \sigma(w \cdot x)^y (1 - \sigma(w \cdot x))^{1-y}$$

Očito, kada se umesto y uvrsti 1 ili 0, dobijaju se očekivane vrednosti verovatnoće. Kako modeluje uslovnu raspodelu, ovaj model je očigledno probabilistički diskriminativni.

Obučavanje ovog modela, odnosno ocena njegovih parametara se i u ovom slučaju zasniva na principu maksimalne verodostojnosti. Funkcija verodostojnosti je data izrazom

$$\mathcal{L}(w) = p_w(y_1, \dots, y_N | x_1, \dots, x_N)$$

Kao i u slučaju linearne regresije, uz prepostavku nezavisnosti instanci, dolazi

se do jednostavnije forme ove funkcije:

$$\mathcal{L}(w) = \prod_{i=1}^N p_w(y_i|x_i)$$

i potrebno je rešiti problem

$$\max_w \mathcal{L}(w)$$

Prelaskom na negativnu vrednost logaritma funkcije verodostojnosti, dobija se

$$\begin{aligned} -\log \mathcal{L}(w) &= -\sum_{i=1}^N \log(\sigma(w \cdot x_i)^{y_i} (1 - \sigma(w \cdot x_i))^{1-y_i}) = \\ &= -\sum_{i=1}^N y_i \log \sigma(w \cdot x_i) + (1 - y_i) \log(1 - \sigma(w \cdot x_i)) \end{aligned}$$

Za ovu funkciju se može pokazati da je konveksna po w . Obično ima (globalni) minimum, koji se obično pronađe pomoću Njutnove metode ili bilo kojom gradijentnom metodom.

Primetimo da $L(u, v) = -u \log v - (1 - u) \log(1 - v)$ predstavlja funkciju greške koja se naziva *unakrsnom entropijom* i koristi se često u kontekstu probabilističke klasifikacije. Lako je videti da ova funkcija ima smisla u ulozi funkcije greške. Ukoliko je $u = 1$ i $v = 0$, zbog $\log v$, dobija se beskonačna greška. Analogno u slučaju kad je $u = 0$ i $v = 1$. S druge strane, kada je $u = 1$ i $v = 1$ ili $u = 0$ i $v = 0$ uz dogovor $0 \log 0 = 0$, važi $L(u, v) = 0$. Odnosno, kada nema greške vrednost funkcije je 0, a kad je ima vrednost je pozitivna, što je očekivano ponašanje funkcije greške.

Napomenimo i da se logistička regresija tipično ne koristi tačno u prikazanom obliku, već se najčešće koristi regularizovana varijanta. Sledeći primer demonstrira zanimljivo svojstvo neregularizovane logističke regresije.

Primer 1 Neka je dat skup za obučavanje $\mathcal{D} = \{(-1, 0), (1, 1)\}$ takav da postoji jedan atribut i ciljna promenljiva. Minimizacioni problem se može svesti na sledeći:

$$\min_{w_0, w_1} -\log(1 - \sigma(w_0 - w_1)) - \log \sigma(w_0 + w_1)$$

odnosno

$$\min_{w_0, w_1} \log(1 + \exp(w_0 - w_1)) + \log(1 + \exp(-w_0 - w_1))$$

Nije teško uveriti se da ova funkcija nema minimum uprkos konveksnosti, već monotono opada sa povećanjem parametra w_1 . Ovo znači da gradijent nikad neće biti nula i da gradijentne metode optimizacije neće konvergirati osim usled zadovoljenja unapred zadate preciznosti. Očito, to što model za tekuće vrednosti parametara ispravno klasificuje podatke u skupu za obučavanje, ne znači da

greška ne postoji. Ali onda se postavlja pitanje u čemu se sastoji ta greška? Vratimo se na polaznu formulaciju problema. Potrebno je maksimizovati verodostojnost bernulijeve promenljive. Za dati skup podataka, verodostojnost se maksimizuje kada važi $p(y = 1|x = -1) = 0$ i $p(y = 1|x = 1) = 1$. Međutim, model pretpostavlja da važi $p(y = 1|x) = \sigma(w_0 + w_1x)$. Sigmoidna funkcija ne može uzeti vrednosti 0 i 1, ali im se može približiti proizvoljno blizu. Otud je jasno da data formulacija kažnjava nesigurnost modela u situaciji u kojoj bi posmatrajući skup za obučavanje mogao biti potpuno siguran u svoja predviđanja. Takođe, što se procesom optimizacije više uveća koeficijent w_1 , to je model sigurniji u predviđanja na skupu za obučavanje. Da li je ovakvo poнаšanje poželjno? Čitalac bi trebalo da je primetio da ovo zapravo predstavlja preprilagođavanje. Nema smisla očekivati od modela koji je obučen na dve instance da bude potpuno siguran u predviđanje. Tako nešto je vrlo nepoželjno. To je još jedna ilustracija potrebe za korišćenjem regularizacije.

Osvrnamo se na dato ponašanje na još jedan način. Optimalna vrednost za w_0 je 0, tako da vredi posmatrati samo funkciju $\sigma(w_1x)$. Podešavanjem parametra w_1 kontroliše se strmost uspona sigmoidne funkcije u okolini nule. Kako w_1 teži beskonačnosti, tako data funkcija teži indikatorskoj funkciji $I(x \geq 0)$.¹ To bi značilo da je model siguran da su sve instance za koje je $x < 0$ instance klase 0, a sve instance za koje je $x > 0$ instance klase 1, uprkos tome što u skupu za obučavanje nema ni jedne instance čija je vrednost atributa x unutar intervala $(-1, 1)$ i moglo bi se desiti da je prava granica između dve klase bilo gde unutar tog intervala, kao i da uopšte ne postoji jasna granica, pa osnova za potpunu sigurnost modela nema.

Jedan čest pogled na logističku regresiju je taj da modeluje logaritam kočionika verovatnoće (eng. *log odds ratio*) dve različite klase linearnim modelom. Naime, važi

$$\log \frac{p(y = 1|x)}{p(y = 0|x)} = \log \frac{\frac{1}{1+\exp(-w \cdot x)}}{\frac{\exp(-w \cdot x)}{1+\exp(-w \cdot x)}} = w \cdot x$$

3.3 Multinomijalna logistička regresija

Multinomijalna logistička regresija predstavlja metod višeklasne klasifikacije. Uprkos nazivu, ipak ne počiva na multinomijalnoj, već na *kategoričkoj raspodeli*. Kao što binomna raspodela predstavlja raspodelu broja uspeha u N realizacija Bernulijeve promenljive, tako multinomijalna raspodela predstavlja raspodelu broja različitih ishoda pri realizacijama kategoričke raspodele. Kategorička raspodela je raspodela koja svakom ishodu i , iz konačnog skupa ishoda, dodeljuje verovatnoću p_i . Stoga, multinomijalna logistička regresija

¹Potpuno precizno, teži funkciji koja je jednaka datoj indikatorskoj funkciji svuda osim u nuli, gde ima vrednost 0.5.

ocenjuje kategoričku raspodelu, odnosno:

$$p(y|x) = \text{Cat}(p_1, \dots, p_C) = \begin{cases} p_1, & y = 1 \\ p_2, & y = 2 \\ \vdots & \vdots \\ p_C, & y = C \end{cases}$$

gde je C broj klasa, pri čemu mora važiti $p_1 + \dots + p_C = 1$ i $p_i \geq 0$ za svako $i = 1, \dots, C$. Očito, ova metoda je u stanju da vrši klasifikaciju u više klase.

Ova raspodela se može modelovati analogno modelovanju koje vrši model logističke regresije. Krenimo od poslednje relacije u prethodnom odeljku. Modelujmo linearnim modelom logaritam odnosa verovatnoća svih klasa u odnosu na jednu privilegovanu. Na primer, poslednju.

$$\log \frac{p(y = i|x)}{p(y = C|x)} = w_i \cdot x \quad i = 1, \dots, C \quad (3.1)$$

Specifično za $i = C$ mora važiti

$$0 = \log \frac{p(y = C|x)}{p(y = C|x)} = w_C \cdot x$$

za svako x , odnosno $w_C = 0$, odnosno $\exp(w_C \cdot x) = 1$. Poslednja činjenica će biti korišćena više puta u nastavku. Primenom eksponencijalne funkcije na obe strane relacija datih jednakostima 3.1 dobija se:

$$p(y = i|x) = p(y = C|x) \exp(w_i \cdot x)$$

Sumirajući ovakve jednakosti po $i = 1, \dots, C - 1$, dobija se

$$1 - p(y = C|x) = p(y = C|x) \sum_{i=1}^{C-1} \exp(w_i \cdot x)$$

$$p(y = C|x) = \frac{1}{1 + \sum_{i=1}^{C-1} \exp(w_i \cdot x)} = \frac{\exp(w_C \cdot x)}{\exp(w_C \cdot x) + \sum_{i=1}^{C-1} \exp(w_i \cdot x)} = \frac{\exp(w_C \cdot x)}{\sum_{i=1}^C \exp(w_i \cdot x)}$$

Uvrštavanjem u formulu za $p(y = i|x)$ dobija se

$$p(y = i|x) = \frac{\exp(w_i \cdot x)}{\sum_{i=1}^C \exp(w_i \cdot x)} \quad i = 1, \dots, C$$

Za $C = 2$, dobija se baš logistički model.

Izvođenje optimizacionog problema je analogno izvođenju za logističku regresiju uz jedan dogovor. Neka je vrednost ciljne promenljive $y = i$ predstavljena vektorom dužine C , koji ima sve elemente 0, osim na mestu i na kojim ima vrednost 1. Funkcija verodostojnosti je onda data izrazom:

$$\mathcal{L}(w) = \prod_{i=1}^N \prod_{j=1}^C \left(\frac{\exp(w_j \cdot x_i)}{\sum_{k=1}^C \exp(w_k \cdot x_i)} \right)^{y_{ij}}$$

Optimizacioni problem je onda

$$\max_w \mathcal{L}(w)$$

Prelaskom na negativnu vrednost logaritma funkcije verodostojnosti, dobija se

$$\begin{aligned} -\log \mathcal{L}(w) &= -\sum_{i=1}^N \sum_{j=1}^C y_{ij} \log \left(\frac{\exp(w_j \cdot x_i)}{\sum_{k=1}^C \exp(w_k \cdot x_i)} \right) = \\ &= -\sum_{i=1}^N \left[\sum_{j=1}^C y_{ij} w_j \cdot x_i - \sum_{j=1}^C y_{ij} \log \left(\sum_{k=1}^C \exp(w_k \cdot x_i) \right) \right] \end{aligned}$$

Kako $\log \sum_{k=1}^C \exp(w_k \cdot x_i)$ ne zavisi od j i kako je $\sum_{j=1}^C y_{ij} = 1$, dobija se minimizacioni problem

$$\min_{w_1, \dots, w_{C-1}} -\sum_{i=1}^N \left[\sum_{j=1}^C y_{ij} w_j \cdot x_i - \log \left(\sum_{k=1}^C \exp(w_k \cdot x_i) \right) \right]$$

Podsećamo da je vektor koeficijenata w_C jednak 0, pa se po njemu ne minimizuje. Logaritam sume eksponencijalnih funkcija je konveksna funkcija, pa je i ceo problem konveksan i stoga pogodan za gradijentne metode optimizacije.

Primetimo da je model multinomijalne logističke regresije manje interpretabilan nego model standardne logističke regresije. Naime, potrebno je interpretirati koeficijente više modela, a u različitim modelima mogu dominirati parametri različitih atributa. Tada je teže proceniti koji atributi su važniji od drugih.

3.4 Uopšteni linearni modeli

Ispostavlja se da se svi pomenuti probabilistički modeli mogu predstaviti kao instance jedne opštije vrste modela – *uopštenih linearnih modela*. Uopšteni linearni modeli se sastoje iz tri komponente. Prva je forma raspodele $p(y|x)$, druga je linearni model i treća je *funkcija veze* (eng. *link function*) ili samo *veza* koja povezuje taj linearni model sa očekivanjem raspodele. Odnosno, pretpostavlja se da važi

$$g(\mu) = w \cdot x$$

U slučaju linearne regresije, upravo je prosek bio modelovan linearnim modelom, pa se linearna regresija može videti kao instance uopštenog linearog modela kod koje je uslovna raspodela normalna, a veza je identitet $g(\mu) = \mu$. U slučaju logističke regresije, veza je bila

$$g(\mu) = \log \left(\frac{\mu}{1-\mu} \right)$$

a raspodela je bila Bernulijeva. Slična relacija se može izvesti i za multinomijalnu logističku regresiju, samo što će sve veličine biti vektorske.

Forma raspodele se obično bira iz *eksponencijalne familije*, što je familija kojoj pripadaju sve pomenute raspodele, ali i mnoge druge poput Puasonove, eksponencijalne, χ^2 , Dirihelove, beta i gama raspodele. Eksponencijalna familija ima mnoga poželjna svojstva, ali u njih ovde nećemo ulaziti. Forma gustine raspodele za ovu familiju je sledeća²

$$p(x) = \exp\left(\frac{\theta \cdot x - A(\theta)}{B(\sigma)} + c(x, \sigma)\right)$$

Nenegativan parametar σ naziva se parametrom disperzije, a vektor θ vektorom prirodnih parametara.

Može se pokazati da važi

$$\nabla_{\theta} A(\theta) = \mathbb{E}[x] \quad \nabla_{\theta}^2 A(\theta) = \text{cov}[x]B(\sigma)$$

Za raspodele koje smo do sada koristili se lako može pokazati da pripadaju eksponencijalnoj familiji. Za normalnu raspodelu važi

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = \exp\left(\frac{x\mu - \mu^2/2}{\sigma^2} - \frac{1}{2}\left\{\frac{x^2}{\sigma^2} + \log(2\pi\sigma^2)\right\}\right)$$

pa je $\theta = \mu$, $A(\theta) = \frac{\mu^2}{2} = \frac{\theta^2}{2}$, $B(\sigma) = \sigma^2$ i $c(x, \sigma) = -\frac{1}{2}\left(\frac{x^2}{\sigma^2} + \log(2\pi\sigma^2)\right)$

Za Bernulijevu raspodelu važi

$$p(x) = \mu^x(1-\mu)^{(1-x)} = \exp(x \log \mu + (1-x) \log(1-\mu))$$

pa je $\theta = \log\left(\frac{\mu}{1-\mu}\right)$, $A(\theta) = \log\left(\frac{1}{1-p}\right) \log(1 + \exp(\theta))$, $B(\sigma) = 1$ i $c(x, \sigma) = 0$. Otud je $\mu = \sigma(\theta)$.

Slično se može pokazati i za kategoričku raspodelu.

Veza za koju važi $g(\mu) = \theta$ naziva se *kanonskom vezom*. U tabeli 3.1 prikazane su neke funkcije eksponencijalne familije i njihove kanonske veze. Ipak, veza koja se koristi ne mora nužno biti kanonska.

Uopšteni linearni modeli obično se ocenjuju metodom maksimalne verodostojnosti.

3.5 Naivni Bajesov algoritam

Naivni Bajesov algoritam se zasniva na modelovanju raspodele ciljne promenljive y pri datim vrednostima promenljive x , korišćenjem Bajesove formule:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

²Postoje i nešto drugačije formulacije.

Raspodela	Domen	Upotreba	Veza
Normalna	\mathbb{R}	neprekidne promenljive	$g(\mu) = \mu$
Eksponencijalna	\mathbb{R}^+	neprekidne pozitivne promenljive	$g(\mu) = \mu^{-1}$
Puasonova	\mathbb{N}	broja događaja u jedinici vremena	$g(\mu) = \log(\mu)$
Bernulijeva	$\{0, 1\}$	binarne promenljive	$g(\mu) = \log\left(\frac{\mu}{1-\mu}\right)$

Tabela 3.1: Raspodele eksponencijalne familije, njihovi domeni, vrste podataka na koje se primenjuju i kanonske funkcije veze.

Najčešće se primenjuje na problem klasifikacije, pa se najčešće govorio o naivnom Bajesovom klasifikatoru. Iako to nije neophodno, u nastavku ćemo pretpostaviti da su sve promenljive kategoričke. Pitanje koje bi odmah trebalo da se nametne je zašto je lakše modelovati desnu stranu jednakosti od leve i odgovor nije očigledan. Načelno je lako modelovati raspodele jedne promenljive. Ukoliko je promenljiva diskretna, kao što smo pretpostavili, moguće je oceniti njenu raspodelu pomoću frekvencija njenih različitih vrednosti u skupu za obučavanje. Ipak, raspodela sa leve strane je uslovna i to nije lako jer bismo morali brojati pojavljivanja vrednosti promenljive y u slučajevima u kojima su odgovarajuće vrednosti atributa baš one date vektorom x . Ukoliko se baš te vrednosti atributa nisu našle u skupu za obučavanje ili su vrlo retke, onda nije lako adekvatno oceniti raspodelu promenljive y za dato x . Ako je vektor x visokodimenzionalan, to je i vrlo verovatno. Sa desne strane, lako je oceniti raspodelu promenljive y , pošto je ona jednodimenzionalna, ali x je visokodimenzionalno i prokletstvo dimenzionalnosti predstavlja prepreku za ocenu $p(x|y)$ i $p(x)$. Primetimo da $p(x)$ uopšte ne zavisi od y . Ta vrednost je ista za sve vrednosti promenljive y . Kako je u predviđanju potrebno samo naći najverovatniju vrednost ciljne promenljive, $p(x)$ se uopšte ne mora izračunavati, već je dovoljno izračunati vrednost u brojiocu, odnosno nije važna tačna vrednost, već proporcionalnost:

$$p(y|x) \sim p(x|y)p(y)$$

Ipak, ovo ne olakšava problem zato što je i dalje potrebno modelovati raspodelu $p(x|y)$. Kako bi se prevazišlo prokletstvo dimenzionalnosti, pretpostavlja se uslovna nezavisnost atributa kada je data vrednost ciljne promenljive, odnosno da važi:

$$p(x|y) = \prod_{i=1}^n p(x_i|y)$$

Obratimo pažnju da x_i ne predstavlja i -tu instancu, već vrednost i -tog atributa. Uslovne raspodele $p(x_i|y)$ su jednodimenzionalne i lako se mogu modelovati – za dato y , na osnovu instanci iz skupa za obučavanje koje imaju vrednost ciljne promenljive baš y , modeluje se jednodimenzionalna raspodela promenljive x_i . Ukoliko je promenljiva x_i kategorička, to je jednostavno – svodi se na računanje frekvencija njenih različitih vrednosti. Kao što je rečeno u delu 17,

uslovna nezavisnost je slabija pretpostavka od nezavisnosti, pa se uz takvu pretpostavku ne gubi sva informacija koja bi bila izgubljena uz punu pretpostavku o nezavisnosti. Ipak, ne možemo očekivati da će ova pretpostavka u praksi biti ispunjena, pa je rešenje aproksimativno, a epitet *naivni* potiče upravo od ove pretpostavke.

Pun model je dat relacijom

$$p(y|x) \sim p(y) \prod_{i=1}^n p(x_i|y)$$

Kako modeluje uslovnu raspodelu, deluje da se može reći da je model diskriminativni. Ipak, ova uslovna raspodela se modeluje tako što se modeluje $p(x|y)p(y)$, što je jednako $p(x, y)$. Odnosno, da bi se rešio laki problem, rešava se teži – modelovanje zajedničke raspodele, pa se zapravo radi o generativnom modelu! Pretpostavka uslovne nezavisnosti ne znači da model nije generativni, već samo da možda nije dobar generativni model.

Primer 2 *Kod lekara dolazi pacijent koji se žali na osećaj hladnoće i blagu glavobolju. Lekar nije bas najstručniji i pokušava da prepostavi dijagnozu preturajući po kartonima drugih pacijenata, pokušavajući da ustanovi kakve su simptome imali pacijenti koji su imali grip, a kakve oni koji nisu. Kako nema mnogo vremena, mora da odlučuje na osnovu malog uzorka. U tabeli 3.2 dati su podaci o nekoliko pacijenata čije je kartone našao. Da je dobar matematičar (što je, imajući u vidu kakav je lekar, malo verovatno), lekar bi na osnovu ovog uzorka izračunao odgovarajuće proizvode koji su proporcionalni uslovnim verovatnoćama da pacijent ima grip i da nema grip:*

$$p(Da|Da, Ne, Blaga, Ne) \sim p(H = Da|Da)p(C = Ne|Da)p(Gl = B|Da)p(Gr = Ne|Da)p(Grip = Da)$$

$$= \frac{3}{5} \cdot \frac{1}{5} \cdot \frac{2}{5} \cdot \frac{1}{8} = \frac{3}{500}$$

$$p(Ne|Da, Ne, Blaga, Ne) \sim p(H = Da|Ne)p(C = Ne|Ne)p(Gl = B|Ne)p(Gr = Ne|Ne)p(Grip = Ne)$$

$$= \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{1}{3} \cdot \frac{2}{8} = \frac{1}{54}$$

Iako nije izračunao tačne verovatnoće da pacijent ima ili nema grip, lekar može da zaključi da je verovatnije da ga nema.

Prednost naivnog Bajesovog algoritma je da se ocene verovatnoća lako ažuriraju kako pristižu novi podaci – prebrojavanjem različitih vrednosti promenljivih. Takođe, iako je ovde diskutovan u kontekstu kategoričkih atributa, može se primeniti i nad kontinualnim ukoliko se diskretizuju ili ukoliko se pretpostavi neka forma raspodele atributa i uradi ocenu parametara te raspodele. Pored pretpostavke uslovne nezavisnosti atributa, postoji još jedna važna miana ovog algoritma. Ukoliko je neka vrednost nekog atributa malo verovatna,

Hladnoća	Curenje iz nosa	Glavobolja	Groznica	Grip
Da	Ne	Blaga	Da	Ne
Da	Da	Ne	Ne	Da
Da	Ne	Jaka	Da	Da
Ne	Da	Blaga	Da	Da
Ne	Ne	Ne	Ne	Ne
Ne	Da	Jaka	Da	Da
Ne	Da	Jaka	Ne	Ne
Da	Da	Blaga	Da	Da

Tabela 3.2: Tabela podataka za problem dijagnostifikovanja gripa.

može se desiti da se ona ne pojavi u skupu za obučavanje. U tom slučaju verovatnoća te vrednosti je 0. Kada se u predviđanju pojavi ta vrednost, njen prisustvo u proizvodu verovatnoća $\prod_{i=1}^n p(x_i|y)$ čini da ceo proizvod bude nula. Ovakvo ponašanje očito nije poželjno, pa se umesto nule, ovakvim vrednostima dodeljuje neka vrlo mala pozitivna vrednost. Postoje i neke složenije tehnike za rešavanje ovog problema, ali se njima nećemo baviti.

Glava 4

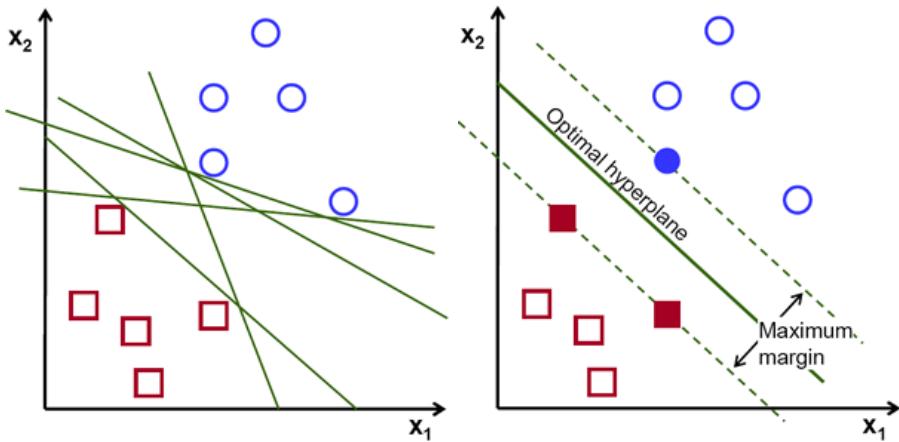
Modeli zasnovani na širokom pojasu

Pouzdanost predviđanja je od kritičnog značaja u mašinskom učenju. Pojam pouzdanosti se verovatno može definisati na više različitih načina. U ovoj glavi će biti razmotren jedan način dizajna algoritama mašinskog učenja koji se zasniva baš na razmatranju kada se predviđanje može smatrati pouzdanim. Radi se o *modelima zasnovanim na širokom pojasu* (eng. *large margin*).

Razmotrimo sledeći neformalan primer. Neka se održava trka na 100 metara. Pobednik se, pouzdanosti radi, proglašava na osnovu foto finiša. Ipak, kamera je omanula baš u toku trke. Pod kojim uslovima sudija može biti pouzdan da je ispravno proglašio pobednika? Ukoliko su prva dva trkača zajedno prošla kroz cilj, teško je reći ko je prvi. Odluka je pouzdana tek ako je prvi trkač ostavio drugog na bezbednom odstojanju! Razmotrimo još jedan primer. Dve populacije iste vrste mrava sakupljaju hranu relativno blizu jedni od drugih. Ukoliko entomolog uoči mrava na nekoj lokaciji, pod kojim uslovima može samo na osnovu lokacije (vrsta je ista, pa mravi isto izgledaju) biti siguran kojoj populaciji mrav pripada? Možda onoč čiji je mravinjak najbliži? Mrav bi verovatno trebalo da bude blizu svog mravinjaka, ali to nije sigurno, pošto jedna populacija može pokrivati nešto veću površinu, a mogu i zalistiti jedni drugima u teritoriju. Ukoliko bi utvrdio da ove dve populacije izbegavaju jedna drugu, odnosno da se drže jedna od druge na bezbednom odstojanju, mogao bi lako utvrditi koja teritorija pripada kojoj populaciji. U oba slučaja, od značaja je koncept bezbednog odstojanja, odnosno praznog prostora između objekata koje treba razlikovati. Način na koji se to može matematički definisati zavisi od konteksta, ali ključno je voditi se tom intuicijom.

4.1 Metod potpornih vektora za klasifikaciju

Metod potpornih vektora (eng. *support vector machine*) je jedan od važnijih metoda mašinskog učenja. Zasnovan je na jasnoj geometrijskoj intuiciji. Pret-



Slika 4.1: Prave koje razdvajaju dve klase

postavimo da imamo dve klase tačaka u ravni i neka su klase takve da se između elemenata te dve klase može povući prava, tako da su svi elementi jedne klase sa jedne strane, a elementi druge klase sa druge strane. Ovaj uslov linearne razdvojivosti nije realističan uslov, ali ćemo za sad pretpostaviti da važi. Ako nacrtamo različite rasporede takvih tačaka, primetićemo da prava koja ih razdvaja praktično nikad nije jedna, već da je moguće povući više njih. Ovo je prikazano na slici 4.1. Ipak, neke prave nam deluju bolje od ostalih. Na istoj slici je prikazana i optimalna prava, što je prava sa najvećim rastojanjem do najbliže joj tačke podataka, odnosno sa najširim pojasom praznog prostora oko nje. Intuitivno, posmatrajući sliku, prava koja bi bila pod drugačijim uglom i prolazila bliže nekoj od tačaka podataka bi nosila veći rizik da neka tačka koja nije u datim podacima završi sa pogrešne strane prave. Sada je prikazani princip potrebno formalizovati.

Jednačina hiperravnji je

$$w \cdot x + w_0 = 0$$

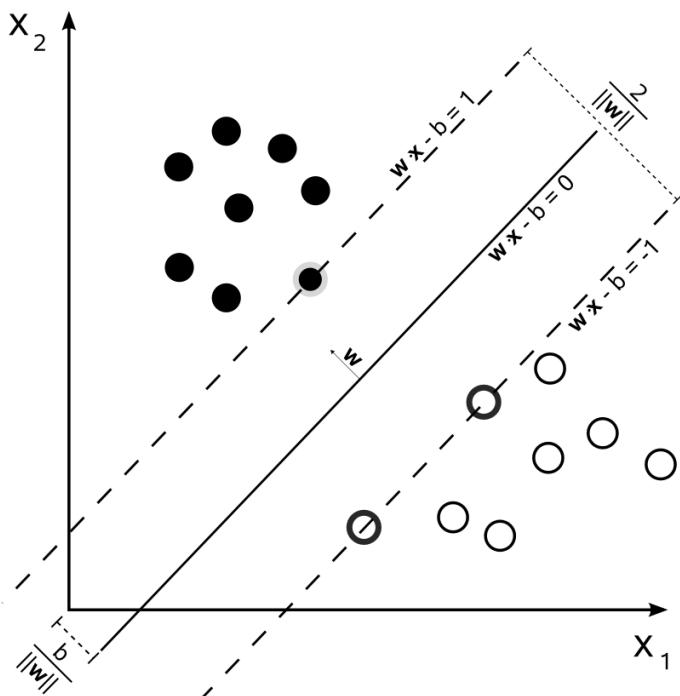
gde je w_0 slobodni član. *Optimalna hiperravan*, odnosno *hiperravan najšireg pojasa* je podjednako udaljena od najbližih predstavnika obe klase. Ako bi bila bliže jednoj klasi, mogla bi se udaljiti ka drugoj, kako bi se povećalo minimalno rastojanje. Stoga, hiperravnji paralelne optimalnoj imaju jednačine

$$w \cdot x + w_0 = c$$

$$w \cdot x + w_0 = -c$$

Deljenjem svih jednačina sa c , za neke nove koeficijente w i w_0 za koje ćemo zadržati iste oznake, dobijaju se jednačine sve tri hiperravnji:

$$w \cdot x + w_0 = 0$$



Slika 4.2: Optimalna hiperravan i paralelne joj hiperravnini koje leže na potpornim vektorima.

$$w \cdot x + w_0 = 1$$

$$w \cdot x + w_0 = -1$$

kao što je prikazano na slici 4.2. Tačke podataka koje se nalaze na pomenutim hiperravnima paralelnim optimalnoj nazivaju se potpornim vektorima, pošto deluju kao da pružaju potporu datom sistemu od tri hiperravnini – tako da ne može da mrdne ni levo ni desno! Po njima je ova metoda i dobila ime. Rastojanje između optimalne hiperravni i jedne od pomenutih hiperravnini koje su joj paralelne je upravo pojas koji treba da bude što veći. Na osnovu jednačine rastojanja tačke od hiperravnini

$$\frac{|w \cdot x + w_0|}{\|w\|_2}$$

i činjenice da za svaku od tačaka sa ovih hiperravnini važi $|w \cdot x + w_0| = 1$, dobija se da je ukupno rastojanje između klasa, u pravcu normalnom u odnosu na optimalnu hiperravan $2/\|w\|$. Otud se optimalna hiperravan dobija pronalaženjem koeficijenata koji maksimizuju ovaj izraz pod uslovima da su sve tačke sa pravih strana te hiperravnini. Ako se izrazimo u terminima minimizacije, umesto

maksimizacije dobijamo sledeći optimizacioni problem:

$$\min_{w, w_0} \frac{\|w\|_2}{2}$$

$$y_i(w \cdot x_i + w_0) \geq 1 \quad i = 1, \dots, N$$

pri čemu se podrazumeva da važi $y_i \in \{-1, 1\}$. Dodatni uslovi izražavaju potrebu da sve tačke budu na većem rastojanju od optimalne hiperravnih nego što su potporni vektori koji su na rastojanju 1. Za rešavanje ovog optimizacionog problema i problema izvedenih iz njega, koriste se posebno konstruisani algoritmi.¹

Suštinski problem sa ovom formulacijom predstavlja činjenica da u praksi retko možemo očekivati linearnu razdvojivost klasa. Prosto, stvarni problemi su komplikovani. Otud je neophodno prihvatići neke greške, uz zahtev da budu što manje. Do nove formulacije se dolazi uvođenjem novih promenljivih ξ_i za svaku instancu u skupu za obučavanje, koje mere koliko je svaka instanca daleko od hiperravnih određene potpornim vektorima njene klase, ali samo pod pretpostavkom da je sa pogrešne strane. Ovakav metod naziva se metodom potpornih vektora sa *mekim pojasmom* (eng. *soft margin*), a optimizacioni problem izgleda ovako:

$$\min_{w, w_0} \frac{\|w\|_2}{2} + C \sum_{i=1}^N \xi_i$$

$$y_i(w \cdot x_i + w_0) \geq 1 - \xi_i \quad i = 1, \dots, N$$

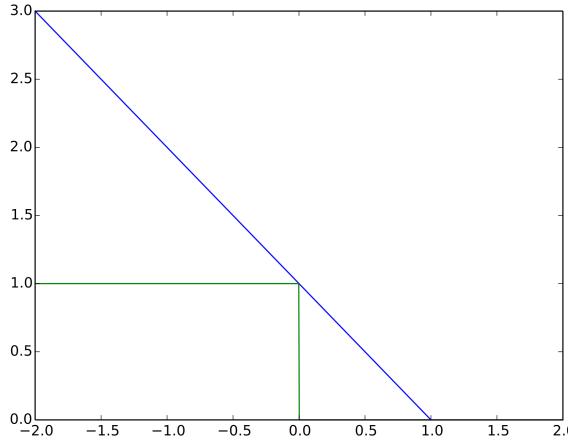
$$\xi_i \geq 0 \quad i = 1, \dots, N$$

Uloga hiperparametra C koji mora biti nenegativan je da kontroliše koliko težine se pridaje greškama. Razmotrimo ponašanje ovog problema za različite vrednosti hiperparametra C . Ukoliko važi $C = 0$, greške uopšte nisu važne i mogu biti proizvoljno velike. Otud je optimalno rešenje $w = 0$. Ukoliko je C ogromno, onda su greške izuzetno važne, a pravac hiperravnih i širina pojasa koji joj odgovara nisu mnogo važni.

U svetu ranije pomenute sheme dizajna algoritama nadgledanog učenja, zanimljivo je razmisiliti gde se u strukturi ovog problema krije koji od elemenata, konkretno funkcija greške i regularizacija. Primetimo da promenljiva ξ_i (bilo koja) ima vrednost veću od nule ako važi $y_i(w \cdot x_i + w_0) < 1$ i da je u tom slučaju njena vrednost najmanje $1 - y_i(w \cdot x_i + w_0)$. Kako se u minimizaciji insistira na njenim što manjim vrednostima, onda možemo smatrati da važi baš jednakost $\xi_i = 1 - y_i(w \cdot x_i + w_0)$. Dodatno, kako ova promenljiva ne može biti negativna, već je u slučaju da važi $y_i(w \cdot x_i + w_0) > 1$ jednaka 0, zaključujemo da važi

$$\xi_i = \max(0, 1 - y_i(w \cdot x_i + w_0))$$

¹Na primer SMO (eng. *sequential minimal optimization*).



Slika 4.3: Greška u vidu šarke kao aproksimacija indikatorske funkcije. Vrednost greške $L(u, v)$ data je u odnosu na proizvod uv .

odnosno da se prethodni problem može predstaviti kao:

$$\min_{w, w_0} \frac{\|w\|_2}{2} + C \sum_{i=1}^N \max(0, 1 - y_i(w \cdot x_i + w_0))$$

što je ekvivalentno sa

$$\min_{w, w_0} \sum_{i=1}^N \max(0, 1 - y_i(w \cdot x_i + w_0)) + \lambda \|w\|_2$$

što nam je već dobro poznata forma. Primetimo da greška nije srednja nego ukupna, ali kako su te dve greške direktno proporcionalne, to ne menja problem, osim utoliko što će druga vrednost parametra λ biti optimalna. Takođe, za regularizaciju smo ranije umesto norme koristili njen kvadrat. Zaista, i u metodu potpornih vektora se češće koristi kvadrat norme, ali to nije od suštinskog značaja.

Razmotrimo funkciju greške $L(u, v) = \max(0, 1 - uv)$. Radi se o takozvanoj *funkciji greške u vidu šarke* (eng. *hinge loss*) koja, kao što prikazuje slika 4.3, predstavlja konveksnu aproksimaciju greške klasifikacije $L(u, v) = I(u \neq v)$.

Za Vapnik-Červonenkisovu dimenziju metoda potpornih vektora važi

$$h \leq \min(R^2 \|w\|^2, n) + 1$$

gde je R radijus sfere koja obuhvata sve podatke u skupu za obučavanje. Kako metod potpornih vektora minimizuje $\|w\|^2$, posredno minimizuje i Vapnik-

Červonenkisovu dimenziju skupa funkcija iz kojih bira najbolju, čime se objašnjavaju njegove dobre performanse. Još jedno bitno zapažanje je da ukoliko je $R^2 \|w\|^2$ manje od n , što tipično jeste, data gornja granica ne zavisi direktno od dimenzionalnosti prostora (iako neka zavisnost postoji kroz normu vektora w), zbog čega je ovaj metod često korišćen u primenama vezanim za visokodimenzionalne prostore, poput obrade prirodnog jezika i bioinformatike. U praksi se ponaša bolje od većine drugih metoda, zahvaljujući čemu je (uz izmene o kojima govorimo kasnije) bio najpopularniji algoritam mašinskog učenja devesetih, dok primat nisu preuzele duboke neuronske mreže.

Do sada smo razmotrili dva klasifikaciona algoritma koji počivaju na razdvajanju klase hiperravnima – logističku regresiju i metod potpornih vektora. Bitno je razumeti u čemu je razlika. Poučno je razmisliti i o tome kako doći do odgovora na to pitanje. Već je rečeno da opšta shema dizajna algoritama mašinskog učenja koja je skicirana u delu 2.9 može pomoći u razumevanju specifičnosti ponašanja algoritama. Oslonimo se na nju. Po pitanju vrste modela, razlika ne postoji – oba su diskriminativna. Forma modela je ista – u oba slučaja radi se o hiperravni. Funkcija greške je različita. Postoji razlika i u regularizaciji – ona je u metod potpornih vektora ugrađena po konstrukciji, a u logističku regresiju nije. Ipak, to nije suštinska razlika jer se regularizacija uvek može dodati u model logističke regresije. Optimizacioni metod je bitan za brzinu obučavanja, ali ukoliko je optimizacija uspešno završena približnim nałożenjem minimuma, nema posledica po ponašanje modela u predviđanju. Stoga to nije ni važno. Zaključujemo da je jedina razlika u funkciji greške. Stoga će se dalja analiza fokusirati na nju. Kako bi takva analiza bila moguća, potrebno je eliminisati određene razlike u formulacijama. Metod potpornih vektora podrazumeva da su oznake klasa 1 i -1 , dok logistička regresija podrazumeva oznake 0 i 1. Nije teško pokazati (eto korisne vežbe!) da u slučaju oznaka 1 i -1 problem logističke regresije ima formu

$$\min_w \sum_{i=1}^N \log(1 + e^{-y_i w \cdot x_i})$$

odnosno da je funkcija greške

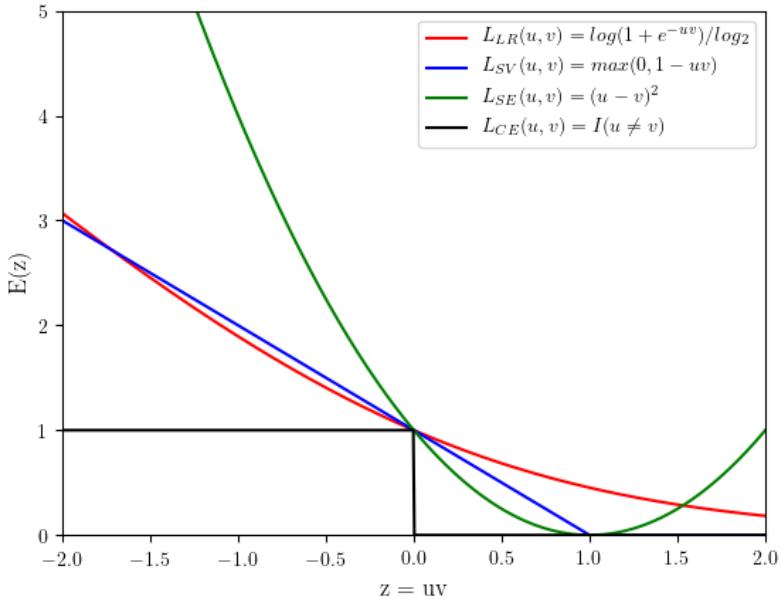
$$L_{LR}(u, v) = \log(1 + e^{-uv})$$

a u slučaju metoda potpornih vektora, to je

$$L_{SV}(u, v) = \max(0, 1 - uv)$$

Kako bi se lakše uporedile i kako je množenje konstantom pri minimizaciji nebitno (zato je recimo nebitno i da li se minimizuje prosek ili suma grešaka), podelimo funkciju greške logističke regresije sa $\log 2$, tako da obe prolaze kroz tačku $(0, 1)$. Poređenja radi, dodajmo u poređenje i metod koji bi radio na osnovu kvadratne funkcije greške

$$L_{SE}(u, v) = (u - v)^2$$



Slika 4.4: Grafici četiri funkcije greške – logističke (crveno), metoda potpornih vektora (plavo), kvadratne (zeleno) i greške klasifikacije (crno).

pošto ju je zaista moguće primeniti i u slučaju binarne klasifikacije kada su oznake klasa numeričke. Uzmimo u obzir i standardnu grešku klasifikacije

$$L_{CE}(u, v) = I(u \neq v)$$

Grafici ovih funkcija prikazani su na slici 4.4. Na horizontalnoj osi je prikazana vrednost poizvoda uv , a na vertikalnoj odgovarajuća vrednost greške. Pozitivne vrednosti uv odgovaraju tačno klasifikovanim instancama, a negativne pogrešno klasifikovanim. Očigledno, logistička funkcija greške kažnjava ne samo pogrešno, već i ispravno klasifikovane instance, čak i ako su daleko od razdvajajuće hiperravnih. To vodi tome da se ne bira hiperravan najšireg pojasa, kao i da sve instance učestvuju u definisanju modela. Upravo to što funkcija greške metoda potpornih vektora dostiže nulu, vodi kako maksimalnom odstojanju, tako i zanemarivanju većine podataka. Obe ove funkcije predstavljaju nekakvu aproksimaciju (indikatorske) funkcije greške klasifikacije. Kvadratna funkcija greške očito predstavlja vrlo lošu aproksimaciju, pošto jako kažnjava i ispravno klasifikovane tačke, čim je proizvod uv veći od 1.

4.2 Metod potpornih vektora za regresiju

Metod potpornih vektora se možda još prirodnije formuliše za regresiju. Funkcija koja se minimizuje je i dalje $\|w\|_2^2$. Uslove tačnog predviđanja bi trebalo prilagoditi kontekstu regresije i model bi mogao da izgleda ovako:

$$\min_w \|w\|_2^2$$

$$|w \cdot x + w_0 - y| = 0$$

Ipak, jedna važna tehnička razlika u odnosu na klasifikaciju je u tome što čak ni u osnovnoj varijanti metoda nema smisla tražiti tačna predviđanja, što je u linearu razdvojivom slučaju kod klasifikacije bio zahtev. Naime, kod binarne klasifikacije postoje dva moguća ishoda – 1 i –1 i sve vrednosti koje linearni model daje zaokružuju se na njih. Nije potrebno da model da baš vrednost 1 ili –1. U slučaju regresije postoji kontinuum ishoda i zahtev za tačnom jednakostu je prejak. Dodatno, često bi mogao biti i štetan. Naime, podaci retko predstavljaju merenja promenljivih veličina sa savršenom tačnošću. Ako podaci sadrže grešku, nema smisla insistirati da se ta greška nauči. Stoga, uvodi se parametar tolerancije ε koji izražava razliku između predviđanja i stvarne vrednosti koja se smatra potpuno prihvatljivom. Osnovni model izgleda ovako:

$$\min_w \|w\|_2^2$$

$$|w \cdot x_i + w_0 - y_i| \leq \varepsilon \quad i = 1, \dots, N$$

odnosno

$$\min_w \|w\|_2^2$$

$$w \cdot x_i + w_0 - y_i < \varepsilon \quad i = 1, \dots, N$$

$$y_i - w \cdot x_i - w_0 < \varepsilon \quad i = 1, \dots, N$$

Ilustracija je data na slici 4.5.

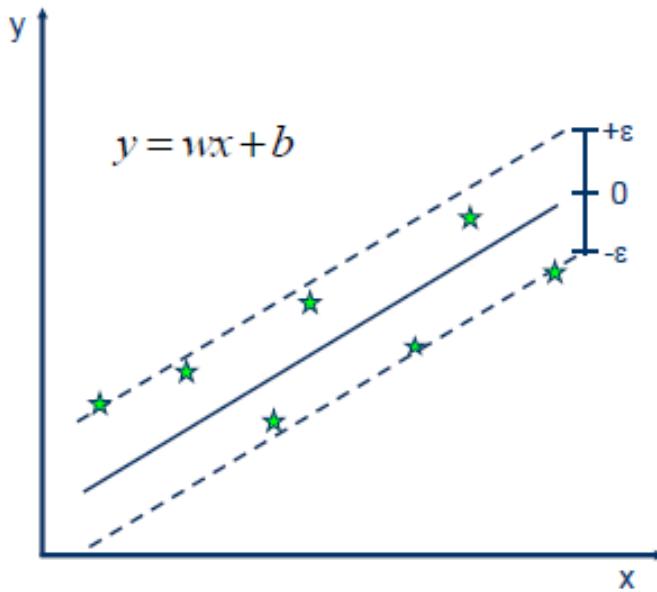
Primetimo da se ova formulacija može jednostavno interpretirati. Ograničenja nalažu da predviđanja ne mogu biti daleko od pravih vrednosti, dok minimizacija norme sprečava izbor modela koji brzo menja vrednosti, odnosno umanjuje prilagodljivost.

Ova formulacija, kao i u slučaju klasifikacionog problema, ima nedostatak da ne dozvoljava greške u predviđanjima (osim za fiksiranu vrednost ε). Slično slučaju mekog pojasa, tolerancija na greške se omogućava uvođenjem novih promenljivih:

$$\min_w \|w\|_2^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*)$$

$$w \cdot x_i + w_0 - y_i < \varepsilon + \xi_i \quad i = 1, \dots, N$$

$$y_i - w \cdot x_i - w_0 < \varepsilon + \xi_i^* \quad i = 1, \dots, N$$



Slika 4.5: Osnovni pristup regresiji pomoću metoda potpornih vektora.

$$\xi_i \geq 0 \quad i = 1, \dots, N$$

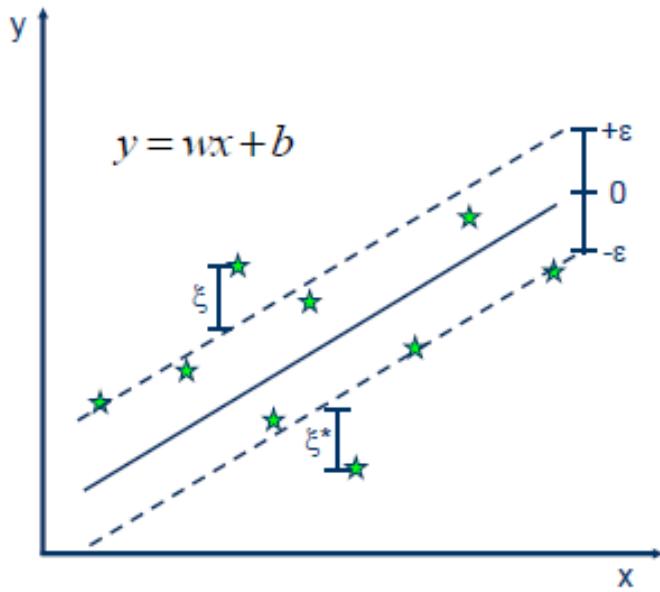
$$\xi_i^* \geq 0 \quad i = 1, \dots, N$$

Ilustracija je data na slici 4.6.

Rešenje ovog problema ima vrlo sličnu formu rešenju klasifikacionog problema.

Jedno pitanje koje zaslužuje razmatranje je – gde je u slučaju ovog metoda široki pojas? U ε okolini modela ne bi mogao biti, pošto se u idealnom slučaju baš tu nalaze svi podaci, a u idealnom slučaju, baš u njemu ne bi trebalo da budu. Razmislimo² kako funkcija greške kažnjava instance u slučaju klasifikacije. Potporni vektori i instance koje su dalje od njih u odgovarajućem poluprostoru ne doprinose grešci. Instance koje zađu u pojas doprinose grešci u skladu proporcionalno udaljenosti od hiperravnini na kojoj leže potporni vektori. U regresionom slučaju, tačke čija se vrednost razlikuje od vrednosti modela za manje od ε , ne doprinose grešci. Čim se razlikuju za više od ε , doprinose proporcionalno toj dodatnoj razlici. To nas navodi na ideju da je u regresionom slučaju široki pojas zapravo prostor tačaka koje se po y osi razlikuju od regresione krive za više od ε .

²I zahvaljimo za ovo razmišljanje kolegi Milošu Jovanoviću.



Slika 4.6: Regresija pomoću metoda potpornih vektora sa mogućnošću grešaka.

4.3 Algoritam k najbližih suseda zasnovan na širokom pojasu

Algoritam najbližih suseda verovatno je najjednostavniji algoritam mašinskog učenja. Može služiti za klasifikaciju sa proizvoljnim brojem klasa, kao i za regresiju. Osnovna pretpostavka ovog algoritma je postojanje rastojanja nad *prostором атрибута* (eng. *feature space*). Najčešće se pretpostavlja vektorska reprezentacija instanci i euklidsko rastojanje, ali moguće su i opštije pretpostavke.

Algoritam k najbližih suseda klasificuje nepoznatu instancu tako što pronađe k instanci iz skupa za obučavanje koje su joj najbliže u smislu neke izabrane metrike i pridružuje joj klasu koja se najčešće javlja među tih k instanci. U slučaju regresije, za predviđanje se uzima prosečna vrednost k najbližih suseda iz skupa za obučavanje. Ovaj algoritam retko predstavlja najbolji izbor za rešavanje nekog problema, ali neretko daje relativno dobre rezultate, a izuzetno lako se implementira i primenjuje. Više reči o detaljima ovog algoritma biće kasnije. Sada ćemo se fokusirati na jednu njegovu vrlo specifičnu klasifikacionu varijantu.

Jedan od problema vezanih za algoritam k najbližih suseda je činjenica da se funkcija rastojanja bira nezavisno od podataka. U slučaju da domenski ekspert zna nešto više o svojstvima podataka, moguće je napraviti meru rastojanja koja će biti prilagođena datom problemu, ali možda još bolji način bi bio da se ta

mera rastojanja uči. Ukoliko je M pozitivno semidefinitna matrica, funkcija

$$d_M(x, x') = (x - x')^T M (x - x')$$

je funkcija rastojanja. Ako važi $M = I$, mesto tačaka jednakog rastojanja od neke fiksirane tačke C je sfera sa centrom u tački C . Ukoliko je matrica M dijagonalna, mesto tačaka jednakog rastojanja od C je elipsoid sa centrom u tački C , čije su ose paralelne koordinatnim osama. U opštem slučaju, radi se o proizvolnjom elipsoidu sa centrom u tački C .³ Razmotrimo još jedan način razumevanja matrice M . Kako je matrica M pozitivno semidefinitna, može se predstaviti kao $M = Q^T Q$. Tada se metrika predstavlja kao

$$d_M(x, x') = (x - x')^T Q^T Q (x - x') = (Qx - Qx')^T (Qx - Qx')$$

Zamenom koordinata $t = Qx$ dobija se

$$d_M(x, x') = (t - t')^T (t - t') = d_I(t, t')$$

Drugim rečima, matrica Q transformiše prostor atributa tako da se metrika d_M u novim koordinatama može računati kao standardna euklidска metrika.

Umesto da se matrica M zada unapred, poželjno je učiti je iz podataka. Ipak, postavlja se pitanje kriterijuma u odnosu na koji se uči. Dobra matrica rastojanja bi bila ona za koju su tačke iz iste klase blizu, dok su sve tačke iz različitih klasa međusobno daleko. Ilustracija je data na slici 4.7. Leva slika prikazuje okolinu tačke u odnosu na euklidsku metriku, dok desna prikazuje okolinu u odnosu na metriku naučenu tako da instance iz iste klase budu u toj okolini, a da instance drugih klasa budu van, razdvojene od te okoline širokim pojasmom.

Neka je

$$\mathcal{Z} = \{(x, x', x'') | (x, y), (x', y), (x'', y'') \in \mathcal{D} \wedge y \neq y''\}$$

skup svih trojki vektora atributa takvih da prva dva pripadaju istoj klasi, kojoj treći ne pripada. Elemente skupa \mathcal{Z} , označavaćemo z_i za $i = 1, \dots, |\mathcal{Z}|$. Jedna formulacija metoda za određivanje matrice M bi mogla biti:

$$\min_M \sum_{(x,y),(x',y) \in \mathcal{D}} d_M(x, x')$$

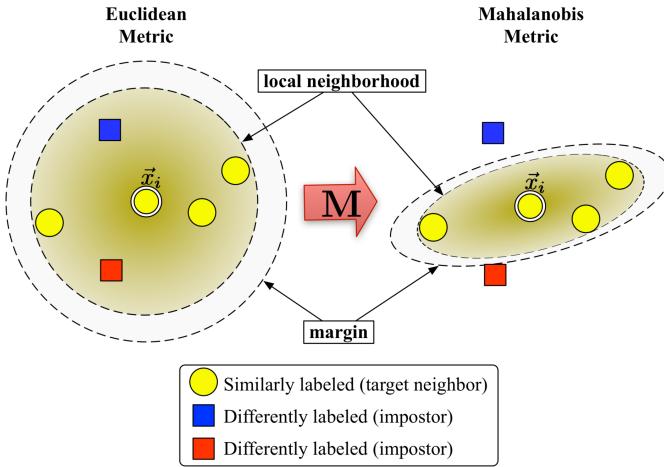
$$d_M(x, x') < d_M(x, x'') \text{ za sve } (x, x', x'') \in \mathcal{Z}$$

$$M \succeq 0$$

gde poslednji uslov označava pozitivnu semidefinitnost matrice M . Bolji pristup, koji insistira na postojanju širokog pojasa između elemenata klase je sledeći

$$\min_M \sum_{(x,y),(x',y) \in \mathcal{D}} d_M(x, x')$$

³Strogo gledano, ako je matrica M pozitivno semidefinitna, ali nije pozitivno definitna, pomenuti elipsoid može biti i degenerisan.



Slika 4.7: Ilustracija algoritma k najbližih suseda zasnovanog na širokom pojusu.

$$d_M(x, x') + 1 \leq d_M(x, x'') \text{ za sve } (x, x', x'') \in \mathcal{Z}$$

$$M \succeq 0$$

Konstanta 1 izgleda kao proizvoljan izbor i to i jeste. Kao i u slučaju metoda potpornih vektora za klasifikaciju, to bi mogla biti bilo koja stroga pozitivna vrednost, ali deljenjem svih izraza tom vrednošću dobija se data formulacija.

Kao i u slučaju metoda potpornih vektora, krutost ograničenja se prevazi-lazi mekim pojasom, odnosno uvođenjem novih promenljivih koje čine model tolerantnijim na greške. Finalna formulacija glasi:

$$\min_M \sum_{(x,y),(x',y) \in \mathcal{D}} d_M(x, x') + C \sum_{i=1}^{|\mathcal{Z}|} \xi_i$$

$$d_M(x_i, x'_i) + 1 \leq d_M(x_i, x''_i) + \xi_i \text{ za sve } (x_i, x'_i, x''_i) \in \mathcal{Z}$$

$$\xi_i \geq 0 \quad i = 1, \dots, |\mathcal{Z}|$$

$$M \succeq 0$$

Specifičnost ovog problema je upravo u zahtevu pozitivne semidefinitnosti. Jedina način postizanja ovog svojstva je da se matrica M predstavi kao $W^T W$ za neku matricu parametara W .

Glava 5

Modeli zasnovani na instancama

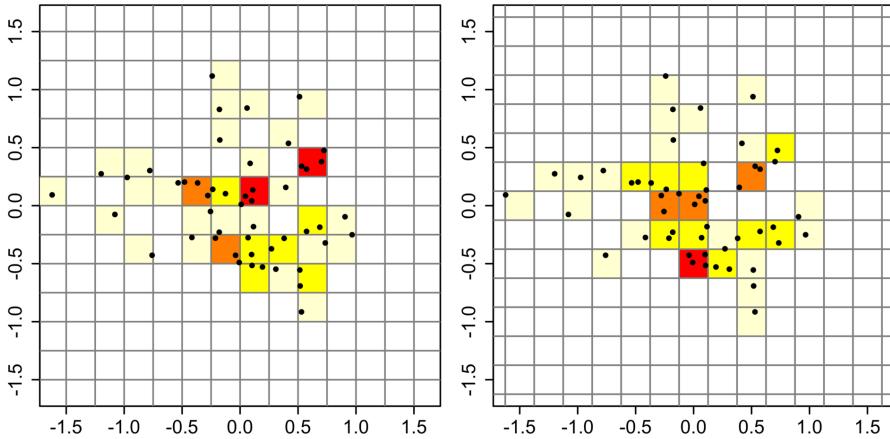
Modeli zasnovani na instancama predstavljaju *neparametarski* pristup mašinskom učenju. Pritom, izraz neparametarski zahteva dodatno objašnjenje. *Parametarski statistički modeli* pretpostavljaju postojanje konačnog skupa parametara čije vrednosti definišu model. *Neparametarski modeli* su modeli koji se ne opisuju konačnim skupom parametara i čiji broj parametara može zavisiti od veličine skupa za obučavanje i stoga je neograničen. Ovakvi metodi često moraju da čuvaju skup podataka za obučavanje kako bi davali predviđanja na noviminstancama, jer su modeli često i izraženi u terminima tih podataka. Zbog toga i predviđanje na osnovu ovakvih metoda može biti računskih zahtevno. To je očito mana ove vrste metoda, ali njihova prednost je da ne pretpostavljaju formu modela tako striktno kao parametarski modeli (npr. pretpostavka normalne raspodele), već ta forma može slobodnije da zavisi od podataka.

5.1 Osnove neparametarske ocene gustine raspodele

Ocena gusitne raspodele predstavlja najteži problem mašinskog učenja. Osnovna intuicija iza metoda ocene gustine raspodele je da regioni prostora atributa u kojima se nalazi više tačaka podataka imaju više vrednosti gustine raspodele, dok oni u kojima se nalazi manji broj tačaka imaju manju vrednost gustine. Ipak, svaka tačka u skupu podataka svedoči samo o svom pojavljivanju i bilo bi moguće da se samo tim tačkama pridruži nenula vrednost gustine raspodele, a da sve ostale tačke dobiju vrednost nula. Ovakva ocena predstavlja ekstreman slučaj preprilagođavanja. Umesto toga, moguće je smatrati da svaka tačka podataka povećava ne samo vrednost gustine raspodele u toj tački, već i u tačkama u nekoj njenoj okolini. Naravno, postavlja se pitanje definisanja takve okoline, a i definisanja načina na koji se takav model konstruiše.

Jedan pristup oceni gustine raspodele koji odgovara prethodnom opisu je histogram. Ipak, taj pristup nije dovoljno dobar iz više razloga:

1. tako ocenjena gustina raspodele zavisi od pozicija korpica, koje bi se pri konstrukciji histograma mogle proizvoljno translirati (ovo je ilustrovano



Slika 5.1: Dva histograma čije su korpice translirane. Boje izražavaju vrednost funkcije.

slikom 5.1),

2. tačke prekida histograma nisu posledica gustine podataka, već izbora lokacija korpica,
3. oblik histograma drastično zavisi od širine, odnosno broja korpica (ovo je ilustrovano slikom 5.2),
4. zbog prokletstva dimenzionalnosti, većina korpica će biti prazna u slučaju prostora veće dimenzionalnosti.

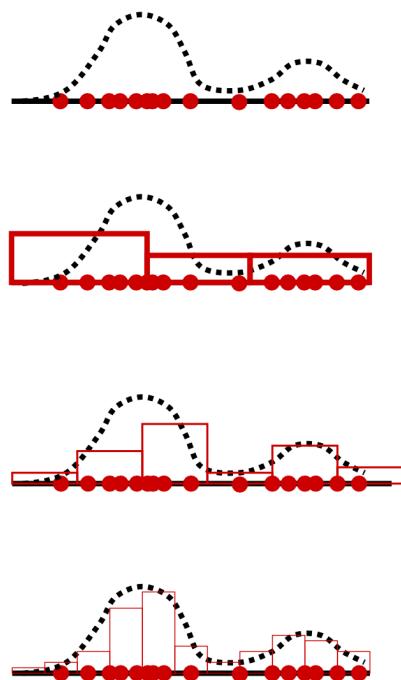
Ovi razlozi čine upotrebu histograma vrlo pipavom i u svrhe preliminarne analize podataka, a za ocenu gustine raspodele ga ne treba koristiti. Ipak, ne znači ni da bolje alternative mogu dati odgovor na sve naznačene probleme.

Držeći se i dalje principa iz prvog paragrafa, razmislimo o alternativama. Jedan razlog za pomenute mane histograma je što se pri njegovoј konstrukciji polazi od particonisanja prostora koje se nakon toga smatra fiksiranim. Alternativa bi bila krenuti od pozicija tačaka na osnovu kojih se gustina određuje. Ilustracija jednog (ali ne jedinog) načina na koji je to moguće uraditi je prikazana na slici 5.3.

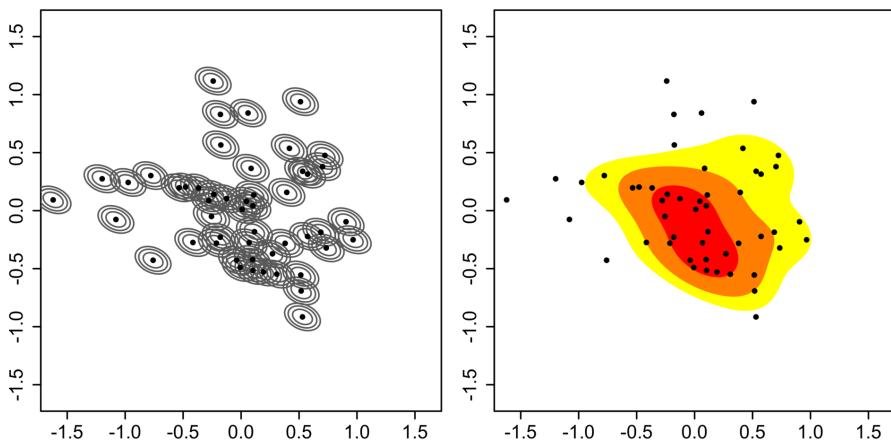
Razmotrimo oblast \mathcal{R} koja sadži tačku x u čijoј okolini treba oceniti gustinu raspodele $p(x)$. Verovatnoća pridružena ovoј oblasti je

$$P = \int_{\mathcal{R}} p(x)dx$$

Ako N opažanja dolazi iz raspodele $p(x)$, svako ima verovatnoću P pripadanja oblasti \mathcal{R} . Broj tačaka k koje pripadaju ovoј oblasti je raspodeljen u skladu sa



Slika 5.2: Histogrami sa različitim brojem korpica konstruisani nad istim podacima.



Slika 5.3: Alternativa histogramu – ocena gustine zasnovana na doprinosima tačaka svojim okolinama. Doprinos tačke opada sa udaljavanjem od nje.

binomnom raspodelom:

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

Važi $\mathbb{E}(k) = NP$ i $\text{var}[k] = NP(1-P)$. Na osnovu toga, očekivanje uđela tačaka koje upadaju u region \mathcal{R} je $\mathbb{E}[k/N] = P$, a varijansa je $\text{var}[k/N] = P(1-P)/N$. Sa povećanjem broja N , varijansa veličine k/N se smanjuje, pa je sama veličina sve bliža proseku, odnosno važi

$$\frac{k}{N} \approx P$$

Slično, ukoliko je zapremina oblasti \mathcal{R} dovoljno mala, tako da se funkcija $p(x)$ ne menja mnogo u njoj, važi

$$P \approx p(x)V$$

gde je V zapremina oblasti \mathcal{R} . Prethodne relacije daju ocenu ¹

$$p(x) \approx \frac{k}{NV}$$

Među veličinama k i V postoji očigledna zavisnost. U većoj zapremini V , očekuje se više tačaka k . Slično, kako bi bio dosegnut veliki broj tačaka k , potrebna je velika zapremina V . Otud, pristupi oceni gustine raspodele mogu biti formulisani oslanjajući se na neku od ove dve veličine, prema čemu razlikujemo *pristupe zasnovane na kernelima* i *pristupe zasnovane na najbližim susedima*. Metodi zasnovani na kernelima formulišu se u odnosu na zapreminu V , a metodi zasnovani na najbližim susedima u terminima broja k . Preciznije, u slučaju metoda zasnovanih na kernelima, broj tačaka na osnovu kojih se vrši ocena gustine raspodele u tački x zavisi od toga koliko ih ima u okolini tačke x unapred izabrane zapremine V . U slučaju metoda zasnovanih na najbližim susedima, zapremina okoline tačke x u kojoj se nalaze tačke na osnovu kojih se vrši ocena gustine u tački x zavisi od unapred izabranog broja tačaka k .

¹Osvrnamo se na uslove pod kojima ova ocena konvergira stvarnoj gustini raspodele $p(x)$. Naime, ako se fokusiramo na jednu tačku uzorka, njena dovoljno mala okolina će sadržati samo tu tačku, a okolina se može proizvoljno smanjivati. Kako njena zapremina teži nuli, naša ocena teži beskonačnosti. Stoga, u slučaju konačnog uzorka okolina očito ne sme biti proizvoljno mala. Razmotrimo situaciju u kojoj broj tačaka N teži beskonačnosti. Neka je gustina $p(x)$ neprekidna u tački x i, jednostavnosti radi, neka je $p(x) \neq 0$. Neka je \mathcal{R}_N lopta sa centrom u tački x , zapremine V_N , koja sadrži k_N tačaka i koja se koristi za ocenu gustine pri uzorku veličine N . Tada ocena $p_N(x) = \frac{k_N}{NV_N}$ teži vrednosti $p(x)$ ako i samo ako važe uslovi

- $\lim_{N \rightarrow \infty} k_N = \infty$
- $\lim_{N \rightarrow \infty} V_N = 0$
- $\lim_{N \rightarrow \infty} k_N/V_N = 0$

Prvi uslov omogućava da k_N/N teži P_N (verovatnoću regiona \mathcal{R}_N). Drugi uslov omogućava da se okolina na osnovu koje se ocenjuje $p(x)$ smanjuje. U suprotnom bi ovakva ocena gustine uprosečavala vrednost gustine u okolini tačke x . I konačno, ako V_n teži nuli, onda i P_N/V_N mora težiti nuli, inače bi ocena divergirala.

5.2 Metodi zasnovani na kernelima

Veliki broj metoda zasnovanih na instancama počiva na upotrebi *kernela*. Neformalno *kernel* predstavlja funkciju sličnosti. Što je vrednost kernela za neke dve instance veća, to se one mogu smatrati sličnijim. Što je manja, to se te dve instance mogu smatrati različitim. U različitim kontekstima, prave se vrlo različite pretpostavke vezane za kernele, pa se i sami koncepti koji se pod tim izrazom podrazumevaju vrlo razlikuju. Ovde neće biti formalna definicija, već će pretpostavke koje se prave biti navedene u odgovarajućim odeljcima.

Kerneli najčešće imaju određene hiperparametre, kojima se finije podešava njihovo ponašanje. Ovi hiperparametri su u vezi sa zapreminom okoline neke tačke u čijoj se okolini ocenjuje gustina raspodele.

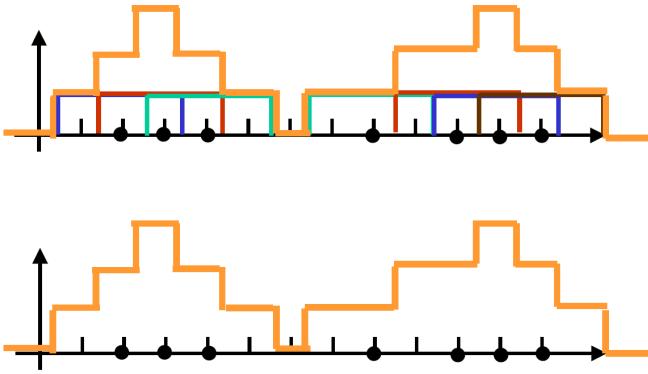
Najavimo jedno svojstvo metoda zasnovanih na kernelima koje ih razlikuju od ranije viđenih metoda. Ranije izlagani metodi izražavaju zavisnost ciljne promenljive od atributa pomoću koeficijenata koji kontrolisu dejstvo promene tog atributa na promenu ciljne promenljive. Na primer, veće vrednosti atributa vode većim vrednostima ciljne promenljive, a male manjim, ili suprotno. Nasuprot tome, pristup zasnovan na kernelima, kao merama sličnosti, se zasniva na bliskosti vrednosti atributa. Ukoliko su vrednosti atributa nove instance bliske vrednostima atributa neke instance iz skupa za obučavanje, i vrednost ciljne promenljive nove instance treba da bude bliska ciljnoj vrednosti te instance, bez obzira da li su vrednosti atributa nove instance same za sebe visoke ili niske. Ovaj pristup vodi *lokalnosti* modela. Ranije diskutovani modeli pokušavaju da okarakterišu globalne zavisnosti, dok u slučaju modela zasnovanim na kernelima, u različitim delovima prostora atributa, te zavisnosti mogu biti vrlo različite, a vrednost ciljne promenljive nove instance će biti određena na osnovu zakonitosti koja važi u njenoj okolini.

5.2.1 Ocena gustine raspodele zasnovana na kernelima

U kontekstu ocene gustine raspodele, obično se polazi od pojma *glačajućeg kernela* (eng. *smoothing kernel*), funkcije za koju važi:

- $K(x) \geq 0$
- $\int K(x)dx = 1$
- K je simetrična funkcija
- K ne raste sa udaljavanjem od nule

Skalirani *kernel* se definiše kao $K_\sigma = \frac{1}{\sigma^n} K\left(\frac{x}{\sigma}\right)$ za $\sigma > 0$, gde je n dimenzija vektora x . Skalirani kernel zadovoljava sve uslove koje zadovoljava i polazni kernel. Hiperparametar σ naziva se *širinom* (eng. bandwidth) kernela.



Slika 5.4: Ocena gustine jednodimenzionalne raspodele pomoću Parzenovih prozora širine 3.

Neka je oblast \mathcal{R} hiperkocka sa centrom u tački x . Definišimo karakterističnu funkciju jedinične hiperkocke sa centrom u koordinatnom početku

$$K(x) = \begin{cases} 1, & |x_i| \leq 1/2, \\ 0, & \text{inače} \end{cases} \quad i = 1, \dots, n$$

Funkcija K je primer kernela i naziva se *Parzenovim prozorom*. Broj tačaka k koji se nalazi u hiperkocki strane σ sa centrom u tački x može se izraziti kao

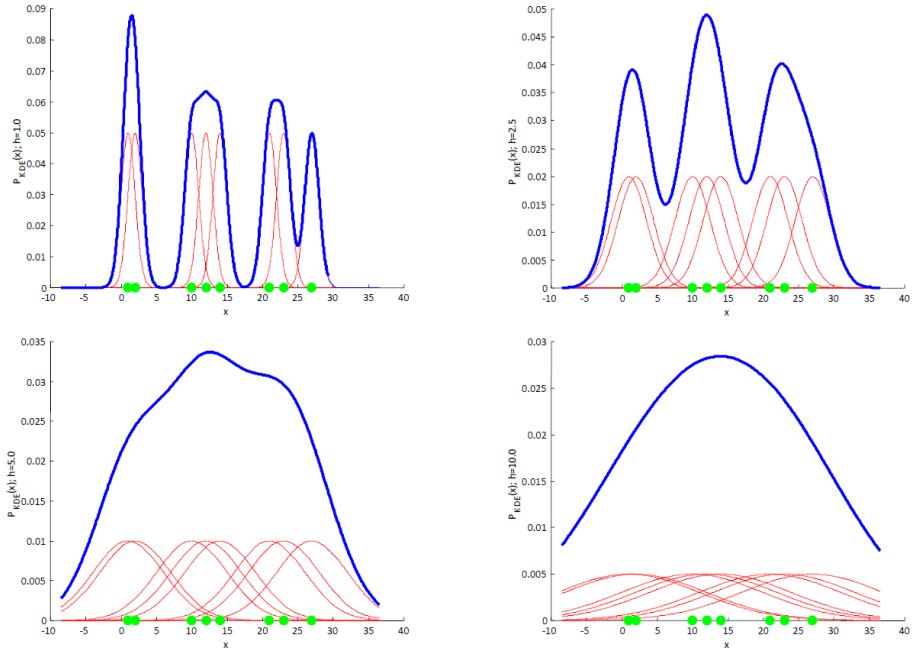
$$k = \sum_{i=1}^N K\left(\frac{x - x_i}{\sigma}\right)$$

Na osnovu ovog zapažanja i prethodno izvedene aproksimacije $p(x) \approx \frac{k}{NV}$ važi

$$p(x) = \frac{1}{N} \frac{1}{\sigma^n} \sum_{i=1}^N K\left(\frac{x - x_i}{\sigma}\right) = \frac{1}{N} \sum_{i=1}^N \frac{1}{\sigma^n} K\left(\frac{x - x_i}{\sigma}\right) = \frac{1}{N} \sum_{i=1}^N K_\sigma(x - x_i)$$

Prikaz ocene jednodimenzionalne gustine raspodele pomoću Parzenovih prozora dat je na slici 5.4

Ovim se dobija prekidna ocena gustine vrlo slična histogramu. Ipak, postoje razlike. Razmotrimo je u jednodimenzionalnom slučaju. Kod histograma su podeoci na x osi, odnosno tačke promene vrednosti histograma, fiksirani, dok u ovom slučaju zavise od podataka, jer se mogu desiti samo u tačkama koje su na rastojanju $\sigma/2$ od neke tačke podataka. Ovaj pristup ima nešto poželjnija svojstva od histograma. Naime, za fiksirane podeoke, odnosno korpice, tačka x koja pripada jednoj korpici, može biti bliže većini tačaka druge korpice, nego većini tačaka iz svoje. Stoga, tačke iz njene korpice od kojih je daleko, određuju vrednost gustine raspodele u tački x , a tačke iz druge korpice kojima je bliža,



Slika 5.5: Ocena gustine jednodimenzionalne raspodele pomoću Gausovih kernela različitih širina.

nemaju nikakav uticaj. U slučaju date ocene pomoću kernela, ovaj problem ne postoji. Bliže tačke uvek daju više doprinosu od udaljenijih. Moguće je prevazići i problem prekidnosti korišćenjem neprekidnih kernela, poput Gausovog:

$$K_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\|x\|^2}{2\sigma^2}\right)$$

Na slici 5.5, prikazana je ocena gustine pomoću Gausovog kernela za različite širine.

Kao i u slučaju regularizacije, izbor hiperparametara σ ćemo diskutovati kasnije, ali generalno, mala širina kernela tipično vodi preprilagođavanju, dok velika vodi potprilagođavanju i uprosečavanju informacije. To se vidi i na slici 5.5.

Mana ocene gustine raspodele zasnovane na kernelima je ta što je širina kernela nezavisna od tačke x , pa jedan skup parametara u oblastima visoke gustine može rezultovati preširokim kernelom, čijim se dejstvom usrednjava struktura u podacima, dok bi u oblastima u kojima je gustina značajno niža taj isti krenel mogao biti premale širine i voditi preprilagođavanju.

5.2.2 Metod Nadaraja-Votson zasnovanu na kernelima

Metod Nadaraja-Votson se najčešće diskutuje u kontekstu regresije, pa ćemo i mi krenuti tim putem. Kao što je ranije rečeno, regresiona funkcija, čija aproksimacija se traži u problemu regresije, definisana je kao uslovno očekivanje ciljne promenljive pri datim vrednostima atributa:

$$r(x) = \mathbb{E}[y|x] = \int yp(y|x)dy = \int y \frac{p(x,y)}{p(x)} dy$$

Jedan način da se izvede ocena ove funkcije je da se u datom integralu upotrebe ocene gustina raspodela koje figurišu u njemu. Ukoliko su definisani neki glačajući kerneli na prostoru atributa \mathcal{X} i na skupu vrednosti ciljne promenljive \mathcal{Y} , njihov proizvod predstavlja kernel na prostoru $\mathcal{X} \times \mathcal{Y}$, pa je poslednji integral moguće aproksimirati sledećim modelom:

$$\begin{aligned} f_\sigma(x) &= \int y \frac{\sum_{i=1}^N K_\sigma(x - x_i)K_\sigma(y - y_i)}{\sum_{j=1}^N K_\sigma(x - x_j)} dy = \\ &= \frac{\sum_{i=1}^N K_\sigma(x - x_i) \int y K_\sigma(y - y_i) dy}{\sum_{j=1}^N K_\sigma(x - x_j)} = \\ &= \frac{\sum_{i=1}^N K_\sigma(x - x_i) y_i}{\sum_{j=1}^N K_\sigma(x - x_j)} \end{aligned}$$

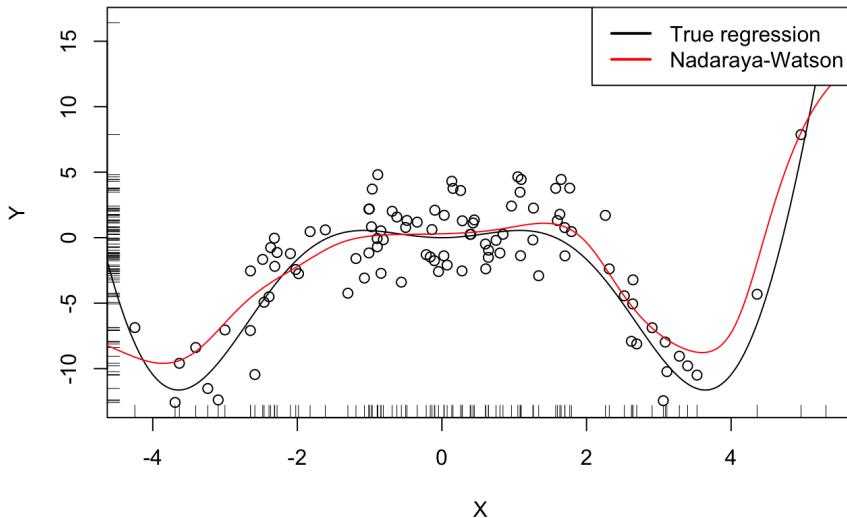
gde poslednja jednakost važi na osnovu simetričnosti kernela. Ovo je model regresije Nadaraja-Votson. Razmotrimo njegov smisao. Težina

$$\frac{K_\sigma(x - x_i)}{\sum_{j=1}^N K_\sigma(x - x_j)}$$

koja se pridružuje vrednosti ciljne promenljive y_i je proporcionalna sličnosti tačke x sa tačkom x_i . Kako se ove težine sabiraju na 1, zaključujemo da je dobijeno rešenje težinski prosek vrednosti y_i , pri čemu se pridaje veća težina sličnijiminstancama.

Na slici 5.6, prikazan je primer regresije pomoću ovog metoda. Povećavanje širine kernela vodilo bi manjem, a smanjivanje većem prilagođavanju podacima. Aproksimacije koje smo prikazali u slučaju linearne regresije polinomom izgledale su slično. Ipak, linearni model pretpostavlja formu modela, a korišćenje polinoma, formu baznih funkcija. Pritom, to je samo jedan mogući izbor i možda bi drugačiji izbori vodili boljim rezultatima. Odnosno, parametarski pristup očekuje da smo u stanju da donešemo dobru pretpostavku vezanu za formu modela. Neparametarski pristup to ne zahteva. Ali, s druge strane, traži izbor kernela i izražava rešenje u terminima celog skupa za obučavanje, koji je stoga potrebno čuvati i koji se ceo koristi prilikom predviđanja.

Sa date slike moguće je uočiti da je kriva predviđena od strane modela glađa od prave regresione krive i da stoga potcenjuje varijansu podataka. Ovo



Slika 5.6: Primer regresije pomoću metoda Nadaraja-Votson. Tačke su generisane na osnovu crne krive dodavanjem slučajnog šuma. Crvena kriva predstavlja ocenu regresije.

je očekivano svojstvo ovog metoda jer se predviđanja dobijaju uprosečavanjem opažanja. Sličan efekat se javlja i kod drugih metoda zasnovanih na glaćajućim kernelima.

Primetimo da se izloženi metod može prilagoditi i kontekstu klasifikacije. Ukoliko se ciljna promenljiva kodira binarnim vektorom čija dimenzija odgovara broju klasa i koji ima jedinicu tačno na mestu koje odgovara pravoj klasi, formula koja definiše metod Nadaraja-Votson se bez izmena može primeniti i u kontekstu klasifikacije. Pritom, predviđanje koje se dobija modelom ne mora biti u formi binarnog vektora, ali se lako može pokazati da su koordinate dobijenog vektora nenegativne i da se sabiraju na 1, odnosno da predstavljaju raspodelu po klasama. Stoga se za predviđenu klasu može uzeti najverovatnija u skladu sa tom raspodelom.

5.2.3 Kernelizovani metod potpornih vektora

U ovom odeljku se fokusiramo na metod potpornih vektora za klasifikaciju. Njegov ključni problem je činjenica da hiperravan ne mora predstavljati adekvatnu granicu među klasama. Oblici granica u praksi mogu biti proizvoljni. Pritom, ovaj problem nije rešen formulacijom sa mekim pojasom, pošto me-

ki pojas pretpostavlja da je hiperravan ugrubo dobar oblik granice, samo što podaci nekada završe sa pogrešne strane. Moglo bi biti potrebno da granica bude kružna i tada očito prava ne predstavlja ni približno dobru aproksimaciju. Takođe, i tad se može očekivati da podaci nekada završe sa pogrešne strane kružnice. Stoga su pitanja oblika razdvajajućih hiperpovrši i potrebe za mekim pojasmom nezavisna.

Podsetimo se problema optimizacije vezanog za metod potpornih vektora za linearno razdvojive klase:

$$\min_{w, w_0} \frac{\|w\|_2}{2}$$

$$y_i(w \cdot x_i + w_0) \geq 1 \quad i = 1, \dots, N$$

Jednostavnosti radi, u nastavku ćemo se držati ovog problema, ali analogno izvođenje važi i za formulaciju sa mekim pojasmom. Jedan način rešavanja ovog problema je prelazak na formulaciju bez ograničenja uz primenu gradijentnih metoda optimizacije. Ipak, zbog mogućnosti uopštavanje metoda na nelinearne rezdvajajuće hiperpovrši, poželjno je promeniti formulaciju optimizacionog problema i posmatrati takozvani *dualni problem*. Lagranžova funkcija polaznog problema je:

$$\mathcal{L}(w, \alpha) = \frac{\|w\|_2^2}{2} - \sum_{i=1}^N \alpha_i [y_i(w \cdot x_i + w_0) - 1]$$

pri čemu se podrazumeva da važi $\alpha_i \geq 0$ za svako i . U terminima Lagranžove funkcije, polazni problem može se izraziti kao

$$\min_{w, w_0} \max_{\alpha_i \geq 0} \frac{\|w\|_2^2}{2} - \sum_{i=1}^N \alpha_i [y_i(w \cdot x_i + w_0) - 1]$$

Dualni problem dobija se zamenom minimuma i maksimuma u prethodnom, *primalnom problemu*. Dualni i primalni problem nisu uvek ekvivalentni, već moraju biti ispunjeni određeni uslovi u koje ovde ne ulazimo. Ti uslovi su ispunjeni u slučaju posmatranog problema. Dualni problem glasi:

$$\max_{\alpha_i \geq 0} \min_{w, w_0} \frac{\|w\|_2^2}{2} - \sum_{i=1}^N \alpha_i [y_i(w \cdot x_i + w_0) - 1]$$

Rešenje unutrašnjeg minimizacionog problema je funkcija promenljivih α_i i može se dobiti analitički, postavljanjem parcijalnih izvoda po w i w_0 na 0:

$$\frac{\partial L}{\partial w_j} = w_j - \sum_{i=1}^N \alpha_i y_i x_i = 0 \quad \frac{\partial L}{\partial w_0} = \sum_{i=1}^N \alpha_i y_i = 0$$

Iz prve jednakosti se dobija da je rešenje problema oblika:

$$w = \sum_{i=1}^N \alpha_i y_i x_i$$

Uvrštavanjem i korišćenjem druge jednakosti u dualnom problemu dobija se problem:

$$\max_{\alpha_1, \dots, \alpha_N} -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) + \sum_{i=1}^N \alpha_i$$

$$\alpha_i \geq 0$$

$$\sum_{i=1}^N \alpha_i y_i = 0$$

Odavde se vidi da je rešenje moguće dobiti minimizacijom po Lagranžovim množiocima α_i . Nakon što su određeni optimalni parametri modela formulom:

$$w = \sum_{i=1}^N \alpha_i y_i x_i$$

može se pokazati da se vrednost w_0 može izračunati izrazom:

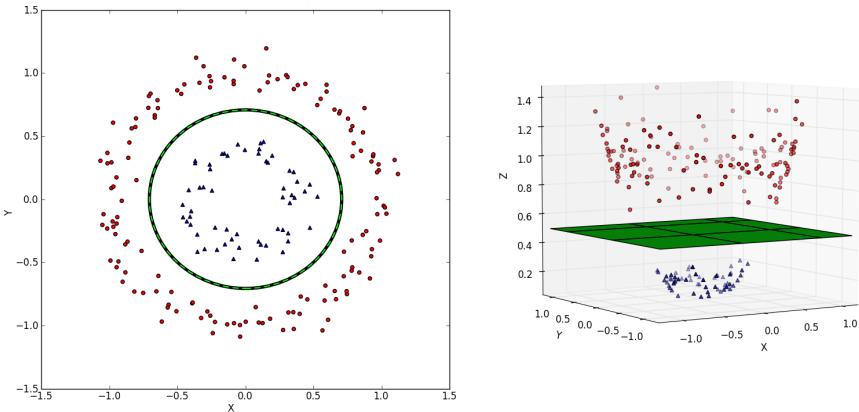
$$w_0 = -\frac{\min_{(x, 1) \in \mathcal{D}} w \cdot x + \max_{(x, -1) \in \mathcal{D}} w \cdot x}{2}$$

Intuicija ovakvog izbora koeficijenta w_0 je da je optimalna hiperravan tačno na sredini između potpornih vektora dve klase. Model je onda dat funkcijom

$$f_{w, w_0}(x) = \sum_{i=1}^N \alpha_i y_i (x_i \cdot x) + w_0$$

Ključno zapažanje koje će voditi uopštavanju modela je da su i optimizacioni problem i njegovo rešenje, to jest model, izraženi u terminima skalarnih proizvoda instanci jednih sa drugima i da mimo tog proizvoda ne zavise od samih instanci. Ali razmotrimo prvo kako bi uopšte mogli izraziti nelinearne razdvajajuće površi između klasa. I u ranijim primerima, recimo u klasifikacionom primeru za regularizaciju videli smo da je to moguće. Čak i linearni modeli mogu izraziti takve razdvajajuće površi ukoliko im se dodaju proizvodi i stepeni polaznih atributa, što možemo smatrati pretprocesiranjem podataka. Razmotrimo sledeći primer.

Primer 3 Neka su podaci raspoređeni tako da elementi jedne klase unutar lopte određenog poluprečnika u prostoru atributa, dok su elementi druge klase van te lopte, što je u dvodimenzionalnom slučaju prikazano na slici 5.7. Definišimo pretprocesiranje podataka sledećim preslikavanjem $\Phi(x) = (x_1, x_2, x_1^2 + x_2^2)$. Ovo preslikavanje očigledno slika podatke u prostor veće dimenzije u kojem se mogu razdvojiti pomoću hiperravnih, kao što je prikazano na slici. U polaznom prostoru, inverzna slika razdvajajuće hiperravnih izgleda kao kružnica.



Slika 5.7: Preslikavanje kojim se problem koji je linearno nerazdvojiv u dve dimenzije preslikava u problem koji je linearno razdvojiv u tri dimenzije.

Ukoliko se podaci preprocesiraju nekim preslikavanjem Φ , dualni optimizacioni problem trivijalno postaje:

$$\begin{aligned} \max_{\alpha_1, \dots, \alpha_N} & -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j (\Phi(x_i) \cdot \Phi(x_j)) + \sum_{i=1}^N \alpha_i \\ & \alpha_i \geq 0 \\ & \sum_{i=1}^N \alpha_i y_i = 0 \end{aligned}$$

a model dobija formu

$$f_{w, w_0}(x) = \sum_{i=1}^N \alpha_i y_i (\Phi(x_i) \cdot \Phi(x)) + w_0$$

Sledeći korak u uopštavanju metoda potpornih vektora je uvođenje kernela. U kontekstu metoda potpornih vektora, bitan je pojam pozitivno semidefinitnog kernela (a ne glačajućeg kernela). Neka je \mathcal{X} neprazan skup i neka je data simetrična funkcija $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$. Ako je za svako $n \in \mathbb{N}$ i svako $x_1, \dots, x_n \in \mathcal{X}$ matrica dimenzija $n \times n$ sa elementima $k(x_i, x_j)$ pozitivno semidefinitna, funkcija k je *pozitivno semidefinitan kernel* ili *Mercerov kernel*. U nastavku se pod izrazom kernel podrazumeva Mercerov kernel. Važan podatak je da za svaki kernel k postoji preslikavanje Φ_k iz \mathcal{X} u neki vektorski prostor \mathcal{H}_k sa skalarnim proizvodom, tako da važi

$$k(x, x') = \Phi_k(x) \cdot \Phi_k(x')$$

Drugim rečima, svaki kernel se može posmatrati kao skalarni proizvod u nekom vektorskom prostoru. Otud se kernel može smatrati merom sličnosti nad elementima skupa \mathcal{X} . Pritom, za taj skup nije pretpostavljena nikakva struktura. Odnosno, za proizvoljne objekte nad kojima možemo definisati kernele, možemo imati značajan deo tehničkih pogodnosti koje pruža skalarni proizvod. Na primer, moći ćemo primenititi metod potpornih vektora. Kerneli imaju određena poželjna svojstva, kao da je linearna kombinacija kernela takođe kernel, da je proizvod kernela takođe kernel i slično. Ova svojstva se mogu upotrebiti za konstrukciju novih kernela od već poznatih.

Primetimo da korišćenje $k(x, x') = \Phi_k(x) \cdot \Phi_k(x')$ jednakosti omogućava uvođenje kernela u metod potpornih vektora. Time dualni problem postaje

$$\begin{aligned} \max_{\alpha_1, \dots, \alpha_N} & -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j k(x_i, x_j) + \sum_{i=1}^N \alpha_i \\ & \alpha_i \geq 0 \\ & \sum_{i=1}^N \alpha_i y_i = 0 \end{aligned}$$

a model dobija formu

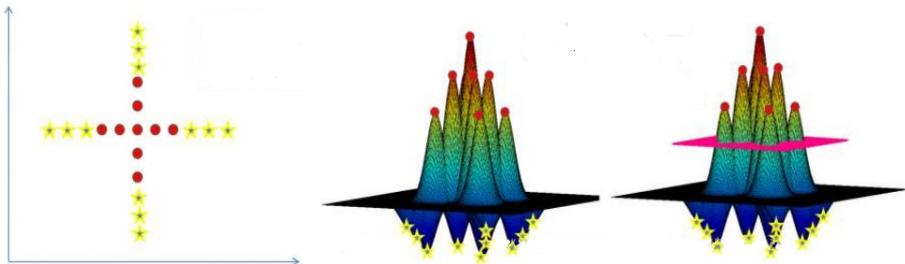
$$f_{w, w_0}(x) = \sum_{i=1}^N \alpha_i y_i k(x_i, x) + w_0$$

Zanimljivo zapažanje je da smo prelaskom na kernele dobili više nego što bismo mogli dobiti pretprocesiranjem. Preslikavanje Φ_k koje odgovara kernelu k može slikati podatke u beskonačno dimenzionalni postor. Pretprocesiranje koje daje beskonačne izlaze se ne može implementirati na računaru, ali nam korišćenje kernela ipak omogućava da izračunamo skalarne proizvode tako pretprocesiranih podataka!

Ipak, i dalje nije jasno da smo ovim na dobitku. Naime, kako ne znamo kako izgledaju podaci u novom prostoru, ne znamo ni da li će metod u njemu raditi išta bolje nego u polaznom prostoru. U prethonom primeru konstruisali smo preslikavanje $\Phi(x) = (x_1, x_2, x_1^2 + x_2^2)$ koje definiše jedan kernel nad tačkama ravni, ali taj primer je polazio od pretpostavke da nam je raspored instanci u prostoru poznat, što je prejaka pretpostavka. Pitanje je da li postoji kernel koji bi nam omogućio rad sa proizvoljno raspoređenim podacima. Ispostavlja se da je jedna vrsta kernela za to dovoljna (iako ne nužno uvek najpogodnija). To je takozvani Gausov kernel:

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{\gamma}\right)$$

Veće vrednosti parametra γ vode širim Gausovim zvonima, a manje užim. Objasnjimo univerzalnu primenljivost ovog kernela primerom. Na slici 5.8, prikazan



Slika 5.8: Podaci za obučavanje (levo), model u slučaju Gausovog kernela kada je svaka instanca potporni vektor (sredina) i zajedničko dejstvo funkcije sgn na taj model pod pretpostavkom da je slobodni član negativan, usled čega je ljubičasta površ izdignuta (desno).

je skup tačaka i odgovarajući model koji ih tačno klasificuje. Ključni uvid je da za dovoljno malo γ svaka tačka može biti potporni vektor sa pridruženom okolinom u kojoj nema drugih tačaka i da se množenjem Gausovog zvona znakom klase može postići njena tačna klasifikacija. Ukoliko se dozvole kerneli sa dovoljno malom širinom, ovo je moguće postići na svakom neprotivrečnom skupu podataka, što znači da skup svih modela zasnovanih na Gausovom kernelu proizvoljne širine ima beskonačnu Vapnik-Červonenkisovu dimenziju i da se u njemu može naći model saglasan sa bilo kojim podacima. Ipak, u praksi se nikad ne omogućava korišćenje proizvoljnih vrednosti parametra γ , već se γ podešava kao hiperparametar, nalik regularizacionom hiperparametru.

Vratimo se na formu modela metoda potpornih vektora:

$$f_{w,w_0}(x) = \sum_{i=1}^N \alpha_i y_i k(x_i, x) + w_0$$

U slučaju linearног kernela, odnosno običnog skalarnog proizvoda, koeficijenti linearног modela su se mogli eksplisitno izračunati zahvaljujući distributivnosti skalarnog proizvoda u odnosu na sabiranje:

$$w = \sum_{i=1}^N \alpha_i y_i x_i$$

U ovom, opštijem slučaju to nije moguće, što znači da je čuvanje instanci neophodno, kako bi se izračunale vrednosti kernela i dobilo predviđanje modela. Ovo je uobičajeno za metode zasnovane na kernelima. Međutim, u ovom kontekstu već pomenuto svojstvo metoda potpornih vektora da model zavisi samo od nekih instanci, dobija na važnosti. Za razliku od drugih metoda zasnovanih na kernelima, u slučaju ovog metoda dovoljno je čuvati samo neke instance, a prilikom predviđanja, zahvaljujući manjem broju instanci za koje se računaju vrednosti kernela, izračunavanje je brže.

Na isti način, odnosno zamenom skalarnog proizvoda kernelom, može se kernelizovati i metod potpornih vektora za regresiju. I šire, mnogi metodi mašinskog učenja kod kojih se model može izraziti u terminima skalarnih proizvoda sainstancama iz skupa za obučavanje, mogu se kernelizovati, što često vodi boljim prediktivnim performansama.

5.3 Metodi zasnovani na najbližim susedima

Kao što metodi zasnovani na kernelima počivaju na upotrebi funkcija sličnosti, tako metodi zasnovani na najbližim susedima počivaju na upotrebi rastojanja. Pritom, veza između rastojanja i sličnosti je tesna, tako da ovo ne predstavlja razliku između ove dve grupe metoda, već sličnost. Tipičan hiperparametar ovakvih metoda je broj suseda koji se razmatra.

5.3.1 Ocena gustine raspodele zasnovana na najbližim susedima

Do metoda k najbližih suseda za ocenu gustine raspodele dolazi se kada se umesto fiksiranja zapremine, kao u slučaju Parzenovog prozora, kao princip konstruisanja ocene gustine raspodele uzme fiksiranje broja tačaka k koje treba da budu obuhvaćene okolinom tačke u kojoj se ocenjuje gustina raspodele. Tada se računa sfera najmanje zapremine V_k koja sadrži k tačaka i dolazi se do ocene

$$p(x) = \frac{k}{NV_k}$$

Primetimo da oblik sfere zavisi od izabrane metrike.

Problem sa datom ocenom gustine je taj što za konačne uzorke zapravo ne predstavlja validnu gustinu raspodele, zato što integral po x divergira. Recimo, u slučaju $k = 1$ će u tačkama uzorka zapremina najmanje sfere biti nula, a ocena gustine beskonačna. Ipak, ova ocena može biti korisna, makar za izvođenje drugih metoda, kao što će biti urađeno u narednom odeljku.

Male vrednosti broja k vode preprilagođavanju tako što se masa verovatnoće raspoređuje sve više na tačke iz skupa za obučavanje, a sve manje na druge tačke, dok velike vrednosti broja k vode usrednjavanju i gubljenju informacije, tako što se masa verovatnoće raspoređuje ravnomernije nego što bi trebalo, čime se gubi struktura raspodele podataka koja u konkretnom slučaju možda i nije ravnomerna.

5.3.2 Algoritam k najbližih suseda

Algoritam k najbližih suseda verovatno je najjednostavniji algoritam mašinskog učenja. Može služiti za klasifikaciju sa proizvoljnim brojem klasa, kao i za regresiju. Osnovna pretpostavka ovog algoritma je postojanje rastojanja nad prostorom atributa. Najčešće se pretpostavlja vektorska reprezentacija instanci i euklidsko rastojanje, ali takve pretpostavke nisu neophodne.

Hladnoća	Curenje iz nosa	Glavobolja	Groznica	Grip
Da	Ne	Blaga	Da	Ne
Da	Da	Ne	Ne	Da
Da	Ne	Jaka	Da	Da
Ne	Da	Blaga	Da	Da
Ne	Ne	Ne	Ne	Ne
Ne	Da	Jaka	Da	Da
Ne	Da	Jaka	Ne	Ne
Da	Da	Blaga	Da	Da

Tabela 5.1: Tabela podataka za problem dijagnostifikovanja gripa.

Procena verovatnoće klase na osnovu vrednosti atributa može se uraditi korišćenjem Bajesove formule i ocene gustine raspodele pomoću k najbližih suseda:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{\frac{k_y}{N_y V_k} \frac{N_y}{N}}{\frac{k}{N V_k}} = \frac{k_y}{k}$$

gde je N_y broj podataka koji pripada klasi y , a k_y broj tačaka iz k najbližih suseda koji pripadaju klasi y . Na osnovu izvedene uslovne raspodele, predviđanje se vrši na uobičajen način:

$$f(x) = \operatorname{argmax}_y p(y|x)$$

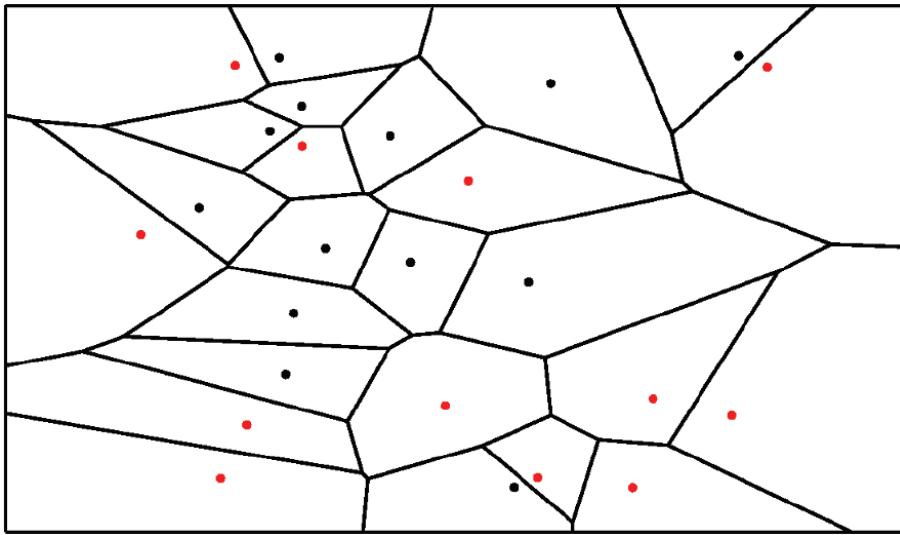
Drugim rečima, algoritam k najbližih suseda klasificuje nepoznatu instancu tako što pronalazi k instanci iz skupa za obučavanje koje su joj najbliže u smislu neke izabrane metrike i pridružuje joj klasu koja se najčešće javlja među tih k instanci. Ovaj algoritam je očito diskriminativni. Retko predstavlja najbolji izbor za rešavanje nekog problema, ali neretko daje relativno dobre rezultate, a izuzetno lako se implementira i primenjuje.

Primer 4 U tabeli 5.1 prikazan je primer primene ovog algoritma. U odnosu na date podatke, potrebno je klasifikovati instancu (Da, Ne, Blaga, Ne). Kako su vrednosti atributa kategoričke, potrebno je definisati specifičnu funkciju rastojanja. Neka je izabrana funkcija:

$$d(x, x') = \sum_{i=1}^n I(x_i \neq x'_i)$$

Ako se u obzir uzima 1 najbliži sused, najbliži je prvi primer iz tabele, pa je predviđena klasa Ne.

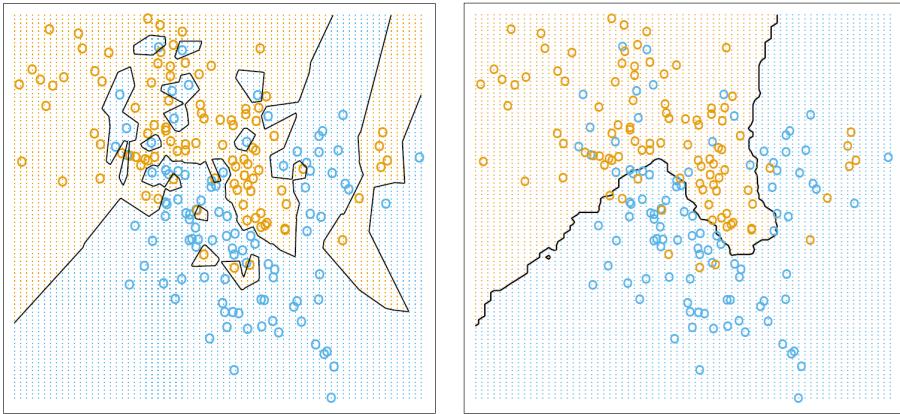
Primetimo da ovaj algoritam nema eksplicitnu formu modela, funkciju greške u odnosu na koju bi se model obučavao, pa ni fazu obučavanja uopšte, osim



Slika 5.9: Način na koji algoritam k najbližih suseda deli prostor atributa na podoblasti različitih klasa.

izbora vrednosti hiperparametra k . Zapravo, slično važi i za metod Nadaraja-Votson, pa i za mnoge druge (ali ne sve) metode zasnovane na instancama. U slučaju metoda k najbližih suseda, sav rad se obavlja u fazi predviđanja. Kao i u slučaju mnogih metoda zasnovanih na kernelima, kako ne proizvodi model u nekoj funkcionalnoj parametrizovanoj formi, koji se može čuvati radi kasnije primene, potrebno je čuvati sve instance ili, eventualno, neki podskup predstavnika ukoliko je broj instanci preveliki. Iako algoritam ne daje eksplicitan model, on je implicitno definisan skupom instanci i može se vizualizovati. Na slici 5.9 prikazan je način na koji algoritam k najbližih suseda deli prostor atributa na podoblasti koje pripadaju različitim klasama za $k = 1$. U tom slučaju, svakoj instanci je pridružen skup tačaka koje su bliže toj instanci nego drugiminstancama iz skupa za obučavanje. Za razliku od prethodnih algoritama klasifikacije koji su prepostavljali da je razdvajajuća granica među klasama hiperravan, ovaj algoritam dozvoljava značajno komplikovaniju razdvajajuću granicu među klasama. Štaviše regioni koji pripadaju istoj klasi mogu biti i nepovezani.

Na slici 5.10, dato je poređenje razdvajajućih granica algoritma k najbližih suseda u slučaju $k = 1$ i $k = 15$. Primećuje se da je ona u prvom slučaju komplikovanija nego u drugom. U ekstremnom slučaju, kada je k jednako veličini skupa za obučavanje, ceo prostor se klasificiše u istu – većinsku klasu. Ovo je slično kako ponašaju metoda zasnovanih na kernelima u odnosu na izbor hiperparametra σ , tako i ponašaju regularizacije kod parametarskih metoda. To nije slučajno. Svi ti hiperparametri igraju vrlo slične uloge. Što je vrednost



Slika 5.10: Podela prostora atributom k najbližih suseda, za $k = 1$ i $k = 15$.

k manja, ovaj algoritam je u većoj opasnosti od preprilagođavanja podacima za obučavanje. Što je vrednost k veća, ta opasnost je manja, ali lakše dolazi do potprilagođavanja.

Za algoritam k najbližih suseda, za $k = 1$, postoji zanimljiv teorijski rezultat. Definišimo prvo pojam *Bajesovog rizika*. Neka je $p(y|x)$ prava (ne ocenjena) uslovna verovatnoća klase za date vrednosti atributa. Klasifikator koji vrši klasifikaciju na sledeći način

$$y = \operatorname{argmax}_y p(y|x)$$

naziva se *Bajesov klasifikator*. Pritom, kako raspodela $p(y|x)$ nije poznata, ovaj klasifikator se ne može konstruisati, ali može poslužiti za teorijska razmatranja. Rizik Bajesovog klasifikatora je *Bajesov rizik*. Neka je R stvarni rizik algoritma 1 najbližeg suseda, neka je R^* Bajesov rizik i neka je M broj klasa. Tada važi

$$R^* \leq R \leq R^* \left(2 - \frac{M}{M-1} R^* \right)$$

Ugrubo, stvarni rizik ovog algoritma je ne više od dva puta veći od teorijski najmanjeg rizika koji se može postići bilo kakvim algoritmom.

Bitno je znati kada se algoritam k najbližih suseda ponaša posebno loše. U visokodimenzionalnim prostorima, rastojanja se ponašaju drugačije nego što po intuiciji trodimenzionalnog prostora očekujemo. Razmotrimo primer dva niza zapremina n dimenzionalnih kocki – sa stranicom dužine 1 i stranicom dužine 0.99. Važi

$$\lim_{n \rightarrow \infty} \frac{V_{0.99}^n}{V_1^n} = \lim_{n \rightarrow \infty} \frac{0.99^n}{1^n} = 0$$

Slično se može pokazati i za lopte vrlo bliskih poluprečnika, sa centrom u istoj tački. Zaključujemo da su u visokodimenzionalnim prostorima praktično

sve tačke lopte vrlo daleko od njenog centra. Štaviše da su uglavnom na istom rastojanju od centra (npr. na rastojanju većem od $0.99r$, a manjem od r). Nekoliko teorijskih rezultata pokazuje da je u visokodimenzionalnim prostorima varijacija distanci između nasumice generisanih tačaka mala. Ovi uvidi su obeshrabrujući za primenu algoritma k najbližih suseda, pošto se on zasniva na razlikovanju bliskih i dalekih suseda, a ako su te razlike male, njegova predviđanja nisu pouzdana. Ovo se primećuje i u praksi.

Još jedan problem ovog algoritma u slučaju visokodimenzionalnih prostora je prokletstvo dimenzionalnosti. Ukoliko se tačka klasificuje na osnovu bliskih suseda, očekuje se da postoje bliski susedi. To je moguće ako je prostor gusto popunjeno podacima za obučavanje. Međutim kako dimenzionalnost prostora raste, broj tačaka potreban za održavanje nekog konstantnog stepena gustine raste eksponencijalno u odnosu na dimenziju i nije za očekivati da će dovoljna količina podataka biti na raspolaganju, a ako i jeste, može se ispostaviti da predviđanja budu računski skupa.

Postoje još neki problemi primene ovog algoritma koji, na sreću, nisu fundamentalni, odnosno mogu se otkloniti određenim preprocesiranjima. Prvo pitanje je pitanje atributa koji se mere na različitim skalamama. Razmotrimo konkretnije slučaj euklidske metrike

$$d(x, x') = \sqrt{\sum_{i=1}^n (x_i - x'_i)^2}$$

Ukoliko atributi x_1 i x_2 predstavljaju dužine, ali je prvi meren u milimetrima, a drugi u metrima, razlika $(x_1 - x'_1)^2$ će tipično biti mnogo veća od razlike $(x_2 - x'_2)^2$, prosto zbog različite skale na kojoj se vrednosti izražavaju, a ne zbog razlike u stvarnim dužinama. Otud će atribut x_1 više uticati na ishod klasifikacije, nego atributa x_2 , što ne mora biti opravданo. Otud se pre primene algoritma k najbližih suseda obavezno primenjuje neki vid preprocesiranja koji svodi različite atrbute na istu skalu. Na primer, standardizacija.

Drugo pitanje je pitanje ponovljenih atributa ili njihove visoke koreliranoosti. Iako ponavljanje baš istih atributa nije verovatno, razmatranje tog slučaja očiglednije ukazuje na problem. U praksi problem ne mora biti toliko drastičan, jer atributi nisu ponovljeni, ali visoka korelacija vodi vrlo sličnoj vrsti problema. Neka su podaci opisani pomoću dva atributa. Neka je potom jedan umnožen 100 puta. U euklidskom rastojanju će razlika po prvom atributu figurisati jednom, a po drugom 100 puta. Otud će uticaj prvog atributa na predviđanje biti zanemarljiv u odnosu na uticaj drugog, a oba mogu biti podjednako informativna, ili čak prvi može biti informativniji. Ovakvi problemi se rešavaju ili ublažavaju tehnikama smanjenja dimenzionalnosti podataka poput *analyze glavnih komponenti* (eng. principal component analysis).

Da sumiramo, prednosti algoritma k najbližih suseda su jednostavnost, laka implementacija i primena i proizvoljni oblici granica između klasa. Negativne strane su loše ponašanje predviđanja u visokodimenzionalnim prostorima, neotpornost na ponovljene i visoko korelirane atrbute, nedostatak interpretabil-

nosti, kao i standardne mane algoritama zasnovanih na instancama – potreba za čuvanjem instanci iz skupa za obučavanje i više vreme primene modela.

Algoritam k najbližih suseda se može primeniti i na problem regresije. U tom slučaju, uzimanje većinske vrednosti među vrednostima ciljne promenljive suseda nema smisla, pošto se vrednosti na kontinualne promenljive često neće ponavljati. Prirodniji pristup je za nepoznatu instancu naći k najbližih suseda i uprosečiti vrednosti ciljne promenljive tih instanci. I u slučaju regresije važe prethodne primedbe vezane za prednosti i mane algoritma.

Glava 6

Ansambli

Jedan od najuspešnijih pristupa mašinskom učenju su *ansambls* (eng. *ensemble*), odnosno skupovi većeg broja modela koji zajednički donose odluke. Ključni uvid na kojem počivaju ovakvi složeni modeli je da veći broj modela, konstruisan na adekvatan način, može dati značajno bolju preciznost od samo jednog modela. Poučna je anegdota o poznatom naučniku Frencisu Galtonu, koji je početkom 20 veka prisustvovao stočnom sajmu na kojem je 800 učesnika pokušavalo da pogodi težinu jednog vola. Iako su pojedinačni učesnici grešili, prosek njihovih pogađanja od 1197 funti, predstavljao je tačnu težinu vola. Ova anegdota sadrži ključne elemente bar jedne vrste ansambla – veći broj modela koji samostalno možda i nisu dovoljno dobri, ali prave relativno dobro informisana predviđanja i greše na nezavisne načine. Uprosečavanjem, nezavisne greške se poništavaju, što značajno popravlja preciznost predviđanja.

Postoje različiti pristupi konstrukciji ansambla, ali vredi naglasiti da su neki od njih po preciznosti koju nude dosledno među najboljim metodama za rešavanje različitih problema. Tipično u slučaju da su podaci predstavljeni u vektorskom obliku, ansambls predstavljaju najpouzdaniji pristup postizanju visoke preciznosti u predviđanju. U takvom slučaju obično su i precizniji i efikasniji i jednostavniji za obučavanje od neuronskih mreža. Naravno, njihov kvalitet i efikasnost zavise i od vrste modela koji sačinjavaju ansambl.

U nastavku se diskutuju dve familije modela kojima pripadaju najkorишćeniji modeli ovog tipa.

6.1 Prosta agregacija

Prosta agregacija (eng. *bootstrap aggregation, bagging*) podrazumeva obučavanje većeg broja modela koji pojedinačno ne moraju imati visoku preciznost, ali čije su greške nezavisne. Prilikom predviđanja, svi modeli nude svoja predviđanja, koja se agregiraju kako bi se dobilo predviđanje ansambla. Ukoliko se radi o regresiji, agregacija se tipično vrši uprosečavanjem (a moguće je koristiti i medijanu), a u slučaju klasifikacije glasanjem. Snaga ovakvog modela počiva na ideji

da će se prilikom agregacije greške koje modeli nezavisno prave poništiti. Na primer, ukoliko se rešava problem regresije i ako se modeli posmatraju kao nezavisne slučajne promenljive X_1, \dots, X_m sa istom raspodelom koja ima prosek μ i varijansu σ^2 , na osnovu centralne granične teoreme sledi da

$$\sqrt{m} \left(\frac{1}{m} \sum_{i=1}^m X_i - \mu \right) \rightarrow \mathcal{N}(0, \sigma^2)$$

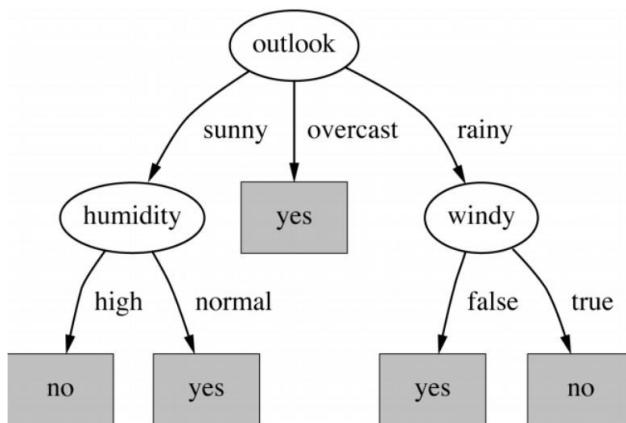
u raspodeli. Drugim rečima, prosek modela približno ima očekivanje μ i varijansu σ^2/m . Ukoliko su predviđanja modela nepričasna, odnosno ako je μ jednakoj tačnoj vrednosti koju je potrebno predvideti, prosta agregacija nudi mogućnost smanjenja varijanse bez povećanja sistematskog odstupanja (!), što vodi povećanju preciznosti. Slična analiza se može izvesti i u slučaju klasifikacije. Prepostavimo, jednostavnosti radi, problem binarne klasifikacije, pri čemu su obe klase podjednako zastupljene među instancama. Tada je verovatnoća da se slučajnim pogadanjem pogodi prava klasa jednaka 0.5. Neka su modeli koje razmatramo malo bolji od slučajno pogadanja, odnosno, neka su X_1, \dots, X_m Bernulijeve slučajne promenljive takve da je verovatnoća uspeha P (pogadanja prave klase) strogo veća od 0.5. Tada je ukupan broj tačnih predviđanja k raspodeljen u skladu sa binomnom raspodelom

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

čije je očekivanje jednako Pm , odnosno kako broj modela u ansamblu teži beskonačnosti, ideo tačnih predviđanja teži P . Kako je P strogo veće od 0.5, za dovoljno veliki ansambl, verovatnoća pogrešnog predviđanja postaje proizvoljno mala. Naravno, prepostavka o nezavisnosti grešaka modela predstavlja idealizaciju koja ne može biti sasvim ispunjena u praksi kada imamo konačnu količinu podataka. Stoga, problemi vezani za nezavisnost grešaka modela će se odraziti i na domete ansambla, pa stoga u praksi ne možemo očekivati proizvoljno smanjenje greške pri povećavanju broja broja modela u ansamblu.

Primetimo da je u prethodnoj diskusiji pojam dobro informisanih predviđanja formalizovan prepostavkom o nepričasnosti u slučaju regresije, a prepostavkom da model predviđa bolje od slučajnog predviđanja u slučaju klasifikacije. Pritom, prihvatljivi su modeli čija predviđanja imaju visoku varijansu u slučaju regresije i proizvoljno malu (ali veću od nule) prednost u odnosu na slučajno pogadanje u slučaju klasifikacije. Ovakvi modeli koji samostalno nisu dovoljno dobri, ali predstavljaju dovoljno dobru osnovu za kreiranje ansambala nazivaju se *slabi modeli* (eng. *weak learners*). Značaj ovakvih modela je u tome što ih je obično lako dobiti, što čini izgradnju ansambla mogućom.

Tipično, rezultati su bolji što je broj modela veći. Ipak, sa povećanjem broja modela, raste i računska zahtevnost obučavanja i predviđanja. Modeli koji se koriste pri agregaciji ne moraju biti iste reprezentacije, iako često jesu. Jedan od najuspešnijih i praktično najčešće korišćenih primera proste agregacije i ansambla uopšte su *slučajne šume*. Pre njih, potrebno je opisati bazni model čijom agregacijom nastaju – *stabla odlučivanja*.



Slika 6.1: Primer stabla odlučivanja.

6.1.1 Stabla odlučivanja

Stabla odlučivanja predstavljaju jednostavan i interpretabilan model mašinskog učenja, primenljiv i na regresiju i na klasifikaciju. U svakom čvoru stabla, osim listova, nalazi se po jedan test. Svaki test ima više od jednog ishoda. Svakom ishodu odgovara jedna grana stabla koja vodi do sledećeg čvora. Listovi su označeni vrednostima koje predstavljaju predviđanja stabla. Primer stabla odlučivanja dat je na slici 6.1. Na instancu za koju je potrebno dati predviđanje prvo se primenjuje test u korenu stabla. U zavisnosti od ishoda, instance se prosleđuje duž grane koja odgovara tom ishodu ka sledećem čvoru i postupak se rekurzivno nastavlja dok instance ne stigne do lista, čija vrednost predstavlja traženo predviđanje.

Postoji ogroman broj varijanti stabala odlučivanja i teško je reći nešto više što važi za sve varijante. Ipak, probaćemo da približimo još neke njihove aspekte. Testovi koji se vrše u svakom od čvorova su tipično (ali ne nužno) provjeravaju vrednosti pojedinačnih atributa. U slučaju kategoričkih atributa, test obično predstavlja proveru jednakosti sa nekom konkretnom vrednošću atributa, proveru pripadnosti nekom skupu vrednosti, itd. U slučaju neprekidnih atributa, tipičan test predstavlja proveru da li je vrednost atributa veća ili manja od neke vrednosti. Ipak, neke varijante stabala odlučivanja sadrže testove nad više promenljivih odjednom, na primer ispitivanje vrednosti neke njihove linearne kombinacije.

Važna razlika u odnosu na većinu prethodno pomenutih modela je diskretna struktura stabala odlučivanja, koja čini gradjentne metode optimizacije neprimenljivim. Alternativu predstavljaju kombinatorne metode optimizacije. Takav pristup je tipično previše zahtevan za praktičnu upotrebu, pa se pribegava pohlepnim pristupima optimizacije, koji u skladu sa nekom heuristikom

formulišu najbolji test u korenu stabla, dele instance skupa za obučavanje na više grupe kojima odgovaraju isti ishodi tog testa, potom rekurzivno konstruišu stabla nad tim podskupovima i povezuju tako dobijena stabla na koren ukupnog stabla. Kako bi se ovakav postupak implementirao, potrebno je precizirati način na koji se formuliše test u nekom čvoru, kriterijum zaustavljanja i kako se određuju vrednosti u listovima. Ovi izbori su heuristički i ima ih raznih. U nastavku diskutujemo neke izbore.

Prilikom dizajna algoritma za učenje stabala odlučivanja, prvo se određuje klasa testova koje je uopšte moguće vršiti. Jednostavnosti radi, prepostavimo da su svi atributi kategorički i da je test ispitivanje vrednosti nekog atributa i da svakoj vrednosti atributa odgovara po jedna grana. Postavlja se pitanje kako najbolje, makar u pohlepnom smislu, izabrati atribut koji će biti testiran u korenu stabla za dati skup instanci. Osnovna ideja je izabrati atribut koji u nekom smislu najbolje razvrstava instance, tako da dobijene podgrupe budu što homogenije u odnosu na klase. Na primer, idealan atribut bi razvrstao instance tako da su klase instanci u svakoj podgrupi iste. Takav atribut bi očito bio vrlo koristan u klasifikaciji. Ipak, tako nešto nije verovatno i potrebno je definisati neku meru kvaliteta atributa u odnosu na razvrstavanje koje nudi, odnosno meru čistoće (eng. *purity*) ili homogenosti dobijenih podgrupa. Homogenost se može meriti na različite načine. Neki od poznatih pristupa kvantifikovanju homogenosti skupa instanci su *entropija* i *Ginijev indeks*. Oba pristupa počivaju na udelima instanci različitih klasa u ukupnom skupu. Na primer, ukoliko se u skupu od 10 instanci javljaju 4 instance klase A , tri instance klase B i tri instance klase C , ti udeli su $p_A = 0.4$, $p_B = 0.3$ i $p_C = 0.3$. U opštem slučaju, za C klase, entropija se definiše kao:

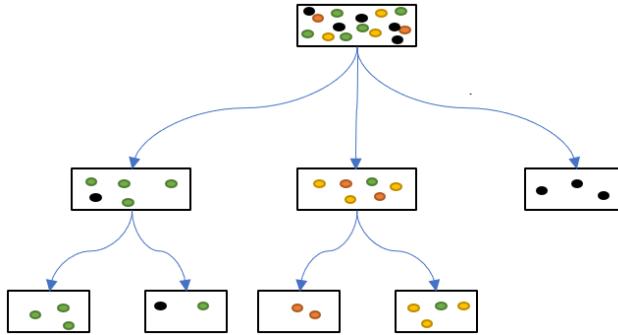
$$H(p_1, \dots, p_C) = - \sum_{i=1}^C p_i \log p_i$$

Lako se uočava da ovaj izraz dostiže maksimalnu vrednost ako za svako i važi $p_i = 1/C$, a vrednost 0, ako je $p_i = 1$ za neko i . Odavde razumemo da entropija zapravo predstavlja meru nehomogenosti, odnosno da su manje vrednosti poželjnije, a da je vrednost nula idealna. Ginijev indeks se definiše kao

$$G(p_1, \dots, p_C) = 1 - \sum_{i=1}^C p_i^2$$

U slučaju da sve instance pripadaju jednoj klasi, njegova vrednost je takođe 0, dok ako je raspodela klasa uniformna, dostiže se maksimalna vrednost, kao i u slučaju entropije. Na osnovu neke od ovih ili neke slične mere, atribut se obično evaluira tako što se izračuna smanjenje nehomogenosti nekon particionisanja skupa instanci po vrednostima tog atributa sledećim izrazom (u slučaju korišćenja entropije):

$$H(\mathcal{D}) - \sum_{i=1}^C \frac{|\mathcal{D}_i|}{|\mathcal{D}|} H(\mathcal{D}_i)$$



Slika 6.2: Ilustracija homogenosti instanci u čvorovima stabla.

gde je \mathcal{D}_i skup svih instanci za koje je vrednost atributa A jednaka i . Ovaj izraz očito izražava razliku između nehomogenosti polaznog skupa i prosečne nehomogenosti njegovih podskupova dobijenih particionisanjem po vrednosti datog atributa. Smanjenje nehomogenosti duž putanje od korena do lista ilustrovano je na slici 6.2.

Što se tiče kriterijuma zaustavljanja, postoje razni. U kontekstu kategoričkih atributa, kako je opisano u prethodnom paragrafu, očito dubina stabla ne može biti veća od broja atributa. Takođe, dalje deljenje nema smisla ukoliko sve instance imaju identične vrednosti atributa ili identične vrednosti ciljne promenljive. Nekada se dubina stabla eksplicitno ograničava i rekurzivni postupak se zaustavlja na nekoj unapred određenoj dubini. Nekada se postupak zaustavlja kada mera nehomogenosti podskupa instanci koje je pridružen tekućem čvoru dostigne neku nisku, unapred definisani vrednost. Mogući su i drugi načini.

Određivanje vrednosti koje odgovaraju listovima je nešto principijeljnije i jednostavnije od prethodnih aspekata izgradnje stabla. U slučaju regresije, ta vrednost se tipično izračunava kao prosek vrednosti koje su pridružene datom listu, a u slučaju klasifikacije, kao većinska klasa među tim instanicama. Zanimljivo je da je moguće dobiti i grube procene pouzdanosti ovih predviđanja u vidu standardne devijacije u slučaju regresije i verovatnoće date klase u slučaju klasifikacije. Obe vrednosti se ocenjuju na osnovu instanci koje su pridružene odgovarajućem listu. Ipak, ovakve ocene treba uzeti sa rezervom jer mogu biti formirane na osnovu malog broja instanci.

Stabla odlučivanja se tipično ne smatraju modelima visoke preciznosti. Duboka stabla su tipično skloni preprilagođavanju, pošto primenom velikog broja testova mogu da opišu i nebitne specifičnosti podataka na kojima su obučavana. Plitka stabla tipično imaju problem potprilagođavanja. Očito, dubina stabla predstavlja regularizacioni hiperparametar, ali uprkos mogućnosti regularizacije, u praksi ispoljavaju visoku varijansu. Jedan od razloga za ovakvo ponašanje

je pohlepna priroda algoritama njihovog obučavanja.

Dobra strana stabala odlučivanja je njihova interpretabilnost. Ukoliko je stablo malo, testovi koje vrši jasno ukazuju na to na osnovu čega stablo donosi odluke. Štaviše, svaka putanja kroz stablo odlučivanja od korena do lista može se videti kao jedno `if ... then` pravilo, u kojem uslov predstavlja konjunkciju svih ishoda testova duž putanje, dok je odluka vrednost u listu. Još neke dobre strane su da stabla prirodno kombinuju atribute različitih vrsta (kategoričke i neprekidne), neosetljiva su na monotone transformacije atributa i vrlo su efikasna u vreme predviđanja. Ovo poslednje ne mora važiti na primer za duboke neuronske mreže kada je potrebno vršiti predviđanja veliki broj puta u sekundi.

6.1.2 Slučajne šume

Metod slučajnih šuma se zasniva na prostoj agregaciji stabala odlučivanja. Ansambl se sastoji od m stabala treniranih na različitim podskupovima skupa za obučavanje. Jedno stablo se obučava tako što se izabere podskup skupa za obučavanje određene veličine, pri čemu je moguće koristiti i samo podskup ukupnog skupa atributa. Stabla se obučavaju na različitim podskupovima kako bi njihove greške bile što slabije korelisane, što ostavlja prostor za popravku agregacijom. Hiperparametri su očito broj stabala m i veličine podskupova instanci i atributa.

Broj stabala m se može posmatrati i kao regularizacioni parametar, sa vrlo zanimljivim svojstvom – visoke vrednosti su tipično bolje. Korišćenje većeg broja stabala tipično smanjuje preprilagođavanje i daje bolje rezultate. Naravno, po cenu računskog vremena.

Slučajne šume su jedan od najprimjenjenijih algoritama mašinskog učenja. Njihovo obučavanje je relativno efikasno, a preciznost predviđanja obično među najboljim za vektorski predstavljene podatke. Slučajna šuma, kao i drugi ansambl nije interpretabilna.

6.2 Pojačavanje

Prilikom proste agregacije, modeli se konstruišu potpuno nezavisno. Opravdanje za ovakav pristup je već dato. Ipak, to ne znači da nije moguće postići više, ako bi modeli nekako uzimali u obzir ponašanje drugih modela. Osnovna ideja *pojačavanja* (eng. *boosting*) je da se ansambl gradi dodajući model po model, pri čemu se svaki od modela obučava tako da što bolje nadomesti slabosti tekućeg skupa modela, odnosno da ga pojača. Veliki broj algoritama pojačavanja počinje izgradnju ansambla od modela $F_0(x) = 0$ i prepostavljući da je dostupan ansambl F_{m-1} gradi ansambl F_m dodavanjem još jednog modela na sledeći način:

$$(\beta_m, f_m) = \underset{\beta, f}{\operatorname{argmin}} \sum_{i=1}^N L(y_i, F_{m-1}(x_i) + \beta f(x_i))$$

$$F_m = F_{m-1}(x) + \beta_m f_m(x)$$

Broj modela od kojih se sastoji ansambl je hiperparametar.

Primetimo da se u slučaju regresije, ako je $L(u, v) = (u - v)^2$, dobija vrlo intuitivno rešenje:

$$L(y_i, F_{m-1}(x_i) + \beta f(x_i)) = ((y_i - F_{m-1}(x_i)) - \beta f(x_i))^2$$

odnosno, novi model βf pokušava da što bolje predvodi grešku predviđanja postojećeg ansambla F_{m-1} .

6.2.1 AdaBoost

Prvi i najpoznatiji algoritam pojačavanja, pod nazivom AdaBoost, izведен je za problem binarne klasifikacije i podrazumeva eksponencijalnu funkciju greške:

$$L(u, v) = \exp(-uv)$$

pri čemu se podrazumeva da važi $y \in \{-1, 1\}$. Izbor eksponencijalne funkcije greške nije dobar zbog osetljivosti na odudaraće podatke, ali je računski pogodan. Često se dobijaju bolji rezultati korišćenjem robustnije funkcije greške u vidu šarke

$$L(u, v) = \max(0, 1 - uv)$$

Ipak, prikazaćemo izvođenje algoritma AdaBoost u izvornom obliku. Problem koji treba rešiti je:

$$\begin{aligned} \operatorname{argmin}_{\beta, f} \sum_{i=1}^N \exp(-y_i(F_{m-1}(x_i) + \beta f(x_i))) &= \\ \operatorname{argmin}_{\beta, f} \sum_{i=1}^N \exp(-y_i F_{m-1}(x_i)) \exp(-y_i \beta f(x_i)) &= \\ \operatorname{argmin}_{\beta, f} \sum_{i=1}^N w_i^m \exp(-y_i \beta f(x_i)) \end{aligned}$$

pri čemu je $w_i^m = \exp(-y_i F_{m-1}(x_i))$. Kako w_i^m ne zavisi od β i f , može se smatrati težinom koja se pridružuje instanci (x_i, y_i) u koraku m . Kako klasifikator f uzima vrednosti u skupu $\{-1, 1\}$, prethodni problem se može zapisati kao

$$\begin{aligned} \operatorname{argmin}_{\beta, f} \left[\sum_{i|f(x_i)=y_i} w_i^m \exp(-\beta) + \sum_{i|f(x_i)\neq y_i} w_i^m \exp(\beta) \right] &= \\ \operatorname{argmin}_{\beta, f} \left[(\exp(\beta) - \exp(-\beta)) \sum_{i=1}^N w_i^m I(y_i \neq f(x_i)) + \exp(-\beta) \sum_{i=1}^N w_i^m \right] \end{aligned}$$

Za bilo koje $\beta > 0$, rešenje po f se dobija rešavanjem problema:

$$\arg \min_f \sum_{i=1}^N w_i^m I(y_i \neq f(x_i))$$

što je standardan problem mašinskog učenja koji se može rešiti metodama koje su diskutovane u prethodnim glavama. Kada je rešavanjem ovog problema određeno f_m , rešavanjem problema:

$$\arg \min_\beta \sum_{i=1}^N w_i^m \exp(-y_i \beta f_m(x_i))$$

dobija se

$$\beta_m = \frac{1}{2} \log \frac{1 - E_m}{E_m}$$

gde je

$$E_m = \frac{\sum_{i=1}^N w_i^m I(y_i \neq f_m(x_i))}{\sum_{i=1}^N w_i^m}$$

Novi ansambl je

$$F_m(x) = F_m(x) + \beta_m f_m(x)$$

a težine se ažuriraju na sledeći način

$$w_i^m = w_i^m \exp(-\beta_m y_i f_m(x_i))$$

odnosno

$$w_i^m = w_i^m \exp(2\beta_m I(y_i \neq f_m(x_i))) \exp(-\beta_m)$$

Faktor $\exp(-\beta_m)$ se može zanemariti pošto je isti za sve instance, pa se dobija:

$$w_i^m = w_i^m \exp(2\beta_m I(y_i \neq f_m(x_i)))$$

Odavde se vidi da se uvećavaju težine onih instanci koje su pogrešno klasifikovane, pa se model koji se dodaje u ansambl u sledećoj iteraciji fokusira na te instance.

6.2.2 Gradijentno pojačavanje

Po pitanju preciznosti, metode gradijentnog pojačavanja su među najboljim metodama pojačavanja i mašinskog učenja uopšte. Obučavanje se takođe može smatrati relativno efikasnim. Zbog toga su vrlo popularne u primenama. Osnovna ideja dolazi iz gradijentnih metoda optimizacije, koje počivaju na popravljanju tekućeg rešenja optimizacionog problema dodavanjem vektora proporcionalnog negativnoj vrednosti gradijenta funkcije koja se minimizuje. Ovo ima smisla, pošto negativna vrednost gradijenta pokazuje smer opadanja funkcije. U kontekstu pojačavanja, ova ideja se realizuje tako što se model

f_m kojim se dopunjuje ansambl F_{m-1} odredi tako da aproksimira gradijent greške po funkciji F_{m-1} . Primetimo da koncept gradijenta po funkciji netrivialno uopštenje koncepta gradijenta po realnom vektoru. Ipak, to u kontekstu pojačavanja ne predstavlja značajnu komplikaciju.

U nastavku će biti skiciran algoritam gradijentnog pojačavanja pomoću stabala, koji se u praksi najčešće koristi. Prvo, primetimo da svakom čvoru stabla i odgovara jedan region prostora atributa R_i i vrednost v_i koja je tom regionu pridružena. Otud se stablo može zapisati kao

$$f(x) = \sum_{i=1}^M v_i I(x \in R_i)$$

gde je M broj regionala. Prvi model f_0 se određuje standardnim obučavanjem. Model f_m kojim se dopunjuje ansambl F_{m-1} se dopunjuje tako što se za svaku instancu (x_i, y_i) izračuna vrednost

$$r_i = - \left[\frac{\partial L(y, F(x))}{\partial F(x)} \right]_{F=F_{m-1}, (x,y)=(x_i,y_i)}$$

a potom se obuči regresiono stablo nad svim parovima (x_i, r_i) . Ovo stablo definiše regione R_i koji su od značaja, međutim, vrednosti v_i ne moraju biti relevantne. Naime, stablo je regresiono, a ansambl može biti i klasifikacioni. Otud se ove vrednosti nanovo izračunavaju, rešavanjem problema

$$v_i = \arg \min_v \sum_{(x,y) \in \mathcal{D}, x \in R_i} L(y, F_{m-1}(x) + v_i)$$

Potom se ansambl ažurira na standardan način:

$$F_m(x) = F_{m-1}(x) + \sum_{i=1}^M v_i I(x \in R_i)$$

Glava 7

Evaluacija i izbor modela

Evaluacija modela predstavlja kvantifikaciju njegove sposobnosti predviđanja. Ukoliko imamo na raspolaganju konačan broj modela, od kojih je potrebno koristiti jedan, očigledno se postavlja pitanje *izbora modela*, koje se obično rešava tako što se na neki način evaluiraju svi raspoloživi modeli i izabere se najbolji. Iako navedeno zvuči trivijalno, to nije slučaj, o čemu svedoči i činjenica da se u praksi, bilo akademskoj, bilo industrijskoj, često prave greške baš u ovim poslovima. Tehnike kojima se ovi poslovi vrše mogu biti netrivijalne za razumevanje, a nekad i za implementaciju, a često bivaju potcenjene, upravo zbog utiska jednostavnosti koje ove teme na prvi pogled ostavljaju.

Evaluacija modela počiva na *merama kvaliteta modela* i na *tehnikama evaluacije modela*. Kako izbor modela počiva na evaluaciji modela, tehnike koje se koriste su slične, što doprinosi konfuziji i potencijalnim greškama.

7.1 Mere kvaliteta modela

Mere kvaliteta modela zavise od vrste problema koji se rešava, kao i od željenih ishoda. Mogu se osmisliti za konkretni problem, ali mićemo se baviti opštim meraima kvaliteta modela u različitim kontekstima.

Mere koje se najčešće koriste za klasifikaciju su *tačnost klasifikacije* (eng. *classification accuracy*), *preciznost i odziv* (eng. *precision and recall*), F_1 mera i površina ispod ROC krive (eng. *area under the curve – AUC*).

Praktično sve često korišćenje mere kvaliteta klasifikacije počivaju na *matrici konfuzije* (eng. *confusion matrix*) i pojmovima vezanim za nju. Ovo je matrica C čiji element c_{ij} predstavlja broj elemenata klase i koji su klasifikovani u klasu j . Klasifikacija je očito najbolja kada je ova matrica dijagonalna, što znači da je klasifikacija potpuno ispravna. Nedijagonalni elementi označavaju greške. U slučaju binarne klasifikacije, obično se jedna klasa naziva *pozitivnom*, a druga *negativnom*. Tada matrica konfuzije ima specifičan oblik prikazan tabelom 7.1. *Stvarno pozitivne* (eng. *true positive*) instance su pozitivne instance koje su od strane modela prepoznate kao pozitivne. Broj takvih instanci skraćeno

		Pozitivno	Negativno
Stvarno/Predviđeno	Pozitivno	stvarno pozitivno (TP)	lažno negativno (FN)
	Negativno	lažno pozitivno (FP)	stvarno negativno (TN)

Tabela 7.1: Matrica konfuzije

označavamo TP . **Stvarno negativne** (eng. *true negative*) instance su negativne instance koje su od strane modela prepoznote kao negativne. Broj takvih instanci skraćeno označavamo TN . **Lažno pozitivne** (eng. *false positive*) instance su negativne instance koje su od strane modela proglašene pozitivnim. Broj takvih instanci skraćeno označavamo FP . **Lažno negativne** (eng. *false negative*) instance su pozitivne instance koje su od strane modela proglašene negativnim. Broj takvih instanci skraćeno označavamo FN .

Tačnost klasifikacije predstavlja udeo tačno klasifikovanih instanci u ukupnom broju instanci. U slučaju binarne klasifikacije, može se izraziti kao

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

Iako vrlo intuitivna, tačnost klasifikacije ne mora uvek biti pogodna mera kvaliteta. Jedan razlog je njena neinformativnost u slučaju da klase imaju vrlo različit broj instanci. Ukoliko jednoj klasi pripada 99% instanci, a drugoj 1%, naizgled impresivna tačnost od 0.99 može biti postignuta tako što će sve instance biti klasifikovane u prvu klasu. Ipak, takav klasifikator je beskorisan. Pritom, klase će često biti neizbalansirane, a čak i ekstremni slučajevi su realistični. Na primer, u slučaju detekcije prevara sa kreditnim karticama, detekcije retkih bolesti i slično.

Preciznost i odziv se obično koriste pri evaluaciji sistema za pronalaženje informacija. Ipak i taj problem se može smatrati problemom klasifikacije – razdvajanja bitnih od nebitnih informacija pri čemu se bitne prikazuju korisniku. Preciznost je udeo pozitivnih instanci u svim instancama koje su proglašene pozitivnim, odnosno

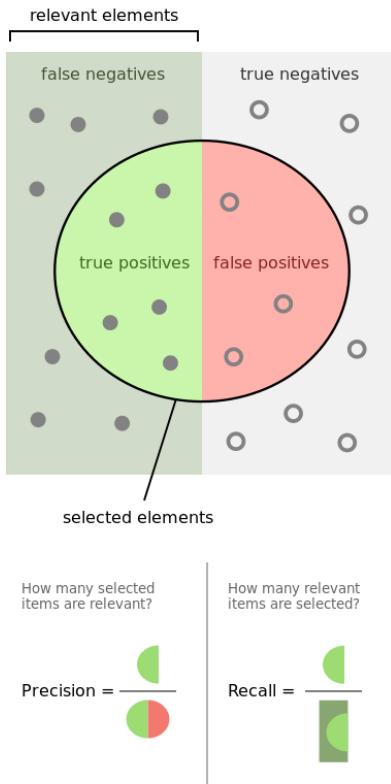
$$Prec = \frac{TP}{TP + FP}$$

i odgovara na pitanje koliko puta smo bili u pravu kad smo tvrdili da je nešto relevantno. Odziv je udeo pronađenih pozitivnih instanci u svim pozitivnim instancama, odnosno

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

i odgovara na pitanje koliko relevantnih informacija smo našli od svih relevantnih informacija. Preciznost i odziv su ilustrovani na slici Figure 7.1.

Preciznost i odziv pojedinačno nisu korisne mere. Naime, proglašavajući sve instance za pozitivne, odziv je maksimalan, a ne proglašavajući nijednu, ne pravi se nijedna greška, pa je preciznost maksimalna. Stoga ih ima smisla



Slika 7.1: Ilustracija pojmljivačkih mera preciznosti i odziva.

posmatrati samo zajedno. Način na koji se to najčešće radi je tako što se izračuna F_1 mera – njihova harmonijska sredina

$$F_1 = 2 \frac{Prec \cdot Rec}{Prec + Rec}$$

Treba imati u vidu da ova mera nije simetrična u odnosu na izbor neke klase kao pozitivne. U slučaju prethodno pomenutog klasifikatora koji klasificiše sve instance u prvu klasu, ako se ona smatra pozitivnom preciznost je 0.99, a odziv 1, što daje F_1 meru od 0.99, što deluje sjajno. Ali ako se ista mera računa u odnosu na drugu klasu, preciznost je 0.01, a odziv 1, što daje F_1 meru 0.02, što uopšte ne deluje sjajno. Uprosečavanjem ova dva slučaja, dobija se 0.51, što daje bolju sliku o kvalitetu klasifikacije nego tačnost.

Jedna mera koja se češće koristi u slučaju binarne klasifikacije sa neizbalansiranim klasama je površina ispod ROC krive. Pretpostavlja se da klasifikator počiva na nekom modelu koji različitim instancama pridružuje neke skore. Logistička regresija i metod potpornih vektora su primjeri takvih metoda.

Obično se takav skor prevodi u klasu nekom vrstom zaokruživanja u odnosu na neki prag. Recimo, u pomenutim metodima, vrednosti linearne modela koje su manje od nule označavaju jednu klasu, a one koje su veće od nule, označavaju drugu. Osnovni smisao površine ispod ROC krive, koja će tek biti definisana, je da se proveri da li ovaj metod dodeljuje niže skorove elementima jedne klase, a više elementima druge klase, nevezano od nekog konkretnog praga – moguće je da izbor praga nije idealan i da bi se čak mogao bolje podesiti, pa ga zanemarujemo. Klasu kojoj odgovaraju niže vrednosti skora nazovimo negativnom a onu kojoj odgovaraju više vrednosti nazovimo pozitivnom. **Površina ispod ROC krive** je ideo parova instanci iz dve klase takvih da je instanci iz negativne klase pridružen manji skor nego instanci iz pozitivne klase u ukupnom broju parova instanci iz različitih klasa. Formalnije, neka su C_1 i C_2 skupovi instanci iz različitih klasa, neka je $N_1 = |C_1|$ i $N_2 = |C_2|$ i neka je f model koji pridružuje skor instancama. Neka je $r(i)$ funkcija koja svakoj instanci i dodeljuje rang u sortiranom nizu instanci u odnosu na vrednost modela f . Ukoliko su instance izjednačene, za rang se uzima prosek svih indeksa (koji počinju od 1) koji odgovaraju tim instancama u sortiranom nizu. Na primer, ukoliko su skorovi 1, 2, 3, 3, 3, 4, 4, 5, rangovi su 1, 2, 4, 4, 4, 6.5, 6.5, 8. Ako se C_1 smatra negativnom, a C_2 pozitivnom klasom, tada važi

$$AUC = \frac{1}{N_1 N_2} \left(\sum_{i \in C_2} r(i) - \frac{N_2(N_2 + 1)}{2} \right)$$

U zagradi se nalazi razlika sume rangova instanci druge klase i sume rangova koja bi im pripadali da svi imaju manje skorove od svih instanci prve klase. Ukoliko ne bi bilo izjednačenih instanci, važila bi i sledeća jednakost

$$AUC = \frac{|\{(i_1, i_2) | i_1 \in C_1 \wedge i_2 \in C_2 \wedge f(i_1) < f(i_2)\}|}{N_1 N_2}$$

Odavde je očigledno da AUC ocenjuje verovatnoću da pri slučajnom izboru dve instance iz različitih klasa ona iz negativne klase ima manji skor od one iz pozitivne klase. Očito, ako je vrednost AUC velika, postoji neki prag koji dobro razdvaja dve klase. U suprotnom, ne postoji. Minimalna vrednost mere AUC je 0.5. Ukoliko bi se dobila manja vrednost, jednostavnom promenom znaka skora, dobila bi se veća. Ova mera nije osetljiva na neizbalansiranost klasa. Razmotrimo pomenuti klasifikator koji sve instance klasificuje u istu klasu, recimo tako što svima daje vrednost skora 1. Tada svima pripada rang $(N + 1)/2$, pa je AUC

$$\begin{aligned} \frac{1}{0.99 \cdot 0.01 N^2} \left(\sum_{i=1}^{0.99N} \frac{N+1}{2} - \frac{0.99N(0.99N+1)}{2} \right) &= \\ \frac{1}{0.99 \cdot 0.01 N^2} \left(\frac{0.99N(N+1)}{2} - \frac{0.99N(0.99N+1)}{2} \right) &= \end{aligned}$$

$$\frac{1}{0.99 \cdot 0.01N^2} \left(0.99N \frac{0.01N}{2} \right) = 0.5$$

što je minimalna vrednost, pa ukazuje na to da je kvalitet klasifikatora najgori.

Jedna stvar koja bi trebalo da je privukla pažnju je neobičan naziv ove mere. Način na koji je definisana ne uključuje nikakvu krivu. Zapravo, izabrali smo da je definišemo kao normiranu vrednost statistike Man-Vitni-Vilkoksonovog testa zbog jednostavne interpretacije u terminima verovatnoće. Alternativna definicija je drugačija. *ROC kriva* (eng. *receiver operating characteristic curve*) je kriva u koordinatnom sistemu udela FP instanci (x osa) u skupu svih instanci i udela TP instanci (y osa) u skupu svih instaci koji zaprema jedinični kvadrat. Za svaku vrednost praga na skorovima koje klasifikator daje, definisana je jedna tačka u ovom koordinatnom sistemu, a menjući ovaj skor od minimalnog (na datim instancama) do maksimalnog, dobija se kriva koja spaja koordinatni početak sa tačkom (1, 1). Površina ispod te krive je jednaka prethodno definisanoj veličini *AUC*, ali intuicija iza takve definicije nije očigledna.

Mere koje se najčešće koriste za regresiju su *srednjekvadratna greška* i njen koren (eng. *mean square error*, *root mean square error*), *srednja relativna greška* izražena u procentima (eng. *mean absolute percentage error*) i *koeficijent determinacije*, poznatiji pod oznakom R^2 . Važi

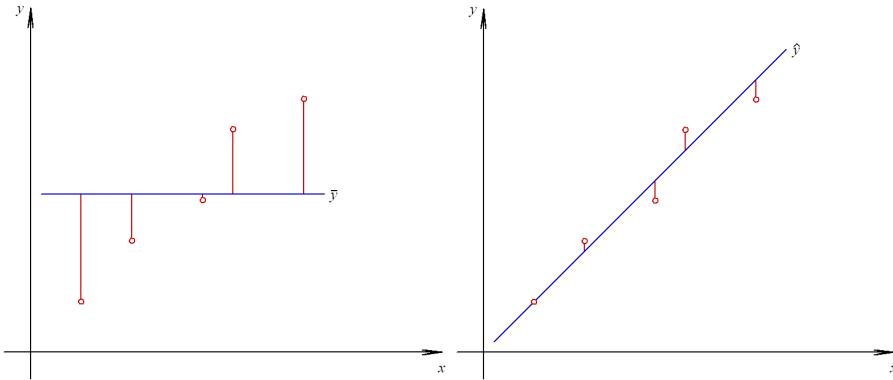
$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - f(x_i))^2$$

Značaj *MSE* je očigledan, pošto se radi o veličini koja se direktno minimizuje u mnogim regresionim problemima. *RMSE*, koren veličine *MSE* je nešto korisniji, pošto je ova mera izražena na istoj skali na kojoj i ciljna promenljiva, što olakšava interpretaciju dobijenih vrednosti. Zapravo, interpretabilnost je važna prednost ove mere. Ukoliko se vrši predviđanje troškova u nekom poslu i procenom kvaliteta modela dobije se $RMSE = 10.000$ u dinarima. Jasno je da su šanse male da model pogreši za milion dinara. Ako nam je greška reda veličine $10.000 - 30.000$ dinara prihvatljiva, možemo koristiti model.

Zbog jakog uticaja odudarajućih podataka na ovu grešku i zbog potrebe da se greška nekada izrazi u odnosu na vrednost ciljne promenljive, koristi se i srednja relativna greška, najčešće izražena u procentima, koja se definiše kao

$$MAPE = \frac{100}{N} \sum_{i=1}^N \left| \frac{y_i - f(x_i)}{y_i} \right|$$

Prethodne mere daju informaciju o redu veličine greške u apsolutnim jedinicama ili relativno u odnosu na vrednost ciljne promenljive. Drugi način evaluacije bi bio da se odmeri koliki je napredak ostvaren učenjem u odnosu na neki trivijalan prediktivni metod koji bi nam bio dostupan i bez učenja. Takođe prediktivnih metoda bi moglo biti raznih. Na primer, slučajno pogađanje. Međutim takav „metod“ nema veze sa podacima. Jedan koji ima, a ipak je trivijalan je uzimanje prosečne vrednosti ciljne promenljive na nekom uzorku



Slika 7.2: Greške kojima se definišu varijansa i koeficijent determinacije. Mogu se posmatrati kao promena promenljive y koju odgovarajući model nije uspeo da objasni promenom promenljive x .

za buduće konstantno predviđanje. Takav metod već ima smisla ne samo zbog jednostavnosti, već zato što pokušava da predviđa vrednost ciljne promenljive bez uspostavljanja veze sa vrednostima atributa, pa je zanimljivo videti koliko uspostavljanje takve veze pomaže u predviđanju u odnosu na metod koji takvu vezu ne koristi. Koeficijent determinacije, odnosno R^2 , nekog modela predstavlja ideo varijanse ciljne promenljive koji je objašnjen tim modelom. Šta to znači? Podsetimo se da modele mašinskog učenja, nalik fizičkim modelima, možemo razumeti kao pokušaj da se objasni variranje neke promenljive na osnovu promena u drugim promenljivim. U fizici, ukoliko dođe do promene temperature gasa, doći će i do promene pritiska. Drugim rečima, promena pritiska je objašnjena promenom temperature. Slično možemo posmatrati i modele mašinskog učenja. Udeo objašnjene varijanse ciljne promenljive se može izraziti preko udela preostale varijanse koju ne uspevamo da objasnimo modelom, što je zapravo MSE . Stoga važi

$$R^2 = 1 - \frac{MSE}{\text{var}[y]}$$

Na slici 7.2 prikazane su greške koje pravimo ukoliko za predviđanje koristimo prosek i greške koje pravimo ukoliko za predviđanje koristimo neki model. Srednjekvadratna greška u odnosu na prosek je $\text{var}[y]$, a u odnosu na model je MSE . R^2 upravo poredi ove greške.

7.2 Tehnike evaluacije i izbora modela

Tehnike izbora i evaluacije variraju po svojoj složenosti u zavisnosti od željene pouzdanosti ocene, količine podataka i svojstava algoritama učenja čiji se modeli evaluiraju. Koja god tehnika izbora ili evaluacije da se koristi, mora

biti ispoštovano glavno načelo evaluacije modela, a to je da **podaci korišćeni u evaluaciji modela ni na koji način ne smeju biti korišćeni prilikom njegovog obučavanja**. Iako deluje jednostavno, ne može se dovoljno naglasiti potreba za pažnjom pri sprovođenju ovog načela u delo. Naime, greške koje se prave u evaluaciji su najčešće ove prirode, a da ljudi koji evaluaciju vrše toga nisu ni svesni. Probleme evaluacije i izbora modela ćemo u nastavku prikazati u dva slučaja. Prvi je jednostavniji slučaj u kom je na raspolaaganju velika količina podataka, a drugi je slučaj male količine podataka koji zahteva nešto komplikovanije metode.

7.2.1 Izbor i evaluacija modela u slučaju dostupnosti velike količine podataka

Osnovni razlog za jednostavnost metoda izbora i evaluacije u slučaju dostupnosti velike količine podataka leži u prepostavci da je u tom slučaju podskup podataka koji bi se koristio za evaluaciju reprezentativan i da su ocene mera kvaliteta na njemu bliske svojim stvarnim vrednostima. U slučaju malih količina podataka, to ne smemo prepostaviti.

Prepostavimo za početak da algoritam nije konfigurabilan, odnosno da nije moguće birati vrednosti regularizacionog hiperparametra, vrstu kernela i slično. U tom slučaju izbor modela je trivijalan - ništa nije moguće birati.

Ključno pitanje evaluacije modela je kako ga evaluirati, a da se u evaluaciji ne koriste podaci na kojima je model obučavan. U slučaju nekonfigurabilnog algoritma i ovo je jednostavno – koristi se *evaluacija pomoću skupa za testiranje*. Ukupni podaci se dele na dva skupa – skup za obučavanje i skup za testiranje. Skup za obučavanje je veći – ugrubo od dve trećine do 90% (u zavisnosti od konteksta) i na njemu se obučava model koji se potom primenjuje na skup za testiranje, čime se dobijaju njegova predviđanja, koja se onda nekom merom kvaliteta mogu oceniti u odnosu na tačne vrednosti ciljne promenljive. Imajući u vidu da podataka ima puno, prepostavlja se da su ovakve ocene pouzdane. Podela na trening i test skup je ilustrovana tabelom 7.2.

Pitanje izbora test skupa je pipavo. Neke instance su izabrane da budu u skupu za obučavanje, a neke da budu u skupu za testiranje. Od načina na koji je izvršena podela zavisiće i ocena kvaliteta. Za različite podele, ova ocena možbiti različita. Pritom, greška ocene može biti vrlo velika ako se skup za testiranje pristrasno izabere. Na primer, ako sadrži samo instance jedne klase ili, u slučaju regresije, ako sadži samo instance sa najvišom (ili najnižom) vrednošću ciljne promenljive. Očigledno, treba voditi računa da se ne napravi ovakva podela. Više reči o tome biće nakon opisa svih tehnika, ali jednostavan način za izbegavanje ovih problema u slučaju velikih količina podataka je slučajni izbor skupa za testiranje.

Ipak, u praksi se nikad ne može govoriti o nekonfigurabilnim algoritmima. Algoritam učenja može biti konfigurabilan po različitim aspektima. Recimo, mogu imati hiperparametre poput regularizacionih hiperparametara, parametra tolerancije kod metoda potpornih vektora za regresiju, parametara kernela,

x_1	x_2	x_3	y
1	9	0	8
0	6	2	1
1	3	1	5
4	9	7	6
1	1	6	7
7	2	3	4
2	9	9	9
3	3	4	6
7	2	1	7
6	5	1	5

Tabela 7.2: Primer podele podataka na skup za obučavanje (plavo) i skup za testiranje (crveno).

itd. Takođe, izbor podskupa atributa nad kojim se uči predstavlja vid konfigurabilnosti. U slučaju neuronskih mreža, moguće je birati različite arhitekture koje se razlikuju po različitim aspektima. Čak i izbor algoritma (npr. neuronska mreža ili metod potpornih vektora) se može smatrati konfigurabilnošću procesa učenja. Skup izbora za sve ove aspekte učenja nazivaćemo konfiguracijom. Značaj konfigurabilnosti je u tome što za dati skup podataka, različite konfiguracije vode različitim modelima.

Postavlja se pitanje kako izabratи najbolju konfiguraciju, a time i model. Odgovor nije težak – za različite konfiguracije, potrebno je obučiti modele koji im odgovaraju, evaluirati ih, i izabratи najbolji, koji će biti korišćen u budućnosti. Ipak, pitanje evaluacije nije trivijalno i predstavlja jedan od najčešćih izvora grešaka u mašinskom učenju. Osnovna greška sastoji se u tome da se ocena kvaliteta dobijena prilikom izbora modela prijavi kao ocena kvaliteta tog modela. **To je pogrešno.** Naime, osnovno načelo izbora i evaluacije modela je da podaci korišćeni u evaluaciji modela ni na koji način ne smeju biti korišćeni prilikom njegovog obučavanja. Međutim, proces obučavanja uključuje sve korake dolaženja do modela. Kako izbor konfiguracije uslovjava izbor modela, i izbor konfiguracije je deo obučavanja modela. Kako je izbor konfiguracije rađen na osnovu svih podataka, to znači da je navedeno načelo prekršeno! Stoga su potrebne sofisticirane tehnike.

U slučaju velikih količina podataka rešenje se sastoji u malom unapređenju prethodno izložene tehnike evaluacije. Radi se o *evaluaciji pomoću skupova za validaciju i testiranje*. U njoj će umesto podele na dva skupa biti korišćenja podela na tri skupa. Ako je \mathcal{K} skup svih konfiguracija, tehnika se sprovodi kroz naredne korake:

- Podeliti skup podataka na skupove za obučavanje, validaciju i testiranje
- Za svaku konfiguraciju $K \in \mathcal{K}$

- Obučiti model na skupu za obučavanje
- Evaluirati ga na validacionom skupu
- Izabratи najbolji model od prethodno obučenih u dnosu na njegov kvalitet na validacionom skupu
- Testirati taj model na skupu za testiranje i koristiti kao finalni model

Na ovaj način izbegnuto je bilo kakvo odlučivanje o izboru finalnog modela na osnovu test skupa.

7.2.2 Izbor i evaluacija modela u slučaju dostupnosti male količine podataka

U slučaju dostupnosti male količine podataka, ne možemo se osloniti na pretpostavku da je podskup podataka reprezentativan i da su ocene kvaliteta modela pouzdane. Osnovna ideja kojom se zaobilazi ovaj problem je korišćenje svih podataka za evaluaciju. Naizgled, ova ideja je u direktnoj kontradikciji sa raniјe navedenim načelom da se podaci na kojima se vrši evaluacija modela ne smeju koristiti za njegovo obučavanje. Naime, neki podaci moraju biti korišćeni za obučavanje, a ako se svi koriste za evaluaciju, deluje da mora doći do preklapanja. Naravno, poenta je u inteligentnom dizajnu tehnike evaluacije kako bi se to ipak izbeglo.

Razmotrimo opet pojednostavljeni slučaj nekonfigurabilnog algoritma. Kao i pre, pitanje izbora modela je trivijalno. Bitno je samo pitanje evaluacije. Za to se u slučaju male količine podataka koristi tehnika *K-slojne unakrsne validacije* (eng. *K-fold cross-validation*). Sprovodi se na sledeći način:

- Podatke \mathcal{D} podeliti na K približno jednakih podskupova, takozvanih *slojeva* (eng. *folds*) $\{S_1, \dots, S_K\}$
- Za $i = 1, \dots, K$
 - Obučiti model na podacima $\mathcal{D} \setminus S_i$
 - Izvršiti predviđanja dobijenim modelom na sloju S_i
- Izračunati ocenu kvaliteta na osnovu svih predviđanja na celom skupu \mathcal{D}

Ilustracija podele podataka pri unakrsnoj validaciji data je tabelom 7.3.

Očigledno, nijedan od modela koji se obučava u unakrsnoj validaciji nije testiran na skupu na kom je obučavan, tako da je osnovno načelo evaluacije ispoštovano. Ipak, u unakrsnoj validaciji se ne obučava jedan model, već njih K . Najčešće postavljano pitanje je koji od ovih modela treba nadalje koristiti. Blisko tom pitanju je pitanje koji je to model čiji je kvalitet ocenjen finalnom ocenom kvaliteta. Odgovor na to pitanje je da se radi o modelu koji bi bio obučavan na svim podacima. On nije eksplicitno evaluiran, već je za ocenu njegovog kvaliteta uzeta ocena dobijena na osnovu K njemu sličnih modela

x_1	x_2	x_3	y
1	9	0	8
0	6	2	1
1	3	1	5
4	9	7	6
1	1	6	7
7	2	3	4
2	9	9	9
3	3	4	6
7	2	1	7
6	5	1	5

Tabela 7.3: Prikaz podele podataka pri unakrsnoj validaciji. Različite boje predstavljaju različite slojeve.

(sličnih jer su obučavani na približno istom skupu podataka). Onda je upravo to model koji se može nadalje koristiti, nakon što se obuči.

Napomenimo da se ocena kvaliteta računa tek nakon što su izračunata sva predviđanja. U praksi se često pravi greška da se mere kvaliteta računaju na svakom sloju pojedinačno, pa se na kraju uproseče. To može biti podjednako dobro u slučaju nekih mera poput MSE koje predstavljaju zbirove, ali u slučaju drugih, poput R^2 , ne.

Očigledna prednost unakrsne validacije u odnosu na prethodne metode je manja varijansa ocene greške usled toga što se ocena greške računa na većoj količini podataka. Drugim rečima, ocena je pouzdanija. S druge strane, model se ne obučava jednom, već K puta, što može biti vremenski zahtevno. Takođe, postavlja se pitanje i izbora broja K . U praksi se za K koriste vrednosti 5 i 10. U slučaju malih podataka, nekad se koristi vrednost $K = N$ (eng. *leave one out*). Ipak, teorijski je pokazano da ovaj pristup vodi optimističnoj proceni kvaliteta, pa se ne preporučuje. Unakrsnom validacijom nije rešen problem pristrasnog izbora podskupova, a o tome će biti reči kasnije.

Vratimo se sada realističnom pitanju izbora i evaluacije konfigurabilnog modela. Izbor se, kao što je i intuitivno, vrši tako što se isproba veći broj konfiguracija i izabere najbolja:

- Za svaku konfiguraciju $K \in \mathcal{K}$
 - Uraditi unakrsnu validaciju algoritma za konfiguraciju K i zapamtitи ocenu kvaliteta.
- Pomoću konfiguracije za koju je ocena kvaliteta najbolja, obučiti model na svim podacima \mathcal{D} .

Ipak, kako je model biran na svim podacima, ne može se verovati oceni kvaliteta koja je u ovom procesu dobijena, već je potrebno nezavisno uraditi evaluaciju

koja liči na evaluaciju pomoću validacionog i test skupa, ali je značajno komplikovanija jer se vrši u kontekstu unakrsne validacije. Takav postupak naziva se *ugnežđena¹ unakrsna validacija*. Sprovodi se na sledeći način:

- Podatke \mathcal{D} podeliti na K približno jednakih podskupova (tzv. slojeva) $\{S_1, \dots, S_K\}$
- Za $i = 1, \dots, K$
 - Izvršiti izbor modela pomoću unakrsne validacije na podacima $\mathcal{D} \setminus S_i$ i zapamtitи najbolju konfiguraciju
 - Pomoću te konfiguracije obučiti model na podacima $\mathcal{D} \setminus S_i$ i izvršiti predviđanje na podacima iz skupa S_i
- Izračunati ocenu kvaliteta na osnovu svih predviđanja na celom skupu \mathcal{D}

Ovako dobijena ocena kvaliteta predstavlja približnu ocenu kvaliteta modela koji se dobija prethodno opisanim postupkom izbora modela.

Ugnežđena unakrsna validacija u odnosu na evaluaciju pomoću skupova za validaciju i testiranje ima iste prednosti koje unakrsna validacija ima u odnosu na evaluaciju pomoću skupa za testiranje. Takođe, ima i iste mane. Pritom, one su još izraženije jer je broj obučavanja modela za K struku ugnežđenu unakrsnu validaciju K^2 . Tim pre se koriste male vrednosti parametra K , poput 5 ili 10.

7.3 Napomene vezane za pretpresiranje

Već je pomenut problem pristrasnog izbora skupa za testiranje. Unakrsna validacija ne rešava ovaj problem. Isto važi za ugnežđenu unakrsnu validaciju. Naime, ako se recimo u problemu regresije instance sortiraju po ciljnoj promenljivoj i onda u tom poretku podele na K slojeva, prvi sloj će sadržati instance sa najmanjim vrednostima ciljne promenljive. Takve vrednosti se ne nalaze u preostalih $K - 1$ slojeva i stoga model obučen na njima ekstrapolira kada se primeni na prvi sloj. U takvim slučajevima se ne očekuju dobre performanse. Razlog je što je raspodela na test skupu značajno različita od raspodele na skupu za obučavanje, pa se od algoritma učenja i ne može očekivati mnogo. Otud se prilikom bilo kakvih podela podataka često vodi računa o tome da svi podskupovi imaju sličnu raspodelu kao i ukupan skup podataka. Pritom, to je teško ili nemoguće postići, osim ako imamo puno podataka male dimenzionalnosti. U tom slučaju i slučajan uzorak često zadovoljava pomenute zahteve. Tehnike koje pokušavaju da zadovolje zahtev da podskupovi imaju istu raspodelu atrributa i ciljne promenljive nazivaju se tehnikama *stratifikacije*. Ovakve tehnike predstavljaju vid pretpresiranja podataka. Pojednostavljena varijanta, koju je uvek moguće izvesti je *stratifikacija po ciljnoj promenljivoj*. Podela na K delova stratifikovana po ciljnoj promenljivoj sprovodi se na sledeći način:

¹Da, možete proveriti u pravopisu da je ovo ispravan oblik ove reči. Alternativno, može i ugnježđena.

x_1	x_2	x_3	y
0	6	2	1
7	2	3	4
1	3	1	5
6	5	1	5
4	9	7	6
3	3	4	6
1	1	6	7
7	2	1	7
1	9	0	8
2	9	9	9

Tabela 7.4: Ilustracija podele podataka stratifikovane po ciljnoj promenljivoj. Različite boje predstavljaju različite podskupove.

- Sortirati instance u odnosu na ciljnu promenljivu. Ako je ciljna promenljiva kategorička, to se može uraditi tako što se svakoj njenoj vrednosti pridruži razičit broj.
- Za $i = 1, \dots, K$
 - Instance sa indeksima $i + j * K$ za $j = 0, 1, \dots$, svrstati u podskup P_i .

Na ovaj način svi podskupovi imaju približne raspodele ciljne promenljive. Ilustracija je data tabelom 7.4.

Česta greška u praksi mašinskog učenja je u lošoj primeni različitih vidova pretprocesiranja. Takva greška je primena standardizacije na ceo skup podataka, pre podele na podskupove koji se koriste u evaluaciji. Grešku je najlakše uočiti na primeru evaluacije pomoću skupa za testiranje. Prilikom standardizacije, vrednosti atributa na skupu za testiranje utiču na prosek i standardnu devijaciju koji se koriste pri standardizaciji. S druge strane, podaci na kojima će se ubuduće model primenjivati nisu dostupni da daju svoj doprinos ovim veličinama i mogu nas iznenaditi i dovesti do veće greške modela. Izbegavanje zajedničke standardizacije celog skupa podataka omogućava baš taj efekat – daje šansu podacima iz skupa za testiranje da nas iznenade i time ocenu greške modela učine realističnijom. Ova diskusija je data na primeru evaluacije pomoću skupa za testiranje i pretprocesiranju pomoću standardizacije. Ipak, ona je opštija i odnosi se i na ostale tehnike evaluacije i na druge tehnike pretprocesiranja, poput smanjenja dimenzionalnosti i izbora podskupa atributa.

Glava 8

Regularizacija

Značaj regularizacije je već naglašen u kontekstu smanjenja fleksibilnosti modela i predupređivanja preprilagođavanja. Iako je to najznačajnija uloga regularizacije, ipak nije jedina. Regularizacija nam može pomoći i u nametanju određene strukture modelu, što može biti vrlo korisno, kao što ćemo se uveriti uskoro, u uključivanju domenskog znanja u model i slično. U nastavku će biti diskutovano nekoliko poznatih i široko primenljivih vrsta regularizacije.

8.1 Proređeni modeli

Jedno specifično svojstvo modela mašinskog učenja na koje se obraća posebna pažnja je *proređenost modela* (eng. *model sparsity*). Model se smatra utoliko proredenijim ukoliko ima veći broj koeficijenata sa vrednošću nula. Ne-kada se za model samo kaže da je proređen, ali ne postoji definisan prag vezan za broj koeficijenata koji bi trebalo da imaju vrednost nula da bi se model smatrao proređenim. Stoga, ovakva kvalifikacija je neformalna, a u strogom smislu može se samo reći da je neki model proređeniji od drugog modela nad istim skupom atributa.

U slučaju linearnih modela, primetimo da proređenost modela¹ znači da je njegovim obučavanjem obavljen i posao *izbora atributa* (eng. *feature selection*), koji predstavlja čest posao u mašinskom učenju. Ovaj posao može biti obavljan odvojenim metodama, ali se obično najefikasnije obavlja ukoliko je tendencija ka proređenim modelima ugrađena u metod obučavanja.

Značaj proređenosti modela (pa time i izbora atributa) je višestruk. Prvo, ukoliko je neki koeficijent modela nula, to znači da postoji nešto u strukturi modela što nije bitno i što se ne mora izračunavati. U slučaju linearog modela, koeficijent sa vrednošću nula znači da neki atribut nije bitan, pa se ne mora ni meriti. Treba imati u vidu da merenje nekih atributa može biti skupo ili nepoželjno iz drugog razloga. Na primer, neke hemijske analize su skupe i poželjno je da model ne zavisi od njihovih rezultata, kako se ne bi vršile. Slično,

¹Evo prethodno pomenute neformalne upotrebe izraza proređenost!

neke medicinske analize su bilo skupe, bilo neugodne ili čak štetne za pacijente. Dalje, ukoliko je neki koeficijent nula, to obično znači da neka od zavisnosti koje model može da izrazi ne postoji, odnosno da je model jednostavniji, što je generalno poželjno svojstvo modela u kontekstu moći generalizacije. I konačno, model sa manje parametara je lakše analizirati i razumeti, odnosno proređeniji model je interpretabilniji.

Treba imati u vidu da se nekada za metod potpornih vektora kaže da daje proređene modele. Ti modeli su proređeni u smislu da ne zavise od svih instanci skupa podataka, već samo od nekih, pošto su Lagranžovi koeficijenti koji im odgovaraju jednaki nuli, ali to nije proređenost u smislu o kojem sada govorimo.

Proređeni modeli se često dobijaju tako što se neki metod učenja modifičuje posebnim vidom regularizacije – najčešće tako što se u minimizacionom problemu kao regularizacioni izraz upotrebi ℓ_1 norma:

$$\|w\|_1 = \sum_{i=1}^n |w_i|$$

Ovaj metod se naziva *lasso* (eng. *lasso – least absolute shrinkage and selection operator*) regularizacijom.

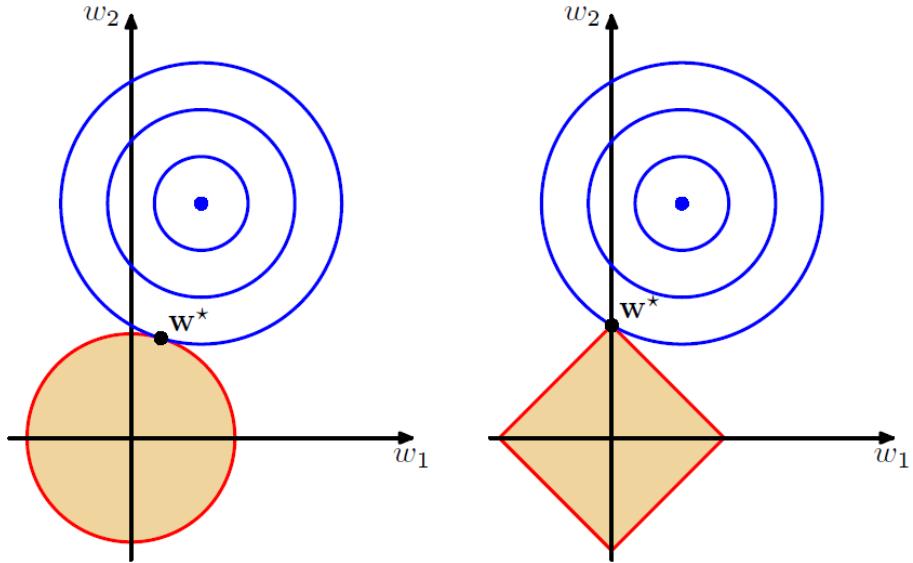
Ključna razlika ℓ_1 regularizacije u odnosu na ℓ_2 regularizaciju je da dok ℓ_2 regularizacija vodi smanjivanju apsolutnih vrednosti koeficijenata, često tako da veliki broj koeficijenata postane mali, ali i dalje različit od nule, ℓ_1 regularizacija vodi tome da neki, manje važni koeficijenti postanu baš jednaki nuli. U specijalnom slučaju linearne regresije u kojem za matricu podataka važi $X^T X = I$, mogu se preciznije okarakterisati efekti ove dve regularizacije. Naglasimo da dati uslov znači da su atributi normirani i nekorelirani vektori, kao i da postoje metode za njegovo obezbeđivanje (iako se to u praksi ne radi). Neka je w vektor vrednosti parametara modela koji se dobija bez regularizacije, neka je w' vektor vrednosti parametara koji se dobija pri ℓ_1 regularizaciji, a w'' vektor vrednosti parametara koji se dobijaju pri ℓ_2 regularizaciji. Tada važi:

$$w'_i = \text{sgn}(w_i) \max \left(|w_i| - \frac{\lambda}{2}, 0 \right) \quad w''_i = \frac{w_i}{1 + \lambda} \quad i = 1, \dots, n$$

gde je λ vrednost regularizacionog parametra. Iako prvi izraz na prvi pogled deluje komplikovano, on samo kaže da vrednost parametra u slučaju ℓ_1 regularizacije ostaje istog znaka, ali apsolutne vrednosti umanjene za λ . Ukoliko je λ dovoljno veliko da $|w_i| - \lambda/2$ postane negativno, taj koeficijent će imati vrednosti nula. Pod pomenutim specijalnim uslovima, odavde je jasno zašto lasso regularizacija daje proređene modele – koeficijent može postati baš nula ako se od njegove apsolutne vrednosti oduzima konačna vrednost, ali ne i ako se njegova apsolutna vrednost deli konačnom vrednošću. Ipak, intuicija iza ovih izraza nije očigledna. Stoga ćemo se osvrnuti na geometrijski prikaz.

Svaki problem oblika

$$\min_w E(w, \mathcal{D}) + \lambda \|w\|_q$$



Slika 8.1: ℓ_2 i ℓ_1 lopta (crveno) i konture srednje greške (plavo). Optimalni parametri u drugom slučaju leže na w_2 osi, što znači da je model 50% proređen.

za $q \in \mathbb{N}^+$ ekvivalentan je problemu oblika

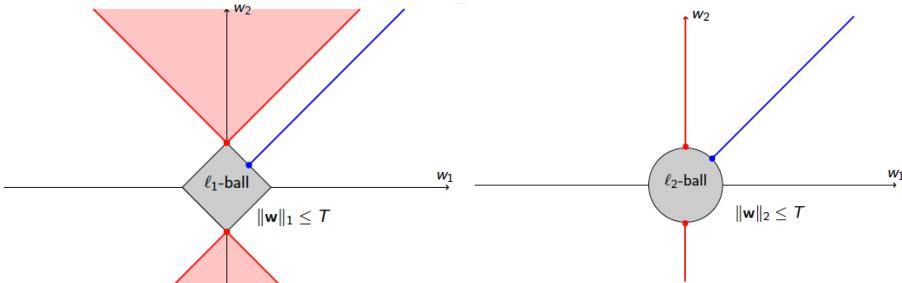
$$\min_w E(w, \mathcal{D})$$

$$\|w\|_q \leq t$$

odnosno, za svaku vrednost hiperparametra λ postoji neka vrednost hiperparametra t , tako da je rešenje oba problema isto i obratno. Na slici 8.1, prikazani su oblici lopti $\|w\|_q \leq t$ za $q = 1$ i $q = 2$ i konture jednakih vrednosti funkcije E . Kao što se sa slike vidi, zahvaljujući špicastom obliku ℓ_1 lopte, optimalno rešenje u slučaju ℓ_1 regularizacije ima 50% nenula parametara, dok optimalno rešenje u slučaju ℓ_2 lopte ima 100% nenula parametara. Treba primetiti i da je kvalitet rešenja u slučaju ℓ_1 regularizacije nešto gori. Uz dozu opreza, ova dva zapažanja se daju generalizovati. Modeli koji se dobijaju pri ℓ_2 regularizaciji obično nisu proređeni, ali često prave nešto manju grešku od modela koji se dobijaju pri ℓ_1 regularizaciji, dok su ovi drugi obično u manjoj ili većoj meri proređeni.

Na slici 8.2 prikazani su regioni u kojima će ℓ_1 i ℓ_2 regularizacije proizvesti proređene modele za koje važi $w_1 = 0$, ukoliko je minimum funkcije E sa kružnim konturama u njima. Očigledno, ℓ_2 regularizacija će proizvesti proređeni model samo ukoliko je to baš optimalan model. U svim drugim slučajevima koeficijenti će biti nenula.

Još jedan pogled koji doprinosi razumevanju zašto ℓ_1 regularizacija vodi proređenim modelima, a ℓ_2 ne, je vezan za gradiente ovih normi. Optimiza-



Slika 8.2: Regioni (crveno) u kojima će ℓ_1 i ℓ_2 regularizacija proizvesti model za koji je $w_1 = 0$.

cioni metodi obično počivaju na uzastopnom umanjivanju tekućih parametara modela za neki vektor proporcionalan gradijentu ciljne funkcije. U slučaju ℓ_1 regularizacije, doprinos regularizacionog izraza gradijentu ciljne funkcije je

$$\frac{\partial \|\mathbf{w}\|_1}{\partial w_i} = \text{sgn}(w_i)$$

kad god važi $w_i \neq 0$. U slučaju ℓ_2 regularizacije važi

$$\frac{\partial \|\mathbf{w}\|_2^2}{\partial w_i} = 2w_i$$

Očito, kolika god da je vrednost parametra w_i , ℓ_1 regularizacija podjednako doprinosi smanjenju apsolutne vrednosti koeficijenta w_i . S druge strane, ℓ_2 regularizacija doprinosi utoliko manje, što je vrednost parametra manja, tako da kako se vrednost bliži nuli, regularizacija sve manje pomaže u daljem smanjenju.

Očigledan problem vezan za ℓ_1 regularizaciju je njena nediferencijabilnost. Nediferencijabilnost se u mašinskom učenju često ignoriše, pošto u nekim problemima nije mnogo verovatno da će optimizacioni algoritam zasnovan na gradijentima naleteti na tačku u kojoj je funkcija nediferencijabilna. Čak i ako naleti i preduzme korak u pogrešnom pravcu, taj korak će u mnogim slučajevima biti kompenzovan daljim tokom optimizacije. Ipak, u slučaju ℓ_1 regularizacije, ovaj problem se ne može ignorisati. Naime, tačka u kojoj je funkcija nediferencijabilna je baš tačka rešenja kojoj se teži, a ne neka tačka na koju optimizacija može nabasati, a u koju se ne mora vraćati. Otud se uz ovu vrstu regularizacije koriste i posebni optimizacioni algoritmi.

Još jedna mana laso regularizacija je ta što vodi nestabilnim rešenjima. Naime, ukoliko postoje dva jako korelirana atributa, laso regularizacija će verovatno iz modela isključiti jedan od tih atributa. Kako su atributi jako korelirani, nije velika razlika da li će biti izbačen jedan ili drugi i lako se može desiti da male promene u podacima vode isključivanju različitih atributa. Ovo očito predstavlja problem za interpretaciju modela. Jedno rešenje ovog problema je

korišćenje *elastične mreže* (eng. *elastic net*), što je regularizacija kojoj odgovara izraz

$$\Omega(w) = \mu\|w\|_1 + (1 - \mu)\|w\|_2^2$$

pri čemu važi $\mu \in [0, 1]$.

8.2 Modeli složenije strukture i uključivanje domenskog znanja

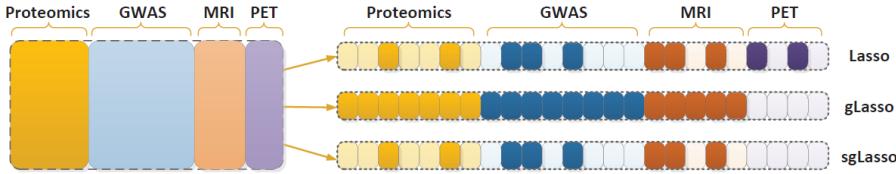
Neka je potrebno predvideti prinos pšenice na jednom podneblju na osnovu nekih atributa, a poznat je model na nekom drugom podneblju. Modeli verovatno neće biti isti, ali mogu biti vrlo slični. Posebno ukoliko je količina podataka na kojima je potrebno odrediti model mala, oslanjanje na već poznati model može biti vrlo korisno. Ovo predstavlja vid uključivanja prethodnog znanja u proces učenja. Dodatno, ne mora ni biti poznat takav model. Moguće je da ekspert za dati domen ima prepostavke o tome kako neki atribut utiče na ciljnu promenljivu. Ukoliko bi mogao ugrubo da kvantifikuje taj uticaj i tačka informacija bi mogla biti korisnija nego nikakva. Postavlja se pitanje kako ukljuciti takvo domensko znanje u model. Jedan jednostavan način je sledeći.

Neka G označava grupu atributa i neka w_G označava vektor vrednosti parametara koji odgovaraju tim atributima. Ukoliko je za grupu nekih atributa G ugrubo poznato kako utiču na ciljnu promenljivu, proces učenja može biti unapređen uključivanjem tog domenskog znanja. Neka je w'_G vektor pretpostavljenih vrednosti parametara za attribute iz grupe G . Tada se te vrednosti mogu upotrebiti za konstrukciju sledećeg regularizacionog izraza:

$$\Omega(w) = \|w_G - w'_G\|_2^2$$

Pritom, moguće je u ciljnu funkciju dodati i druge vrste regularizacije zajedno sa ovom. Očigledan je problem kvaliteta pretpostavljenih vrednosti w'_G , ali to nije suštinski problem. Ukoliko je pretpostavka loša, prilikom izbora najbolje konfiguracije biće izabrana vrednost koeficijenta λ koja je mala ili jednaka nuli. To je istovremeno i indikacija kvaliteta same pretpostavke ili, u kontekstu problema predviđanja prinosa pšenice, sličnosti zakonitosti između ciljne promenljive i atributa na dva različita podneblja.

Jedna vrlo korisna vrsta regularizacije je *grupna lasso regularizacija* (eng. *group lasso*). Vrlo često, atributi se mogu grupisati prema nekom kriterijumu. U medicinskim primenama, to recimo mogu biti merenja na osnovu uzorka krvi, rendgenskog snimka, snimanja magnetnom rezonancicom, biopsije i slično. Neke od ovih analiza su neprijatne, neke skupe, na neke se može dugo čekati i slično. Već je istaknuto da je kvalitet proređenih modela što omogućavaju da se merenja koja odgovaraju određenim atributima ne vrše. Ipak, zamislimo da ℓ_1 regularizacija pridruži koeficijent 0 merenju hemoglobina, ali nenula koeficijent merenju triglicerida u krvi. To što nije potrebno meriti hemoglobin, ipak ne znači mnogo, pošto je svejedno potrebno da pacijent da krv i da se izvrše



Slika 8.3: Ilustracija dejstva obične laso regularizacije (desno gore), grupne laso regularizacije (desno u sredini) u skladu sa definisanim grupama (levo) i proređene grupne laso regularizacije (desno dole).

neke analize. Suštinski dobitak bi bio da nijednu od analiza krvi nije potrebno raditi. U suprotnom, u ovom slučaju, mogu se uraditi sve. Slično, ukoliko se uradi snimanje magnetnom rezonancom, nebitno je da li se računaju neki atributi snimka ili svi. Grupna laso regularizacija rešava ovaj problem. Neka je dat skup grupa $\{G_1, \dots, G_M\}$ atributa koje mogu, ali ne moraju biti disjunktnе. Grupna laso regularizacija se vrši upotrebot regularizacionog izraza

$$\Omega(w) = \sum_{i=1}^M \|w_{G_i}\|_2$$

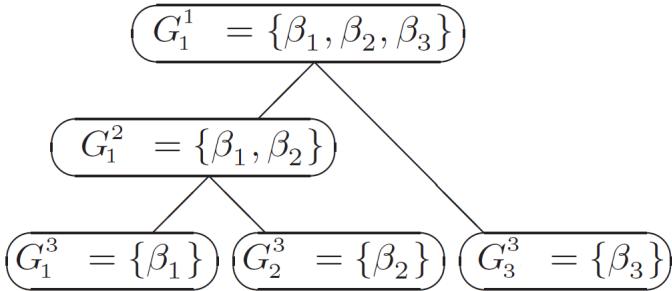
Primetimo da norma nije kvadrirana. Ovakva regularizacija teži tome da anulira norme pojedinačnih grupa, a anuliranjem normi grupa anuliraju se svi koeficijenti u grupi. Pored anuliranja celih grupa, i dalje može biti poželjno da se i u okviru relevantnih grupa model što više proredi. U tom slučaju moguće je koristiti kombinaciju grupne i obične laso regularizacije:

$$\Omega(w) = \mu\|w\|_1 + (1 - \mu) \sum_{i=1}^M \|w_{G_i}\|_2$$

za neko $\mu \in [0, 1]$. Ovaj vid grupne laso regularizacije naziva se *proređena grupna laso* (eng. *sparse group lasso*) regulizacija. Slika 8.3 ilustruje ove vidove regularizacije.

Još jedan kontekst u kojem je grupna laso regularizacija vrlo korisna je postojanje kategoričkih atributa. Kao što je već rečeno kategorički atribut se najčešće uključuje u model tako što se kodira pomoću više binarnih promenljivih. Isključivanje samo nekih od tih binarnih promenljivih neće omogućiti potpuno izbacivanje atributa iz modela i eliminisati potrebu za njegovim menjanjem. S druge strane, korišćenje grupne laso regularizacije, vodiće tome da su sve binarne promenljive uključene u model ili da su sve isključene iz njega.

Još jedan vid regularizacije koja daje specifičnu strukturu modela, a i uključivanje domenskog znanja je *hijerarhijska laso regularizacija* (eng. *tree group lasso*), pri čemu pretpostavljamo da je hijerarhija data stablom u čijim su listovima (i samo u listovima) atributi. Ovakve hijerarhije nisu neuobičajene.



Slika 8.4: Organizacija grupa atributa u vidu stabla.

Recimo, medicinske dijagnoze se prirodno organizuju u stabla, pri čemu u korenu stabla može biti najopštija dijagnoza *bolest*, na nešto nižim nivoima mogu biti recimo grupe dijagnoza *bolest pluća*, *bolest srca* i tako dalje, dok u listovima mogu biti konkretnе dijagnoze poput *grip*, *upala pluća*, *artritis* i tako dalje. U ovom slučaju regularizacioni izraz ima istu formu kao i izraz za grupnu lasku regularizaciju. Hijerarhijski efekat se postiže tako što svakom čvoru stabla odgovara grupa svih atributa koji su u listovima potomcima datog čvora, kao što je prikazano slikom 8.4. Na taj način, ukoliko se regularizacijom iz modela eliminisu plućne bolesti, automatski su eliminisane i sve pojedinačne plućne bolesti.

8.3 Učenje više poslova odjednom

Kao što je već diskutovano u prethodnom odeljku, nekada se nedovoljna količina podataka može nadomestiti jačim prepostavkama o modelu. U primeru vezanom za prinos pšenice, prepostavka je bila vezana za sličnost sa modelom sa nekog drugog podneblja. Scenario koji će biti diskutovan u nastavku možda ne deluje srođno, ali se zapravo oslanja na slične principe.

Nekada podaci na kojima je potrebno uraditi obučavanje mogu biti podeljeni na nekoliko grupa koje uprkos istim atributima i ciljnoj promenljivoj mogu dolaziti iz nešto različitih raspodela. Na primer, problemi predviđanja roda pšenice u Srbiji, Španiji i Australiji sigurno imaju nešto zajedničko, čak toliko da se mogu smatrati jednim problemom, ali postoje i razlike usled značajno različitih klimatskih, zemljишnih i drugih svojstava ta tri podneblja. Slično, predviđanje budućeg stanja pacijenata (na primer, da li će lečenje uspeti ili će pacijentima ponovo biti potreban tretman) u različitim bolnicama, čak i ako su bolnice iste specijalnosti, ima sličnosti, ali može biti i razlika. Na primer, primena pravih procedura će sigurno voditi boljim rezultatima kod pacijenata iz svih bolnica, ali će pacijenti iz bolnice u rudarskom gradu verovatno imati sporiji oporavak, nego pacijenti iz bolnice u centru velikog grada. Razlog za to je prostо što pacijenti iz prve bolnice u proseku verovatno imaju nepo-

voljnije životne uslove, što se odražava i na njihovo zdravstveno stanje. Kako modelujemo ovakve probleme? Ukoliko je količina podataka velika, verovatno ima smisla napraviti odvojene modele za svaki od potproblema. Ukoliko bismo spojili sve podatke i napravili jedan model, moguće je da bi se različite zakonitosti koje važe u različitim potproblemima uprosećile i da bi ukupni model izgubio na tačnosti. Ipak, često podataka nema dovoljno. Često je raspodela broja podataka neravnomerna. Neki potproblemi mogu imati veliku količinu podataka, a neki malu. Ukoliko bi se svi podaci spojili, specifičnosti potproblema sa malom količinom podataka bi verovatno bile zanemarene u korist specifičnosti potproblema za koje su dostupne velike količine podatka. S druge strane, učenje ne bi bilo uspešno ni u slučaju odvojenog obučavanja, pošto učenje na malom skupu podataka lako može voditi nepouzdanom modelu. Ovo su tipična pitanja *učenja više poslova odjednom* (eng. *multitask learning*). Postoje različite postavke ovakve vrste problema, a i različiti pristupi njihovog rešavanja. Otud narednu formulaciju i diskusiju koja je prati ne treba smatrati najopštijim slučajem.

Pretpostavimo da se skup podataka \mathcal{D} može predstaviti kao unija disjunktivnih skupova \mathcal{D}_i za $i = 1, \dots, T$, gde je T broj različitih, ali srodnih poslova. Svi podaci imaju iste atribute i istu ciljnu promenljivu. Neka je W matrica dimenzija $n \times T$, čije su kolone vektori W_1, \dots, W_T . Jedan jednostavan način zajedničkog obučavanja modela za sve poslove je sledeći:

$$\min_W \sum_{i=1}^T E(W_i, \mathcal{D}_i) + \lambda \sum_{i=1}^T \|W_i - \bar{W}\|_2^2$$

pri čemu važi

$$\bar{W} = \frac{1}{T} \sum_{i=1}^T W_i$$

Drugim rečima, potrebno je minimizovati grešku pri specifičnoj regularizaciji, koja sugerira da nijedan od modela ne treba daleko da odstupi od proseka svih modela. Može se pokazati da je ovo ekvivalentno formulaciji

$$\min_W \sum_{i=1}^T E(W_i, \mathcal{D}_i) + \lambda \sum_{i=1}^{T-1} \sum_{j=i+1}^T \|W_i - W_j\|_2^2$$

u kojoj regularizacija sugerira da svaki model treba da bude blizak svakom drugom. Na ovaj način, vrši se nagodba između minimizacije greške i sličnosti modela, odnosno, svaki model može odstupiti od drugih, ali samo ako to značajno doprinosi smenjenju greške. Na taj način, model koji odgovara poslu sa manjom količinom podataka, može se osloniti na modele sa većom količinom podataka, ali u meri u kojoj mu to potreba za smanjenjem greške dopušta. Kao i obično, relativni značaj ovih faktora se vaga izborom vrednosti parametra λ .

Prethodni model pretpostavlja da su naša apriorna uverenja o sličnosti među različitim poslovima jednaka. Ipak, nekada možemo smatrati da su neka

dva posla sličnija od neka druga dva. U opštem slučaju, moguće je konstruisati graf zavisnosti među poslovima i sličnost poslova i i j kvantifikovati težinama grana grafa α_{ij} i rešiti sledeći problem:

$$\min_W \sum_{i=1}^T E(W_i, \mathcal{D}_i) + \lambda \sum_{i=1}^{T-1} \sum_{j=i+1}^T \alpha_{ij} \|W_i - W_j\|_2^2$$

Nekada se želi da dobijeni modeli budu proređeni. Tada se mogu dodati ℓ_1 regularizacije za svaki od modela. U takvom slučaju bi bilo korisni da svi modeli budu proređeni na isti način. Odnosno, da ako kod jednog pacijenta ne vršimo neku analizu, to ne radimo ni kod drugog. Ovaj problem je već malo komplikovaniji. Neka je $\ell_{p,q}$ norma matrice definisana na sledeći način

$$\|W\|_{p,q} = \left(\sum_{i=1}^n \|W^i\|_p^q \right)^{\frac{1}{q}}$$

gde W^i označava i -tu vrstu matrice W . Tada se rešavanjem sledećeg optimizacionog problema dobija željeno svojstvo jednake proređenosti više modela istovremeno:

$$\min_W \sum_{i=1}^T E(W_i, \mathcal{D}_i) + \lambda \sum_{i=1}^T \|W_i - \bar{W}\|_2^2 + \rho \|W\|_{2,1}$$

Naime, time što će cela ℓ_2 norma vrste W^i biti svedena na nulu, nijedan model neće uključivati merenja odgovarajućeg atributa. Ova regularizacija može pomoći i da se model multinomijalne logističke regresije učini interpretabilnijim.

Glava 9

Optimizacija

Matematička optimizacija se bavi metodama pronalaženja minimuma i maksimuma funkcija. Opšti *problem optimizacije* je obično oblika:

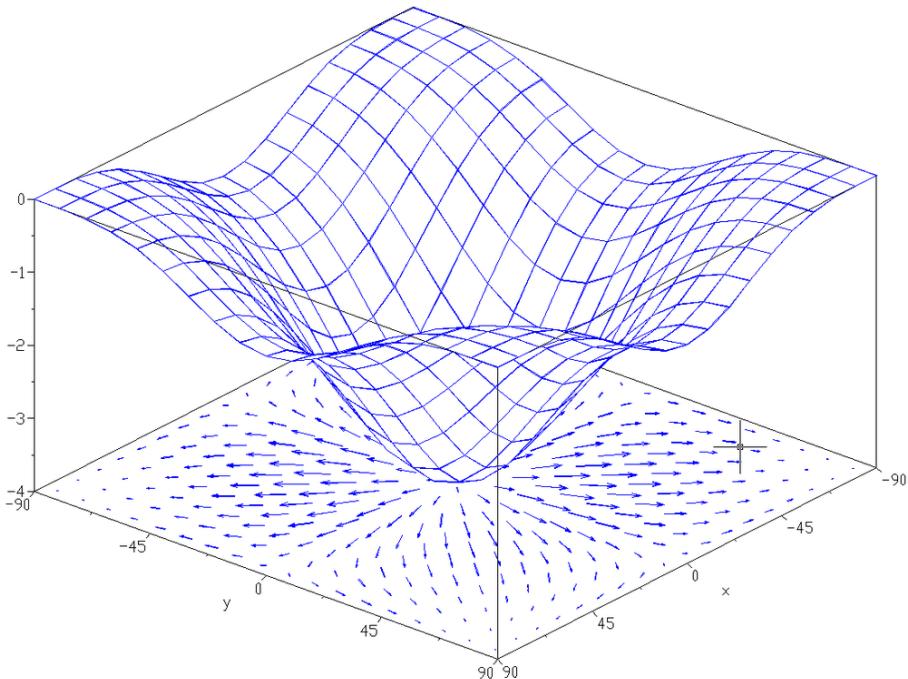
$$\begin{aligned} & \min_{x \in \mathcal{D}} f(x) \\ & \text{pri uslovima } g_i(x) \leq 0 \quad i = 1, \dots, L \end{aligned}$$

pri čemu se funkcija f naziva *ciljnom funkcijom*, skup \mathcal{D} domenom, a uslovi vezani za g_i , takođe i *ograničenjima*. Objekat iz domena koji zadovoljava sva ograničenja, naziva se *dopustivo rešenje*. Potrebno je među svim dopustivim rešenjima naći ono za koje je vrednost ciljne funkcije najmanja. Ova formulacija obuhvata i pronalaženje maksimuma, pošto se pronalaženje maksimuma funkcije f može svesti na pronalaženje minimuma funkcije $-f$. Zato će u nastavku biti reči isključivo o metodama pronalaženja minimuma, odnosno *minimizacije*. Takođe, treba primetiti da se i jednakosna ograničenja lako uklapaju u navedeni okvir. Naime, ograničenje $g(x) = 0$ se može predstaviti pomoću dva ograničenja: $g(x) \leq 0$ i $-g(x) \leq 0$.

U mašinskom učenju se koriste najrazličitije varijante ove opšte forme. Ipak, najčešće se sreću problemi bez ograničenja ili problemi koji se lako mogu transformisati u takve probleme, pa će u nastavku biti diskutovani algoritmi za optimizaciju tog tipa.

9.1 Gradijentni spust

Najjednostavnija i najpoznatija metoda optimizacije prvog reda za diferencijabilne funkcije je *gradijentni spust* (eng. gradient descent). Ova metoda, kao i većina metoda optimizacije, zasniva se na postepenom, iterativnom, približavanju rešenju problema. Gradijent ukazuje na pravac najbržeg uspona, što je ilustrovano slikom 9.1. Stoga, negativna vrednost gradijenta ukazuje na pravac najbržeg spusta. Osnovna ideja gradijentnog spusta je da se, polazeći od



Slika 9.1: Gradijenti funkcije u različitim tačkama

neke nasumice izabrane tačke, nizom koraka u pravcu gradijenta dođe vrlo blizu rešenju. Ako je polazna tačka x_0 , svaka naredna se dobija primenom pravila

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

U vezi sa ovakvim pristupom, postavlja se više pitanja. Prvo je kako se bira dužina koraka α_k koji se preduzima u pravcu suprotnom gradijentu. Postoje različiti pristupi. Jedan jednostavan izbor je korišćenje konstantne vrednosti koraka $\alpha_k = \alpha$, za neko α , za svako i . Drugi pristup je oslanjanje na Robins-Monroove uslove

$$\sum_{k=1}^{\infty} \alpha_k = \infty \quad \sum_{k=1}^{\infty} \alpha_k^2 < \infty$$

Intuitivno, smisao prvog uslova je da su koraci dovoljno veliki da se može dostići rešenje problema. Smisao drugog uslova je da su koraci dovoljno mali da niz tačaka x_k konvergira rešenju, umesto da osciluje. Jedan od izbora koji zadovoljava ove uslove je $\alpha_k = \frac{1}{k}$. Pored ovih pristupa, postoje i drugi. Drugo pitanje je kada se staje sa izračunavanjem. Kriterijuma zaustavljanja koji se u praksi koriste ima više. Najčešći su zaustavljanje nakon unapred zadatog broja iteracija, nakon što razlika između susednih koraka $\|x_{k+1} - x_k\|$ postane manja od unapred zadate vrednosti ε , nakon što razlika između vrednosti funkcije

u susednim koracima $|f(x_{k+1}) - f(x_k)|$ postane manja od ε ili nakon što ova razlika u odnosu na polaznu vrednost funkcije $|f(x_{k+1}) - f(x_k)|/|f(x_0)|$ postane manja od ε . Moguće je kombinovati i više ovakvih kriterijuma.

U slučaju konstantnog koraka, moguće je dokazati konvergenciju metoda ka pravom rešenju, ali tek sa određenom nesavladivom greškom, koja je utoliko veća ukoliko je veličina koraka veća. U slučaju oslanjanja na Robins-Monroove uslove, za konveksne funkcije sa Lipšic neprekidnim¹ gradijentom, greška metode $\|x_k - x^*\|$ u koraku k , gde je x^* tačka minimuma, je reda $O\left(\frac{1}{k}\right)$, što očito implicira konvergenciju. Za jako konveksne funkcije sa Lipšic neprekidnim gradijentom, greška je reda $O(c^k)$ za neko $0 < c < 1$. U slučaju nekonveksnih funkcija, gradijentni spust i njegove varijante prikazane u nastavku konvergiraju, ali navedene brzine konvergencije ne važe. Konvergencija gradijentnog spusta se smatra relativno sporom. Razmotrimo realističnost jake konveksnosti.

Primer 5 *Funkcija greške u problemu linearne regresije $\|Xw - y\|_2^2$ je konveksna. Naime,*

$$(Xw - y)^T(Xw - y) = w^T X^T X w - w^T X^T y - y^T X w + y^T y$$

Matrice oblika $X^T X$ su uvek pozitivno semidefinitne. Pošto je matrica $X^T X$ hesijan funkcije $w^T X^T X w$, onda je ona konveksna funkcija. Ostale funkcije su linearne, pa otud i konveksne. Zbir konveksnih funkcija je konveksna funkcija.

Srednja greška u problemu regularizovane linearne regresije

$$\|Xw - y\|_2^2 + \lambda \|w\|_2^2$$

je jako konveksna. Naime,

$$(Xw - y)^T(Xw - y) + \lambda w^T w = w^T X^T X w - w^T X^T y - y^T X w + y^T y + \lambda w^T w$$

Hesijan ove funkcije je

$$X^T X + \lambda I$$

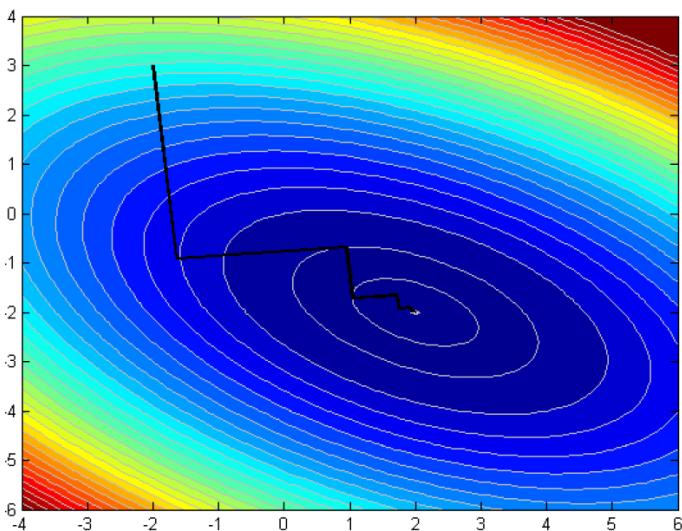
Znamo da je funkcija f jako konveksna ukoliko je matrica $\nabla^2 f(x) - mI$ pozitivno semidefinitna. Za $m = \lambda$, dobija se

$$X^T X + \lambda I - \lambda I = X^T X$$

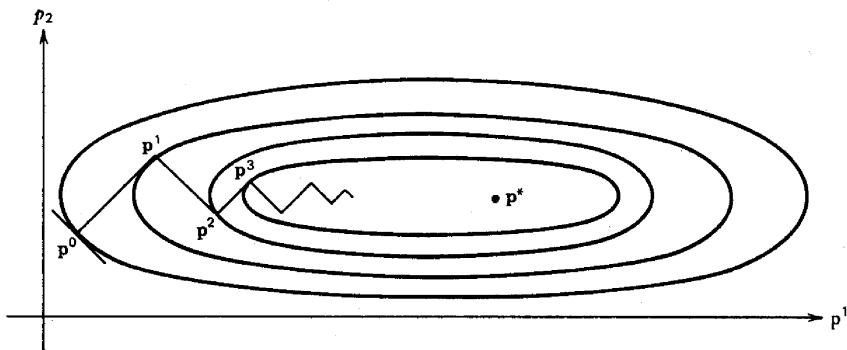
što je pozitivno semidefinitna matrica. Stoga, polazna funkcija mora biti jako konveksna. Otud se zaključuje da ℓ_2 regularizacija ne utiče pozitivno samo na prediktivne performanse dobijenog modela, već i na brzinu optimizacije.

Poznato je da je gradijent u svakoj tački normalan na konturu (poput izohipse na geografskoj karti) funkcije sa istom vrednošću koju funkcija ima u toj tački. Ovo ponašanje je ilustrovano slikom 9.2. Imajući ovo u vidu, ne čudi

¹Funkcija f je Lipšic neprekidna ako postoji konstanta L , takva da za svake dve tačke x i y važi $|f(x) - f(y)| \leq L\|x - y\|$.



Slika 9.2: Pravac gradijenta u nekoj tački je normalan na odgovarajuću konturu funkcije.



Slika 9.3: Ponašanje gradijentnog spusta u slučaju funkcije sa izduženim konturnama.

da se gradijentni spust ne ponaša dobro u slučajevima funkcija čije su konture izdužene, kao na slici 9.3. U takvim situacijama, gradijentni spust bira tačke koje leže duž cik-cak putanje ka minimumu i broj koraka do zadovoljavajućeg rešenja može biti veliki. Očito, pravac najbržeg spusta uopšte ne mora biti pravac najbržeg kretanja ka minimumu.

U sumi, prednosti metode gradijentnog spusta su njena jednostavnost i široki uslovi primenljivosti, a mane su spora konvergencija, to što je izabrani pravac samo lokalno optimalan, što dodatno usporava konvergenciju cik-cak

kretanjem i to što se u mnogim slučajevima za izračunavanje tog neoptimalnog pravca troši puno vremena.

9.2 Metod inercije

Kao što je rečeno, pri gradijentnom spustu gradijent u nekim situacijama naglo menja pravac, što dovodi do cik-cak kretanja i sporije konvergencije. *Metod inercije* se zasniva na ideji akumuliranja prethodnih gradijenata, pri čemu je značaj starijih gradijenata manji, a novijih veći, a onda se umesto gradijenta u dатој тачки koristi ukupan akumulirani gradijent. Kako prosek nekih vrednosti, manje varira nego same vrednosti, ovakva tehnika dovodi do manjih promena pravca u gradijentu i često do povećanja brzine konvergencije. Metod inercije je definisan na sledeći način:

$$d_0 = 0$$

$$d_{k+1} = \beta_k d_k + \alpha_k \nabla f(x_k)$$

$$x_{k+1} = x_k - d_{k+1}$$

pri čemu važi $0 \leq \beta_k < 1$. U vektoru d_k se akumuliraju gradijenti prvih k koraka. Pritom, kako se d_k u svakoj iteraciji množi brojem manjim od 1, uticaj ranijih gradijenata eksponencijalno brzo opada, tako da skoriji gradijenti dosta više utiču na pravac koraka. Ovaj metod se često koristi za obučavanje neuronskih mreža.

9.3 Nesterovljev ubrzani gradijentni spust

Nesterovljev ubrzani gradijentni spust je modifikacija metoda inercije, koja predstavlja asimptotski optimalan algoritam prvog reda za konveksne funkcije. Ukoliko je funkcija konveksna sa Lipšic neprekidnim gradijentom, greška je reda $O\left(\frac{1}{k^2}\right)$, naspram $O\left(\frac{1}{k}\right)$ u slučaju običnog gradijentnog spusta, pod istim uslovima.

$$d_0 = 0$$

$$d_{k+1} = \beta_k d_k + \alpha_k \nabla f(x_k - \beta_k d_k)$$

$$x_{k+1} = x_k - d_{k+1}$$

Algoritam definiše specifičan izbor vrednosti α_k i β_k , ali o njemu neće biti reči. Ovaj algoritam je posebno pogodan u slučaju podataka visoke dimenzionalnosti. Naime, u tom slučaju je teško primeniti metode drugog reda, zbog toga što veličina hesijana može biti ogromna. Zbog svoje brzine, Nesterovljev algoritam je tada najbolja alternativa. I on se često koristi u obučavanju neuronskih mreža.

9.4 Adam

Najkorišćeniji algoritam za obučavanje neuronskih mreža je Adam (eng. adaptive moment estimation). Zasniva se na ocenama prvog i drugog momenta gradijenata koje su date narednim formulama:

$$m_0 = 0$$

$$v_0 = 0$$

$$m_{k+1} = \beta_1 m_k + (1 - \beta_1) \nabla f(x_k)$$

$$v_{k+1} = \beta_2 v_k + (1 - \beta_2) \nabla f(x_k) \odot \nabla f(x_k)$$

Očito, ocena prvog momenta m_{k+1} predstavlja akumulirani pravac kretanja, kao što je bio slučaj i sa prethodnim algoritmima. Ocena drugog momenta slično akumulira kvadrat norme gradijenta. Kako ove procene uključuju proizvoljne težinske koeficijente β_1 i β_2 , pristrasne su ka nuli i potrebno ih je korigovati:

$$\hat{m}_{k+1} = m_{k+1} / (1 - \beta_1^{k+1})$$

$$\hat{v}_{k+1} = v_{k+1} / (1 - \beta_2^{k+1})$$

Ažuriranje parametara vrši se na sledeći način:

$$x_{k+1} = x_k - \alpha_{k+1} \frac{\hat{m}_{k+1}}{\sqrt{\hat{v}_{k+1}} + \varepsilon}$$

pri čemu sabiranje vektora $\hat{v}_k + 1$ i skalara ε predstavlja dodavanje tog skalara na sve koordinate vektora i pri čemu je deljenje pokoordinatno.

Suština algoritma je da dužina koraka koji se preduzima zavisi od svojstava tekućeg regiona u kojem se funkcija optimizuje. Ukoliko je tekući korak preveliki u odnosu na širinu minimuma, tipično dolazi do oscilovanja optimizacionog procesa preko minimuma. To dovodi do čestih promena pravca, usled čega dolazi do poništavanja gradijenata pri njihovoј akumulaciji, pa se prvi moment, a time i korak, smanjuje i optimizacioni proces lakše silazi ka minimumu. U slučaju stabilnog kretanja niz neku nizbrdicu, zbog nedostatka promena pravca, pri oceni prvog momenta nema poništavanja, pa je ta ocena velika, a time i korak gradijenta. Koja je uloga normiranja ocenom drugog momenta? Naime, vrednost prvog momenta nekad nije mala zbog poništavnja usled promena pravca, već prosti zato što je norma gradijenta mala, iako je pravac stabilan. U takvoj situaciji deljenje malom ocenom drugog momenta vodi ubrzavanju kretanja što je uvek poželjno ako nema promena pravca. U slučaju velikih gradijenata, čak i kad se osciluje, rezultujuća vrednost može biti velika. Deljenje velikom ocenom drugog momenta vodi smanjenju koraka i bržem zaustavljanju oscilacija.

Jedno važno svojstvo algoritma je da se zahvaljujući tome što su veličine m_k i v_k vektori i tome što se računske operacije ažuriranja vektora x_k izvode

pokoordinatno, svaka njegova koordinata ima zasebnu promenljivu veličinu koraka. Ovo je od izuzetnog značaja za obučavanje neuronskih mreža čije konture mogu biti vrlo izdužene, pa otud jednake dužine koraka po svim dimenzijama nisu odgovarajuće.

9.5 Stohastički gradijentni spust

Jedna, vrlo široko primenjena, modifikacija gradijentnog spusta je *stohastički gradijentni spust* (eng. stochastic gradient descent). Analogna modifikacija se može primeniti i na druge diskutovane metode. Stohastički gradijentni spust se intenzivno primenjuje u obučavanju modela mašinskog učenja sa velikim količinama podataka. Modifikacija se sastoji u tome da je umesto gradijenta dovoljno koristiti neki slučajni vektor čije je očekivanje kolinearno sa gradijentom i istog je smera. Ovakva modifikacija ima smisla pre svega kada se funkcija koja se optimizuje može predstaviti kao prosek drugih jednostavnijih funkcija:

$$f(x) = \frac{1}{N} \sum_{i=1}^N f_i(x)$$

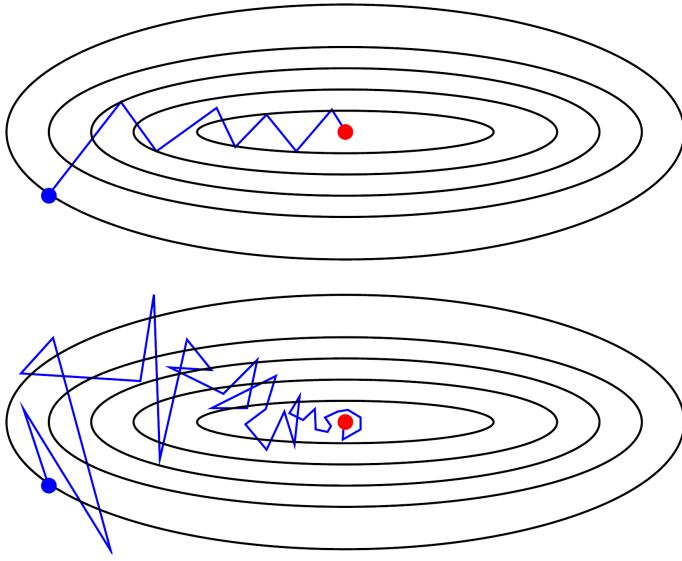
Ovo je tipičan slučaj u kontekstu mašinskog učenja, gde se minimizuje funkcija greške koja je zbir grešaka na pojedinačniminstancama. Tada je pravilo izračunavanja novog koraka moguće zameniti sledećim pravilom:

$$x_{k+1} = x_k - \alpha_k \nabla f_i(x_k)$$

Jasno, kako je funkcija f , prosek funkcija f_i , ako se i bira u skladu sa uniformnom raspodelom, očekivanje slučajnog vektora $\nabla f_i(x)$ je baš $\nabla f(x)$. Obično se i bira tako da bude jednak ($k \bmod N$) + 1, odnosno tako da se u svakom koraku koristi naredna funkcija f_i dok se ne dođe do poslednje, a onda se ponovo nastavlja od prve. Ovaj pristup predstavlja jeftinu aproksimaciju gradijenta. Ipak, ona može biti prilično neprecizna, kao što se može videti sa slike 9.4. Stoga se kao kompromis često, umesto samo jedne od funkcija f_i , koristi prosek nekog podskupa ovih funkcija (eng. minibatch). Ovo je praktično uvek pristup koji se koristi u obučavanju neuronskih mreža.

Brzina konvergencije stohastičkog gradijentnog spusta merena u broju iteracija je dosta manja nego kod običnog gradijentnog spusta. U slučaju konveksnih funkcija sa Lipšić neprekidnim gradijentom, greška je reda $O\left(\frac{1}{\sqrt{k}}\right)$, a u slučaju jako konveksnih funkcija sa Lipšić neprekidnim gradijentom, greška je reda $O\left(\frac{1}{k}\right)$. Uprkos ovome, u mašinskom učenju, u kojem se danas često koriste ogromne količine podataka, vreme jedne iteracije gradijentnog spusta, koji u svakoj iteraciji koristi sve podatke, je drastično veće nego u slučaju stohastičkog gradijentnog spusta, koji u svakoj iteraciji koristi samo po jednu instancu iz skupa podataka.

U odnosu na gradijentni spust, prednosti stohastičkog gradijentnog spusta su mnogostrukе. Gradijent, koji inače može biti skup za izračunavanje, jeftino



Slika 9.4: Ponašanje gradijentnog spusta i stohastičkog gradijentnog spusta.

se aproksimira. U kontekstu metoda mašinskog učenja nad velikim količinama podataka, to često vodi bržem učenju. Greška aproksimacije gradijenta može poslužiti i kao vid regularizacije, pošto sprečava preciznu konvergenciju ka minimumu, koja u slučaju vrlo fleksibilnih modela ili male količine podataka može voditi ka preprilagođavanju. Manje je podložan problemu redundantnosti podataka prilikom obučavanja. Pod redundantnošću se podrazumeva ponavljanje istih ili sličnih instanci u skupu podataka. Poslednja poenta zahteva opširnije obrazloženje, koje je dato narednim primerom. Mana je očito veći broj iteracija do konvergencije, što u slučaju da korak gradijentnog spusta nije vremenski skup, vodi sporijem zaustavljanju stohastičke varijante u odnosu na izvornu.

Primer 6 Neka se skup podataka sastoji od instanci $\{(x_1, y_1), \dots, (x_N, y_N)\}$. Neka se minimizuje srednjekvadratna greška

$$E(w) = \frac{1}{N} \sum_{i=1}^N (y_i - f_w(x_i))^2$$

Onda je pravilo ažuriranja koeficijenata u skladu sa stohastičkim gradijentnim spustom:

$$w_{k+1} = w_k - 2\alpha_k (y_i - f_{w_k}(x_i)) \nabla f_{w_k}(x_i)$$

dok je u slučaju gradijentnog spusta to

$$w_{k+1} = w_k - 2\alpha_k \frac{1}{N} \sum_{i=1}^N (y_i - f_{w_k}(x_i)) \nabla f_{w_k}(x_i)$$

Ukoliko se ceo skup podataka uveća tako što ponovi za redom M puta u poretku

$$\underbrace{(x_1, y_1), \dots, (x_N, y_N), \dots, (x_1, y_1), \dots, (x_N, y_N)}_{M \times N}$$

pravac koraka u gradijentnom spustu se neće promeniti ni u jednom koraku, pa je stoga i broj koraka u primeni algoritma isti. Kako svaki korak zahteva M puta više vremena, ceo proces M puta duže traje. Stohastički gradijentni spust u ovom slučaju ne zahteva ništa više vremena nego inače, zahvaljujući tome što u svakom koraku koristi samo po jednu instancu, pa korak košta jednako vremena, i što se instance u uvećanom skupu za obučavanje nižu na isti način na koji ih stohastički gradijentni spust i inače smenjuje.

Očigledno, ovo je ekstreman primer redundantnosti podataka, koji se ne očekuje u praksi, ali je ova prednost stohastičkog gradijentnog spusta osetna i u manje ekstremnim slučajevima.

Glava 10

Neuronske mreže i duboko učenje

Neuronske mreže (eng. neural networks) predstavljaju najpopularniju i jednu od najprimjenjenijih metoda mašinskog učenja. Njihove primene su mnogobrojne i pomeraju domete veštačke inteligencije, računarstva i primjenjene matematike. Neke od njih su kategorizacija teksta, medicinska dijagnostika, prepoznavanje objekata na slikama, autonomna vožnja, igranje igara poput igara na tabli (tavla i go) ili video igara, mašinsko prevođenje prirodnih jezika, modelovanje semantike reči prirodnog jezika i slično. Neuronske mreže zapravo predstavljaju parametrizovanu reprezentaciju koja može poslužiti za aproksimaciju drugih funkcija. Kao i u slučaju drugih metoda učenja, pronađenje odgovarajućih parametara se vrši matematičkom optimizacijom nekog kriterijuma kvaliteta aproksimacije i može biti računski vrlo izazovno.

Postoje različite vrste neuronskih mreža. Osnovnu varijantu predstavljaju *potpuno povezane neuronske mreže* (eng. *fully connected*). U obradi slika i drugih vrsta signala, pa i teksta, vrlo su popularne *konvolutivne neuronske mreže* (eng. *convolutional neural networks*). Za obradu podataka nalik nizovima promenljive dužine, najčešće se koriste *rekurentne neuronske mreže* (eng. *recurrent neural networks*), a za obradu podataka koji se predstavljaju grafovima koriste se *grafovske neuronske mreže* (eng. *graph neural networks*). U nastavku će biti diskutovane prve tri vrste.

U svetu njihovih izvanrednih uspeha i velike popularnosti, u laičkim krugovima postoji tendencija poistovećivanja mašinskog učenja, pa čak i veštačke inteligencije sa neuronskim mrežama. Ovakav pogled je prosto pogrešan. Takođe, postoji tendencija da se neuronska mreža razmatra kao prvi izbor metoda učenja nevezano od toga o kom se problemu radi. Ovo bi bio vrlo loš praktičan savet. Stoga, pre nego što pređemo na diskusiju neuronskih mreža, naglašavamo u kakvim situacijama su superiorne u odnosu na druge modele. Dok je sasvim moguće da će neuronska mreža preći druge modele i u drugačijim problemima, problemi zahvaljujući kojima su se neuronske mreže proslavile imaju određena zajednička svojstva. To su velika količina podataka i učenje na osnovu sirove reprezentacije podataka. Male količine podataka u slučaju neuronskih mreža lako vode preprilagođavanju, a učenje nad do sada diskutovanim vektor-

skim reprezentacijama podataka ne koristi u dovoljnoj meri ključnu prednost neuronskih mreža – sposobnost da same konstruišu nove atribute nad sirovom reprezentacijom podataka. Naime, iako domenski eksperti nekad mogu pretpostaviti koji su atributi najinformativniji za predviđanje ciljne promenljive, njihovi izbori nekada mogu biti i pogrešni, a neretko lošiji od onoga što bi algoritam učenja mogao da detektuje u sirovoj informaciji ako bi bio primenljiv na nju. Neuronske mreže su karakteristične po tome što su u stanju to da rade. Stoga, ukoliko je skup podataka mali i u vektorskom obliku, nema razloga da očekujemo posebni benefit od primene neuronskih mreža, a moguće je da ćemo imati problema sa njihovim nedostacima. Ukoliko je veliki i u sirovom obliku, verovatno je dobra ideja primeniti neku varijaciju neuronske mreže. Ukoliko je ispunjen samo jedan od ovih uslova, teško je napraviti procenu a priori.

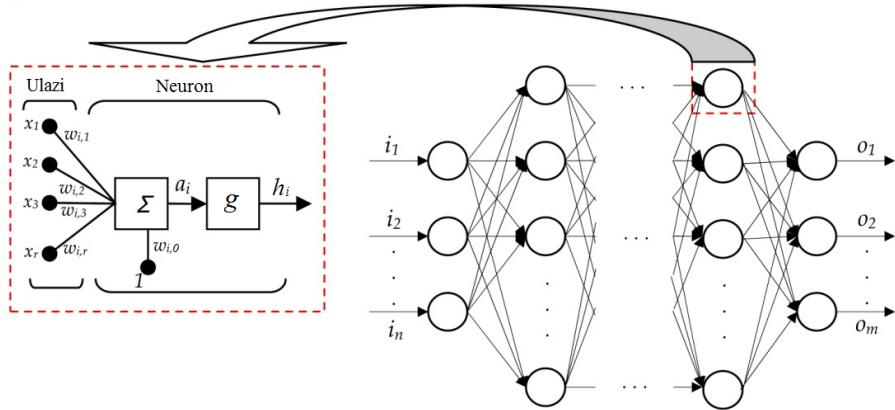
10.1 Potpuno povezane neuronske mreže

Potpuno povezana neuronska mreža (u nastavku samo – neuronska mreža) se sastoji od osnovnih računskih jedinica koje se nazivaju *jedinicama* ili *neuronima*, koje predstavljaju jednostavne parametrizovane funkcije. Svaka jedinica računa linearnu kombinaciju svojih argumenata i nad njom računa neku nelinearnu transformaciju, takozvanu *aktivacionu funkciju* (eng. *activation function*). Ove jedinice su organizovane u slojeve, tako da jedinice jednog sloja primaju kao svoje argumente, odnosno *ulaze*, vrednosti, odnosno *izlaze*, svih jedinica prethodnog sloja i sve jedinice prosleđuju svoje izlaze jedinicama narednog sloja. Zato se zovu potpuno povezanim. Svi slojevi čije jedinice prosleđuju svoje izlaze drugim jedinicama se nazivaju *skrivenim slojevima*. Ulazi jedinica prvog sloja se nazivaju ulazima mreže. Izlazi jedinica poslednjeg sloja se nazivaju izlazima mreže. Mreže sa valikim brojem skrivenih slojeva nazivaju se *dubokim neuronskim mrežama* (eng. *deep neural networks*). Šta je tačno veliki broj zavisi od vremena u kome se kvalifikacija koristi. Tekući kriterijumi su značajno različiti od onih od pre deset godina. Vrednosti neurona skrivenih slojeva mreže se mogu smatrati novim atributima tih objekata, nad kojima ostatak neuronske mreže uči aproksimaciju ciljne funkcije. Drugim rečima, neuronska mreža konstruiše nove atribute u svojim skrivenim slojevima. Svaki sloj je u stanju da nadograđuje nad prethodnim i tako gradi složenije i složenije atribute. Ovo svojstvo je posebno uočljivo kod konvolutivnih neuronskih mreža i smatra se da je ova mogućnost konstrukcije novih atributa jedan od glavnih razloga za uspešnost dubokih neuronskih mreža.

10.1.1 Formulacija modela

Formalno, model se definiše na sledeći način:

$$\begin{aligned} h_0 &= x \\ h_i &= g(W_i h_{i-1} + w_{i0}) \quad i = 1, 2, \dots, L \end{aligned}$$



Slika 10.1: Struktura potpuno povezane neuronske mreže.

gde je x vektor ulaznih promenljivih, L je broj slojeva, W_i je matrica čija j -ta vrsta predstavlja vektor vrednosti parametara jedinice j u sloju i , $w_{i,j}$ predstavlja vektor slobodnih članova linearnih kombinacija koje jedinice i -tog sloja izračunavaju, a g je nelinearna aktivaciona funkcija. Za vektor v , $g(v)$, predstavlja vektor $(g(v_1), g(v_2), \dots, g(v_m))^T$, gde je m dimenzionalnost vektora. Skup svih parametara modela će se ukratko označavati sa w . Važi $f_w(x) = h_L$. Shema neuronske mreže, prikazana je na slici 10.1.

Naredna teorema izražava važno svojstvo neuronskih mreža – da se svaka neprekidna funkcija može proizvoljno dobro aproksimirati neuronskom mrežom sa jednim skrivenim slojem i konačnim brojem neurona.

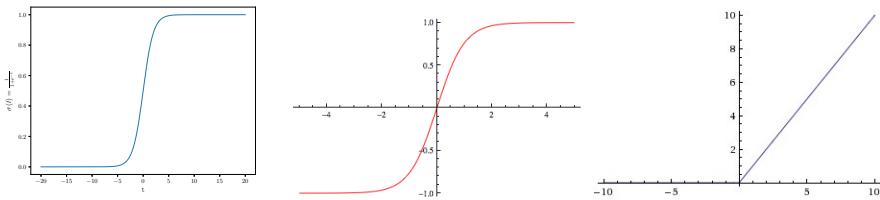
Teorema 4 (Teorema o univerzalnoj aproksimaciji) *Neka je g ograničena i strogo rastuća neprekidna funkcija. Tada za svaku funkciju $f \in C[0, 1]^n$ i sva-ko $\varepsilon > 0$, postoji broj $m \in \mathbb{N}$, matrica $W \in \mathbb{R}^{m \times n}$, vektor $w_0 \in \mathbb{R}^m$ i vektor $v \in \mathbb{R}^m$, tako da za svako $x \in [0, 1]^n$ važi*

$$|v^T g(Wx + w_0) - f(x)| < \varepsilon$$

Odnosno, skup svih neuronskih mreža sa jednim skrivenim slojem je svuda gust na skupu funkcija neprekidnih na intervalu $[0, 1]^n$. Ova teorema predstavlja osnovno teorijsko opravdanje za korišćenje neuronskih mreža u aproksimaciji funkcija. Ipak, ova teorema samo ustanovljava da postoji neuronska mreža sa datim svojstvima. To nikako ne znači da ju je lako naći na osnovu uzorka podataka.

10.1.2 Aktivacione funkcije

U datoј formulaciji modela neuronske mreže nije precizirano koja aktivaciona funkcija se koristi. Moguće je izabrati različite funkcije, ali se u praksi



Slika 10.2: Najčešće korišćene aktivacione funkcije – sigmoidna, tangens hiperbolički i ispravljajuća linearna jedinica.

najčešće koristi nekoliko narednih funkcija:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

$$rlu(x) = \max(0, x)$$

Grafici sve tri funkcije su prikazani na slici 10.2. Prva, *sigmoidna funkcija* je široko korišćena u mašinskom učenju i dugo je bila najčešće korišćena aktivaciona funkcija u neuronskim mrežama. Ipak, ova funkcija u praksi nije najpogodnija za upotrebu, pre svega zbog problema u optimizaciji. Naime, ona je skoro pa konstantna osim u okolini nule, pa stoga dovodi praktično do anuliranja gradijenta, što otežava ili praktično onemogućava učenje. Druga, tangens hiperbolički je takođe bila u širokoj upotrebi, obično sa nešto većim uspehom od sigmoidne funkcije sa kojom je vrlo srodnja (važi $\tanh(x) = 2\sigma(2x) - 1$). Treća, *ispravljena linearna jedinica* (eng. *rectified linear unit*) predstavlja trenutno najčešće korišćenu aktivacionu funkciju. Razlog za njenu popularnost su pogodnija svojstva pri optimizaciji, uprkos svojoj nediferencijabilnosti. Naime, sanse da se naleti na tačku nediferencijabilnosti u procesu optimizacije nisu velike, a i ako se to desi, u kasnijem toku optimizacije greška će tipično biti nadomešćena (ovo recimo ne bi važilo za ℓ_1 regularizovane funkcije pošto je tada tačka nediferencijabilnosti baš tačka kojoj se teži). S druge strane, izvod u linearom delu funkcije je konstantno 1, što vodi bržoj konvergenciji nego kad se gradijent smanjuje u nekim delovima domena funkcije. Ipak, ravan deo funkcije levo od nule predstavlja problem za optimizaciju. Naime, instance za koje je vrednost aktivacione funkcije nula, ne doprinose obučavanju, jer se na njima i gradijent anulira zahvaljujući konstantnosti funkcije. Stoga se ova aktivaciona funkcija često modifikuje tako da se to izbegne. Jedna modifikovana varijanta je *nakošena ispravljena linearna jedinica* (eng. *leaky rectified linear unit*), koja levo od nule uzima vrednost αx , za malu vrednost parametra α , poput 0.01.

10.1.3 Izlazne jedinice i greška

Neuronske mreže se mogu koristiti kako za regresiju funkcija sa vrednostima iz \mathbb{R}^n , tako i za klasifikaciju.

U slučaju regresije, jedinice poslednjeg nivoa ne koriste aktivacionu funkciju (kao što je predviđeno i u formulaciji teoreme o univerzalnoj aproksimaciji). Uz pretpostavku da je dat skup parova $\mathcal{D} = \{(x_i, y_i) | i = 1, \dots, N\}$, gde su x_i argumenti, a y_i vrednosti aproksimirane funkcije, aproksimacija se sprovodi rešavanjem narednog optimizacionog problema:

$$\min_w \sum_{i=1}^N (f_w(x_i) - y_i)^2 + \lambda \|w\|_2^2$$

Regularizacija formalno nije neophodna, ali se u praksi uvek koristi neki vid regularizacije (ne nužno ℓ_2), jer su neuronske mreže vrlo fleksibilni modeli, što znači da je problem učenja često loše uslovjen.

U slučaju klasifikacije se na linearne kombinacije jedinica poslednjeg nivoa primenjuje takozvana funkcija *mekog maksimuma* (eng. *softmax*) koja preslikava vektor dimenzije C u vektor iste dimenzije:

$$\text{softmax}(x) = \left(\frac{e^{x_1}}{\sum_{i=1}^C e^{x_i}}, \dots, \frac{e^{x_C}}{\sum_{i=1}^C e^{x_i}} \right)$$

Vrednosti novog vektora se očigledno sumiraju na 1 i stoga se mogu koristiti kao raspodela verovatnoće. Takođe, ova funkcija naglašava razlike među koordinatama polaznog vektora. Najveća pozitivna vrednost će biti transformisana u novu vrednost koja još više odskače od drugih. Za vrednost aproksimacije se uzima kategorija koja odgovara izlazu sa najvišom vrednošću. Za potrebe obučavanja, kao u slučaju drugih probabilističkih metoda, pribegava se primeni principa maksimalne verodostojnosti. Potrebno je maksimizovati verovatnoću opaženih vrednosti y_i za date vrednosti x_i . Uz pretpostavku uslovne nezavisnosti vrednosti y_i za date vrednosti x_i , važi:

$$P_w(y_1, \dots, y_N | x_1, \dots, x_N) = \prod_{i=1}^N P_w(y_i | x_i)$$

gde w predstavlja vektor parametara neuronske mreže od kojeg vrednosti na izlazu, koje predstavljaju verovatnoće, očigledno zavise. Za verovatnoću jednog podataka važi

$$P_w(y_i | x_i) = \prod_{j=1}^C \left(\frac{e^{a_j}}{\sum_{k=1}^C e^{a_k}} \right)^{t_{ij}}$$

gde promenljiva t_{ij} ima vrednost 1 ako vrednosti y_i odgovara kategoriji j , a 0 u suprotnom. Kao i pre, umesto maksimizacije date verovatnoće, vrši minimi-

zacija negativne vrednosti logaritma te verovantoće

$$-\log P_w(y_1, \dots, y_N | x_1, \dots, x_N) = -\sum_{i=1}^N \log P_w(y_i | x_i) = -\sum_{i=1}^N \sum_{j=1}^C t_{ij} \log \frac{e^{a_i}}{\sum_{k=1}^C e^{a_k}}$$

U oba slučaja, za rešavanje ovog problema potrebno je koristiti metode matičke optimizacije.

10.1.4 Algoritam propagacije unazad

Problem optimizacije neuronske mreže je težak zbog svoje nekonveksnosti, što znači da je moguće završiti u lokalnim optimumima ili da neke metode optimizacije nisu lako primenljive ili da sporije rade. U praksi, uvek se koriste gradijentne metode optimizacije. Nekad se koriste i metode drugog reda, koje se oslanjaju na hesijan, ali je to u slučaju većeg broja parametara nemoguće, jer je broj elemenata hesijana kvadratan u odnosu na broj parametara. U slučaju neuronske mreže, već je izračunavanje gradijenta netrivijalan problem i vrši se algoritmom *propagacije unazad* (eng. *backpropagation*) koji će biti opisan u nastavku.

Algoritam propagacije unazad, prikazan na slici 10.3, je jedan od malog broja najznačajnijih algoritama mašinskog učenja. Zasniva se na pravilu izračunavanja parcijalnog izvoda složene funkcije. Za funkcije $g : \mathbb{R}^m \rightarrow \mathbb{R}^n$ i $f : \mathbb{R}^n \rightarrow \mathbb{R}$, parcijalni izvod se računa prema formuli:

$$\partial_i(f \circ g) = \sum_{j=1}^n (\partial_j f \circ g) \partial_i g_j$$

Krećući se po slojevima neuronske mreže od poslednjeg ka prvom, algoritam u svakoj iteraciji obavlja tri posla:

- proširivanje do tada izračunatog parcijalnog izvoda izvodom aktivacione funkcije u skladu sa pravilom za računanje izvoda složene funkcije,
- izračunavanje vrednosti gradijenta po parametrima jedinica na tekućem nivou, zarad čega se do tada izračunati parcijalni izvod množi ulazima jedinica koje ti parametri množe i
- proširivanje do tada izračunatog parcijalnog izvoda izvodom linearne kombinacije po ulazima, u skladu sa pravilom za računanje izvoda složene funkcije.

Trivijalan primer takvog proširivanja parcijalnog izvoda dat je narednim izvođenjem:

$$f(g(h(x)))' = \underbrace{f'(g(h(x)))}_{d} g(h(x))' = \underbrace{f'(g(h(x)))}_{d} \underbrace{g'(h(x))}_{d} h(x)' = \underbrace{f'(g(h(x)))}_{d} \underbrace{g'(h(x))}_{d} \underbrace{h'(x)}_{d}$$

Potpuniji primer rada algoritma, dat je narednim primerom.

```

1  $d = \nabla_{h_L} E$ 
2 repeat
3    $d = d \odot g'(a_k)$ 
4    $\nabla_{w_{k_0}} E(w) = d + \lambda \nabla_{w_{k_0}} \Omega(w)$ 
5    $\nabla_{W_k} E(w) = dh_{k-1}^T + \lambda \nabla_{W_k} \Omega(w)$ 
6    $d = W_k^T d$ 
7    $k = k - 1$ 
8 until  $k = 0;$ 

```

Slika 10.3: Algoritam propagacije unazad. Simbol \odot označava pokoordinatno množenje vektora.

Primer 7 Neka je potrebno naći gradijent funkcije

$$E(w) + \Omega(w)$$

gde je $E(w)$ funkcija odstupanja koju treba minimizovati izračunata za jedan par (x, y) , a $\Omega(w)$ regularizacioni izraz. Recimo, u primeru regresije, važi

$$E(w) = (h_L - y)^2$$

$$\Omega(w) = \|w\|^2$$

gde h_L jasno zavisi od w . Gradijent za ceo skup \mathcal{D} se dobija sumiranjem gradijenata za pojedinačne instance. Izvršavanje algoritma propagacije unazad je prikazano u tabeli 10.1, na primeru izračunavanja gradijenta naredne funkcije:

$$E(w) = (h_2 - y)^2$$

$$f_w(x) = h_2 = \sigma(w_{20} + w_{21}\sigma(w_{10} + w_{11}x))$$

Algoritam propagacije unazad nije lako primenljiv na duboke neuronske mreže, zbog velikog broja uzastopnih množenja. Konkretno, zbog toga često dolazi do toga da vrednosti parcijalnih izvoda koje se računaju budu praktično nula (kada se množe brojevi manji od jedan) ili da budu ogromne ili čak da dođe do prekoračenja ili prosto nestabilnosti optimizacionog metoda (kada se množe brojevi veći od jedan). Ovaj problem se naziva problemom nestajućih i eksplodirajućih gradijenata (eng. *exploding and vanishing gradients*). Ovaj problem je blaži ukoliko se kao aktivaciona funkcija koristi nakošena ispravljena linearna jedinica, pošto za pozitivne vrednosti ima vrednost izvoda 1.

10.1.5 Kvaliteti i mane

Kao što je na početku rečeno, neuronske mreže su se izvanredno pokazale u rešavanju praktičnih problema. Za razliku od klasičnih metoda, za koje

#	Aktivni deo formule	Akumulirano	Izračunato
1	$(\textcolor{red}{h}_2 - y)^2$	$2(h_2 - y)$	
3	$\sigma(w_{20} + \underbrace{w_{21}\sigma(w_{10} + w_{11}x)}_{a_2})$	$2(h_2 - y)\sigma'(a_2)$	
4	$\sigma(\textcolor{red}{w}_{20} + w_{21}\sigma(w_{10} + w_{11}x))$	$2(h_2 - y)\sigma'(a_2)$	$2(h_2 - y)\sigma'(a_2)$
5	$\sigma(w_{20} + \textcolor{red}{w}_{21}\sigma(\underbrace{w_{10} + w_{11}x}_{a_1}))$	$2(h_2 - y)\sigma'(a_2)$	$2(h_2 - y)\sigma'(a_2)\sigma(a_1)$
6	$\sigma(w_{20} + w_{21}\sigma(\textcolor{red}{w}_{10} + \textcolor{red}{w}_{11}x))$	$2w_{21}(h_2 - y)\sigma'(a_2)$	
3	$\sigma(w_{20} + w_{21}\sigma(w_{10} + w_{11}x))$	$2w_{21}(h_2 - y)\sigma'(a_2)\sigma'(a_1)$	
4	$\sigma(w_{20} + w_{21}\sigma(\textcolor{red}{w}_{10} + w_{11}x))$	$2w_{21}(h_2 - y)\sigma'(a_2)\sigma'(a_1)$	$2w_{21}(h_2 - y)\sigma'(a_2)\sigma'(a_1)$
5	$\sigma(w_{20} + w_{21}\sigma(w_{10} + \textcolor{red}{w}_{11}x))$	$2w_{21}(h_2 - y)\sigma'(a_2)\sigma'(a_1)$	$2w_{21}2w_{21}(h_2 - y)\sigma'(a_2)\sigma'(a_1)x$
6	$\sigma(w_{20} + w_{21}\sigma(w_{10} + w_{11}x))$	$2w_{11}w_{21}(h_2 - y)\sigma'(a_2)\sigma'(a_1)$	

Tabela 10.1: Ilustracija izvršavanja algoritma propagacije unazad. Kolone predstavljaju redni broj koraka u algoritmu, deo formule po kojem se radi diferenciranje, akumulirani parcijalni izvod i izračunate vrednosti parcijalnih izvoda, za koje se vidi kom parametri odgovaraju po tome koji parametar je istaknut u drugoj koloni.

je potrebno pažljivo definisati atribute, neuronske mreže su često u stanju da nelinearnim transformacijama uče korisne atribute i da onda uče nad tim reprezentacijama. Ipak, imaju i značajne mane. Da bi njihovi kvaliteti došli do izražaja potrebna je velika količina podataka. Vreme potrebno za optimizaciju mreže može biti vrlo veliko (npr. može se meriti danima) i mogu zahtevati specijalizovani hardver da bi optimizacija bila izvršena u prihvatljivom vremenu. Postoji veliki broj izbora koje je potrebno učiniti pre rešavanja optimizacionog problema, poput broja nivoa mreže, broja jedinica u nivou, izbora vrednosti regularizacionog parametra, izbora algoritma za optimizaciju, itd. Za pravljenje ovih izbora često ne postoje jasne smernice, pa se često određuju empirijski – velikim brojem rešavanja optimizacionog problema dok se ne postignu zadovoljavajući rezultati, što je vremenski vrlo zahtevno. Zbog izrazite fleksibilnosti modela, moguće je da se model preterano prilagodi podacima na kojima je vršena optimizacija i da njegove performanse pri upotrebi budu nezadovoljavajuće. Javljuju se različiti problemi u procesu optimizacije, poput problema nestajućih i eksplodirajućih gradijenata. Zbog svega ovoga upotreba neuronskih mreža zahteva i veliko iskustvo.

10.1.6 Primer primene – prepostavljanje naredne izgovorene reči

Kao primer primene neuronske mreže, poslužio bi bilo koji problem regresije ili klasifikacije u kojem je za svaki vektor podataka x data vrednost y koju treba aproksimirati. Ipak, neki primeri, poput primena u prepoznavanju govora, su posebno zanimljivi.

Primer 8 Prepoznavanje govora, odnosno identifikacije reči koje su izgovorene je izazovan problem iz mnogo razloga, kao što su različite boje glasa različitih

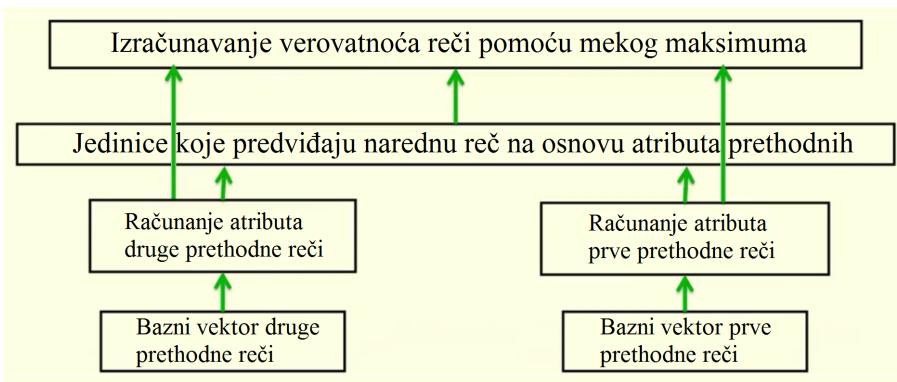
ljudi, različita brzina govora, različit intenzitet glasa, različit akcenat i artikulacija, prisustvo šuma prilikom snimanja, itd. Ljudi nisu svesni težine ovog problema u svakodnevnom govoru, pošto se u velikoj meri oslanjaju na razumevanje konteksta, koje vodi tome da očekuju pojavljivanje nekih reči, pre nego nekih drugih. Analogno tome, sistem za prepoznavanje govora bi mogao značajno poboljšati svoje performanse ukoliko bi mu bila dostupna makar približna informacija o verovatnoćama pojavljivanja reči, recimo na osnovu prethodne dve koje su već identifikovane. Naime, ukoliko sistem koji analizira govor pridružuje približne verovatnoće dvema rečima samo na osnovu njihovog zvučnog zapisa, ali se na osnovu prethodne dve reči može zaključiti da je verovatnoća pojavljivanja jedne od njih nakon prethodne dve približna nuli, to je važan indikator da je izgovorena druga reč.

Predviđanje treće reči na osnovu prve dve u principu je izvodljivo ukoliko je poznat ogroman korpus teksta, iz kojeg se verovatnoće pojavljivanja različitih trojki mogu oceniti na osnovu njihovih frekvencija. Ovo se može uraditi birajući reč koja ima najvišu uslovnu verovatnoću, koja se ocenjuje iz frekvencija f kombinacija reči u korpusu:

$$P(w_3|w_2, w_1) = \frac{P(w_3, w_2, w_1)}{P(w_2, w_1)} \approx \frac{f(w_3, w_2, w_1)}{f(w_2, w_1)}$$

U današnjem dobu, veliki korupsi teksta su dostupni, ali određivanje verovatnoća trojki reči nije pouzdano zbog toga što se mnoge trojke i u velikim korpusima mogu javiti vrlo retko ili ne uopšte, zbog toga što je veličina vokabulara koji se koristi u nekom jeziku vrlo velika. Na primer ako se broj reči koje osoba koristi meri desetinama hiljada, onda se broj svih trojki tih reči meri hiljadama milijardi. Zbog toga bi verovatoće mnogih trojki bile verovatno netačno ocenjene kao izuzetno male ili čak jednake nuli. Osnovna mana ovog pristupa je da tretira reči kao potpuno različite nedeljive celine i ne uočava potencijalne sličnosti među njima, poput recimo sinonimije ili srodnosti po smislu. Na primer, ukoliko čovek na osnovu prethodnog konteksta očekuje da se u nastavku rečenice javi reč novac, lakše će prepoznati i reč pare. Ukoliko očekuje da se u nastavku rečenice javi kućni ljubimac, lakše će prepoznati reči poput psa i mačke. Ukoliko očekuje da se javi dan u nedelji, lakše će prepoznati reči poput ponedeljka i utorka.

Ako se umesto ovog pristupa, prepostavi da je svaka reč okarakterisana numeričkim vektorom atributa određene dimenzije m, koji opisuje njena sintaksna i semantička svojstva, dolazi se do mnogo kompaktnije reprezentacije reči jer broj m može biti drastično manji od ukupnog broja reči. Dodatno, ukoliko se model predviđanja formuliše nad takvim reprezentacijama, on može da nauči da se nakon reči sa nekim svojstvima očekuje reč koja označava kućnog ljubimca i da na osnovu toga pridruži višu verovatnoću rečima pas i mačka koje će imati visoku vrednost atributa kućni ljubimac. Ovakav pristup omogućava i korišćenje dosta dužeg konteksta reči nego što je moguće u pristupu koji se oslanja na same reči. Razlog za to je što se smisao kućnog ljubimca, predstavljen nekom od reči koje označavaju kućne ljubimce, mnogo češće javlja u tekstovi-



Slika 10.4: Shema neuronske mreže koja predviđa narednu reč na osnovu prethodnog konteksta.

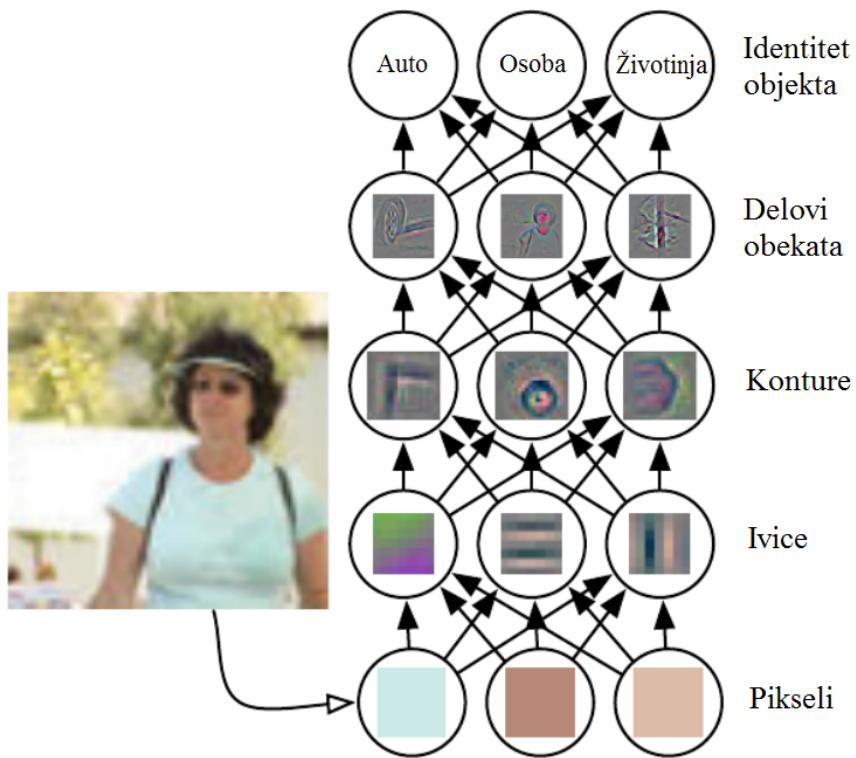
ma od bilo koje pojedinačne reči koja ga označava, pa su i ocene verovatnoća pouzdanije. Naravno, postavlja se pitanje, otkud dolaze sintaksna i semantička svojstva reči. Da li ih je potrebno osmislići unapred i koliko to košta u terminima vremena i novca? Odgovor je da će atributi koji opisuju reči biti definisani u procesu optimizacije, tako što će u tom procesu biti određeni parametri jedinica u skrivenim slojevima mreže, a izlazi tih jedinica će predstavljati vrednosti tih atributa. Vrlo je verovatno da će biti teško ili nemoguće odrediti značenje tako konstruisanih atributa, ali to nije mnogo važno ako je glavni cilj pogoditi narednu reč.

Na slici 10.4., data je shema neuronske mreže koja predviđa sledeću reč na osnovu prethodnog konteksta. Ukoliko je broj reči u vokabularu m , reči se predstavljaju baznim vektorima v_1, v_2, \dots, v_m , prostora \mathbb{R}^m , takvima da je vrednost i -te koordinate vektora v_j , 1 ukoliko je $i = j$, a 0 u suprotnom.

Ovakvi problemi se još bolje rešavaju rekurentnim neuronskim mrežama, koje ne prepostavljaju fiksiranu dužinu konteksta.

10.2 Konvolutivne neuronske mreže

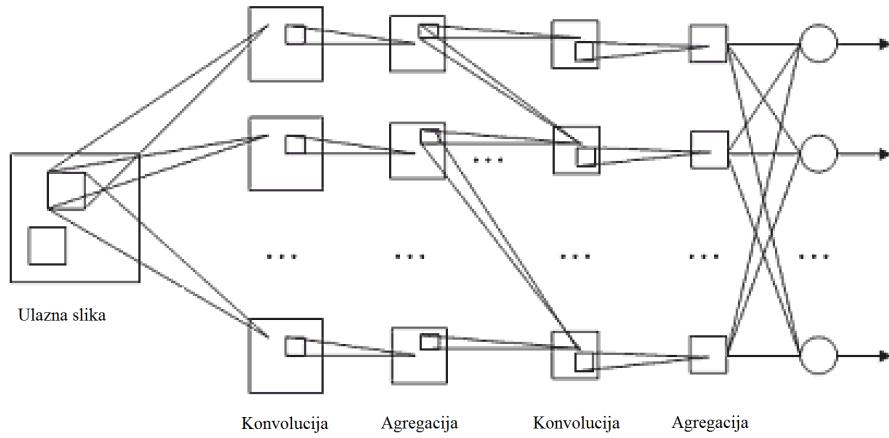
Konvolutivne neuronske mreže se intenzivno koriste u obradi signala poput zvuka i slike, ali takođe i teksta. Zasnivaju se upravo na pomenutoj sposobnosti mreža da konstruišu atribute, ali ne iz već datih atributa, već prvenstveno iz sirovog signala. Nazivaju se konvolutivnim zato što uče filtere, čijom konvolutivnom primenom detektuju određena svojstva signala. U obradi signala su dugo korišćeni filteri dizajnirani od strane inženjera i istraživača (npr. poput filtera za detekciju ivica na slikama). Značaj konvolutivnih mreža je upravo u tome što ne zahtevaju ljudski angažman u definisanju relevantnih svojstava signala, koji verovatno i ne bi proizveo najbolje rezultate, već u zavisnosti od problema koji je potrebno rešiti, same ustanovljavaju koja su svojstva bitna,



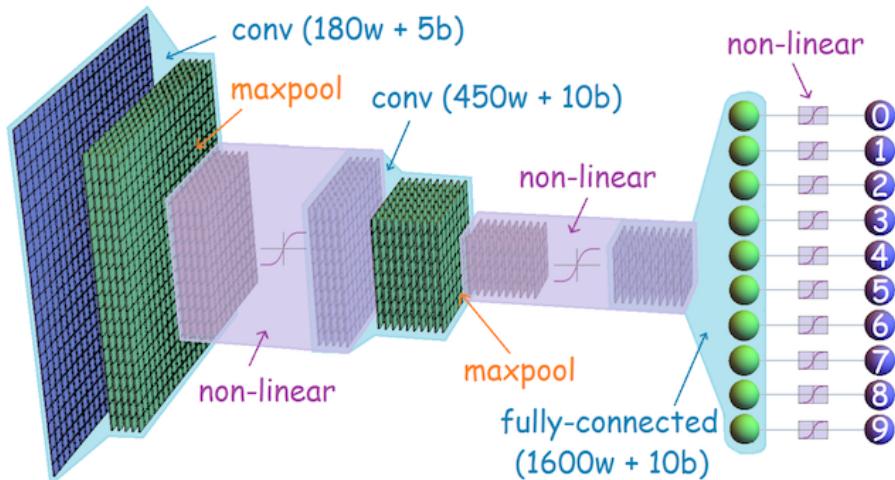
Slika 10.5: Shema konstrukcije složenijih od jednostavnijih atributa u slojevima konvolutivne mreže.

kroz učenje adekvatnih filtera. Ipak, ovakva vrsta fleksibilnosti podrazumeva izazove pri optimizaciji, kao i veliku količinu podataka.

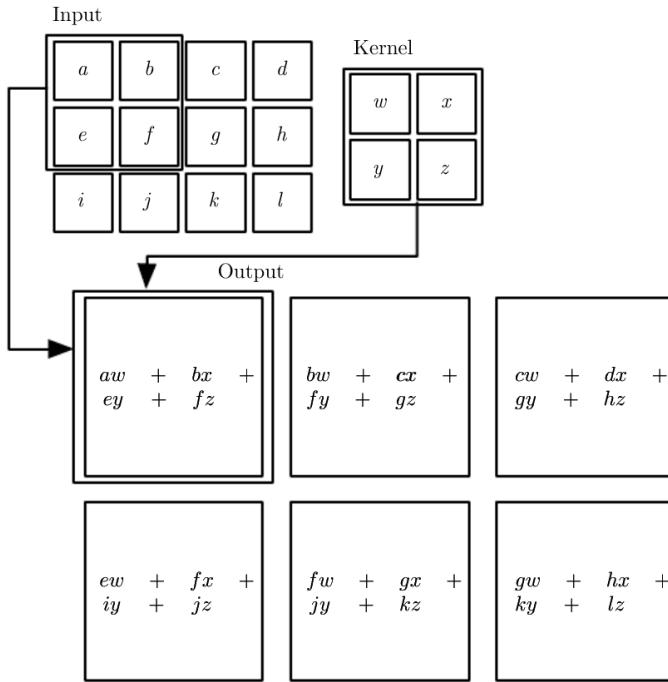
Konvolutivne mreže su praktično uvek duboke neuronske mreže, jer je potrebno od sitnijih detalja, poput uspravnih, kosih i horizontalnih linija, koji obično bivaju detektovani u nižim slojevima mreže, iskonstruisati složenije oblike poput delova lica. Ovo je ilustrovano slikom 10.5. Uobičajena struktura konvolutivne mreže podrazumeva smenjivanje dve vrste slojeva – konvolutivnih slojeva (eng. convolution layer) i slojeva agregacije (eng. pooling layer), kao što je prikazano na slici 10.6, pri čemu je moguće i da se ista vrsta sloja ponovi više puta. Izlazi konvolutivnog sloja se transformišu nelinearnom aktivacionom funkcijom. Na izlaze poslednjeg od tih slojeva se obično nadovezuje potpuno povezana mreža, koja uči nad atributima koje prethodni slojevi konstruišu. Još jedna ilustracija, koja naglašava manje detalja, ali je nakon inicijalnog razumevanja osnovnih principa, zbog svoje kompaktnosti lakša za vizualizaciju, data je na slici 10.7.



Slika 10.6: Shema konvolutivne mreže.



Slika 10.7: Grublja i kompaktnija shema konvolutivne mreže.



Slika 10.8: Konvolucija ulazne slike sa filterom.

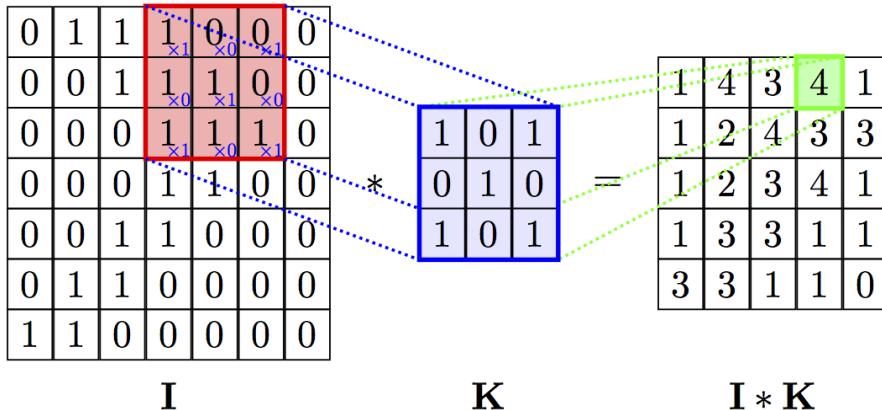
10.2.1 Konvolucija

Neka su f i g dve matrice dimenzija $m \times n$ i $p \times q$. Konvolucija je operacija definisana u diskretnom dvodimenzionalnom slučaju na sledeći način

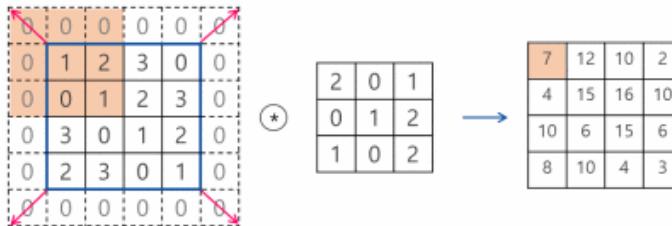
$$(f * g)_{ij} = \sum_{k=0}^{p-1} \sum_{l=0}^{q-1} f_{i-k, i-l} g_{k, l}$$

Oduzimanje u indeksima izraza $f_{i-k, i-l}$ se u praksi često menja sabiranjem pošto u kontekstu konvolutivnih mreža ne dovodi do razlike. Matrica f je obično ulaz, poput slike, dok je matrica g *filter* – pomoćna matrica koja izdvaja neku vrstu informacije iz ulaza. Umesto izraza *filter*, često se koristi i izraz *kernel*, ali ćemo ga izbegavati jer u ovom kontekstu filter nema svojstva kornela na koja smo navikli. Primeri izračunavanja konvolucije dati su na slikama 10.8 i 10.9.

Primetimo da formula konvolucije koju smo dali nije definisana za sve indeksi $i = 0, \dots, m-1$ i $j = 0, \dots, n-1$. Na primer, za $i, j = 0$ i $k, l > 0$, vrednost $f_{i-k, i-l}$ nije definisana. Ukoliko bismo se ograničili na definisane vrednosti, dimenzija konvolucije bi bila manja od dimenzije matrice f , što se vidi i sa slikama 10.8. To nije uvek poželjno, i može se izbegnati tako što se vrši *proširivanje* (eng. *padding*) matrice f , recimo nulama ili, još bolje, vrednostima koje su već



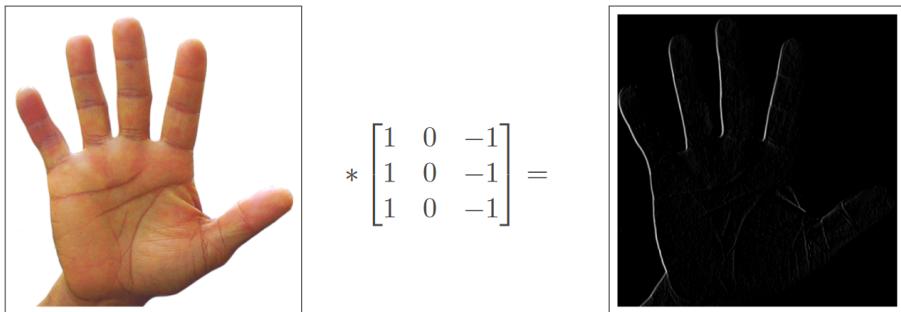
Slika 10.9: Konvolucija ulazne slike sa filterom.



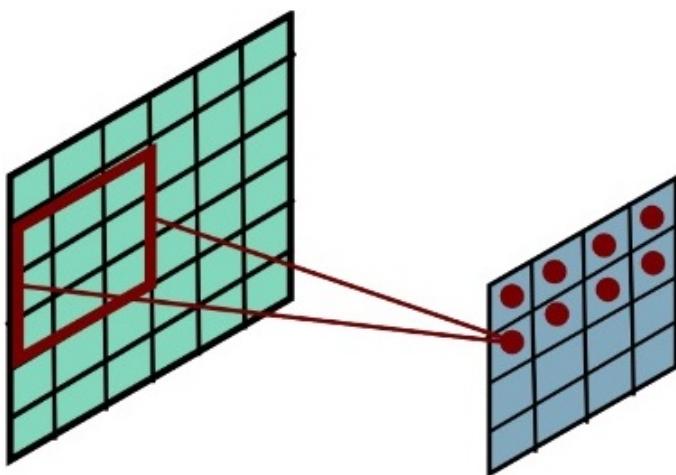
Slika 10.10: Konvolucija slike proširene nulama sa filterom.

na obodu, tako da veličina rezultujuće matrice bude jednaka veličini matrice f pre proširivanja. Ovo je prikazano na slici 10.10. Takođe, prilikom računanja konvolucije, filter se duž slike ne mora pomerati za jedan piksel, već za neki veći *korak* (eng. *stride*).

Konvolutivni sloj ima ulogu konstrukcije novih atributa. Jednostavan primer je filter za detekciju uspravnih ivica koji nije naučen već dizajniran, a koji je prikazan na slici 10.14. Osnovna ideja konvolutivnih mreža, kao što je već rečeno, je da ovakvi (a potencijalno i korisniji) filteri ne moraju biti dizajnirani, već naučeni. Dalje, postojanje nosa na slici se može utvrditi na osnovu specifičnog rasporeda uspravnih, kosih i horizontalnih linija, koje detektuje prethodni konvolutivni sloj. Svaki konvolutivni sloj raspolaže nizom parametrizovanih filtera koji vrše ove poslove. Na primer, ako jedan filter detektuje vertikalne linije, prirodno je imati paralelno, nad istim prethodnim slojem (ili ulazom) i filter koji detektuje horizontalne linije. Upravo, parametri filtera predstavljaju parametre konvolutivne mreže i bivaju naučeni u procesu obučavanja mreže. Konvolucija sa filterom se realizuje jedinicama koje su organizovane u niz (u slučaju jednodimenzionalnih signala poput zvuka) ili matricu (u slučaju dovodi-



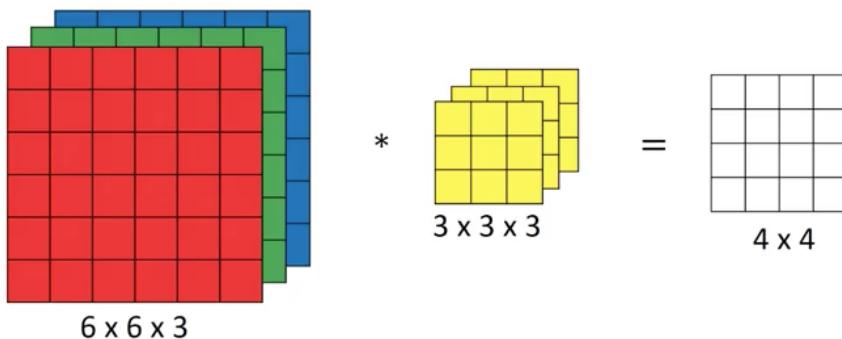
Slika 10.11: Filter koji nalazi uspravne ivice na slici.



Slika 10.12: Konvolutivni sloj. Svaka jedinica, označena crvenim krugom, kao ulaze uzlaze prethodnog sloja koji odgovaraju jedinicama koje se nalaze u crvenom kvadratu. Sve jedinice istog konvolutivnog sloja imaju iste vrednosti parametara i samim tim vrše isti obrazac u prethodnom sloju.

menzionih signala poput slike) i *dele vrednosti parametara*. Obično kao ulaze uzimaju vrednosti susednih jedinica iz prethodnog sloja. Na primer, u slučaju slike, to mogu biti vrednosti 3×3 jedinice iz prethodnog sloja. Ovakva organizacija jedinica je prikazana na slici 10.12. Na taj način, imajući u vidu da sve jedinice u jednom sloju imaju iste koeficijente, dejstvo konvolutivnog sloja se može razumeti kao konvolucija prethodnog sloja sa jedinicom definisanom parametrima tekućeg sloja ili u jednostavnijim terminima, kao prevlačenje filtera duž slike i izračunavanje vrednosti koju filter daje na svakoj poziciji.

U prethodnim razmatranjima nije diskutovano da li filter deluje samo nad tačno jednim kanalom prethodnog sloja ili nad više. U praksi se implementiraju



Slika 10.13: Filter koji deluje na više kanala odjednom.

višekanalne operacije konvolucije koje omogućavaju da filter deluje nad više kanala iz prethodnog sloja odjednom. Ovo je ilustrovano slikom 10.13.

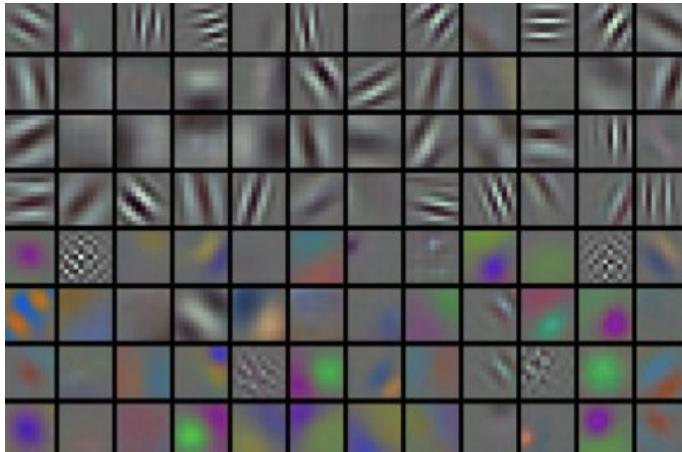
10.2.2 Agregacija

Sloj agregacije ukrupnjuje informacije koje dobija iz prethodnog sloja, obično tako što računa neku jednostavnu funkciju agregacije susednih jedinica prethodnog sloja, poput maksimuma ili proseka. Jednom kanalu konvolucije odgovara jedan kanal agregacije. Ukoliko agregira, na primer, 3×3 piskela, onda je broj izlaza ovog sloja 9 puta manji od broja izlaza prethodnog. Maksimum je najpopularniji izbor funkcije agregacije. Ukoliko 3×3 jedinice prethodnog sloja predstavljaju ulaz u jednu jedinicu sloja agregacije i ako ona računa njihov maksimum, dolazi do zanemarivanje informacije o tome gde je precizno neko svojstvo (poput uspravne linije) pronađeno, ali se ne gubi informacija da je pronađeno. Ovakva vrsta zanemarivanja informacije često ne šteti cilju koji treba postići. Na primer, ako su na slici pronađeni oko, uvo, usta i nos, informacija o tačnoj poziciji najverovatnije nije bitna za odlučivanje da li se na slici nalazi lice. Naime, u realnosti, šanse da slika koja sadrži navedene elemente ne predstavlja lice su izuzetno male. Ipak, ukoliko je potrebno napraviti mrežu koja igra igru u kojoj su pozicije objekata na ekranu bitne, nije poželjno koristiti agregaciju.

Uloga agregacije je smanjenje broja računskih operacija u višim slojevima, a i smanjenje broja parametara (koji može biti ogroman) u potpuno povezanoj mreži koja sledi za slojevima konvolucije i agregacije. Sve to rezultuje smanjenjem računske zahtevnosti pri optimizaciji i smanjenjem fleksibilnosti modela.

10.2.3 Interpretabilnost

Najčešća primena konolutivnih neuronskih mreža je u obradi slika. Pokazale su se izuzetno uspešnim u prepoznavanju objekata na slikama. Kao što



Slika 10.14: Vizualizacija oblika koje jedinice najnižeg sloja mreže prepoznaju.

je rečeno, niži slojevi konvolutivne mreže detektuju jednostavne oblike. Vrlo je lako ustanoviti koji oblik detektuju jedinice prvog konvolutivnog nivoa. To je oblik za koji one daju najviše vrednosti na izlazima. Kako je aktivaciona funkcija rastuća, to je oblik koji daje najvišu vrednost linearne kombinacije koju jedinica računa. Ako su koeficijenti jedinice w , oblik je dat kao rešenje problema

$$\max_{\|x\|=1} w \cdot x$$

Normiranje je važno pošto bi u slučaju da je skalarni proizvod pozitivan, povećavanjem intenziteta vektora x , uvek mogla biti dostignuta proizvoljna vrednost skalarnog proizvoda. Pod tim uslovom, lako se ustanavljava (recimo, postavljanjem parcijalnih izvoda po x na 0) da se maksimalna vrednost dostiže za vrednost $w/\|w\|$. Stoga, da bi se prikazao oblik koji jedinica najnižeg konvolutivnog sloja prepoznaće, dovoljno je vizualizovati njene koeficijente u vidu slike čije dimenzije odgovaraju dimenzijama filtera koji ta jedinica predstavlja. Jedan takav prikaz je dat na slici 10.14. Vizualizacija oblika koje prepoznaaju viši nivoi konvolutivne mreže je komplikovanija i zahteva neki vid optimizacije, kako bi se našli ulazi u mrežu za koje ove jedinice daju najviše vrednosti. Na slici 10.15 dat je prikaz oblika koje prpoznaaju jedinice višeg nivoa mreže koja prepoznaće objekte na slikama. Ova vrsta analize je važna zbog toga što pokazuje koja svojstva analiziranih objekata, u ovom slučaju slika, su važna za obavljanje posla koji mreža obavlja. Ova informacija ne mora uvek biti poznata korisniku i može predstavljati novo saznanje.

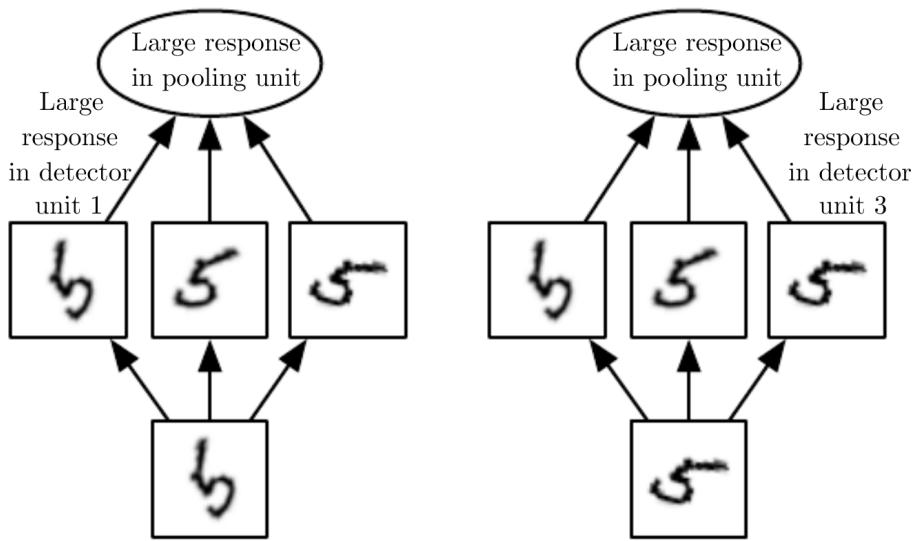


Slika 10.15: Vizualizacija oblika koje prepoznaju jedinice višeg nivoa mreže – jedrenjak, plišani meda i bokal.

10.2.4 Kvaliteti, mane i poboljšanja

Konvolutivne neuronske mreže su vrhunski model za obradu signala mašinskim učenjem – slika, zvuka i vremenskih serija. Imaju nekoliko glavnih kvaliteta koji su zaslužni za njihove dobre performanse. Prvi su *proređene interakcije*, pod čime se podrazumeva da je svaka jedinica povezana samo sa malim brojem jedinica iz prethodnog sloja, umesto sa svim, kao što u slučaju potpuno povezanih mreža. Drugi je *deljenje parametara*, koje se odnosi na to da sve jedinice jednog kanala imaju iste parametre – definisane filterom tog kanala. Kada parametri ne bi bili deljeni, kao kod potpuno povezane mreže, učeći različite parametre za različite delove slike, mreža bi učila da nekim delovima ulaza pridaje posebnu semantiku. To može zvučati dobro, ali u slučaju detekcije lica negde na slici, ne bi bilo dobro da se nauči da nos treba da bude baš na sredini slike, jer bi mogao biti i na nekom drugom mestu ako foto aparat nije bio centriran na lice. Deljenje parametara omogućava da se nauči filter koji traži nos bilo gde na slici. Treći je *neosetljivost na translacije*, odnosno svojstvo da će neki signal koji mreža traži biti nađen bez obzira kako je transliran na slici. Četvrti je *specijalizovanost za topologiju signala*. Naime i prethodno diskutovani modeli mašinskog učenja bi se mogli primenjivati na probleme vezane za zvuk, slike i slično, ali ako bi recimo trebalo da rade nad sirovom reprezentacijom slike u vidu piksela, raspored piksela bi bio potpuno proizvoljan, pošto ovi modeli ne uzimaju u obzir susednost piksela na slici, dok su konvolutivne mreže konstruisane imajući u vidu da to što su pikseli jedni u okolini drugih ima poseban značaj. Još jedan kvalitet, koji nije vezan za performanse je *delimična interpretabilnost*, zahvaljujući tome što se mogu vizualizovati ulazi na koje filteri daju najjači odgovor.

S druge strane, postoje i problemi. Naime, mreža je osjetljiva na neke druge transformacije, poput rotacije i skaliranja (homotetije). Takođe, kako potpuno povezana mreža koja se nalazi na kraju niza slojeva konvolutivne mreže uvek mora imati fiksiran broj ulaza, tako i konvolutivni deo mora proizvesti izlaz tačno određenih dimenzija, što zbog fiksiranog broja slojeva agregacije i fiksi-



Slika 10.16: Agregacija nad više kovolutivnih kanala odjednom. Ukoliko se u nekom od konvolutivnih kanala pronađe relevantni uzorak, agregacioni sloj prijavljuje da je pronađen. Pritom različiti konvolutivni kanali traže na različite načine transformisan uzorak, čime omogućavaju invarijantnost mreže u odnosu na datu transformaciju.

rane rezolucije agregiranja (npr. 3×3 vrednosti se zamenjuju jednom), znači i da ulazi moraju biti istih dimenzija.

Postoje određeni pristupi poboljšanja kojima se ublažuju prethodno pomenuti problemi. U slučaju potrebe za neosetljivošću na određene transformacije, to se može donekle postići time što agregacija ne bi bila vršena nad jednim kanalom konvolucije, već nad više. U tom slučaju, različiti konvolutivni kanali mogu naučiti da traže različito transformisane uzorce od značaja i ukoliko ga jedan od kanala nađe, agregacija maksimumom prijavljuje da je obrazac nađen. Ovo je ilustrovano slikom 10.16. Drugi problem, vezan za veličine ulaza, može se rešiti na više načina. Neki su komplikovani i uključuju izgradnju novih modela i na njih se ne osvrćemo. Neki, poput skaliranja slike, su naivni i vode gubitku informacije. Dodatno, osetljivost na transformacije, uključujući skaliranje, je već konstatovana slabost konvolutivnih mreža. Poluzadovoljavajući pristup bi bio taj da rezolucija aggregacije zavisi od dimenzija ulaza. Obično se ovaj tip aggregacije radi samo na poslednjem nivou. Recimo, ako se za slike dimenzija 1000×500 na poslednjem nivou aggregacije traži maksimum delova dimenzija 3×3 , onda bi se za slike dimenzija 2000×1000 tražio maksimum delova dimenzija 6×6 . Na taj način, dimenzije ulaza u potpuno povezanu mrežu ostaju iste. Još jedna mogućnost je da se na poslednjem nivou, aggregacija uradi uprosećavanjem svih vrednosti kanala.

Pored diskutovanih problema koji su specifični za konvolutivne mreže, problemi diskutovani u slučaju potpuno povezanih mreža su takođe prisutni. To što su ublaženi navedenim kvalitetima konvolutivnih mreža, vodi povećanju ambicija i izgradnji većih mreža, sa većim brojem parametara, pa isti problemi ponovo dolaze do izražaja.

10.3 Rekurentne neuronske mreže

Rekurentne neuronske mreže predstavljaju arhitekturu mreža specijalizovanu za obradu sekvenčnih podataka, poput rečenica prirodnog jezika i vremenskih serija. Postoje i njihova uopštenja na složenije strukture poput stabala i grafova, ali se njima nećemo baviti. Rekurentne neuronske mreže su konstruisane sa idejom da se modeluje zavisnosti među instancama. Na primer, vrsta neke reči može zavisiti od vrste neke od prethodnih reči, jer vrste prethodnih reči nose informaciju o vrsti narednih reči. Na primer, ne očekuje se niz od tri glagola. Pritom, u takvim poslovima, obično nije unapred poznato koliko prethodnih elemenata sekvenca nosi informaciju o narednom elementu. Čak broj prethodnih elemenata koji nose relevantnu informaciju ne mora biti jednak za bilo koji tekući element, već može zavisiti od njihovih svojstava. U suprotnom, kada bi se znalo da je za obradu svakog elementa bitno njegovih k prethodnika, bilo bi moguće primeniti potpuno povezanu neuronsku mrežu koja bi kao informaciju uzimala informacije o tim elementima i vršila predviđanje na osnovu toga. Rekurentne neuronske mreže prevazilaze ovaj problem tako što se odustaje od ideje da se parametrima modela definiše predviđanje na osnovu tekućih ulaza mreže. Umesto toga, elementi ulazne sekvence se obrađuju u koracima, mreža ima skriveno stanje koje akumulira informaciju o elementima sekvence obrađenim u prethodnim koracima, a parametri određuju na koji način se to stanje menja iz koraka u korak na osnovu prethodnog stanja i tekućih ulaza i kako se generiše izlaz mreže u zavisnosti od tekućeg stanja.

Obratimo pažnju na značaj ove ideje na primeru obrade jezika. Ukoliko bi trebalo primeniti potpuno povezanu neuronsku mrežu na problem mašinskog prevođenja rečenica sa jednog jezika na drugi, mreža bi, zarad analize cele rečenice, morala imati onoliko ulaza koliko ima reči u rečenici. Međutim broj reči u rečenici varira od rečenice do rečenice, dok broj ulaza jedne neuronske mreže mora biti fiksiran. Očito, morali bismo prepostaviti dužinu najduže moguće rečenice i ravnati se prema njoj. Takođe morali bismo ustavoviti mehanizam kojim se mreži govori kojim ulazima su pridružene reči a koji su neaktivni, jer je rečenica kraća. Dodatno, kako su reči kategoričke vrednosti iz ogromnog skupa kategorija (desetine hiljanda) i predstavljaju se upotrebom binarnog kodiranja, broj parametara bi bio izuzetno veliki, što povlači opasnost od preprilagođavanja. Dodatno, potpuno povezane mreže su dizajnirane tako da prepostavljuju nekakav fiksiran smisao svojih ulaza, a način na koji se taj smisao odražava na izlaz je određen naučenim parametrima. Na primer, ukoliko su u nekom problemu atributi oblačnost i temperatura, naučeni parametri

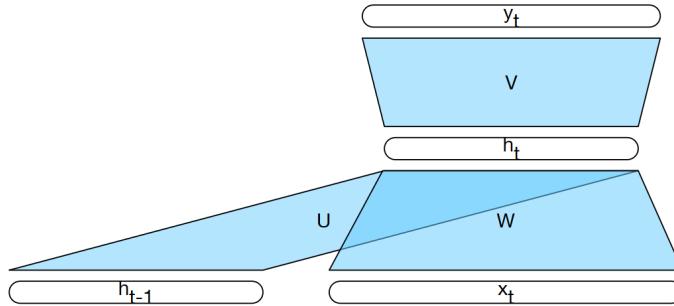
će biti adekvatni baš za te veličine. Ako bismo razmenili redosled atributa u instancama nakon što je model obučen, a zadržali redosled parametara u modelu, od modela ne bismo mogli očekivati dobro ponašanje. Međutim, u primeru obrade jezika, nema nikakvog smisla prepostaviti da različitim pozicijama u rečenici pripadaju fiksna značenja (npr. da je na prvom uvek subjekat, na drugom, uvek objekat, itd.), već se može očekivati da reči koje vrše različitu funkciju u rečenici mogu razmenjivati pozicije, pošto rečenice često možemo preformulisati, tako da je redosled reči drugačiji, a da to ne utiče na značenje. Potpuno povezana neuronska mreža bi očito morala da nauči vrlo komplikovane zavisnosti vezane za najrazličitije moguće redoslede reči, kako bi prepoznala isti smisao u različitim formulacijama rečenice. S druge strane, iz perspektive rekurentnih mreža, situacija je mnogo jednostavnija. Ulazi mreže mogu da odgovaraju kodiranju samo jedne reči, koje se u koracima jedna po jedna prikazuju mreži. Skriveno stanje mreže odražava smisao dela rečenice koji je prezentovan mreži. Parametri mreže upravljaju promenom stanja kada se mreži kao ulaz da naredna reč i generisanjem izlaza (npr. naredne reči prevoda) na osnovu novog stanja. Mreži se očito može, jedna za drugom, pružiti različit broj reči. Dovoljno je samo transformisati stanje pri svakoj promeni ulaza. Time je i broj parametara mnogo manji nego kad mreža analizira odjednom veliki broj reči. Naravno, u svrhe mašinskog prevodenja, potrebno je razrešiti još problema. Na primer, pitanje različitog broja reči u rečenici na polaznom i na ciljnem jeziku (što će biti diskutovano kasnije), ali deluje da je u odnosu na potpuno povezane mreže već učinjen krupan korak.

10.3.1 Formulacija modela

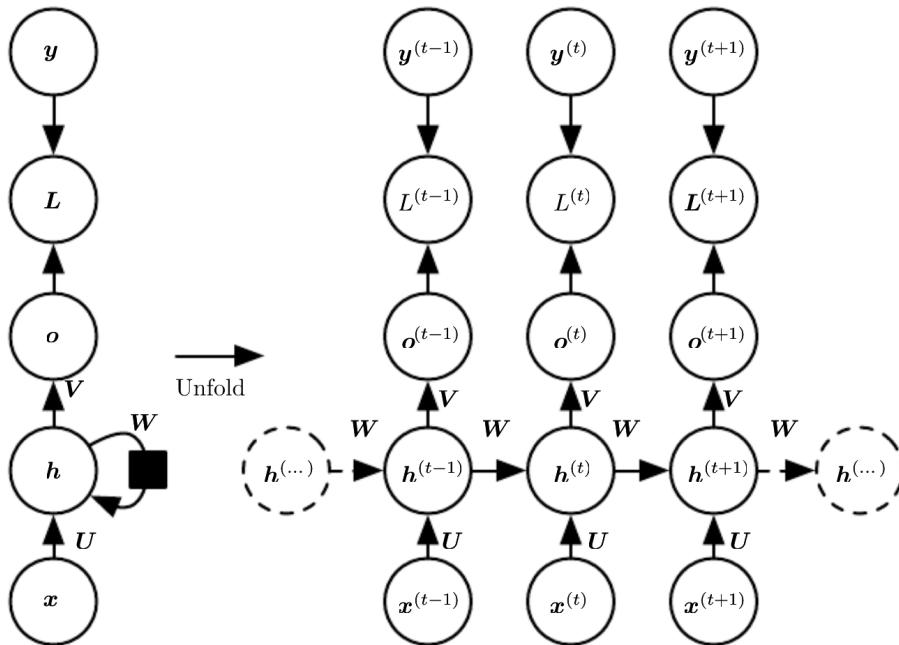
Postoje mnoge varijacije modela rekurentnih mreža. Mi ćemo formalnije opisati osnovni model sa jednim skrivenim slojem:

$$\begin{aligned} h^{(0)} &= 0 \\ h^{(t)} &= g \left(b_h + Wh^{(t-1)} + Ux^{(t)} \right) \\ o^{(t)} &= g \left(b_o + Vh^{(t)} \right) \end{aligned}$$

za $t = 1, 2, \dots, T$, gde je T dužina sekvene i može se razlikovati od sekvene do sekvene. Vektor $x^{(t)}$ predstavlja ulaz u trenutku t , $h^{(t)}$ vektor vrednosti skrivenog sloja, g je nelinearna aktivaciona funkcija, $o^{(t)}$ je vektor izlaza mreže koji daje predviđanje prave vrednosti $y^{(t)}$. Parametri su matrica transformacije stanja u stanje W , ulaza u stanje U , stanja u izlaz V i vektori slobodnih članova b_h i b_o . Svi parametri će skupa biti označavani sa w . Jedna grafička ilustracija rekurentne mreže data je na slici 10.17. Stil ilustracije koji se češće koristi i koji sakriva broj jedinica po slojevima, a naglašava vremenku dimenziju dat je na slici 10.18.



Slika 10.17: Ilustracija rekurentne neuronske mreže. Strelice između različitih jedinica istog sloja, označavaju referisanje na vrednosti tih jedinica iz prošlog trenutka.



Slika 10.18: Ilustracija rekurentne neuronske mreže. Levo je data kompaktna reprezentacija, a desno takozvano razmotavanje mreže kroz vreme. Svaki krug predstavlja ceo sloj jedinica. Takođe su označeni i funkcija greške i prave vrednosti ciljne promenljive. Kvadrat na strelici označava referisanje na vrednosti datog skrivenog sloja u prethodnom trenutku.

Može biti zanimljiva informacija da je dati model rekurentne neuronske mreže Tjuring potpun. Pritom, broj koraka u kojem rekurentna neuronska mreža može da izvrši izračunavanje je asimptotski linearan u odnosu na broj koraka potreban Tjuringovoj mašini. Ipak, treba imati u vidu da ovakva tvrdnja znači samo postojanje nekog modela koji je u stanju da izvrši željeno izračunavanje. To ne znači da je taj model lako naći. Zapravo, kao ni druge vrste neuronskih mreža, rekurentne mreže nisu luke za obučavanje.

Problem učenja može biti različito postavljen u zavisnosti od specifičnosti arhitekture. Ipak, kao i u drugim slučajevima, minimizuje se srednja greška. Stoga, nakon što je definisan model, ključno je definisati funkciju greške. Do sada je to bilo lako zato što su sve instance (parovi vrednosti x i y) bile nezavisne. Sada obrazuju sekvene i u opštem slučaju, greška se računa na osnovu sekvene vrednosti y i sekvene predviđanja, odnosno funkcija greške ima oblik:

$$L([y^{(1)}, \dots, y^{(T)}], [f_w^{(1)}, \dots, f_w^{(T)}])$$

gde se uglasim zagradama označava sekvenca, a $f_w^{(t)}$ predstavlja predviđanje modela u koraku t i moglo bi se zapisati kao

$$f_w^{(t)} = f_w([x^{(1)}, \dots, x^{(t)}])$$

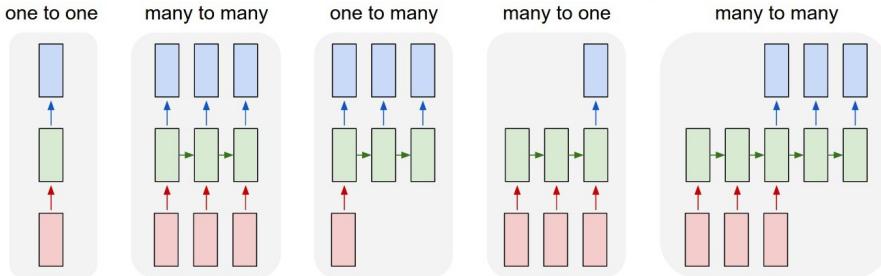
Česta je prepostavka da se funkcija greške može razložiti na sledeći način:

$$L([y^{(1)}, \dots, y^{(T)}], [f_w^{(1)}, \dots, f_w^{(T)}]) = \sum_{t=1}^T L(y^{(t)}, f_w^{(t)})$$

Ipak, postoje modeli u kojima se ne prepostavlja da tekuća vrednost ciljne promenljive zavisi samo od prethodnih vrednosti ulazne sekvene. U nekim slučajevima zavisi i od narednih vrednosti ulaza. Recimo, prirodno je da vrsta reči u rečenici ne zavisi samo od prethodnih reči, već i od narednih.

10.3.2 Graf izračunavanja, njegovo razmotavanje i obučavanje rekurentnih mreža

Graf izračunavanja je važan tehnički pojam koji olakšava razumevanje rekurentnih neuronskih mreža, ali još važnije, predstavlja osnovu implementacija različitih biblioteka za rad sa proizvoljnim neuronskim mrežama. Iako je značaj pojma opštiji, uvodimo ga tek ovde jer je prvi put koristan za objašnjavanje suštinskih koncepata. Računski graf je graf čiji su čvorovi promenljive koje mogu predstavljati bilo ulaze, bilo međurezultate izračunavanja. Izračunavanje se vrši primenom operacija nad promenljivim. Ukoliko se vrednost promenljive y dobija primenom neke operacije na osnovu promenljive x (ali ne nuzno samo na osnovu nje), onda postoji grana od promenljive x do promenljive y . Ovakav graf se ne koristi samo za predstavljanje modela, već svih izračunavanja koja su uključena u postavku optimizacionog problema, što znači i regularizaciju, funkciju greške itd. Prikaz rekurentne neuronske mreže na slici 10.18 upravo



Slika 10.19: Nekoliko različitih arhitektura rekurentnih neuronskih mreža.

predstavlja graf izračunavanja. Alternativni prikaz računskog grafa se dobija *razmotavanjem* (eng. *unfolding*) računskog grafa, koji je prikazan na istoj slici.

Razmotavanje grafa pokazuje da se na jednoj sekvenci rekurentna mreža može posmatrati kao duboka potpuno povezana mreža sa velikim brojem nadovezanih skrivenih slojeva $h^{(0)}, \dots, h^{(T)}$ i velikim brojem deljenih parametara. Ovim zapažanjem je sugerisano i kako se neuronske mreže mogu obučavati – propagacijom unazad po razmotanom računskom grafu. Pritom, različite sekvene koje se istovremeno nalaze u skupu za obučavanje će dati različite gradijente, ali se ti gradijenti jednostavno sabiraju zahvaljujući tome što je greška na celom skupu za obučavanje prosti zbir (ili prosek) grešaka na pojedinačnim sekvencama. U ovom kontekstu, algoritam se naziva *propagacijom kroz vreme* (eng. *backpropagation through time*) i zahteva određena uopštenja, pošto u razmotanom grafu jedinice nisu složene po slojevima u duhu potpuno povezane mreže. Recimo, izlazi nisu samo na poslednjem nivou razmotanog grafa, već na T različitih nivoa. Međutim, ta uopštenja nisu komplikovana.

Deljenje parametara čini da dubina mreže koja se proizvodi razmotavanjem grafa, nije prepreka obučavanju u smislu fleksibilnosti modela, ali i dalje postoje drugi problemi poput velikog broja nadovezanih množenja izvoda pri računanju gradijenta, što lako dovodi do nestajanja ili eksplozije gradijenata. Delimično rešenja za ovakve probleme ćemo videti kasnije.

10.3.3 Varijacije arhitekture

Do sada razmatrana formulacija pretpostavlja da mreža preslikava ulaznu sekvencu u izlaznu sekvencu iste dužine. Ovo ne mora biti tačno. Zapravo, arhitekture koje se koriste u praksi mogu biti raznovrsne. Slika 10.19 prikazuje neke od varijacija. Poređenja radi, prva slika prikazuje trivijalan slučaj bez rekurentnosti, koji odgovara potpuno povezanoj mreži. Druga predstavlja do sada diskutovanu formulaciju. Treća prikazuje mrežu kojoj se daje samo jedan ulaz, a ona generiše izlaz promenljive dužine. Primer primene u kojoj je ova formulacija korisna je opisivanje slika rečenicama. U tom slučaju ulaz je jedan – slika koju treba opisati, dok dužina opisa prirodno može varirati. U vezi ove

formulacije postavljaju se dva pitanja – kako se mreži daje samo jedan ulaz i kako mreža završava sa davanjem izlaza. Prvo pitanje je lako rešiti. Ili će ulaz biti dat kao početno skriveno stanje $h^{(0)}$, a mreža neće imati nikakve ulaze ili će mreži u svakom koraku biti prikazivan isti ulaz. Pitanje zaustavljanja se takođe može rešiti na više načina, ali jedan jednostavan način je uvođenje još jednog, sigmoidnog, izlaza mreže koji će davati indikaciju da li treba zaustaviti generisanje izlaza ili treba nastaviti. U vreme obučavanja, tom izlazu se recimo pridružuje 1 za svaki ulazni korak koji nije poslednji, a 0 za poslednji ulazni korak.

Četvrta varijanta prikazuje mrežu kod koje nizu ulaza odgovara jedan izlaz. Ovakva formulacija može biti korisna na primer u klasifikaciji vremenskih serija, tekstova i slično, kada se celoj sekvenci pridružuje neka klasa. Ovakva arhitektura se može realizovati na osnovu polazne arhitekture uz izmenu funkcije greške, koja će pri obradi sekvence obraćati pažnju samo na poslednju vrednost, odnosno formu funkcije greške je:

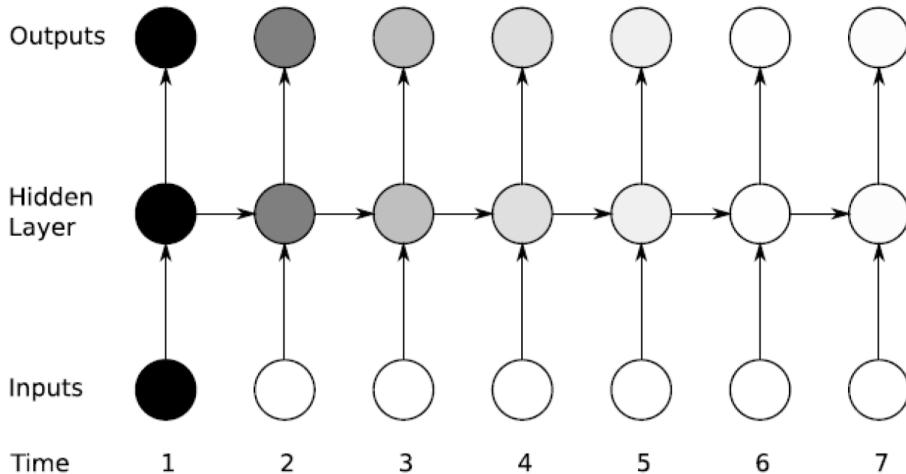
$$L([y^{(1)}, \dots, y^{(T)}], [f_w^{(1)}, \dots, f_w^{(T)}]) = L(y^{(T)}, [f_w^{(1)}, \dots, f_w^{(T)}])$$

Poslednja arhitektura, u kojoj generisanje izlaza počinje od poslednjeg elementa ulaza, naziva se *enkoder-dekoder* arhitekturom i karakteristična je recimo za problem mašinskog prevođenja i dobija se kombinovanjem četvrte i treće arhitekture. Enkoder je deo mreže koji čita ulaz, na primer rečenicu na srpskom jeziku, i od njega generiše skriveno stanje. Kada se čitanje ulaza završi, skriveno stanje predstavlja apstraktu reprezentaciju ulaza, koja se nekad naziva misao. Tu misao treba izreći na drugom jeziku, recimo engleskom. Deo mreže koji generiše izlaz, na isti način kao u trećem slučaju naziva se dekoder. Kao u slučaju četvrte arhitekture, funkcija greške uzima u obzir samo izlaze generisane od poslednjeg koraka ulaza. Moguće su i drugačije arhitekture.

Rekurentna mreža takođe može biti duboka. U tom slučaju dubina se može postići nadovezivnjajem potpuno povezanih slojeva između ulaza i skrivenog sloja ili između skrivenog sloja i izlaza, pa čak i između koraka skrivenog sloja.

10.3.4 Duga kratkoročna memorija

Postoje dva osnovna problema rekurentnih neuronskih mreža u njihovoј osnovnoј formi. Prvi se tiče problema nestajućih i eksplodirajućih gradijenata. Naime, graf izračunavanja rekurentne neuronske mreže je tipično vrlo dubok zbog velike dužine sekvence. Usled toga, prilikom izračunavanja gradijenta propagacijom kroz vreme, dolazi do velikog broja množenja koja neretko čine da koordinate gradijenta ili eksplodiraju ili nestanu. Prvi ishod čini optimizacione metode nestabilnim, a drugi sporim. Drugi problem se odnosi na dugoročno čuvanje informacije i modelovanje dugoročnih zavisnosti u podacima. Kako se skriveno stanje u svakom koraku dobija linearnom kombinacijom prethodnog stanja i ulaza, doprinos starijih ulaza se brzo gubi pod uticajem novih, kao što slika 10.20 ilustruje. Dugoročno čuvanje relevantnih informacija nije moguće. Oba ova problema se prevazilaze upotrebom *duge kratkoročne memorije*



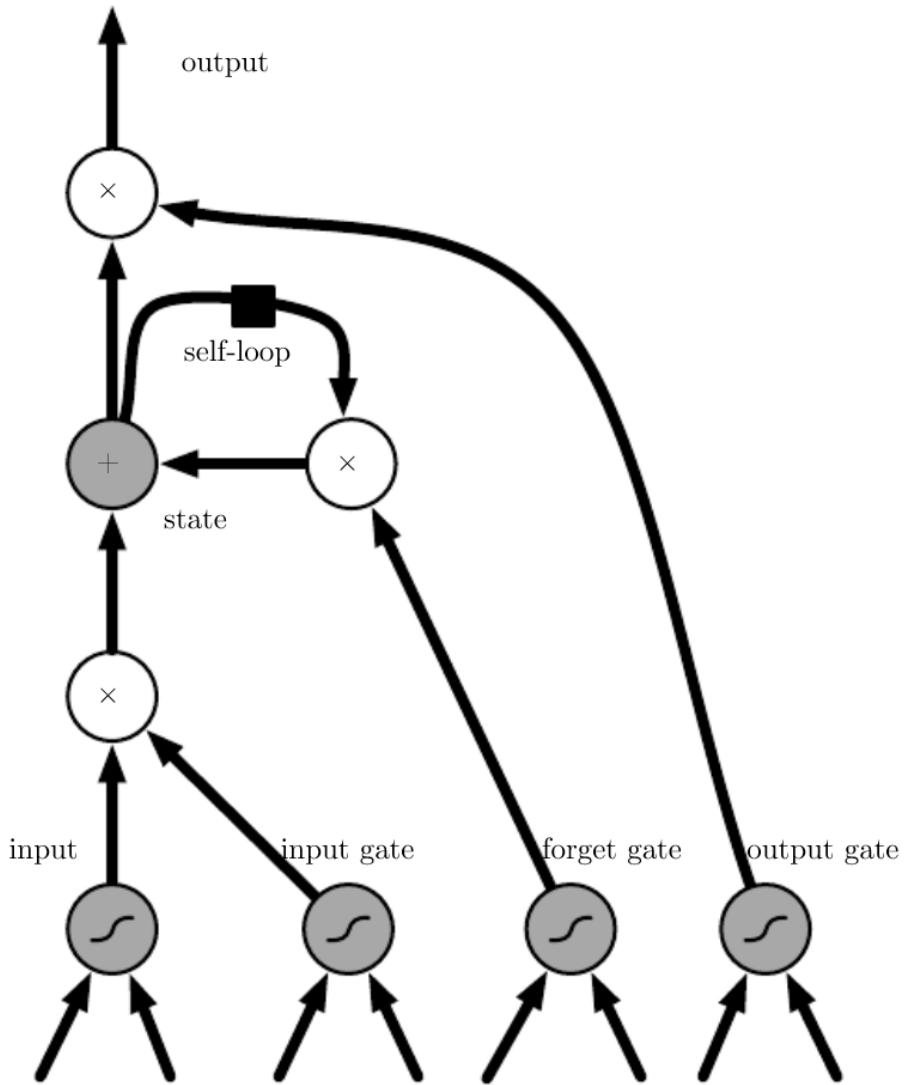
Slika 10.20: Ilustracija nemogućnosti dugoročnog čuvanja informacije u standardnoj rekurentnoj neuronskoj mreži.

(eng. *long short term memory*), skraćeno LSTM, što je složena jedinica mreže sa specifičnom strukturu koja omogućava kontrolu čitanja i upisa u jedinicu. Upravo ova vrsta jedinice dovela je do ključnih uspeha rekurentnih neuronskih mreža i predstavlja standardni izbor prilikom formulisanja modela rekurentne mreže.

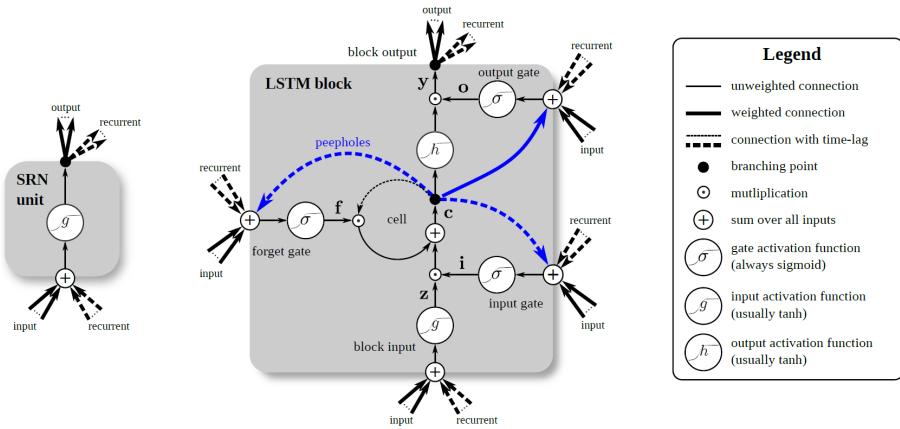
Osnovna ideja LSTM-a je postojanje takozvane ćelije koja čuva skriveno stanje, uz kontrolu pisanja, čitanja i zaboravljanja, koja se vrši na osnovu naučenih pravila. Na slici 10.21 prikazana je struktura LSTM jedinice, koja prikazuje ćeliju u centru i nekoliko kapija koje kontrolišu prethodno pomenute aspekte. Još jedan prikaz dat je na slici 10.22. Formule koje je definišu su sledeće:

$$\begin{aligned}
 c^{(0)} &= 0 && \\
 h^{(0)} &= 0 && \\
 z^{(t)} &= g(W_z x^{(t)} + U_z h^{(t-1)} + b_z) && \text{Transformisani ulaz} \\
 i^{(t)} &= \sigma(W_i x^{(t)} + U_i h^{(t-1)} + b_i) && \text{Ulagna kapija} \\
 f^{(t)} &= \sigma(W_f x^{(t)} + U_f h^{(t-1)} + b_f) && \text{Kapija zaboravljanja} \\
 c^{(t)} &= z^{(t)} \odot i^{(t)} + c^{(t-1)} \odot f^{(t)} && \text{Ćelija} \\
 q^{(t)} &= \sigma(W_q x^{(t)} + U_q h^{(t-1)} + b_q) && \text{Izlazna kapija} \\
 h^{(t)} &= g(c^{(t)}) \odot q^{(t)} && \text{Izlaz}
 \end{aligned}$$

gde c označava ćeliju koja čuva stanje LSTM jedinice, z transformisanu vrednost ulaza, i vrednost ulazne kapije, f vrednost kapije za zaboravljanje, q vrednost izlazne kapije, a h vrednost izlaza LSTM jedinice. Svaka od kapija ima svoj skup parametara označen odgovarajućim indeksom. Iako deluju



Slika 10.21: Struktura LSTM jedinice.



Slika 10.22: Struktura LSTM jedinice (desno) u poređenju sa standardnom rekurentnom jedinicom (levo). Oznaka o na slici, odgovara oznaci q u našim formulama.

rogobatno, suština je prilično jednostavna. Svaka od kapija ima istu strukturu kao jedinica standardne rekurentne mreže i na osnovu ulaza koje dobija i pridruženih im parametara odlučuje o tome da li i u kolikoj meri dozvoljava izvršavanje operacije koju kontroliše. Na primer, ulazna kapija kontroliše da li će propustiti ulaz ka ćeliji i to radi tako što na osnovu ulaza (prva suma u definiciji ulazne aktivacije) i svog stanja u prethodnom koraku (druga suma) računa koeficijent koji se koristi za kontrolu uticaja ulaza na stanje ćelije. Slično, kapija za zaboravljanje na analogan način kontroliše uticaj prethodnog stanja ćelije na novo stanje ćelije. Postoje različite modifikacije LSTM-a, ali se ovde ograničavamo na njegovu osnovnu formu.

LSTM je značajan iz dva razloga. Prvo, zahvaljujući kapijama koje mogu da kontrolišu ulaz u ćeliju, ćelija ne mora da prihvata ulazne signale i stoga može dugo da čuva informaciju o dalekim delovima sekvene. Da li treba da prima ulazne signale ili ne, je nešto što se uči zahvaljujući tome što je ulazna kapija parametrizovana. Treba imati u vidu da se u praksi paralelno koristi veći broj ovakvih jedinica, tako da dok neke čuvaju informaciju iz ranijih delova sekvene, druge mogu da obrađuju tekući ulaz, da ga kombinuju sa informacijom iz onih prvih i slično. Drugi razlog iz kojeg je LSTM značajan je ublažavanje problema nestajućih gradijenata. Naime, u slučaju obične rekurentne mreže, nova vrednost skrivenog sloja se dobija tako što se kao ulaz uzima izlaz tog sloja u prethodnom koraku, koji je transformisan aktivacionom funkcijom. Nadovezivanje aktivacionih funkcija u računskom grafu vodi tome da se prilikom računanja izvoda, množe izvodi aktivacionih funkcija. U slučaju sigmoidne funkcije izvod može biti najviše 0.25, a u slučaju tangensa hiperboličkog je strogo manji od 1. Množenjem ovakvih brojeva gradijent nestaje. S druge strane, prilikom računanja nove vrednosti ćelije, nema primene

aktivacione funkcije. Naravno, prethodna vrednost se množi vrednošću kapije za zaboravljanje, koja uključuje sigmoidnu aktivacionu funkciju, pa se njen izvod mora pojaviti negde u računanju gradijenata, ali postojanje putanja u računskom grafu koje nisu zahvaćene ovim problemom ima osetan efekat. Na primer, standardna rekurentna mreža obično ne modeluje dobro zavisnosti na rastojanju većem od desetak koraka. S druge strane, rekurentnim mrežama sa LSTM jedinicom se uspešno modeluju zavisnosti i na rastojanju od više stotina koraka.

10.3.5 Kvaliteti i mane

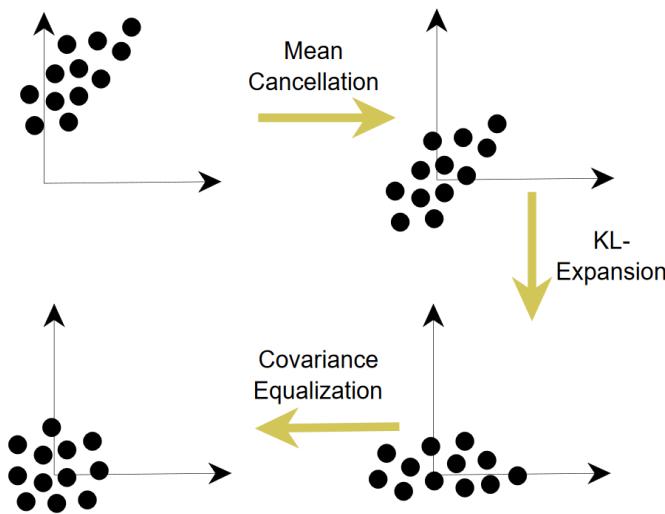
Zahvaljujući tome što modeluju prelaze iz stanja u stanje, umesto direktnе zavisnosti izlaza od celokupnog ulaza, rekurentne mreže su u stanju da predstave vrlo duboke strukture zavisnosti i da obrađuju sekvene promenljive, a velike dužine. Zahvaljujući LSTM jedinicama, takođe su u stanju da ublaže problem nestajućih gradijenata i da modeluju dugoročne zavisnosti. Ipak, zbog kompleksnosti podataka koje modeluju i broja parametara koje LSTM jedinice imaju, nije ih lako obučavati (npr. jedna od poteškoća je haotično ponašanje u slučaju velikih apsolutnih vrednosti parametara). Poznato je da je njihovo obučavanje teže hardverski ubrzati zbog visokog obima memorijskog protoka i sekvencialne prirode izračunavanja (dok hardversko ubrzavanje počiva na paralelizaciji), što je ključ efikasnog obučavanja neuronskih mreža.

10.4 Praktične tehnike i napredni koncepti

Obučavanje neuronskih mreža je ozloglašeno težak posao. Problemi se kriju kako u modelovanju, tako i u optimizaciji. U modelovanju, glavni problem je u nagodbi između potencijala za preprilagođavanje i mogućnosti dovoljnog prilagođavanja podacima, ali i u ostvarivanju drugih, domenski specifičnih ciljeva. U optimizaciji, ima ih više i vezani su za brzinu i stabilnost konvergencije. Dodatno, problemi vezani za modelovanje i problemi vezani za optimizaciju često interaguju. Uprkos tome, pomoću dubokih neuronskih mreža postignuti su zadivljujući praktični uspesi. To otvara i dodatna teorijska pitanja vezana za razumevanje faktora koji vode bržem učenju i boljoj generalizaciji. U nastavku su diskutovani napredni koncepti koji se tiču kako praktičnih tehnika, tako i teorijskih pitanja.

10.4.1 Inicijalizacija parametara

Polazne vrednosti parametara mogu imati značajnog odraza na brzinu konvergencije i na kvalitet dobijenog modela. Naivna prepostavka bi mogla biti da se koeficijenti mogu inicijalizovati na nule. Ova prepostavka je vrlo loša ne samo zbog izbora vrednosti, već prvenstveno zbog izbora istih vrednosti za različite jedinice, što vodi njihovom vrlo sličnom ponašanju makar u polaznim



Slika 10.23: Transformacija podataka – oduzimanje proseka, dekoralacija, de-ljenje standardnom devijacijom.

fazam optimizacije, što usporava proces učenja i smanjuje raznovrsnost atributa koje mreža konstruiše. Otud je bitno da parametri budu različiti i najčešće se koriste slučajno izabrane vrednosti iz raspodela visoke entropije poput normalne ili uniformne. Postoje empirijski i ne preterano uverljivi teorijski razlozi da se vrednosti parametara nasumice generišu iz raspodele

$$U \left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}} \right]$$

10.4.2 Unutrašnja standardizacija

Pri primeni neuronskih mreža, kao i mnogih drugih metoda mašinskog učenja, poželjno je da ulaz bude standardizovan. Zapravo, poželjno je da atributi budu standardizovani i dekorelisani, odnosno da im je matrica kovarijacije identička. U suprotnom se očekuje problem izduženih kontura funkcije i cik-cak kretanje gradijentnih metoda. Na slici 10.23 prikazan je proces transformacije podataka tako da imaju poželjna svojstva. U tom slučaju, konvergencija optimizacionih metoda je obično brža. Pomenuta svojstva će često biti poželjnija ako se na ulaze primeni analiza glavnih komponenti (PCA), ali ona nije praktična za visokodimenzionalne ulaze. Stoga se ulazi obično samo standardizuju, što u nekoj meri takođe pomaže. Kao što je korisno da ulazi mreže budu standardizovani, korisno je i da izlazi svakog od slojeva budu standardizovani, pošto i oni predstavljaju ulaze u nove slojeve i mogu se videti kao ulazi ostatka mreže. Ipak, to u toku obučavanje mreže nije lako obezbediti.

Kako se parametri slojeva mreže menjaju, tako se menjaju i raspodele njihovih izlaza. Ukoliko bi se takvo ponašanje otklonilo, za očekivati je da bi konvergencija bila brža. Upravo u tome je poenta tehnike *unutrašnje standardizacije*¹ (eng. *batch normalization*). Ova tehnika se sastoji u tome da se na nivou podskupa podataka na kome se računa gradijent (što u stohastičkoj varijanti ne mora biti ceo skup podataka za obučavanje) vrši standardizacija izlaza svake jedinice. Za različite ulaze, jedinica u daje različite izlaze. Na podskupu instanci \mathcal{B} , prosek izlaza je $\mu_{\mathcal{B}}^u$, a standardna devijacija je $\sigma_{\mathcal{B}}^u$.² Primetimo da je neophodno da važi $|\mathcal{B}| > 1$. Prilikom obučavanja, ovi parametri se računaju za svaku jedinicu pre izvršavanja algoritma propagacije unazad. Prilikom propagacije unapred, izlaz svake jedinice se standardizuje pomoću ova dva parametra na uobičajen način. Prilikom izračunavanja gradijenata, uzima se u obzir i izvod ove transformacije. Prilikom primene mreže u predviđanju na novoj instanci, skup podataka na koji se mreža primenjuje je jednočlan, pa parametri $\sigma_{\mathcal{B}}^u$ nisu definisani, a parametri $\mu_{\mathcal{B}}^u$ se računaju na pojedinačnim instancama, što nije poželjno. Stoga se prilikom predviđanja koriste ocene ovih parametara koje su dobijene na osnovu proseka vrednosti ovih parametara u završnim iteracijama obučavanja, tipično korišćenjem pokretnog proseka koji je računat sve vreme u toku obučavanja.³

Problem ovako opisane transformacije je u tome da insistiranje na standarizovanosti smanjuje izražajnu moć neuronske mreže, koja stoga ima manju moć učenja. Kako bi se ovo izbeglo, vrši se sledeća neobična transformacija. Ako je \hat{h}^u standardizovana vrednost izlaza jedinice u , umesto te vrednosti, koristi se vrednost $\alpha^u \hat{h}^u + \beta^u$, gde su α^u i β^u parametri koji se uče kao i ostali parametri mreže.⁴ Ovim se omogućava da jedinica nauči proizvoljan prosek i standardnu devijaciju svojih izlaza. Ovakva mreža može da izrazi iste funkcije kao i polazna mreža koja ne koristi unutrašnju standardizaciju. Ali ako ovi me omogućavamo da izlazi jedinica ne budu standardizovani, zašto smo onda uopšte vršili standardizaciju?! Razlog je taj što su u polaznoj mreži proseci i standardne devijacije izlaza jedinica bili proizvod komplikovanih interakcija parametara u prethodnim slojevima mreže. U novoj mreži, ove vrednosti su određene samo parametrima α^u i β^u , zbog čega je mreža koja koristi unutrašnju standardizaciju mnogo lakše obučavati. U praksi, korišćenjem ove tehnikе dobijena su ubrzanja u obučavanju veća od 10 puta, uz povećanje tačnosti dobijanog modela. Ovaj vid standardizacije dozvoljava i veće vrednosti koraka u procesu optimizacije zahvaljujući tome što ublažava problem izduženih kontura. Pоказало se da korišćenje ove tehnike neretko čini i regularizaciju nepotrebnom,

¹Prevod ne odgovara bukvalno izvornom terminu, ali izvorni termin ne odgovara onome što označava.

²Prilikom računanja standardne devijacije, pod korenom se na uzoračku varijansu dodaje mala pozitivna vrednost, poput 10^{-8} , kako se ne bi desilo da standardna devijacija bude nula, što bi onemogućilo njeno korišćenje u standardizaciji.

³Pokretni prosek m_i u trenutku i nekog niza vrednosti a_0, \dots, a_n se računa kao $m_i = \alpha m_{i-1} + (1 - \alpha)a_i$ za neko $0 \leq \alpha \leq 1$.

⁴U slučaju konvolutivnih mreža, parametri α i β se ne pridružuju filterima, ne njihovim pojedinačnim primenama na različitim delovima ulaza u konvoluciju.

pošto se sama može videti kao vid regularizacije zasnovan na unošenju šuma. Ova vrsta regularizacije će biti diskutovana kasnije.

10.4.3 Stohastičko obučavanje

Neuronske mreže se najčešće obučavaju stohastičkim gradijentnim spustom ili stohastičkom varijacijom nekog drugog algoritma optimizacije. Prednosti stohastičkih varijanti su već objašnjene. Kako čist stohastički pristup vodi lošoj oceni gradijenta, gradijent se obično ne računa samo na jednoj instanci, već na nekom podskupu (eng. *minibatch*). Veličina podskupa nije definisana, ali red veličine bi se mogao meriti na primer desetinama. Ipak, ni to nije pravilo koje je neophodno poštovati. Podskupovi se definišu nasumičnim particionisanjem celog skupa za obučavanje. Nasumičnost se često postiže time što se instance izmešaju pre deljenja. Ona je važna jer ukoliko se desi da su instance poređane po nekom pravilu, učenje može biti sporije. Na primer, ako su poređane po vrednosti ciljne promenljive i prvi korak se izvrši na skupu instanci sa najmanjom vrednošću ciljne promenljive, a potom na vrlo sličnim instancama, zbog sličnosti instanci, promena koeficijenata neće biti velika, pa će i konvergencija biti spora.

U slučaju rekurentnih mreža, ne mora biti očigledno kako je potrebno deliti podatke na podskupove. Osnovno je pravilo da sve instance koje pripadaju jednoj sekvenci pripadaju istom podskupu, jer ih je mreži potrebno davati u uzastopnim koracima. Postoji mogućnost i njihovog razdvajanja, ali to vodi komplikovanoj proceduri obučavanja ili gubitku informacije o dugoročnim zavisnostima koje u sekvenci postoje.

10.4.4 Odsecanje gradijenata

Jedan od pomenutih problema u obučavanju neuronskih mreža su eksplodirajući gradijenti. Opasnost koju ovaj problem nosi lako je razumeti. Naime, gradijent daje informaciju o pravcu najbržeg uspona u nekoj tački. Ta informacija je lokalna u smislu da ima značaj u maloj okolini tačke u kojoj se gradijent računa. Daleko od te tačke, funkcija se može ponašati drugačije i vrednost joj čak može biti manja, čak i ako tačka leži u pravcu gradijenta u odnosu na tačku u kojoj se gradijent računa. Stoga, gradijenti velike norme često nisu poželjni.

Jedno jednostavno rešenje ovog problema je *odsecanje gradijenata* (eng. *gradient clipping*). Ova operacija nije jednoznačno definisana i može se vršiti bar na dva načina. Jedan je da se koordinate gradijenta pojedinačno zaokruže na interval $[-c, c]$ za neki meta parametar c , tako što se vrednosti u tom intervalu ne modifikuju, ali se one koje su veće od c zaokružuju na c , a one koje su manje od $-c$ zaokružuju na $-c$. Ova operacija očito menja pravac gradijenta. Druga varijanta, matematički intuitivnija je odsecanje norme gradijenta, tako što se ukoliko važi $\|\nabla E(w)\| > c$, gradijent zameni vektorom

$$c \frac{\nabla E(w)}{\|\nabla E(w)\|}$$

Ovo rešenje deluje dosta bolje od prethodnog, pošto je pravac gradijenta očuvan. Ipak, u praksi se ponašaju približno jednako. Primetimo jedan problem ovog rešenja. U slučaju stohastičkog obučavanja, bilo na pojedinačniminstancama, bilo na podskupovima, odsecanje norme čuva pravac gradijenta na podskupu na kojem se gradijent trenutno rečuna, ali, u smislu pravca, prosek gradijenata odsecene norme nije nepričasna ocena gradijenta na celom skupu podataka. Ukoliko se u nekom podskupu javi jedna instanca koja dovodi do eksplozije gradijenta, zahvaljujući odsecanju, doprinos svih instanci iz istog podskupa ukupnom gradijentu je manji.

10.4.5 Regularizacija

Kako neuronske mreže predstavljaju vrlo fleksibilne modele, razvoj adekvatnih tehnika njihove regularizacije je jedan od najvažnijih pravaca istraživanja u ovoj oblasti. Neuronske mreže koriste kako opšte vrste regularizacije, tako i neke vrlo specifične za ovu vrstu modela. Naglasimo da se različite vrste regularizacije mogu koristiti zajedno.

Od opštih vrsta regularizacije, najkorišćenja je ℓ_2 regularizacija. Pritom, ne koristi se uvek isti regularizacioni parametar za sve nivoe mreže. ℓ_1 regularizacija je manje važna u ovom kontekstu, pošto se prilikom korišćenja neurosnih mreža retko teži interpretabilnosti. Valja napomenuti da se slobodni parametri svih jedinica tipično izostavljaju iz regularizacije. Njihovo regularizovanje tipično vodi potprilagođavanju.

Jedna tehnika regularizacije koja je specifična za neuronske mreže je *rano zaustavljanje* (eng. *early stopping*). Poznato je da u toku procesa učenja, greška na skupu za obučavanje ima trend opadanja, dok bi na odvojenom skupu podataka greška prvo opadala, zahvaljujući učenju, a onda rasla, zahvaljujući preprilagođavanju. Optimalan model se nalazi na minimumu krive koja odgovara tom odvojenom skupu podataka. Tehnika ranog zaustavljanja se zasniva upravo na ovom zapažanju i sastoji se u izdvajaju posebnog validacionog skupa iz skupa za obučavanje, obučavanju modela unapred određeni broj iteracija i pamćenju svakog modela dobijenog u toku optimizacije koji na tom validacionom skupu ima manju grešku od najbolje prethodno zabeležene. Na kraju se bira model koji je dao najbolje rezultate na skupu za validaciju. Metodima regularizacije uvek odgovara neki hiperparametar. U ovom slučaju, to je broj iteracija obučavanja, koji se ponaša obrnuto od regularizacionog hiperparametra u slučaju ℓ_2 regularizacije. Mana ovog pristupa je očito to što se na jednom delu podataka ne vrši obučavanje. Međutim, to se tipično nadoknađuje time što se izvrši naknadno obučavanje na svim podacima. Postavlja se pitanje kako se u tom slučaju zaustavlja obučavanje. Postoji više načina da se to odredi, ali najbolji sa tačke gledišta tačnosti dobijenog modela, ali ne nužno i utrošenog vremena, je da se obučavanje započne iz početka na svim podacima za obučavanje u istom broju koraka u kojem je dobijen najbolji model prilikom upotrebe ranog zaustavljanja. Ima smisla postaviti pitanje u kom smislu je ovo tehnika regularizacije. Odgovora su dva. Intuitivno, sprečava preprilagođavanje

podacima za obučavanje time što ranije zaustavlja obučavanje. Tehnički, u slučaju linearnih modela (što neuronske mreže nisu) može se pokazati ekvivalentnost ove tehnike sa ℓ_2 regularizacijom. Osnova ove ekvivalentnosti je u tome da u slučaju ograničene norme gradijenta i ograničene dužine koraka prilikom optimizacije, ograničavanje broja iteracija ograničava udaljenost krajnjih parametara od polaznih i time smanjuje skup potencijalnih modela iz kojih se bira krajnji model. Slično ℓ_2 regularizacija ograničava skup modela na sve modele čiji se parametri nalaze u lopti određenog poluprečnika.

Još jedna od tehnika specifičnih za neuronske mreže je *unošenje šuma* (eng. *noise injection*) i odnosi se na dodavanje pseudoslučajno generisanih brojeva, najčešće uzorkovanih iz normalne raspodele, na vrednosti atributa, na vrednost ciljne promenljive, na vrednosti parametara ili na vrednosti skrivenih slojeva mreže. Naravno, moguće je i kombinovati ove izbore. Razlog za regularaciono ponašanje ove tehnike je taj što očito ometa mrežu u savršenom prilagođavanju podacima. Primena ove tehnike u praksi je pipava, jer je potrebno pažljivo odrediti intenzitet šuma. Premala količina ne daje efekat regularizacije, dok prevelika ne dozvoljava mreži da nauči korisne zakonitosti.

Jedna od najuspešnijih tehnika regularizacije za neuronske mreže je regularizacija *izostavljanjem* (eng. *dropout*) i aproksimira ansambl (skup modela koji zajedno predviđaju uprosečavanjem ili glasanjem) svih neuronskih mreža koje se mogu dobiti iz polazne mreže postavljanjem izlaza nekih jedinica na nulu, odnosno njihovim izostavljanjem iz mreže. Jedinice čiji se izlazi postavljaju na nulu se biraju nasumice, tako što se svaka bira sa nekom verovatnoćom, koja predstavlja hiperparametar regularizacije. Treba imati u vidu da se ni u jednom trenutku ne obučava veliki broj mreža, već samo jedna, ali se prilikom svakog izračunavanja gradijenta, bira podskup jedinica koje će biti anulirane. Na taj način se svaki korak učenja vrši u odnosu na različitu mrežu. Ipak, ove mreže nisu nezavisne (što bi bilo optimalno) jer dele parametre, ali zahvaljujući tome je proces obučavanja efikasan. Prilikom predviđanja, daju se predviđanja pomoću određenog broja mreža uzorkovanih anuliranjem jedinica iz finalnog modela i njihove vrednosti se uprosečavaju kako bi dale finalno predviđanje. Postoji i drugi, računski efikasniji način predviđanja, ali u njega nećemo ulaziti. Pored interpretacije u smislu ansambla, regularizacija izostavljanjem ima još jednu, možda i direktniju, interpretaciju. Naime, izostavljanjem jedinica, forsira se robusnost mreže. Mreža se ne sme osloniti na neki detalj koji jedna jedinica opaža, što bi moglo biti preprilagođavanje. Na primer, ako u konvolutivnoj mreži, jedan filter detektuje nos, njegovim izostavljanjem, mreža se forsira da prepoznavanje ispravno vrši i samo na osnovu usta i očiju, koje nalaze drugi filteri.

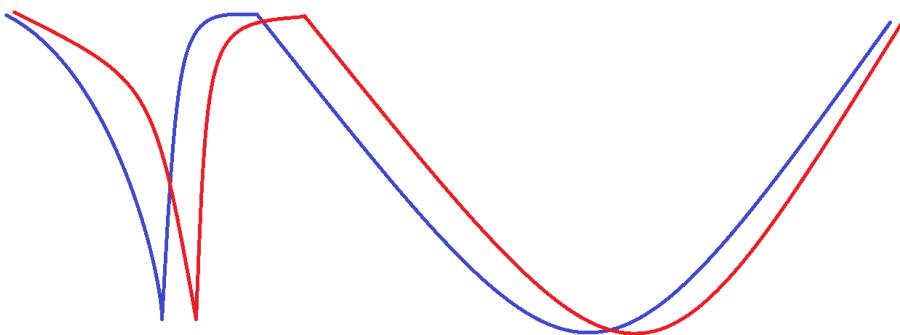
10.4.6 Objasnjenja uspešnosti dubokih neuronskih mreža

Već je rečeno da neuronske mreže imaju svojstvo univerzalne aproksimacije. Ipak, teorema koja to tvrdi ne govori koliko velika mreža treba da bude da bi postigla određenu preciznost. Ispostavlja se da je u nekim slučajevima potrebno

eksponencijalno mnogo jedinica u skrivenom sloju u odnosu na broj atributa, kako bi se ovo postiglo. Pod određenim prepostavkama, može se pokazati da povećanje dubine neuronske mreže vodi eksponencijalnom smanjenju broja neophodnih skrivenih jedinica, što ukazuje na potencijalni razlog za uspešnost dubokih neuronskih mreža. Ipak, nije sigurno koliko su ove prepostavke realistične. Ipak, nekada duboka arhitektura prosto odgovara svojstvima problema, kao što se veruje za konvolutvne i rekurentne mreže u slučaju obrade slika i sekvenci.

Jedno od velikih pitanja vezanih za uspešnost dubokih neuronskih mreža tiče se uspešnosti optimizacionih metoda u pronalaženju dovoljno dobrih minimuma funkcije greške. Naime, imajući u vidu duboku komponovanost ovih mreža, očekuje se da funkcija greške ima veliki broj lokalnih minimuma. S druge strane, uspešnost neuronskih mreža u praktičnim primenama sugerira da optimizacioni algoritmi ipak ne završavaju u lošim lokalnim minimumima. Ovo zapažanje je iniciralo istraživanja svojstava lokalnih minimuma funkcije greške i pod određenim prepostavkama, dokazano je da verovatnoča pojavljivanja loših lokalnih minimuma eksponentijalno opada sa vrednošću funkcije u tom minimumu. Ipak, ove prepostavke nisu realistične. Zanimljivo je da je pod slabijim prepostavkama dokazano da su svi lokalni optimumi istovremeno i globalni optimumi. Iako su pomenute prepostavke dosta slabije i dalje nisu dovoljno realistične. Ipak, postoji uverenje da se ti rezultati u nekoj meri mogu generalizovati i na realističnije kontekste. Jedan intuitivan argument bi bio sledeći. U lokalnom optimumu, funkcija mora biti konveksna, odnosno hesijan funkcije mora biti pozitivno semidefinitan, odnosno mora imati nenegativne sopstvene vrednosti. Iako je realističnost ove prepostavke upitna, zamislimo da se znak sopstvene vrednosti bira bacanjem novčića. Verovatnoča da sve sopstvene vrednosti budu pozitivne (nenegativne) eksponencijalno opada sa dimenzionalnošću prostora nad kojim je funkcija definisana. Odnosno, postojanje velikog broja minimuma u visoko dimenzionalnim prostorima nije mnogo verovatno. Verovatnije je da će se naći neki pravac bekstva iz tog minimuma, definisan sopstvenim vektorom hesijana koji odgovara negativnoj sopstvenoj vrednosti. Zapravo, u skorije vreme je sve čvršeće ubedjenje da glavni problem obučavanja neuronskih mreža nije vezan za lokalne minimume, već pre za platoe malog nagiba na kojima gradijenti nestaju, što dovodi do neefikasnosti gradijentnih metoda.

Još jedan niz skorašnjih uvida vezan je za ponašanje stohastičkog gradijentnog spusta. Naime, empirijski se pokazalo da stohastički gradijentni spust ima efekat regularizacije, čak i kad nikakve eksplisitne regularizacije nema. Jedno od ranih objašnjenja uspešnosti stohastičkog gradijentnog spusta je da usled šuma koji postoji pri oceni pravog gradijenta, stohastički gradijentni spust može da se kreće i u neoptimalnim pravcima koji ga mogu izvaditi iz lokalnog optimuma. Ipak, ovaj argument ne zvuči sasvim uverljivo. Naime, bilo bi zamislivo i da se iz istog razloga iz globalnog optimuma pređe u lokalni. Takođe, kao što je već rečeno, vrlo je moguće da lokalni optimumi i nisu ključni problem obučavanja neuronskih mreža. Nešto uverljiviji argument kojim se objašnjava



Slika 10.24: Rizik (plavo) i empirijski rizik (crveno) koji imaju jedan uzak i jedan širok minimum. Stvarni rizik uskog minimuma empirijskog rizika je vrlo visok, dok je stvarni rizik širokog minimuma empirijskog rizika takođe nizak.

dobro ponašanje stohastičkog gradijentnog spusta je taj da zahvaljujući šumu koji unosi u ocenu pravog gradijenta, ne uspeva da se spusti u uske minimume, već ima tendenciju da pronalazi šroke minimume. Zašto je ovo dobro, ilustrovano je slikom 10.24. Naime, uzorak podataka kojim se raspolaže je uvek konačan i stoga prosečna greška predstavlja tek aproksimaciju stvarnog rizika. U slučaju vrlo uskih minimuma, minimum prosečne greške će biti pomeren u odnosu na minimum stvarnog rizika. Odnosno, stvarni rizik minimuma prosečne greške može biti vrlo visok, jer se vrednost rizika brzo povećava sa udaljavanjem od minimuma. S druge strane, ukoliko su minimumi široki, iako greška aproksimacije dovodi do udaljavanja od stvarnog minimuma, vrednosti stvarnog rizika u obližnjim tačkama su i dalje niske. Ovo zapažanje ima jedan neobičan aspekt. Svojstva naučenog modela trebalo bi da zavise od forme modela i od formulacije problema učenja, a ne od optimizacione metode. Međutim, zbog svoje nemogućnosti da konvergira ka određenim vrstama minimuma, neformalno rečeno, stohastički gradijentni spust efektivno modifikuje funkciju koju optimizuje.

Glava 11

Šta ako ne radi?

Mašinsko učenje često ne uspeva. Preciznije, rezultati koji se u praksi dobijaju, posebno u inicijalnim eksperimentima, često nisu dobri. Nekada se mogu popraviti uz razuman obim dodatnog truda, a nekada ni to nije dovoljno. Razlozi za neuspeh učenja mogu biti raznovrsni. U nastavku razmatramo neke greške i načine da se uoče.

11.1 Preprilagođavanje i potprilagođavanje

Dva česta razloga neuspeha su preprilagođavanje i potprilagođavanje modela. Recimo, linearni model je nekad nedovoljno fleksibilan i ne može uhvatiti trend u podacima. S druge strane, velika neregularizovana neuronska mreža će se lako preprilagoditi. Ključno je ustanoviti da li se radi o nekom od ova dva problema. Prvo, pretpostavka je da je proces optimizacije iskonvergirao, odnosno da greška na podacima za obučavanje opada kroz iteracije i da je od nekog trenutka smanjenje greške malo. Nakon toga, potrebno je proveriti grešku na podacima za obučavanje i na podacima za testiranje. Ukoliko je na prvom greška mala, a na drugom velika, to je indikator preprilagođavanja. Ukoliko su obe greške velike, to je indikator potprilagođavanja. Ni jedan od ova dva indikatora nije potpuno pouzdan. U prvom slučaju, alternativni razlog problema može biti loša stratifikacija podataka. U tom slučaju će i model sa malim brojem parametara obučavan na velikom skupu instanci, koji na skupu za obučavanje pravi vrlo malu grešku, imati veliku grešku na skupu za testiranje. U drugom slučaju, zamislivo je i da je korak optimizacije preveliki i da zbog toga greška ne može da se smanji. Međutim, ovi indikatori su ipak važni i valja ih proveriti.

U slučaju sumnje na preprilagođavanje, dobro je uraditi proveru smanjivanjem fleksibilnosti modela primenom regularizacije, smanjenjem arhitekture ako se radi o neuronskoj mreži, izborom atributa radi smanjenja broja parametara i slično. Ukoliko se ishod promeni tako što se greška na podacima za testiranje smanji, to je indikator da smo na pravom putu. U slučaju da se time

samo poveća greška na podacima za obučavanje, verovatnije je da je problem u nečemu drugom.

U slučaju sumnje na potprilagođavanje, vredi probati sa fleksibilnijim modelima, smanjenjem regularizacije, povećanjem arhitekture u slučaju neuronske mreže, dodavanjem atributa, bilo novih bilo izvedenih iz postojećih (transformacijama pojedinačnih atributa, proizvodima atributa, itd.) i slično. Makar greška na skupu za obučavanje bi trebalo da se smanji, a ako ne, problem je verovatno na drugom mestu.

11.2 Problemi podataka

Osnovni problem podataka je nedovoljna informativnost atributa u odnosu na ciljnu promenljivu. U ekstremnom primeru, zamislimo da pokušavamo da predvidimo uzorak šuma na osnovu drugog, nezavisnog, uzorka šuma. Ovo nije moguće. U realističnjem kontekstu, ukoliko pokušavamo da predvidimo buduće zdravstveno stanje pacijenta, to će teško biti moguće ukoliko ne znamo u kakvim uslovima živi, kakve su mu životne navike i slično. Neinformativnost podataka će se na proces učenja odraziti visokom greškom na test podacima, bez obzira šta radimo sa modelom, prosti zato što informacije u podacima nema. Brz test, koji je posebno koristan u slučaju linearnih modela je ispitivanje matrice korelacija među svim promenljivim, a posebno ciljne sa atributima. Ukoliko nijedan atribut nije jako koreliran sa ciljnom promenljivom, male su šanse da takvi atributi mogu biti korisni u predviđanju. Eventualno, mogu biti korisni ukoliko su vrlo slabo korelirani među sobom, pa daju mnoštvo nezavisnih malih doprinosova. Dodatno, korišćenje standardnog Pirsonovog koeficijenta korelacije je adekvatno u slučaju linearnih modela. Međutim, u slučaju drugih vrsta modela, nije nužno relevantna samo linearna korelacija koju ovaj koeficijent meri, već i monotona korelacija, koja meri koliko se tačke u ravni raštrkavaju oko neke monotone funkcije. Ovu vrstu korelacije meri Spirmanov koeficijent korelacijski koji se računa na isti način kao i Pirsonov, ali nad rangovima vrednosti promenljivih u sortiranom nizu, umesto nad samim vrednostima. Ako postoji više atributa koji su jako korelirani sa ciljnom promenljivom, problem verovatno nije do podataka. Ako ne postoje, vrlo je moguce da jeste. Šta znači izraz „jako“ u ovom kontekstu, nije lako precizirati.

U slučaju klasifikacije, posebno je korisno proveriti matricu konfuzije i proučiti koje klase model najčešće meša. Potrebno je zapitati se da li atributi koji se koriste intuitivno sadrže dovoljno podataka za razlikovanje tih klasa. Ako sadrže, problem verovatno nije do podataka. Ako ne sadrže, onda je potrebno smisliti nove atrubute koji će razlikovati te dve klase.

Drugi potencijalni problem sa podacima je njihova količina u odnosu na broj atributa. Ako je količnik N/n mali, lakše dolazi do preprilagođavanja, uzorak možda nije dovoljno bogat da opiše zakonitost koja postoji u podacima i slično. U tom slučaju, ako nije moguće sakupiti više podataka, posebna pažnja se poklanja regularizaciji ili se koriste metode za generisanje sličnih podataka,

kojima se povećava količina podataka, ali se time ne može nadomestiti njihova mala reprezentativnost. Ako je količina podataka velika, može doći do problema vezanih za raspoloživost računskih resursa. Usled toga se nekad pribegava obučavanju na podskupu podataka, pri čemu se pazi da podskup bude stratifikovan. Ipak, to se radi samo ukoliko nema drugog izbora, pošto su podaci najvredniji element procesa mašinskog učenja.

Treći potencijalni problem podataka je njihova nepotpunost, bilo u smislu da instance ne pokrivaju prostor od interesa, bilo da postoje nedostajuće vrednosti atributa u instancama. Za prvi problem nema opštег rešenja. Ni u konkretnom slučaju ga ne mora biti, iako je nekad moguće ublažiti problem. U drugom slučaju, postoje metode koje interpoliraju nedostajuće vrednosti (eng. *data imputation*) sa manjim ili većim uspehom.

11.3 Greške u pretpresiranju

Već je naglašeno da je prilikom primene metoda mašinskog učenja poželjno uraditi standardizaciju podataka. Razloga ima više i uključuju interpretabilnost parametara modela, izbegavanje neravnomerne regularizacije različitih parametara i brzinu konvergencije optimizacionih metoda. Izostavljanje standardizacije se stoga smatra greškom. Dodatno, grešku predstavlja i računanje proseka i standardne devijacije zarad skaliranja nad celim skupom podataka ili bilo kojim skupom koji je dalje potrebno deliti radi evaluacije. Takođe, greška je i računati prosek i standardnu devijaciju nad skupom za obučavanje i nad skupom za testiranje odvojeno i standardizovati ih odvojeno. Jedino je ispravno računati ih na skupu za obučavanje i onda ih primeniti na oba skupa.

Greška je i zanemarivanje približno linearne zavisnosti atributa, pošto vode lošoj uslovjenosti problema. Ovi atributi se mogu detektovati ispitivanjem korelacija među atributima. Problem se može otkloniti na dva načina. Ukoliko je potrebno da model bude izražen nad polaznim atributima, na primer zarad interpretabilnosti, moguće je izvršiti izbor podskupa atributa tako da se ne koristi više linearne zavisnosti atributa. Drugi način je da se konstruišu novi ne-korelirani atributi analizom glavnih komponenti (PCA). Gubitak informacije je tipično manji u drugom slučaju. Pored ovih rešenja, regularizacija takođe pomaze, ali ako nije potrebna zbog preprilagođavanja, smanjiće kvalitet dobijenog modela.

Još jedna greška je predstavljanje kategoričkih vrednosti jednom numeričkom promenljivom. Umesto toga, koristi se već diskutovano binarno kodiranje.

11.4 Neadekvatnost algoritma

Nijedan algoritam učenja nije pogodan za sve probleme. Jedna indikacija mogućnosti da je problem do izbora algoritma je neuspeh mera vezanih za otklanjanje preprilagođavanja ili potprilagođavanja. Ipak, ova vrsta problema

traži i teorijsko razumevanje algoritma. Ona se najpre tiče lošeg izbora reprezentacije modela ili funkcije greške. Naravno, moguće je i da je poželjno raditi na specifičnim metodama regularizacije, ali za očekivati je da loši izbori vezani za dva pretodno navedena elementa dizajna vode većim problemima nego da li se za regularizaciju koristi ℓ_2 regularizacija ili regularizacija izostavljanjem.

Već smo primetili da su neke funkcije greške osetljive na odudarajuće podatke, dok neke druge nisu. U drugačijim poslovima od regresije i klasifikacija (poput rangiranja dokumenata u odnosu na korisnikove preference i upit) moguće je konstruisati i drugačije funkcije greške. Takođe, već smo skrenuli pažnju da funkcija greške nekada ne treba da bude simetrična, odnosno da ne treba da kažnjava sve greške jednako (na primer u slučaju medicinske dijagnostike). U ovakvim slučajevima, poželjno je napraviti svoju funkciju greške koja je primerena problemu.

U nekim slučajevima reprezentacija modela može predstavljati problem. Lokalne zakonitosti se tipično lakše uče metodima zasnovanim na instacama. Oni će nekoj instanci dodeljivati recimo visoku vrednosti ciljne promenljive ne zato što visoka vrednost ciljne promenljive uvek odgovara recimo visokim vrednostima nekih atributa, što bi bila prirodna zakonitost u slučaju linearog modela, već zato što bliske instance imaju visoku vrednost ciljne promenljive. Zamislimo da su instance jedne klase raspoređene u okolinama dva dijagonalno naspramna temena kvadrata, a instance druge u okolinama druga dva temena i da nema mešanja između instanci klasa. Linearni model nad koordinatama u ravni ne može naučiti ovaku zakonitost. Nasuprot njemu, model zasnovan na instancama će vrlo dobro predviđati.

Nekada se prilikom izvođenja algoritma prave razne aproksimacije, koje pojednostavljaju zavisnosti među promenljivim. Takav je primer aproksimacije u slučaju naivnog Bayesovog algoritma. Aproksimacije mogu biti izvor lošijeg ponašanja algoritma i potrebno je razmotriti da li su adekvatne u datom kontekstu. Takvo razmatranje ne mora biti lako.

Kao što je rečeno, nije lako ustanoviti da su greške uzrokovane svojstvima algoritma učenja, a ne nečim drugim. Određene indikacije bi se mogle dobiti analizom grešaka. Na primer, bilo bi poželjno proučiti instance na kojima je greška najveća ili instance na kojima model daje pogrešna predviđanja, a sa visokom sigurnošću. Onda se može razmotriti da li su one u nekom smislu odudarajući podaci, da li predstavljaju specifične instance koje odražavaju neke zakonitosti kojima model nije namenjen i slično. Ovo je utoliko lakše ako se radi sa podacima koje možemo analizirati čulima, odnosno obradi slika, prostornih podataka (koji se mogu predstaviti kartama), obradi zvuka i slično.

11.5 Neadekvatnost optimizacije

U slučaju nekonveksnih minimizacionih problema optimizacija može predstavljati netrivijalan deo procesa učenja. Neuronske mreže su tipičan primer. Možda najvažniji hiperparametar u procesu obučavanja neuronskih mreža je

korak optimizacije. Premale vrednosti ovog hiperparametra vode presporoj konvergenciji. Kako se ovaj parametar tipično smanjuje sa brojem iteracija, praktično dolazi do zaustavljanja optimizacije značajno pre nego što se dođe do kvalitetnog modela. Velike vrednosti koraka optimizacije u slučaju konstantnog koraka, vode nemogućnosti silaska optimizacionog procesa u minimume. Premale vrednosti se lako prepoznaju po sporoj promeni vrednosti funkcije cilja. Prevelike vrednosti se teže prepoznaju. Tipično se u nekom trenutku uočava oscilatorno ponašanje vrednosti funkcije cilja, kroz naizmenično smanjivanje i povećavanje njene vrednosti. Nekada, ovaj problem vodi još neobičnjem ponašanju. Na primer, monotonom rastu vrednosti funkcije cilja i norme gradijentata, što je potpuno suprotno od onoga što se od gradijentnih metoda očekuje. Ovako nešto se dešava usled prevelikog koraka optimizacije kada je polazna tačka relativno blizu optimalne tačke u relativno uskom minimumu. Prvi korak koji metod preduzima, zbog prevelike dužine, prebacuje preko mimuma i vodi ka tački koja je od njega dalje od polazne. Tu je vrednost funkcije, ali i njena nagnutost veća, pa gradijent postaje još veći i sledeći korak vodi još dalje i još većem gradijentu i tako dalje dok se ne izađe u region veće širine ili dok se korak ne smanji.

Rešenja za prethodne probleme ima više. Jedno je svakako smanjivanje koraka prema nekom unapred utvrđenom pravilu. Još bolji je korišćenje metoda sa adaptivnom dužinom koraka, poput Adama. Čak je moguće primetiti kada u toku optimizacije dolazi do toga da korak koji je do tad bio adekvatan postane neadekvatan i ručno izmeniti postupak optimizacije da od tog trenutka pređe na drugu dužinu koraka ili čak ručno prekinuti optimizaciju uz čuvanje svih relevantnih parametara stanja optimizacije, a onda nastaviti odatle zadajući novu dužinu koraka.

Problem druge vrste su već pominjani nestajući i eksplodirajući gradijenti. Eksplodirajući gradijenti se uočavaju po visokim normama gradijanata i hao-tičnom ponašanju vrednosti funkcije cilja, a već je rečeno da se problem može ublažiti odsecanjem gradijenata. Problem nestajućih se uočava po malim normama gradijenata i malim promenama vrednosti funkcije cilja, a obično rešava ili ublažava promenom funkcije greške (sledeći primer pokazuje da nije dobro koristiti sigmoidne jedinice u izlaznom sloju sa kvadratnom greškom), aktivacione funkcije (na primer nakošena ispravljena linearna jedinica se u ovom smislu bolje ponaša od sigmoidne funkcije) ili prelaskom na LSTM ukoliko to odgovara kontekstu.

Primer 9 *Kako logistički model daje vrednosti u intervalu (0, 1), na prvi pogled, pokušaj njegovog obučavanja pomoću kvadratne greške ne bi delovalo neobično. Prosečna greška bi bila:*

$$E(w) = \sum_{i=1}^N (y_i - \sigma(w \cdot x_i))^2$$

Gradijent greške je

$$\frac{\partial E}{\partial w_j}(w) = -2 \sum_{i=1}^N (y_i - \sigma(w \cdot x_i)) \sigma(w \cdot x_i)(1 - \sigma(w \cdot x_i)) x_{ij}$$

Razmotrimo kolike su vrednosti parcijalnih izvoda ukoliko važi $|y_i - \sigma(w \cdot x_i)| \approx 1$. To je situacija u kojoj je greška maksimalna, pa bismo očekivali da je norma gradijenta velika, a da se smanjuje sa približavanjem minimumu. Međutim, ukoliko važi $|y_i - \sigma(w \cdot x_i)| \approx 1$ ili je $y_i = 1$, a $\sigma(w \cdot x_i)$ blizu nule, što čini da je gradijent blizu nule ili je $y_i = 0$, a $\sigma(w \cdot x_i)$ blizu jedinice, što čini da je izraz $1 - \sigma(w \cdot x_i)$, a time i gradijent, blizu nule. Odnosno, iako je greška velika, gradijent je praktično nula.

11.6 Greške u evaluaciji

Već je dovoljno skrenuta pažnja na potencijalne greške u evaluaciji modela koje proističu iz preklapanja podataka za obučavanje i podataka za testiranje, pre svega u kontekstu konfigurabilnih algoritama. Rešenje za takve probleme je u korišćenju evaluacije pomoću skupova za validaciju i testiranje i ugnezđene unakrsne validacije. Unakrsna validacija pruža još jedan dijagnostički alat. Naime, unakrsna validacija obučava veliki broj modela koji bi trebalo da aproksimiraju model obučen na svim podacima pomoću neke izabrane konfiguracije. Ukoliko je prethodno adekvatno urađeno preprocesiranje i stratifikacija, i ako unakrsna validacija ima recimo 10 slojeva, modeli ne bi trebalo da se mnogo razlikuju među sobom ili u odnosu na finalni model. Jasno, vrednosti koeficijenata će se razlikovati, ali neobično je ako u nekom modelu nekom atributu odgovara vrlo visok pozitivan paramtar, a u drugom vrlo nizak ili čak negativan parametar. Takve situacije obično sugerišu ili neadekvatnu stratifikaciju ili slabu reprezentativnost podataka usled nedovoljne količine.

U nekim specifičnim domenima, moguće je da kvalitetna evaluacija postavlja još neke zahteve osim razdvojenosti skupova za obučavanje i testiranje. Na primer, u slučaju vremenskih serija, poput podataka sa trgovine na berzi, važno je da podaci u skupu za testiranje odgovaraju kasnijim opažanjima. Takođe, potrebna je pažnja da se ne definiše model koji zahteva poznавanje nekih informacija iz budućnosti. Iako zvuči trivijalno, takve greške se ipak dešavaju.

Greške je moguće napraviti i u izboru mera za evaluaciju. Tipičan primer je oslanjanje na tačnost u kontekstu neizbalansiranih klasa i simetričan tretman različitih vrsta grešaka u situacijama u kojima realna cena različitih grešaka nije jednaka. Takođe, različite mere kvaliteta daju informaciju o različitim aspektima kvaliteta modela, pa je uvek poželjno uzeti u obzir što više različitih mera. Na primer, visoka vrednost R^2 ne znači mnogo ukoliko RMSE ili srednja apsolutna greška nisu u okvirima koji su praktično prihvatljeni. Slično, niska vrednost RMSE može biti posledica niske polazne varijanse, a ne posledica učenja. Stoga je dobro proveriti i R^2 .

11.7 Greške u interpretaciji modela

Već je više puta pomenuto da izostavljanje standardizacije vodi besmislenim ishodima u interpretaciji modela. Takođe, istaknuto je da je radi povećanja interpretabilnosti poželjno koristiti ℓ_1 (laso) umesto ℓ_2 norme, pošto su proređeni modeli lakši za analizu. Grupna lasso regularizacija je takođe poželjna, zbog binarno kodiranih kategoričkih atributa. Pomenut je i problem koreliranih atributa, koji vodi nestabilnosti lasso regularizacije. Tada je poželjno koristiti elastičnu mrežu. U slučaju postojanja grupe visoko koreliranih atributa, ti atributi će imati manje koeficijente, nego što bi samo jedan od njih koji nosi približno istu informaciju kao svi zajedno. Na taj način se može steći utisak da su svi relativno nebitni. Otud je poželjno eliminisati visoko korelirane attribute pre obučavanja modela.

11.8 Greške u implementaciji

Greške u implementaciji mogu biti proizvoljne prirode. Zbog toga nema opštih smernica za njihovo nalaženje i ispravljanje. U velikoj meri se mogu izbeći korišćenjem već gotovih dobro testiranih i široko korišćenih biblioteka. Ipak, nekada je potrebno napraviti nov algoritam koji se ne uklapa nužno u funkcionalnosti postojećih biblioteka. U takvim slučajevima se obično izvode prvi (nekad i drugi) izvodi i predaju nekoj funkciji optimizacije. Nekada se i optimizacioni algoritam razvija od nule. Ipak, potencijalna greška je često u izvodima koji su u mašinskom učenju neretko komplikovani. Na sreću, ispravnost izvoda je moguće relativno lako testirati, postupkom takozvane *provere gradijentata* (eng. *gradient check*) koji se sastoji u izračunavanju njegove aproksimacije podeljenim razlikama, odnosno:

$$\frac{\partial E}{\partial w_i}(w) = \frac{E(w + \frac{\varepsilon}{2} e_i) - E(w - \frac{\varepsilon}{2} e_i)}{\varepsilon}$$

pri čemu e_i označava i -ti element ortonormirane baze prostora \mathbb{R}^n , a ε mali broj. Ukoliko je odstupanje između ove aproksimacije, koja se obično jednostavno implementira, i vrednosti implementiranog parcijalnog izvoda velika, jasno je da postoji greška.

U slučajevima specifičnih algoritama mogu postojati određena svojstva koja neki segment procesa učenja mora da poštuje i narušenost tih uslova može poslužiti kao indikator greške u implementaciji. Na primer, svi algoritmi koji predstavljaju instance EM (eng. *expectation maximization*) metaalgoritma imaju svojstvo da u svakom koraku optimizacije povećavaju vrednost funkcije cilja (koja ne predstavlja grešku, već verodostojnost, pa se otud povećava). Ukoliko u bilo kom koraku dođe do njenog pada, sigurno je da postoji greška u implementaciji. Ovakva svojstva se moraju tražiti od algoritma do algoritma.

Deo II

Učenje potkrepljivanjem

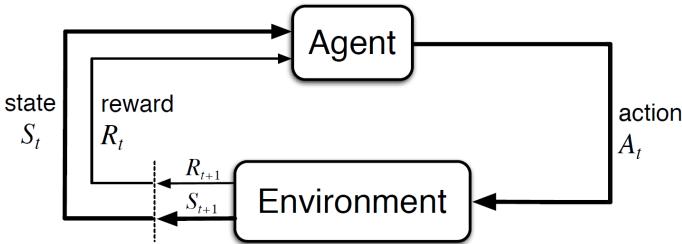
Glava 12

Markovljevi procesi odlučivanja i njihovo rešavanje

Već je rečeno da se učenje potkrepljivanjem koristi u situacijama u kojima je potrebno rešiti neki problem preduzimajući niz akcija, čijim se zajedničkim dejstvom dolazi do rešenja problema. Pretpostavlja se da postoji *agent* koji opaža tekuće *stanje okruženja* i u mogućnosti je da preduzima *akcije* usled kojih dobija *nagrade* predstavljene numeričkom vrednošću i prelazi u novo stanje. Ovaj proces je ilustrovan slikom 12.1. Ishod učenja je *optimalna politika*, odnosno preslikavanje stanja u akcije koje vodi maksimalnoj (ili, u praksi, dovoljno visokoj) dugoročnoj nagradi. Pritom, ključna je prepostavka da nije poznato koja od preduzetih akcija je bila prava u datom kontekstu, a koja nije. U suprotnom, radilo bi se o problemu nadgledanog učenja. Pomenut je primer autonomne vožnje. Agent je sistem koji vozi automobil i koji je u stanju da opaža pozicije drugih automobila, pešaka, saobraćajne znake, svetla semafora i slično (a što čini okruženje), a koji je u stanju da menja smer kretanja i da povećava i smanjuje brzinu kretanja automobila (što su akcije). Agent pritom dobija nagrade koje su recimo 1 za svaki kilometar pređenog puta, 100 za stizanje na cilj, -100 u slučaju sudara i -1000 u slučaju smrtnog ishoda u saobraćaju. Optimalnu politku nije lako opisati na osnovu ličnog znanja. Ona bi verovatno uključivala kočenje na žutom i crvenom svetlu ili pred pešacima, skretanje tamo gde je znakom naznačeno da je obavezno skrenuti itd, ali to je moguće naučiti iz iskustva.

12.1 Osnovni pojmovi

Kako bismo pristupili rešavanju ovakvih problema učenja, prvo ćemo, kao i u slučaju nadgledanog učenja, postaviti teorijski okvir, koji nije nužno realističan. U slučaju nadgledanog učenja, teorijski okvir je postavljen u terminima zajedničke raspodele $p(x, y)$, koja je naravno nepoznata, ali je takav pristup ipak bio koristan. Slično će biti i u ovom slučaju. Teorijski okvir učenja potkreplji-



Slika 12.1: Shema interakcije agenta i okruženja.

vanjem se opisuje Markovljevim procesima odlučivanja (eng. *Markov decision processes*), ili skraćeno MDP. U nastavku se podrazumevaju *konačni* Markovljevi procesi odlučivanja, iako ta distinkcija ubuduće neće biti naglašavana. I u učenju potkrepljivanjem i mimo njega, od velikog značaja su i Markovljevi procesi odlučivanja koji nisu konačni i kojih ćemo se dodataći kasnije, ali bez dublje analize.

Neka agent i okruženje interaguju u diskretnim trenucima $t = 0, 1, \dots$. U svakom trenutku t , agent opaža stanje okruženja S_t iz konačnog skupa stanja \mathcal{S} i preduzima akciju A_t iz konačnog skupa dopustivih akcija $\mathcal{A}(s)$ u konkretnom stanju s (promenljive su označene velikim slovima, a njihove konkretne vrednosti malim). Pritom, dobija nagradu R_{t+1} iz konačnog skupa nagrada \mathcal{R} i prelazi u novo stanje S_{t+1} . MDP i agent zajedno proizvode *putanju*:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \dots$$

Osnovno svojstvo MDP-a je Markovljevo svojstvo, da novo stanje i nagrada zavise samo od prethodnog stanja i preduzete akcije, a ne od cele istorije procesa:

$$P(S_t, R_t | S_0, A_0, R_1, \dots, R_{t-1}, S_{t-1}, A_{t-1}) = P(S_t, R_t | S_{t-1}, A_{t-1})$$

Funkcija prelaska se definiše kao

$$p(s', r | s, a) = P(S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a)$$

Verovatnoće date funkcijom prelaska u potpunosti karakterišu MDP.

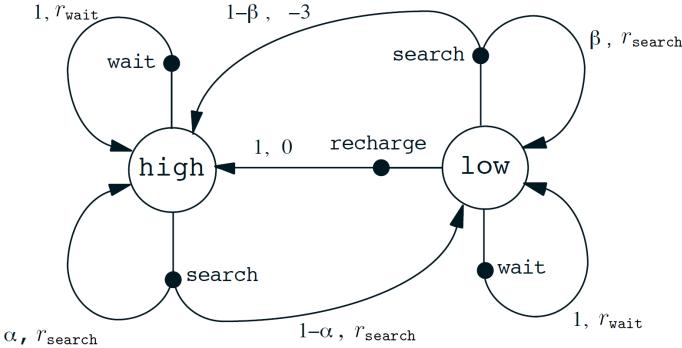
Primer 10 Robot ima zadatok da sakuplja prazne limenke u kancelariji. Raspolaze senzorima za limenke, rukom kojom može da ih sakuplja, skladištem gde ih čuva, mehanizmom za kretanje i baterijom koju je moguće puniti. Robot treba da nauči da donosi odluke visokog nivoa, kao što su da li treba tražiti limenke u toku određenog vremenskog intervala, mirovati i čekati da neko doneše limenknu ili vratiti se na mesto na kojem je moguće punjenje baterije. Poželjan događaj je sakupljanje limenke, a nepoželjan pražnjenje baterije. Modelujmo ovaj proces kao Markovljev proces odlučivanja. To ćemo razumevati na osnovu određenog razumevanja problema. Najefektniji način sakupljanja limenki je

s	a	s'	$p(s' s, a)$	$r(s, a, s')$
high	search	high	α	r_{search}
high	search	low	$1 - \alpha$	r_{search}
low	search	high	$1 - \beta$	-3
low	search	low	β	r_{search}
high	wait	high	1	r_{wait}
high	wait	low	0	r_{wait}
low	wait	high	0	r_{wait}
low	wait	low	1	r_{wait}
low	recharge	high	1	0
low	recharge	low	0	0.

Slika 12.2: Tabelarni prikaz Markovljevog procesa odlučivanja koji opisuje poнаšanje robota koji sakuplja limenke. Poslednja kolona predstavlja očekivane nagrade pri različitim prelazima.

aktivno traženje. Međutim, tako se troši baterija. Vrlo je nepoželjno istrošiti je i čekati da čovek prenese robota na punjenje. Stoga, važan parametar je stanje baterije. Neka je skup stanja $S = \{\text{high}, \text{low}\}$. Kada je baterija puna, nema smisla puniti je, tako da skupovi akcija mogu biti $\mathcal{A}(\text{high}) = \{\text{search}, \text{wait}\}$ i $\mathcal{A}(\text{low}) = \{\text{search}, \text{wait}, \text{recharge}\}$. Period traganja u stanju **high** sa verovatnoćom α vodi prelasku u stanje **low**, a sa verovatnoćom $1 - \alpha$ ostavlja robota u stanju **high**. Period traganja u stanju **low** sa verovatnoćom β vodi praznjenju baterije, nakon čega se ona puni i robot prelazi u stanje **high**, a sa verovatnoćom $1 - \beta$ ostavlja robota u stanju **low**. Svaka sakupljena limenka pruža nagradu 1, a praznjenje -3. Dodatno, pri odlasku na punjenje, robot ne sakuplja limenke. Tabela na slici 12.2 sumira opis datog MDP-a, koji se može predstaviti i grafom na slici 12.3

U praksi iz različitih razloga problem često ne predstavlja MDP kakav smo definisali. Nekada nije moguće precizno opaziti stanje, već samo neke njegove parametre. Na primer, robot koji se kreće kroz nepoznatu sredinu na osnovu očitavanja infracrvenog senzora neće uočiti objekte na putu koji ne reflektuju zračenje u infracrvenom delu spektra. Ukoliko ima na raspolaganju i ultrazvučni senzor, šanse da će ispravno detektovati stanje su veće. Nekada se parametri MDP-a menjaju kroz vreme. Na primer, u problemu autonomne vožnje, iste akcije u istim stanjima mogu voditi različitim stanjima u zavisnosti od parametara saobraćaja poput gužve. Lako je primetiti da takav slučaj možemo svesti na prethodni uzimajući u obzir neke promenljive koje do tada nismo imali u vidu, ali to lako može voditi eksploziji prostora stanja. Nekada Markovljevo svojstvo nije ispoštovano. Na primer, ako je na osnovu slike potrebno razumeti kretanje objekata, poslednja slika često ne daje jasnu informaciju o pravcu kretanja. U ovom slučaju nije teško korigovati model – dovoljno je uzeti u obzir



Slika 12.3: Graf prelaska u Markovljevom procesu odlučivanja koji opisuje ponašanje robota koji sakuplja limenke.

dve ili više slike u dovoljno kratkim intervalima i definisati stanje na osnovu njih. Ipak, u nekim slučajevima to može značajno povećati prostor stanja.

Uloga nagrada je da na implicitan način definišu cilj agenta. Treba da budu tako izabrane da maksimizujući nagradu, što je agentov cilj, agent istovremeno obavlja posao koji želimo da obavi. Jedna od klasičnih grešaka je da se nagrade koriste za usmeravanje agenta kako da nešto uradi, umesto šta da uradi. Na primer, ukoliko agent igra šah, nagradu treba da dobije ukoliko pobedi u partiji, a ne ukoliko recimo pojede protivničku figuru. Naime, u zavisnosti od toga kako su nagrade izabrane, moguće je da agent nađe način da pojede veliki broj protivničkih figura, po cenu gubitka partije. Stoga, nagrada je način da agentu saopštimo šta treba da uradi, a ne kako to treba da uradi.

Već je rečeno da je cilj agenta da maksimizuje dugoročnu nagradu, koju ćemo nazivati *dobitkom*. Dobitak u trenutku t u odnosu na neku putanju definišemo na sledeći način

$$G_t = \sum_{i=0}^{\infty} \gamma^i R_{t+i+1}$$

pri čemu je γ hiperparametar *umanjenja* (eng. *discount*) za koji važi $0 \leq \gamma \leq 1$. U slučaju da je $\gamma = 0$, maksimizacijom dobitka, agent vrši pohlepnu maksimizaciju neposredne nagrade. Što je γ veće, to agent više uzima u obzir buduće nagrade. Što je manje, to je agent pohlepniji, odnosno teži višim neposrednim nagradama, iako to dugoročno ne mora voditi dobrom rezultatu. Ipak, ovaj parametar ima još jednu ulogu. U slučaju beskonačnih putanja, dobitak ne mora konvergirati ukoliko γ nije manje od 1. Ukoliko jeste, pod pretpostavkom ograničene nagrade, konvergencija je garantovana. U slučaju konačnih putanja, koje nazivamo *epizodama* (eng. *episodes*), prirodno je izabrati $\gamma = 1$.

Stanja koja vode višim dobitcima se smatraju poželjnijim od onih koja vode nižim. U tom smislu možemo govoriti o vrednosti stanja. Ipak, ta vrednost očito zavisi od toga koje akcije će biti preduzete u budućnosti od strane agenta. Stoga se vrednost stanja definiše u odnosu na neki konkretan način delanja, odnosno

politiku. $\text{Politika } \pi$ je funkcija koja stanju s pridružuje raspodelu verovatnoće nad skupom akcija $\mathcal{A}(s)$. Ukoliko agent prati politiku π u trenutku t , $\pi(a|s)$ označava verovatnoću da važi $A_t = a$, ako je $S_t = s$. Učenje potkrepljivanjem se bavi pronalaženjem ovakvih politika na osnovu iskustva.

Funkcija vrednosti stanja $s \in \mathcal{S}$, ili kraće *vrednost stanja*, pri politici π , koju označavamo $v^\pi(s)$ je očekivani dobitak kada se agent nalazi u stanju s i prati politiku π . Za dati MDP, važi

$$v^\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i R_{t+i+1} | S_t = s \right]$$

Funkcija vrednosti akcije u stanju, odnosno *vrednost akcije a u stanju s* se definiše kao

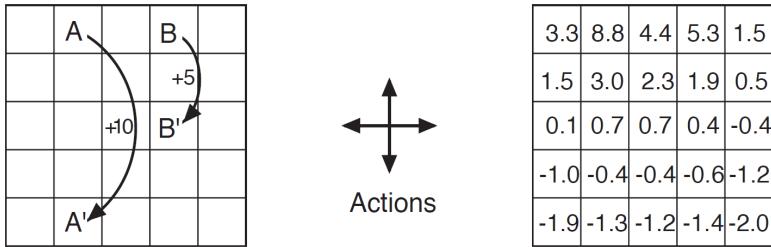
$$q^\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i R_{t+i+1} | S_t = s, A_t = a \right]$$

Razlika između poznavanja funkcije v i funkcije q je u tome da funkcija v često ima kompaktniju reprezentaciju, pošto stanja ima manje nego parova stanja i akcija, ali je odlučivanje o akciji koju treba preduzeti komplikovanije na osnovu funkcije v , nego na osnovu funkcije q , jer funkcija q uključuje tu akciju, dok je u slučaju korišćenja funkcije v potrebno vršiti pretragu po akcijama koje je moguće preduzeti u potrazi za najboljim budućim stanjem.

Primer 11 Na slici 12.4 dat je primer okruženja u kojem važe jednostavna pravila – stanja odgovaraju poljima table, agent se kreće levo, desno, gore i dole, svaki put kada naiđena na polje A dobija nagradu 10 i biva premešten na polje A', a kad naiđe na polje B dobija nagradu 5 i biva premešten na polje B'. Pokušaji izlaska sa teble ostavljaju agenta na istom polju, ali uz nagradu -1. Ostali potezi nose nagradu 0. Pored table, slika prikazuje i funkciju vrednosti stanja pri politici koja dodeljuje jednakе verovatnoće svim smerovima, za vrednost umanjenja $\gamma = 0.9$. Pri toj politici, zbog velike nagrade, najpoželjnije je stanje A. Međutim, ovo stanje vredi manje od neposredne nagrade 10 koju agent u njemu dobija zbog mogućnosti da nakon premestanja na ivicu table dobije negativnu nagradu. S druge strane, te opasnosti nema u polju B, pa je njegova vrednost veća od neposredne nagrade koja se dobija u tom stanju.

Za obe funkcije važe rekurentne relacije, koje se nazivaju Belmanovim jednačinama:

$$\begin{aligned} v^\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) [r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']] \\ &= \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) [r + \gamma v^\pi(s')] \end{aligned}$$



Slika 12.4: Tabela na kojoj se kreće agent, smerovi kretanja i funkcija vrednosti stanja pri uniformnom izboru smera kretanja.

$$\begin{aligned}
 q^\pi(s, a) &= \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\
 &= \sum_{s', r} p(s', r | s, a) [r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']] \\
 &= \sum_{s', r} p(s', r | s, a) \left[r + \gamma \sum_{a'} \pi(a' | s') q^\pi(s', a') \right]
 \end{aligned}$$

koje su od velikog značaja za algoritme.

Nad politikama je moguće definisati parcijalno uređenje. Za politiku π kažemo da je bolja ili jednaka politici π' i pišemo $\pi \geq \pi'$ ako i samo ako za sva stanja $s \in \mathcal{S}$ važi $v^\pi(s) \geq v^{\pi'}(s)$. Uvek postoji bar jedna politika koja je bolja ili jednaka svim ostalim potpolitikama. Takve politike nazivamo *optimalnim politikama* i bilo koju od njih označavamo kao π^* . Kako svim optimalnim politikama odgovara ista funkcija vrednosti stanja, *optimalnu funkciju vrednosti stanja* za svako stanje $s \in \mathcal{S}$ definišemo na sledeći način:

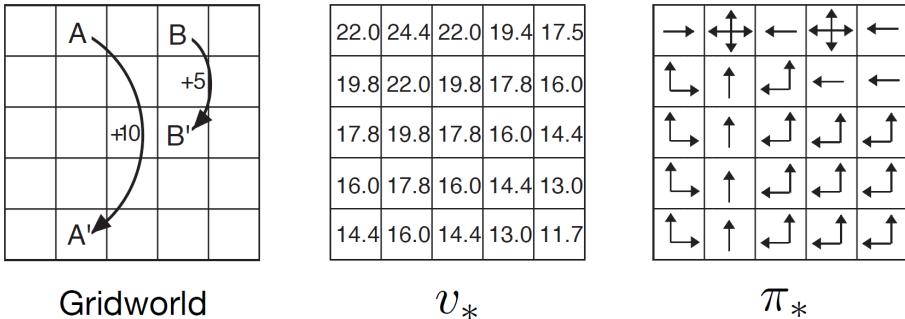
$$v^*(s) = \max_\pi v^\pi(s)$$

Slično, za svako stanje $s \in \mathcal{S}$ i akciju $a \in \mathcal{A}$, definišemo *optimalnu funkciju akcije u stanju*:

$$q^*(s, a) = \max_\pi q^\pi(s, a)$$

Belmanove jednačine za optimalne funkcije vrednosti stanja i vrednosti akcije u stanju se mogu napisati bez referisanja na bilo koju politiku (osim u međukoracima):

$$\begin{aligned}
 v^*(s) &= \max_a q^{*\pi}(s, a) = \max_a \mathbb{E}_{\pi^*}[G_t | S_t = s, A_t = a] \\
 &= \max_a \mathbb{E}_{\pi^*}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\
 &= \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma v^*(s')]
 \end{aligned}$$



Slika 12.5: Tabela na kojoj se kreće agent, optimalna funkcija vrednosti stanja i optimalna politika.

$$q^*(s, a) = \sum_{s', r} p(s', r | s, a) \left[r + \gamma \max_{a'} q^*(s', a') \right]$$

Ukoliko su optimalne funkcije vrednosti stanja i vrednosti akcije u stanju poznate, lako je naći optimalnu politiku. Optimalna politika je recimo bilo koja politika koja je pohlepna u odnosu na ove funkcije. Kao optimalne politike, najčešće ćemo koristiti *determinističke politike* koje za dato stanje jednoj akciji pridružuju verovatnoću 1, a ostalima verovatnoću 0. U tom slučaju, optimalnu politiku možemo pisati kao funkciju stanja:

$$\pi^*(s) = \operatorname{argmax}_a \sum_{s', r} p(s', r | s, a) [r + \gamma v^*(s')]$$

ili

$$\pi^*(s) = \operatorname{argmax}_a q^*(s, a)$$

Primer 12 Na slici 12.5 pored table iz prethodnog primera prikazani su i optimalna funkcija vrednosti stanja i optimalnu politiku za $\gamma = 0.9$. Zanimljivo je da na polju neposredno ispod B optimalna politika ne savetuje pohlepno poнаšanje – prelazak na polje B, već kretanje u pravcu polja A. S druge strane, sa polja desno od B, koje je dva koraka dalje od prethodno pomenutog, se ipak prelazi na B jer je zbog umanjenja nagrada vezana za polje A dovoljno potcenjena.

12.2 Rešavanje MDP-a dinamičkim programiranjem

Određivanje optimalnih funkcija stanja i akcije u stanju, odnosno optimalne politike su centralni problemi učenja potkrepljivanjem. Belmanove jednačine daju osnovu za određivanje ovih veličina ukoliko nam je okruženje, odnosno

MDP koji ga definiše, potpuno poznato. To je nerealistična pretpostavka, ali važna za dalje razumevanje učenja potkrepljivanjem, zato što metode koje uče u nepoznatom okruženju na neki način aproksimiraju ova rešenja.

Važno pitanje za dati MDP se odnosi na kvalitet neke konkretnе politike. *Evaluacija politike* se svodi na aproksimaciju funkcije v^π . Postupak je jednostavan. Pošavši od vektora proizvoljnih vrednosti $v_0 \in \mathbb{R}^{|\mathcal{S}|}$, vršiti izračunavanja u skladu sa Belmanovom jednakošću:

$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) [r + \gamma v_k(s')]$$

za svako s , dok se ne zadovolji neki unapred odabran kriterijum kvaliteta aproksimacije. Ukoliko važi $\gamma < 1$, ovaj postupak uvek konvergira funkciji v^π u polinomijalnom vremenu u odnosu na broj stanja i akcija. Strogo gledano, ova formulacija podrzumeva izračunavanje novih vrednosti na osnovu starih vrednosti za svako s . Ipak, u praksi se ova ažuriranja rade u mestu – novoizračunate vrednosti $v_{k+1}(s)$ za neka stanja se odmah koriste za ažuriranje vrednosti funkcije za druga stanja. Ovaj postupak konvergira još brže.

Analogan postupak se vrši i u slučaju određivanja funkcije v^* , samo što je pravilo ažuriranja drugačije:

$$v_{k+1}(s) = \max_a \sum_{s',r} p(s',r|s,a) [r + \gamma v_k(s')]$$

Slično je moguće aproksimirati i funkciju vrednosti akcije u stanju, bilo vezanu za neku politiku, bilo optimalnu. Kada su funkcije v^* ili q^* poznate, jednostavno je doći do optimalne politike π^* .

U oba postupka, javlja se izraz $r + \gamma v_k(s')$. Ova ideja, da se vrednosti stanja aproksimiraju na osnovu neposrednih nagrada i tekućih aproksimacija vrednosti drugih stanja je sveprisutna u učenju potkrepljivanjem.¹

Prikazani postupci aproksimacije predstavljaju vid dinamičkog programiranja. Rečeno je da je broj iteracija do konvergencije polinomijalan u odnosu na broj stanja i akcija MDP-a. Iako ovo zvuči dobro, treba imati u vidu da broj stanja često može biti ogroman. Naime, problemi koje u praksi rešavamo su retko zaista diskretni, a i tada, kao u slučaju šaha ili igre go, imaju ogromne prostore stanja. Diskretni prostori stanja se češće dobijaju diskretizacijom neprekidnih prostora stanja koji često mogu biti i visokodimenzionalni. Tada je broj stanja eksponencijalan u odnosu na broj dimenzija prostora, što znači da u praksi metode dinamičkog programiranja ne bi bile dobar način učenja, čak i kad bi MPD bio poznat.

¹Ova vrsta aproksimacije se na engleskom naziva *bootstrapping*.

Glava 13

Učenje u nepoznatom okruženju

Postoje različiti metodi za učenje u nepoznatom okruženju. Pored izračunavanja vrednosti funkcija v^* ili q^* ili optimalne politike π^* , ovakvi metodi moraju eksplicitno ili implicitno i da sakupljaju informaciju o funkciji $p(s', r|s, a)$ koja definiše odgovarajući MDP. Ovo sakupljanje informacija se može vršiti na dva načina. Jedan je *u skladu sa politikom* (eng. *on-policy*), pri kojem se dela u skladu sa nekom politikom, koja se istovremeno popravlja na osnovu nagrada dobijenih od okruženja. Drugi je *mimo politike* (eng. *off-policy*), pri kojem se optimalna politika aproksimira, iako se dela na osnovu nekih drugih politika. U oba slučaja biće važno istraživati različite delove prostora stanja i akcija. Jedan od mehanizama koji garantuje postojanje istraživanja je ε -pohlepna politika (eng. ε -greedy policy). ε -pohlepna politika je politika koja svim akcijama osim najisplatljivoj pridružuje verovatnoću $\frac{\varepsilon}{|\mathcal{A}|}$, a najisplatljivoj verovatnoću $1 - \varepsilon + \frac{\varepsilon}{|\mathcal{A}|}$. Isplatljivost se naravno određuje na osnovu tekućih ocena funkcija vrednosti.

13.1 Osnovni algoritmi

Sarsa je osnovni algoritam učenja u skladu sa politikom. U toku učenja, aproksimira se funkcija q^π za tekuću politiku π , koja se ne predstavlja eksplicitno, već je implicitno definisana aproksimacijom funkcije q^π . Politika se iterativno popravlja u svakom koraku. Ako se u nekom trenutku t u stanju s preduzme akcija a , time dobije nagradu r i izvrši se prelaz u stanje s' u kojem se preduzima akcija a' , ažuriranje aproksimacije funkcije q^π se vrši narednim pravilom:

$$q(s, a) \leftarrow q(s, a) + \alpha_t [r + \gamma q(s', a') - q(s, a)]$$

koje se primenjuje kad god agent preduzme akciju i dobije nagradu r , pri čemu se q inicijalizuje nasumično i polazno stanje svake epizode se nasumično birala kad god se neka epizoda završi finalnim stanjem. Izbor akcije se vrši na osnovu ε -pohlepne politike izvedene iz tekuće aproksimacije q . Funksionisanje ovog metoda je kontrolisano pomoću dva hiperparametra – ε koji kontroliše

učestalost istraživanja i α_t koji predstavlja korak učenja i ima sličnu ulogu kao u ranije diskutovanim metodama optimizacije. Intuicija iza ovog algoritma je jednostavna – u svakom koraku se vrši linearna interpolacija između tekuće ocena vrednosti $q(s, a)$ i ocene dobijene na osnovu tekuće nagrade r i ocene dobitka iz narednog stanja $\gamma q(s', a')$. Ovo se još bolje vidi nakon što se delovi izraza pregrupišu na sledeći način:

$$q(s, a) \leftarrow (1 - \alpha_t)q(s, a) + \alpha_t [r + \gamma q(s', a')]$$

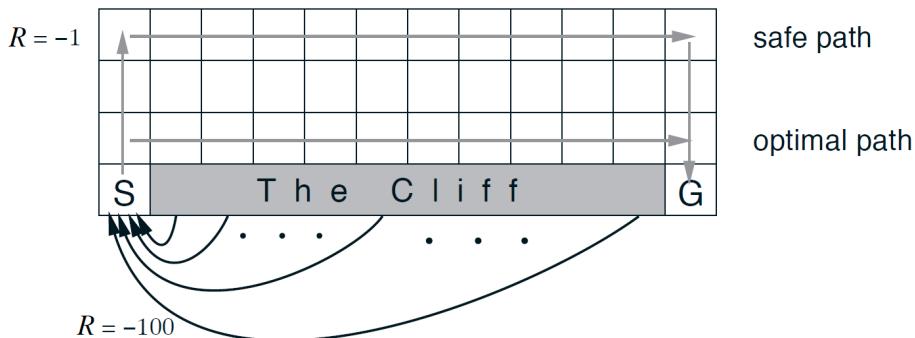
Algoritam je dobio ime prema nizu s, a, r, s', a' na osnovu kojeg ažurira funkciju vrednosti akcije u stanju. Ovaj algoritam konvergira ka optimalno funkciji vrednosti akcije u stanju sa verovatnoćom 1 ukoliko se svi parovi stanja i akcija posećuju beskonačno mnogo puta, ako politika konvergira ka pohlepnoj politici i ako niz α_t zadovoljava Robins-Monroove uslove. Prvi uslov je garantovan izvođenjem ε pohlepnih politika iz funkcije Q , a drugi recimo smanjivanjem vrednosti hiperparametra ε u toku optimizacije, recimo korišćenjem vrednosti $\varepsilon = 1/k$, gde je k broj iteracija.

Već pomenuto pitanje istraživanja prostora stanja i akcija je izuzetno važno u učenju potkrepljivanjem. Ukoliko je optimalna politika dobro aproksimirana, u budućoj primeni je najbolje *eksploatisati* datu politiku. S druge strane, ako bi se politika uvek slepo pratila, ne bismo dobili informacije o boljim alternativama. Otud je važno *istraživati*. Ukoliko se samo vrši nasumično istraživanje, ne akumulira se velika nagrada. Ovo ne zvuči kao problem ako su proces učenja i primene odvojeni. Ipak, ovo ne mora uvek biti slučaj. Recimo, zato što je u realnim okolnostima potrebno što pre početi sa zaradivanjem ili zato što se okruženje menja i nikada ne možemo završiti učenje. Dodatno, neka stanja nisu mnogo važna jer je verovatnoća dolaska u ta stanja mala. Mnogo je važnija brzina konvergencije u stanjima koja se češće sreću. Nasumično istraživanje vodi mnogo sporijoj konvergenciji u slučaju tih stanja, jer pridaje podjednak značaj svim stanjima. Otud je potrebno pažljivo birati vrednost hiperparametra ε koja kontroliše nagodbu između eksplotacije i istraživanja. Prirodno je i da on bude promenljiv – da na počektu ima višu, a kasnije nižu vrednost. Kao i u slučaju drugih hiperparametara, nema opštih preporuka za izbor njegove vrednosti.

Jednostavan i vrlo često korišćen algoritam učenja mimo politike je algoritam Q -učenja (eng. Q -learning) koji se zasniva na sledećem pravilu ažuriranja aproksimacije funkcije q^* :

$$q(s, a) \leftarrow q(s, a) + \alpha_t \left[r + \gamma \max_{a'} q(s', a') - q(s, a) \right]$$

pri čemu se q inizijalizuje nasumično i polazno stanje svake epizode se nasumično bira kad god se neka epizoda završi finalnim stanjem. Kako se bira akcija koja će biti preduzeta nije precizirano algoritmom, pošto je on nezavisan od politike koja se prati u toku učenja. Razlika u odnosu na Sarsa algoritam, koja ovaj algoritam čini algoritmom učenja mimo politike, je u tome što se funkcija vrednosti ažurira prepostavljajući najbolju moguću akciju u narednom koraku,



Slika 13.1: Tabela na kojoj se kreće agent i na kojoj su označene bezbedna i optimalna politika.

a ne onu koja odgovara akciji koja stvarno biva preduzeta u narednom koraku. Time se uči funkcija vrednosti u odnosu na optimalnu politiku, bez obzira što se prilikom učenja možda prati neka druga politika. I ovaj algoritam konvergira optimalnoj funkciji vrednosti akcije u stanju, ukoliko se svi parovi stanja i akcija ponavljaju beskonačno puno puta, a α_t zadovoljava Robins-Monroove uslove.

Primer 13 Ilustrujmo na jednom primeru razliku u ponašanju pomenutih algoritama. Na slici 13.1 dat je primer okruženja u kojem agent treba da stigne od polja S do polja G . Svaki potez nosi nagradu -1 , osim poteza koji završavaju padom sa litice koji nose nagradu -100 . Prilikom učenja, koristi se ε -pohlepna politika za $\varepsilon = 0.1$. Q učenje očito dolazi do optimalne politike, koja vodi kraćim putem duž litice. Sarsa uči politiku koja vodi do cilja daljim, bezbednijim, ali skupljim putem. Otkud ovakva razlika? Pošto uči mimo politike, Q učenje uči zaista optimalnu politiku. Ukoliko bi agent pratio ε pohlepnu politiku izvedenu iz nje, kao što se radi prilikom učenja, to bi ga moglo skupo koštati, ali optimalna politika je deterministička. S druge strane, Sarsa prati politiku koju uči, a koja je nužno ε -pohlepna, tako da ona uči optimalnu politiku iz skupa svih ε -pohlepnih politika! Za takvu politiku, kretanje duž litice je problematično i bolje je ići daljim, a bezbednjim putem. Kako se ε smanjuje i Sarsa uči politiku sve bližu optimalnoj.

13.2 Funkcionalna aproksimacija

Prethodne metode učenja su prepostavljale diskretnost skupa stanja i akcija. Diskretnost skupa stanja nekada može biti i realistična prepostavka, ali ređe i do takvih modela se često dolazi diskretizacijom neprekidnog prostora. Kao što je već komentarisano, to neretko vodi lošim rezultatima zbog prokletstva dimenzionalnosti. Dodatno, legitimno je postaviti pitanje u kom smislu je

učenje nad takvim skupom stanja stvarno učenje. Naime, funkcije vrednosti u nekom stanju se aproksimiraju na osnovu prethodnog iskustva vezanog za to stanje. Međutim, to ne predstavlja generalizaciju u smislu na koji smo navikli. U diskusiji nadgledanog učenja, prepostavljali smo da se predviđanje vrši nadinstancama koje nismo ranije videli, dok prethodno diskutovani metodi učenja potkrepljivanjem nikako ne generalizuju na stanja koja ranije nisu viđena. Ukoliko bi stanja bila opisana nekim atributima, onda bi znanje naučeno u nekim stanjima moglo biti generalizovano na stanja sa sličnim vrednostima atributa, iako konkretna stanja nikad ranije nisu viđena. Odgovor na pomenute probleme je učenje funkcija vrednosti pomoću nekog od već diskutovanih modela nadgledanog učenja. Ipak, samo obučavanje ne može biti vršeno u nadgledanom maniru, tako da će biti potrebno da se algoritmi učenja potkrepljivanjem modifikuju tako da mogu da obučavaju takve modele. Pored rešavanja pomenutih problema, ovaj pristup omogućava prirodnu primenu do sada diskutovanih algoritama na *delimično opazive Markovljeve procese odlučivanja* (eng. *partially observable Markov decision process*) kod kojih se stanje ne može precizno ustanoviti, već se samo opažaju vrednosti nekih njegovih parametara.

Kao i u slučaju nadgledanog učenja, prepostavlja se da se funkcije v i q aproksimiraju parametrizovanim funkcijama v_w i q_w ili da se proces aproksimacije vrši neparametarski. U nastavku će biti prepostavljen parametarski pristup, ali to nije suštinsko ograničenje.

Prvi zadatak je prirodno aproksimacija funkcije vrednosti, na primer, stanja. Problem aproksimacije se može postaviti na već poznat način:

$$\min_w \int_{s \in S} (v(s) - v_w(s))^2 ds$$

odnosno, na konačnom uzorku:

$$\min_w \sum_{t=0}^T (v(s_t) - v_w(s_t))^2 ds$$

Koeficijenti se mogu ažurirati stohastičkim gradijentnim spustom na sledeći način

$$w_{t+1} = w_t - \alpha_t (v(s_t) - v_w(s_t)) \nabla_w v_w(s_t)$$

Ova formula se ne može direktno primeniti pošto vrednost $v(s_t)$ nije poznata. Kao i ranije, ona se može aproksimirati pomoću već viđenog trika koji daje sledeću formulu:

$$w_{t+1} = w_t - \alpha_t (r_{t+1} + \gamma v_w(s_{t+1}) - v_w(s_t)) \nabla_w v_w(s_t)$$

Analogna formula sa može izvesti i za funkciju vrednosti akcije u stanju.

Slično se modifikuju i algoritmi Sarsa i Q-učenje. U prvom slučaju, ažuriranje se vrši po sledećem pravilu:

$$w_{t+1} = w_t - \alpha_t (r_{t+1} + \gamma q_w(s_{t+1}, a_{t+1}) - q_w(s_t, a_t)) \nabla_w q_w(s_t, a_t)$$

a u drugom po pravilu:

$$w_{t+1} = w_t - \alpha_t (r_{t+1} + \gamma \max_a q_w(s_{t+1}, a) - q_w(s_t, a_t)) \nabla_w q_w(s_t, a_t)$$

Sarsa konvergira pod uslovima konvergencije stohastičkog gradijentnog spusta. S druge strane, Q učenje može i da divergira! Naime, oznato je da kombinacija funkcionalne aproksimacije, učenja mimo politike i aproksimacije vrednosti stanja na osnovu tekućih ocena vrednosti drugih stanja dovodi do problema u obučavanju. Izostavljanje bilo kojeg od ovih elemenata se pokazalo kao spasonosno u smislu stabilnosti konvergencije. Ipak, to često nije prihvatljivo. Funkcionalna aproksimacija je od suštinskog značaja za primenljivost učenja potkrepljivanjem, pošto pruža generalizaciju na stanja koja nisu viđena i skalabilnost, koju zbog memorijskih i računskih zahteva diskretne varijante nemaju. Nekada je poželjno učiti više poslova odjednom, koji ne dele istu optimalnu politiku, pa učenje u skladu sa politikom nije moguće. Pomenuta vrsta aproksimacije se može zamjeniti sakupljanjem velikih uzoraka nagrada u različitim stanjima za različite preduzete akcije, čime se može eliminisati korišćenje tekućih ocena vrednosti drugih stanja, ali takav postupak je računski značajno zahtevniji, pa odustajanje od pomenute aproksimacije nije isplativo. Stoga je važno naći algoritam koji radi dovoljno dobro u slučaju ovakve kombinacije. Prvo je važno identifikovati izvore problema, a ima ih više i to su: korelacije u sekvenci prelaza, činjenica da male izmene funkcije q_w mogu voditi značajno različitim optimalnim politikama u odnosu na tu funkciju i korelacije između vrednosti funkcije $q_w(s_t, a)$ i vrednosti $r + \gamma \max_a q_w(s_{t+1}, a)$, pomoću koje se ta vrednost ažurira, koje i jedne i druge zavise od istih parametara w . Dodatan problem je neefikasnost stohastičkog gradijentnog spusta sa pojedinačniminstancama.

Rešavanje svih ovih problema se vrši u malim koracima, traje decenijama i još nije završeno. Jedan algoritam koji se relativno dobro nosi sa ovim problemima je *duboka Q mreža* (eng. *deep Q network*). Naziv očigledno prepostavlja upotrebu neuronske mreže za funkcionalnu aproksimaciju, ali algoritam ne insistira na toj prepostavci. Kompletan formulacija algoritma je na slici 13.2.

hiperparametri algoritma su dužina memorije N , broj epizoda M , veličina podskupa D' , broj iteracija C nakon kojih se pamte parametri w i učestalost istraživanja ε . Razmotrimo na koji način ovaj algoritam odgovara na prethodne probleme. Pamćenjem većeg broja prelaza u nizu D omogućava se da se stohastički gradijentni spust ne vrši na pojedinačnim instancama, već na skupovima instanci. Ipak, to nije jedina korist od čuvanja prelaza. Uzimanjem nasumičnog podskupa, koji se ne sastoji nužno od uzastopnih prelaza, instance nad kojima se vrši gradijentni spust postaju manje korelisane. Ažuriranje funkcije q_w u odnosu na funkciju $q_{w'}$ čiji se parametri tek povremeno menjaju, čini da su aproksimativne i ciljne vrednosti manje korelisane i da je proces obučavanja stabilniji. Na primer, ako bismo se oslonili samo na parametre w , ažuriranje vektora w koje povećava vrednost $q(s_t, a_t)$ često uvećava i $q(s_{t+1}, a)$ za sve akcije a , a time i ciljnu vrednost y_j , što može voditi divergenciji. Fiksiranje pa-

```

1 Alocirati niz  $D$  dužine  $N$ 
2 Nasumično inicijalizovati parametre  $w$ 
3 Inicijalizovati parametre  $w'$  na  $w$ 
4 for  $i = 1, M$  do
5   Inicijalizovati polazno stanje  $s_1$ 
6   for  $t = 1, T$  do
7      $a_t = \begin{cases} \text{nasumična akcija} & \text{sa verovatnoćom } \varepsilon \\ \operatorname{argmax}_a q_w(s_t, a) & \text{sa verovatnoćom } 1 - \varepsilon \end{cases}$ 
8     Izvršiti  $a_t$  i opaziti  $r_{t+1}$  i  $s_t$ 
9     Sačuvati  $(s_t, a_t, r_{t+1}, s_{t+1})$  u  $D$ 
10    Nasumice izabratи podskup  $D'$  из  $D$ 
11    for  $j = 1, |D'|$  do
12       $y_j = \begin{cases} r_{j+1} & s_{j+1} \text{ je završno stanje} \\ r_{j+1} + \gamma \max_a q_{w'}(s_{j+1}, a) & \text{inače} \end{cases}$ 
13    end
14    Izvršiti po  $w$  korak gradijentnog spusta za funkciju

$$\sum_{j=1}^{|D'|} (y_j - q_w(s_j, a_j))^2$$

15    Ako je  $t$  deljivo sa  $C$  postaviti  $w' = w$ 
16  end
17 end

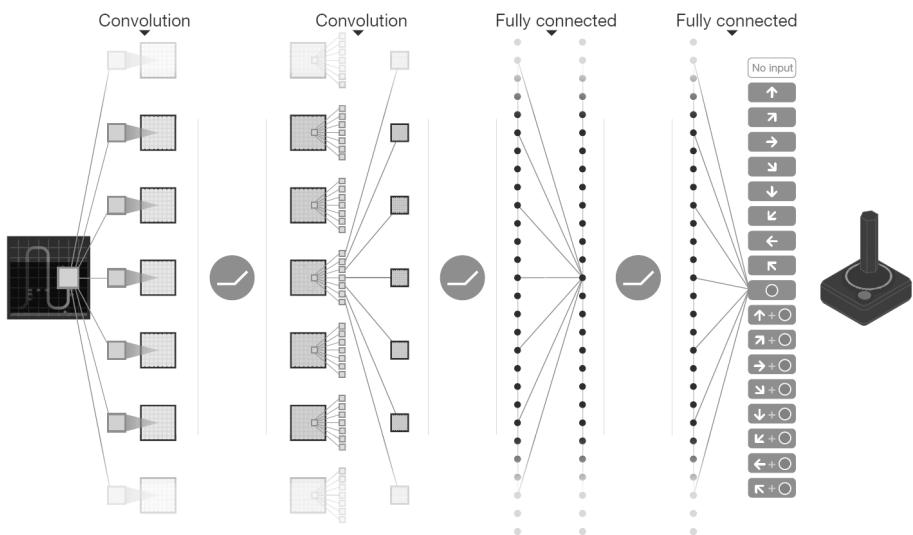
```

Slika 13.2: Duboka Q mreža.

rametara w' prilikom izračunavanja ciljnih vrednosti y_t vodi stabilnim ciljnim vrednostima.

Primer 14 Duboka Q mreža je prvi put objavljena u radu u kojem je poslužila za igranje 49 igara razvijenih za Atari 2600. Arhitektura mreže ilustrovana je slikom 13.3. Kako se mreža ne bi evaluirala za svaku kombinaciju stanja i akcije, za dato stanje, mreža na izlazima paralelno daje vrednosti svih akcija u tom stanju. Ulaz u mrežu se dobija tako što se slike dimenzija 210×160 piksela skaliraju na 84×84 piksela, izdvoji se Y kanal (eng. luminance) i 4 takve uzastopne slike se daju mreži kao istovremeni ulaz. Prvi skriveni sloj uključuje 32 filtera dimenzija 8×8 sa pomerajem 4, drugi 64 filtera dimenzija 4×4 sa pomerajem 2, a treći 64 filtera dimenzija 3×3 sa pomerajem 1. Svi konvolutivni slojevi su praćeni ispravljenim linearnim jedinicama. Potpuno povezana mreža ima jedan sloj od 512 skrivenih jedinica, dok izlaz, u zavisnosti od igre, varira od 4 do 18 akcija.

U toku obučavanja, veličina podskupa na kojem se računa gradijent je bila 32 instance, a ε je smanjivano od 1 do 0.1 u toku prvih milion frejmova, a



Slika 13.3: Skica arhitekture mreže koja igra igre za Atari 2600.

potom je fiksirano na 0.1. Parametar N je imao vrednost milion. Korišćeno je pokoordinatno odsecanje gradijenata na interval $[-1, 1]$. Obučavanje je trajalo 50 miliona frejmova, odnosno ukupno 38 dana. Na preko polovini igara postignuto je preko 75% skora profesionalnih igrača. U međuvremenu, ovi uspesi su nadmašeni.

Deo III

Nenadgledano učenje

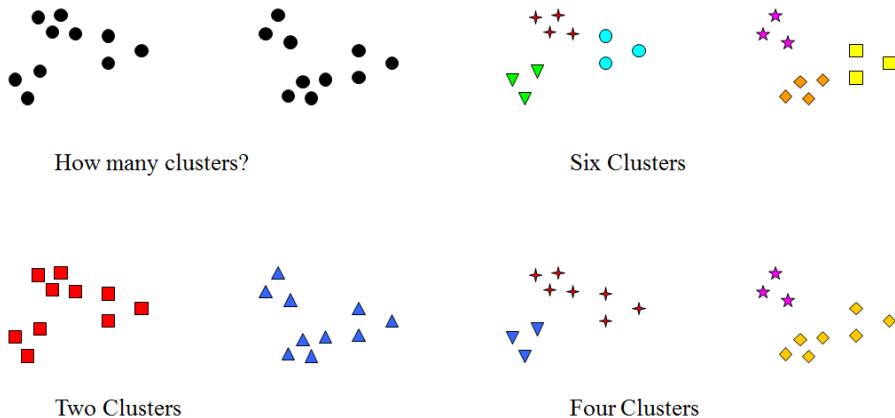
Glava 14

Klasterovanje

Klasterovanje predstavlja identifikaciju grupa u datim podacima. Potreba za rešavanjem ovakvog problema može se javiti u različitim praktičnim problemima, poput identifikacije zajednica u društvenim mrežama (na primer, za potrebe oglašavanja), detekcije raznorodnih tkiva na medicinskim snimcima, ustanovljavanja zajedničkog porekla jezika, živih bića i specifično ljudskih zajednica, i slično. Pored primena koje u sebi direktno kriju problem klasterovanja, ove tehnike su često korisne i zarad pretprecesiranja podataka na koje će biti primenjene metode nadgledanog učenja. Na primer, u slučaju obrade огромнog broja prodataka, cele grupe podataka mogu biti zamjenjene svojim reprezentativnim predstavnicima. Ovo nije idealno sa tačke gledišta kvaliteta dobijenog prediktivnog modela, ali sa tačke gledišta računske i memoriske efikasnosti može biti isplativo. Takođe, kako bi se obezbedilo da prilikom unakrsne validacije različiti slojevi imaju slično raspodeljene podatke, podaci se mogu prvo klasterovati, a potom se n slojeva može formirati tako što se svaki od klastera podeli na n jedakih delova koji se razvrstaju u različite slojeve. Očito, klasterovanje može biti korisno kako kao tehnika rešavanja problema, tako i kao tehnika pretprecesiranja.

Pojam klasterovanja nije jednoznačno definisan. Kao što primer prikazan na slici 14.1 ukazuje, u jednom skupu se može identifikovati više različitih grupisanja, često različite granularnosti. Pritom, takvi slučajevi nisu posledica nedovoljnog promišljanja definicije klasterovanja, već raznovrsnosti konteksta u kojima se grupisanje može vršiti i ciljeva koji se pomoću klasterovanja žele postići. Za očekivati je da je nekada potrebno izvršiti grublje klasterovanje – u manji broj klastera, a nekada finije – u veći broj klastera. Algoritmi klasterovanja obično omogućavaju podešavanje nivoa granularnosti, odnosno broja klastera koji se u podacima pronalazi.

Pojam klasterovanja nije jednoznačno definisan ne samo u odnosu na broj klastera koji se u podacima mogu naći, već i u odnosu na ideju šta jednu grupu tačaka čini klasterom. U odnosu na to, postoji više neformalnih definicija klasterovanja. *Globularni* ili *centrični* klasteri su grupe tačaka koje popunjavaju unutrašnjost lopte ili eventualno elipsoida. *Dobro razdvojeni* klasteri su



Slika 14.1: Različita klasterovanja nad istim podacima.

grupe tačaka koje su bliže drugim tačkama svoje grupe nego bilo kojoj tački iz neke druge grupe. *Gustinski* klasteri su klasteri čije su tačke razdvojene od tačaka drugih klastera regionima manje gustine. *Hijerarhijski* klasteri su ili pojedinačne tačke ili klasteri čije su tačke takođe organizovane u strukturu hijerarhijskih klastera.

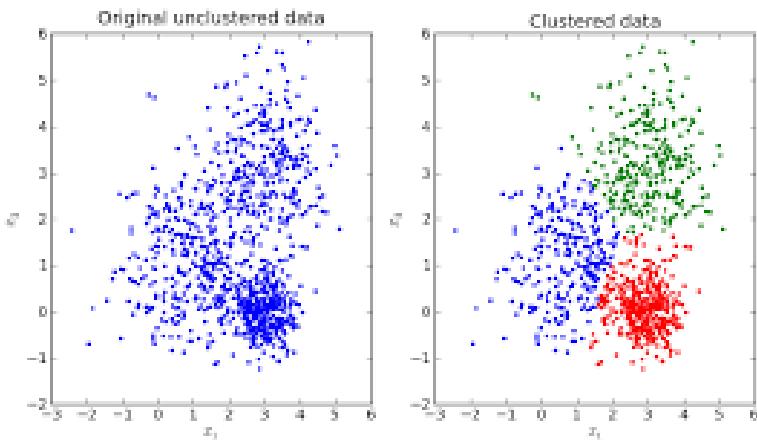
U nastavku su prikazana dva poznata algoritma klasterovanja.

14.1 K sredina

Algoritam k sredina pronalazi k klastera u podacima koje predstavlja pomoću k centroida tih klastera, od kojih se svaka dobija uprosečavanjem elemenata datog klastera. Ova pretpostavka čini algoritam primenljivim samo na podatke koji se mogu uprosečavati, poput vektora. Pod određenim uslovima, postoje uopštenja algoritma i na drugačije vrste podataka, ali o njima neće biti reči. Polaznih k centroida se bira nasumično (mada, ako korisnik zna nešto o strukturi svojih podataka, mogu biti i unapred date), a potom se ponavljaju sledeći koraci:

1. rasporediti sve instance u nove klastere tako što se svaka instanca pridruži najbližoj centroidi
2. izračunati nove centroide kao prosek instanci koje su im pridružene.

Ovi koraci se izvršavaju sve dok se centroide menjaju. Kada su centroide iste u dve uzastopne iteracije, algoritam se zaustavlja.



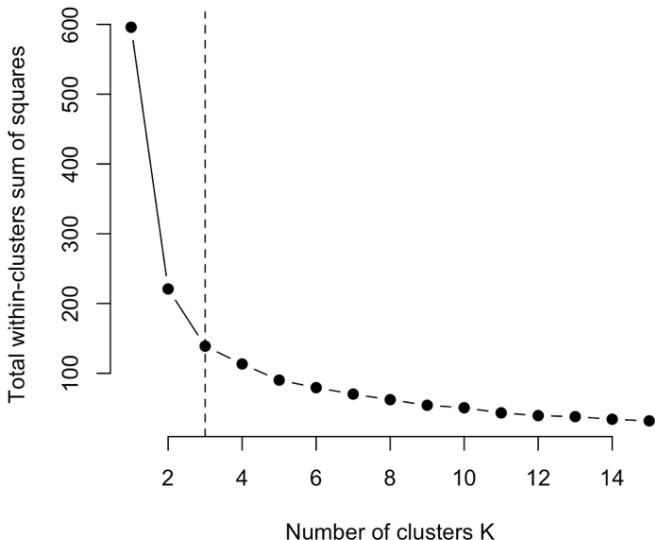
Slika 14.2: Klasteri pronađeni algoritmom k sredina.

Primer klastera koje pronalazi ovaj algoritam, dat je na slici 14.2. Može se pokazati da ovaj algoritam minimizuje funkciju

$$\sum_{i=1}^k \sum_{x \in C_i} d(x, c_i)^2$$

po c_i , gde je d euklidsko rastojanje (ali je moguće koristiti i neko drugo). Na osnovu ovoga se može nešto zaključiti i o njegovom ponašanju. Zahvaljujući tome što je zasnovan na minimizaciji euklidskog rastojanja, algoritam teži pronađenju klastera u obliku lopte. Kako je rastojanje kvadrirano, algoritam je osetljiv na podatke koji značajno odudaraaju od ostalih. U tom slučaju će veće rastojanje uticati na ukupnu grešku neproporcionalno u odnosu na ostala rastojanja i takva tačka će neprpororacionalno uticati na lokaciju centroide. Takođe, ako gustina tačaka ne varira drastično i rastojanja među klasterima nisu velika, algoritam preferira klastera sa sličnim brojem tačaka u njima, pošto bi u tom slučaju brojan klaster morao sadržati i tačke daleko od centroide koje bi značajno povećavale sumu kvadrata rastojanja.

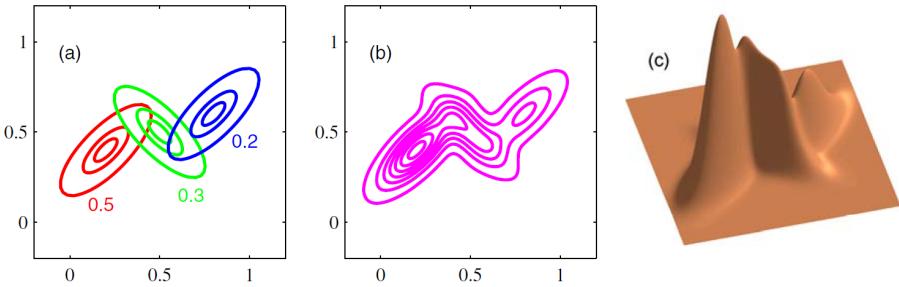
Činjenica da algoritam k sredina minimizuje navedenu sumu navodi na njen dalju analizu. Bitno je pitanje da li ona ima jedan globalni minimum, odnosno da li je najbolje klasterovanje u odnosu na datu sumu kvadrata rastojanja jedinstveno. Odgovor na prvo pitanje je negativan. Moguće je da postoji veći broj klasterovanja jednakog kvaliteta. Jedan primer u kojem bi to bilo i intuitivno je kada su tačke uniformno raspoređene unutar kruga i potrebno ih je podeliti na dva klastera. Rotiranje dobijenih centroida u odnosu na centar kruga daje podjednako dobro klasterovanje. Drugim rečima, u slučaju takvog skupa podataka, postoji puno globalnih, i samim tim podjednako dobrih, minimuma. Takva situacija nije zabrinjavajuća. Ipak, ispostavlja se da mogu postojati i lo-



Slika 14.3: Pravilo lakta sugerije da za broj k treba uzeti vrednost koja odgovara uspravnoj isprekidanoj liniji.

kalni minimumi slabijeg kvaliteta od globalnog i da algoritam može naći takav minimum, što nije dobro. Ovaj problem se ublažava tako što se klasterovanje pokreće veći broj puta sa od različitih inicijalnih tačaka i za rezultat se uzima klasterovanje najmanje vrednosti sume kvadrata rastojanja.

Algoritam k sredina omogućava fleksibilnost pri pronalaženju klastera kroz mogućnost podešavanja broja k . Ipak, u praksi često nije jasno kako izabратi broj k i pomenuta fleksibilnost često vodi nedoumici. Jedno pravilo heurističko je „pravilo lakta“ koje sugerije da se za različite vrednosti broja k izvrši klasterovanje, da se nacrti grafik zavisnosti sume kvadrata rastojanja u zavisnosti od k i da se za izabere klasterovanje koje odgovara broju k koji leži na tački nagle promene brzine opadanja grafika ili na njegovom „laktu“. Ovakva situacija je prikazana na grafiku 14.3. Intuitivno obrazloženje je da su nakon „lakta“ klasteri već homogeni i dodavanje novih centroida ne doprinosi značajno smanjenju sume kvadrata rastojanja.



Slika 14.4: Ilustracija mešavine normalnih raspodela. Na slici (a) prikazane su konture tri normalne raspodele sa pridruženim verovatnoćama da tačka pripada klasteru (brojevima 0.5, 0.3 i 0.2). Na slici (b) prikazane su konture mešavine, a na slici (c) njen grafik.

14.2 Mešavina normalnih raspodela i EM algoritam

Mešavina normalnih raspodela (eng. *mixture of Gaussians*) predstavlja nešto sofisticiraniji model klasterovanja od modela k sredina, ali je zanimljivo primećiti da postoji i veza, koja će biti objašnjena na kraju. Osnovna pretpostavka je da se podaci mogu podeliti u određeni broj relativno kompaktnih globularnih klastera čiji se oblik može dobro opisati normalnim raspodelama sa različitim proseccima i matricama kovarijacije. Proseci, jasno, definišu pozicije klastera u prostoru, dok matrice kovarijacije opisuju njihov oblik i orientaciju u prostoru. Površi jednakog gustine u okviru jednog klastera u tom slučaju predstavljaju elipsoide. U ovom modelu, raspodela je malo složenija nego u dosadašnjim, ali ima prirodnu dekompoziciju na jednostavnije raspodele. Zamislimo kako se može dobiti nasumično generisana tačka iz ovakve raspodele. Prvo se može nametnuti da se izabire klaster, a potom se iz tog klastera izabere tačka u skladu sa normalnom raspodelom koja mu odgovara. Ukoliko je C broj klastera i za brojeve p_1, \dots, p_C važi $p_i \geq 0$ i $\sum_{i=1}^C p_i = 1$, onda (p_1, \dots, p_C) predstavlja multinomijalnu raspodelu nad klasterima. Gustina raspodele nadinstancama se može zapisati kao:

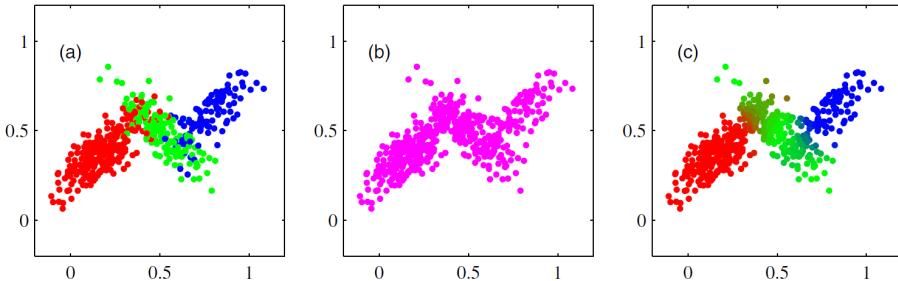
$$p(x) = \sum_{i=1}^C p_i \mathcal{N}(x; \mu_i, \Sigma_i)$$

gde je \mathcal{N} gustina normalne raspodele sa više promenljivih

$$\mathcal{N}(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp(-(x - \mu)^T \Sigma^{-1} (x - \mu))$$

Ovo je ilustrovano slikom 14.4.

Primetimo da ovaj model prestavlja generativni model i mogao bi se koristiti u različite svrhe, pa i u svrhe klasifikacije. Pošto svakoj klasi može odgovarati



Slika 14.5: Ilustracija podataka iz mešavine normalnih raspodela. Na slici (a) prikazani su podaci kao uzorci iz raspodele $p(x, z) = p(x|z)p(z)$, zbog čega se na slici vidi i njihova boja, koja se dobija na osnovu multinomijalne raspodele i pozicija, koja se dobija na osnovu normalne raspodele. Na slici (b), prikazana je informacija koja se može dobiti na osnovu marginalne raspodele $p(x)$ i otud nije poznata informacija o promenljivim z , odnosno o bojama. Na slici (c) tačke su obojene mešavinom boja klastera, pri čemu je svaka od boja zastupljena proporcionalno verovatnoći da tačka pripada tom klasteru.

jedan klaster, uslovna raspodela $p(x|z = c)$ je definisana kao dogovarajuća normalna raspodela, a ona se može koristiti za klasifikaciju, jer Bajesova teorema omogućava da se na osnovu nje izračuna $p(y|x)$. Ipak, time se nećemo baviti. U kontekstu klasterovanja, neka promenljiva z uzima vrednosti od 1 do C u skladu sa multinomijalnom raspodelom (p_1, \dots, p_C) . Verovatnoća da instanca pripada klasteru k je

$$p_w(z = k|x) = \frac{p_w(x|z = k)p_w(z = k)}{p_w(x)} = \frac{p_w(x|z = k)p_w(z = k)}{\sum_{k=1}^C p_w(x|z = k)p_w(z = k)}$$

gde je w vektor parametara koji objedinjuje sve parametre iz svih vektora μ_k i matrica Σ_k , kao i (p_1, \dots, p_C) . Ilustracija podataka kao uzoraka iz mešavine raspodela, data je na slici 14.5

Primetimo da, iako se koriste u modelu, vrednosti promenljive z za različite instance nisu poznate u vreme obučavanja. Primetimo takođe, da bi problem bio vrlo jednostavan ukoliko bi bile. Kada su poznate vrednosti z_i za svaku od instanci, parametri μ_k i Σ_k normalne raspodele klastera k se ocenjuju standardnim empirijskim ocenama – prosekom instanci za koje važi $z_i = k$ matricom kovarijacije između svih atributa izračunatom na osnovu tih instanci. U ovakvim situacijama, kada se može prepostaviti da postoje takozvane *latentne*, odnosno *skrivene* promenljive z koje nisu opažene i kada je ocena parametara raspodele $p_w(x)$ teška, a ocena parametara raspodele $p_w(x, z)$ laka (kada su dati empirijski podaci i za x i za z), pribegava se algoritmu *maksimizacije očekivanja* (eng. *expectation maximization*) ili, skraćeno – *EM algoritmu*.

EM algoritam ćemo prikazati prvo u opštem obliku, pa ćemo ga precizirati

za model mešavine normalnih raspodela. Za logaritam funkcije verodostojnosti parametara važi:

$$\ell(w) = \log \mathcal{L}(w) = \sum_{i=1}^N \log p_w(x_i) = \sum_{i=1}^N \log \int_z p_w(x_i, z) dz$$

ili u diskretnom slučaju

$$\ell(w) = \log \mathcal{L}(w) = \sum_{i=1}^N \log p_w(x_i) = \sum_{i=1}^N \log \sum_{k=1}^C p_w(x_i, k)$$

Ovaj problem nije lak za rešavanje, pošto logaritam ne može da prođe kroz sumu. Rešenje se sastoji u posmatranju funkcije verodostojnosti u odnosu na zajedničku raspodelu promenljivih x i z . Imajući u vidu da svakoj instanci pored vrednosti opaženih promenljivih x odgovaraju i vrednosti latentnih promenljivih z , ovo je potpuno legitimno, ali kako vrednosti z nisu poznate, takva funkcija verodostojnosti se ne može izračunati, već predstavlja slučajnu promenljivu. Iako se ona ne može izračunati, može se izračunati njeno očekivanje u odnosu na promenljive z , tako da optimizacija može biti vršena po njemu. Konkretno, logaritam funkcije veodostojnosti koja uzima u obzir i promenljive z je

$$\ell(w) = \sum_{i=1}^N \log p_w(x_i, z_i)$$

Neka je Q pomoćna funkcija

$$Q(w, w^{t-1}) = \mathbb{E}_z[\ell(w)|w^{t-1}] = \int \ell(w)p_{w^{t-1}}(z)dz$$

Nova vrednost parametara w se dobija maksimizacijom funkcije Q . Konkretno

$$w^t = \arg \max_w Q(w, w^{t-1})$$

Ovako definisan algoritam se može podeliti u dva koraka. U prvom se oceњuje funkcija Q . Ovaj korak se naziva E, pošto se u tom postupku zapravo vrši ocena očekivanja. U drugom koraku se vrši maksimizacija, pa se zato naziva korak M. Ovaj algoritam ima jedno zanimljivo svojstvo, a to je da se vrednost funkcije Q nikad ne smanjuje, što može biti korisno pri debagovanju njegove implementacije.

U slučaju problema klasterovanja pomoću mešavine normalnih raspodela,

funkcija Q ima sledeći oblik:

$$\begin{aligned}
Q(w, w^{t-1}) &= \mathbb{E} \left[\sum_{i=1}^N \log p_w(x_i, z_i) \right] \\
&= \sum_{i=1}^N \mathbb{E} \left[\log \left[\prod_{k=1}^C p_k p_w(x_i)^{I(z_i=k)} \right] \right] \\
&= \sum_{i=1}^N \sum_{k=1}^C \mathbb{E}[I(z_i = k)] \log[p_k p_w(x_i)] \\
&= \sum_{i=1}^N \sum_{k=1}^C p_{w^{t-1}}(z_i = k | x_i) \log[p_k p_w(x_i)] \\
&= \sum_{i=1}^N \sum_{k=1}^C p_{w^{t-1}}(z_i = k | x_i) \log p_k + \sum_{i=1}^N \sum_{k=1}^C p_{w^{t-1}}(z_i = k | x_i) \log p_w(x_i)
\end{aligned}$$

Korak E se sastoji od izračunavanja veličine $p_{w^{t-1}}(z_i = k | x_i)$ na sledeći način:

$$p_{w^{t-1}}(z_i = k | x_i) = \frac{p_k \mathcal{N}(x; \mu_k^{t-1}, \Sigma_k^{t-1})}{\sum_{j=1}^C p_j \mathcal{N}(x; \mu_j^{t-1}, \Sigma_j^{t-1})}$$

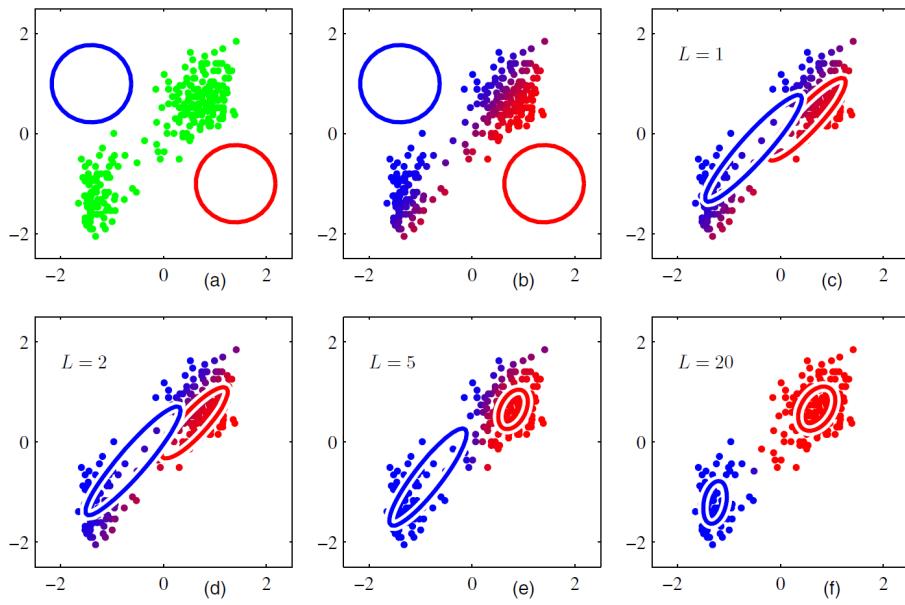
Kako su sve vrednosti koje izraz uključuje definisane u prethodnom koraku, ovaj izraz je lako izračunati.

Korak M se sastoji od maksimizacije funkcije Q po w , odnosno po veličinama p_k , μ_k i Σ_k . Može se izvesti sledeće rešenje:

$$\begin{aligned}
p_k &= \frac{1}{N} \sum_{i=1}^N p_{w^{t-1}}(z_i = k | x_i) \\
\mu_k &= \frac{\sum_{i=1}^N p_{w^{t-1}}(z_i = k | x_i) x_i}{\sum_{i=1}^N p_{w^{t-1}}(z_i = k | x_i)} \\
\Sigma_k &= \frac{\sum_{i=1}^N p_{w^{t-1}}(z_i = k | x_i) (x_i - \mu_k)^T (x_i - \mu_k)}{\sum_{i=1}^N p_{w^{t-1}}(z_i = k | x_i)}
\end{aligned}$$

Primetimo da su izvedene formule intuitivne. Naime, p_k je praktično udeo instanci koje su pridružene klasteru i , s tom ogradiom da sad svaka instanca pripada svakom klasteru, ali sa težinom $p_{w^{t-1}}(z_i = k | x_i)$. prosek ovih težina za klaster k je ocena p_k . Slično, prosek klastera k je prosek svih instanci, ali otežanih u skladu sa udelom sa kojim pripadaju tom klasteru. Interpretacija formule za kovarijansu je potpuno analogna. Izvršavanje ovog algoritma u više iteracija ilustrovano je na slici 14.6.

Primetimo da se algoritam k sredina može posmatrati kao jednostavnija verzija klasterovanja pomoću mešavine normalnih raspodela koja se karakteriše sledećim prepostavkama:



Slika 14.6: Izvršavanje EM algoritma za mešavinu normalnih raspodela u 20 koraka.

- $\Sigma_k = \sigma^2 I$ za svako k ,
- $p_k = 1/C$ za svako k i
- $p(z_i = k|x_i)$ je 1 ako je među svim prosećima μ_j instanci x_i najbliži baš μ_k , a 0 u suprotnom

Očito, klasterovanje zasnovano na mešavini normalnih raspodela je značajno izražajnije – može prirodno modelovati ne samo loptaste klastere, već i klastere u obliku elipsoida različitih orientacija u prostoru. Pritom njihov oblik se može ustanoviti na osnovu matrice kovarijanse, što znači da ovaj model omogućava i interpretaciju. Takođe, dodele instanci klasterima nisu tvrde, već mogu izražavati i pouzdanost da instanca pripada nekom klasteru.

Glava 15

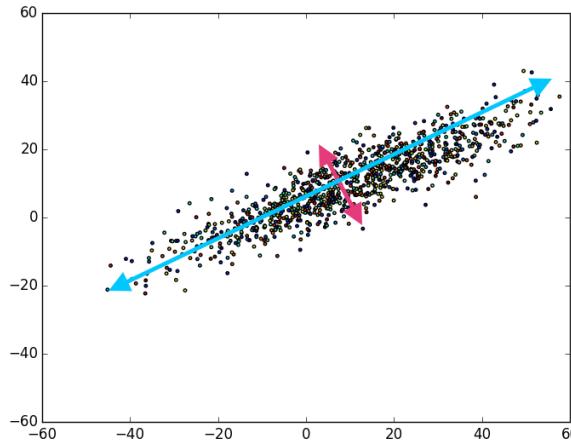
Učenje reprezentacije

Učenje reprezentacije je opšti naziv za mnoštvo pristupa koji se zasnivaju na preslikavanju instanci u neki drugi prostor u kojem relevantna svojstva tih instanci mogu lako biti iskorišćena. Nekada se radi o nenadgledanim pristupima, što znači da se naučeno preslikavanje (tzv. *enkoder*) može koristiti u različitim kontekstima, poput nadgledanog učenja, analize podataka, vizualizacije i slično. Nekada metode nadgledanog učenja kao svoj sastavni deo imaju učenje reprezentacije. Takve reprezentacije su tipično korisne samo za dati problem. U nastavku ćemo se baviti pravopomenutim, nenadgledanim pristupima. Ključni princip na kojem se zasnivaju je da nove reprezentacije instanci u nekom smislu treba da budu kompaktne, ali da se iz njih mogu rekonstruisati polazne instance ili neki njihovi važni aspekti.

15.1 Metod glavnih komponenti

Metod glavnih komponenti zasniva se na zapažanju da usled linearnih zavisnosti ili korelacija među atributima podataka, instance ne popunjavaju uniformno prostor atributa, već, ugrubo, samo neki njegov linearni potprostor, često značajno manje dimenzije od polaznog. Ukoliko bi se instance predstavile u ortogonalnom koordinatnom sistemu tog prostora i ukoliko bi se dalje radio sa tim reprezentacijama, mogli bi se umanjiti negativni efekti koju visoka dimenzionalnost i korelisanost atributa imaju na metode mašinskog učenja.

Metod ćemo objasniti na nivou intuicije. Visoka korelisanost atributa znači da se neki od atributata mogu približno izraziti u funkciji drugih i da nisu svi važni. Kako bi se taj fenomen iskoristio, potrebno je prvo ustanoviti koji atributi su koliko korelirani. Standardni alati kojim se opisuju takve zavisnosti su matrice korelacija i kovarijanse. U tim matricama, element σ_{ij} označava korelaciju, odnosno kovarijansu atributa x_i i x_j . Koja od ovih matrica će se koristiti zavisi od toga da li atributi variraju u sličnim rasponima i da li su slične prirode. Ukoliko jesu, preferira se matrica kovarijanse. Ukoliko nisu, preferira se matrica korelacija, koja predstavlja kovarijansu standardizovanih promenljivih.



Slika 15.1: Primer glavnih komponenti varijacije podataka u dvodimenzionalnom slučaju.

Zato se u nastavku govori o matrici kovarijanse, što obuhvata i slučaj da se radi sa matricom korelacije.

Kako je matrica kovarijanse realna, simetrična i pozitivno definitna, njeni sopstveni vektori čine ortogonalni sistem prostora, a sopstvene vrednosti su nenegativne. Ono sto je još zanimljivije je da se može pokazati da sopstveni vektor koji odgovara po absolutnoj vrednosti najvećoj sopstvenoj vrednosti maksimizuje varijansu projekcije podataka među svim pravcima u polaznom prostoru. Isto važi za svaki sledeći sopstveni vektor kada se podaci prvo projektuju na potprostor ortogonalan na sve prethodne sopstvene vektore. Ovi vektori se nazivaju glavnim komponentama varijacije podataka, ili skraćeno *glavnim komponentama* (eng. *principal components*). Na slici 15.1 dat je primer za dvodimenzionalni slučaj.

Sopstveni vektori matrice kovarijanse atributa i sopstvene vrednosti koje im odgovaraju mogu se dobiti bilo nekim metodom za nalaženje sopstvenih vektora, bilo singularnom dekompozicijom matrice podataka X , što je način na koji se ova metoda tipično implementira. Ukoliko je data matrica podataka X i ako je V matrica čije su kolone koordinate sopstvenih vektora, koordinate u novom koordinatnom sistemu dobijaju se izračunavanjem XV , pošto matrica baznih vektora sistema služi kao matrica transformacije u novi sistem. Kako su sopstveni vektori ortogonalni, projekcije podataka na njih, koje čine nove koordinate, odnosno attribute su nekorelisane, pa se otud varijansa podataka može razbiti na sumu varijansi projekcija podataka na te vektore. Udeo varijanse duž

i -tog sopstvenog vektora je

$$\frac{\lambda_i}{\sum_{j=1}^n \lambda_j}$$

gde je λ_i njemu odgovarajuća sopstvena vrednost matrice kovarijanse. Ovo zapažanje daje jasnu smernicu za izbor dobrog potprostora ukoliko je potrebno sačuvati udeo α polazne varijanse podataka. Matrica V treba da sadrži prvih k sopstvenih vektora takvih da zbir

$$\sum_{i=1}^k \frac{\lambda_i}{\sum_{j=1}^n \lambda_j}$$

bude veći od α . Ukoliko formulacija „sačuvati određeni udeo varijanse podataka” zvuči strano, zapitajmo se na koji način podaci pružaju važnu informaciju za potrebe modelovanja njihovih zavisnosti. Različite tačke podataka svedoče koje vrednosti različite promenljive uzimaju pri kojim vrednostima drugih promenljivih. Otud je različitost, odnosno, raspršenost, odnosno varijansa tačaka podataka bitna. Ako bi se svi podaci projektovali na jednu tačku, ništa od polazne informacije ne bi bilo sačuvano. Ukoliko se linearnom transformacijom projektuju na potprostor u kojem njihove projekcije imaju varijansu od na primer 95% polazne varijanse, praktično sva informacija je sačuvana.

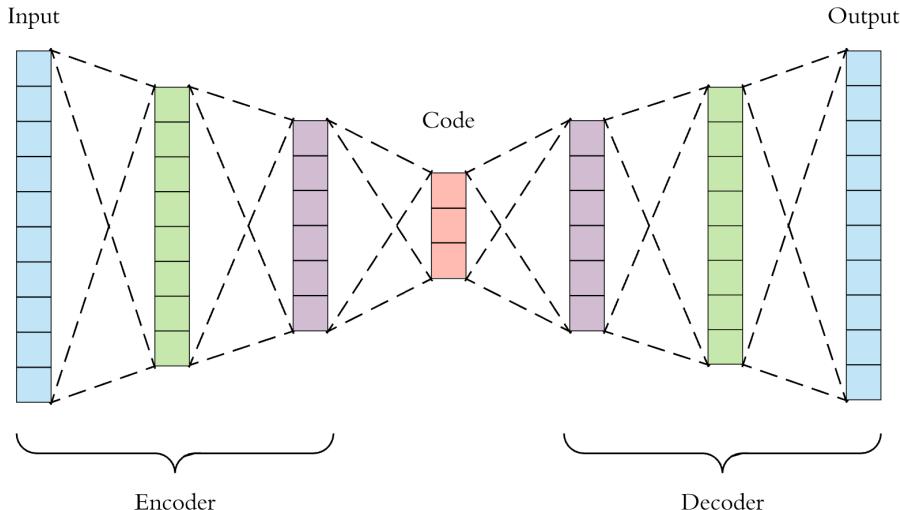
U slučaju da su korelacije među mnoštvom promenljivih jake, novi prostor može sadržati veliku količinu informacije, a imati značajno manju dimenzionalnost. Dodatno, ortogonalnost novog sistema i njom uslovljena nekorelisanost novih atributa, nosi računske pogodnosti za različite metode mašinskog učenja. Ipak, za razliku od polazne, nova reprezentacija je neinterpretabilna. Naime, novi prostor je zarotiran u odnosu na polaznih i svaka njegova osa (novi atribut) je linearna kombinacija starih osa (starih atributa). Otud kad je koeficijent modela koji odgovara nekom novom atributu veliki, nije lako reći šta to znači u terminima polaznih atributa.

15.2 Autoenkoderi

Autoenkoder se može videti kao nelinearni rođak metoda glavnih komponenti. Štaviše, postoje i teorijski rezultati koji bliže karakterišu ovu vezu. Standardni autoenkoder, čija je arhitektura prikazana na slici 15.2, predstavlja neuronsku mrežu koja na svojim izlazima rekonstruiše svoje ulaze. Funkcija greške, poznata kao *greška rekonstrukcije* je:

$$E(w, \mathcal{D}) = \sum_{i=1}^N \|x - f(x)\|_2^2$$

Odavde je jasno da autoenkoder pokušava da nauči funkciju $f(x) = x$, što na prvi pogled deluje beskorisno. Tako deluje zato što smo se navikli da je vrednost modela u njegovim izlazima. Međutim, u slučaju autoenkodera, vrednost



Slika 15.2: Skica arhitekture autoenkodera.

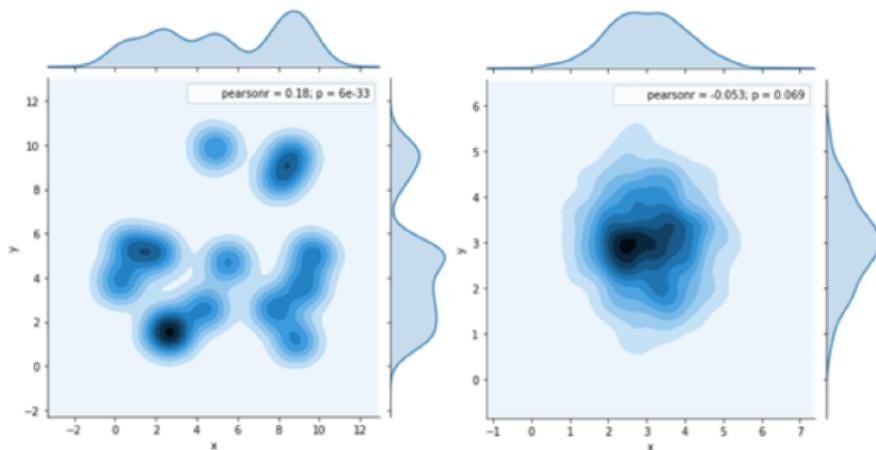
je u preslikavanju iz polazne reprezentacije u reprezentaciju koju daje centralni skriveni sloj. Ta reprezentacija se naziva *latentnim prostorom* (eng. *latent space*). Slojevi enkodera koji realizuju ovo preslikavanje u latentnim prostorom čine *enkoder*, dok slojevi koji realizuju preslikavanje iz latentnog u polazni prostor čine *dekoder*. Dekoder se može posmatrati kao parametrizacija površi u kojoj leže podaci u odnosu na latentni prostor.

Tipično je latentni prostor značajno manje dimenzije od polaznog prostora. Na taj način, efektom uskog grla, forsira se kreiranje kompaktnih, a opet dovoljno informativnih reprezentacija. U praksi se primećuje da podaci koji su bliski u prostoru atributa često bivaju bliski i u latentnom prostoru. To često znači da, ako tačkama a i b u latentnom prostoru odgovaraju na primer slike A i B u prostoru atributa (koji su u tom slučaju pikseli), tačkama na pravoj između a i b odgovara prelaz od slike A do slike B . Ovo je ilustrovano slikom 15.3. Ipak, ovaj prelaz ne mora uvek biti gladak. Naime, latentni prostor ne mora biti jednostavnog oblika (poput lopte ili kocke), tako da je moguće da se pri interpolaciji između tačaka a i b koje odgovaraju poznatim podacima, zalazi u deo latentnog prostora koji uopšte ne odgovara poznatim podacima. Razlika između poželjnog i nepoželjnog izgleda latentnog prostora prikazana je slikom 15.4. Postoje pristupi, na primer takozvani *varijacioni autoenkodери*, koji su konstruisani tako da latentni prostor bude kompaktan u smislu da je skoncentrisan u nekoj lopti i da bliski delovi latentnog prostora odgovaraju sličnim podacima i da prelazi budu glatki. Na taj način je poznat raspon varijacije podataka u latentnom prostoru, a moguće je i interpolirati između tačaka u prostoru atributa. Na primer, između slika.

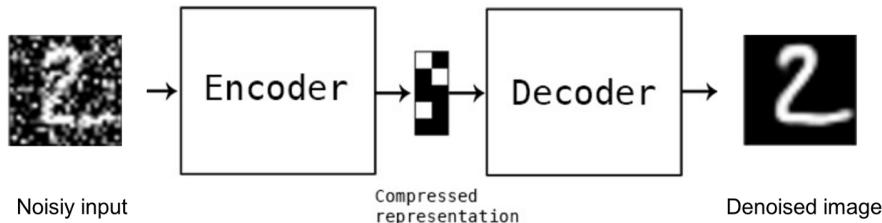
Primene autoenkodera su raznovrsne. Mnogi poslovi, poput klasterovanja

6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
9 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
9 4 2
9 4 2
9 9 4 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 5 5 5 5 5 5 5
9 9 4 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 5 5 5 5 5 5 5
9 9 9 9 9 3 3 3 3 3 3 3 3 3 3 3 3 3 3 5 5 5 5 5 5
9 9 9 9 9 3 3 3 3 3 3 3 3 3 3 3 3 3 3 5 5 5 5 5 5
7 9 9 9 9 9 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
7 9 9 9 9 9 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
7 9 9 9 9 9 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
7 9 9 9 9 9 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
7 9 9 9 9 9 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
7 9 9 9 9 9 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
7 9 9 9 9 9 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
7 9
7 9
7 9
7 9
7 9
7 9
7 9
7 9
7 9
7 9
7 9

Slika 15.3: Vizualizacija slika koje se dobijaju pomeranjem u latentnom prostoru.



Slika 15.4: Nepoželjan oblik latentnog prostora (levo) i poželjan oblik latentnog prostora.

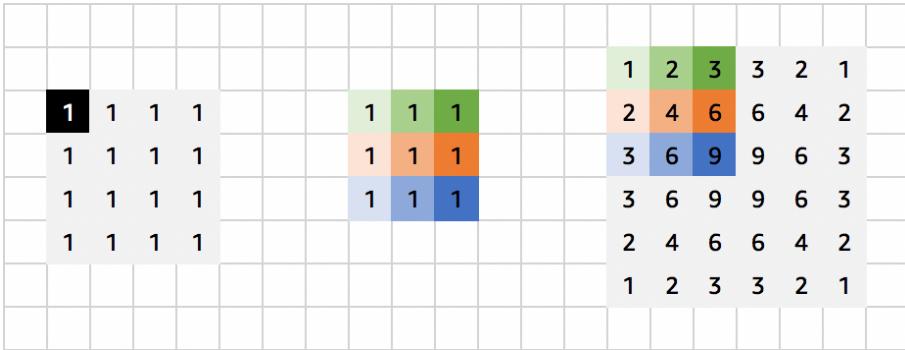


Slika 15.5: Shema autoenkodera za uklanjanje šuma.

ili bilo koje vrste nadgledanog učenja, mogu biti jednostavniji nad niskodimenzionalnim reprezentacijama, koje pritom zadržavaju dovoljno informacije o podacima i vrlo često čuvaju i sličnost. Pretraga nad slikama visoke rezolucije, na primer da bi se našla najsličnija slika zadatoj, može biti značajno olakšana ukoliko se vrši nad niskodimenzionalnim vektorima kojima su poznate slike predstavljene. Tada je dovoljno dobiti novu reprezentaciju zadate slike pomoću enkodera i uraditi pretragu u latentnom prostoru. Takođe, modelovanje gustine raspodele je lakše u niskodimenzionalnim prostorima. Ovo se može upotrebiti za detekciju odudarajućih podataka, na primer tako što se modeluje raspodela podataka u latentnom prostoru, a onda se za nove instance odlučuje da li su odudarajući podaci na osnovu toga koliku vrednost gustina raspodele daje njihovim slikama u latentnom prostoru dobijenim pomoću enkodera. Ukoliko je vrednost niža od nekog unapred zadatog praga, radi se o odudarajućem podatku.

Autoenkoder može poslužiti i uklanjanju šuma (eng. *denoising autoencoder*), na primer tako što se prilikom obučavanja na ulazima daju slike pokvarene šumom, greška rekonstrukcije se izračunava u odnosu na slike bez šuma. Naravno, upotrebljivost autoenkodera u ovakovom kontekstu značajno zavisi od toga da li je šum sa kojim je vršeno obučavanje šum koji će se javiti u praktičnom kontekstu upotrebe. Željeno ponašanje je ilustrovano slikom 15.5.

Kao što se iz prethodne diskusije naslućuje, autoenkoderi se često primenjuju na slike. Kao što znamo, u takvim primenama, poželjno je koristiti konvolutivne mreže. Iz dosadašnje diskusije o konvolutivnim mrežama, lako je zaključiti kako je moguće smanjivati dimenziju reprezentacije i tako izgraditi enkoder. Ipak, nije jasno kako se iz latentne reprezentacije niske dimenzije može dobiti slika visoke rezolucije. Ovo se radi pomoću takozvane *transponovane konvolucije* (eng. *transposed convolution*), koja se u literaturi neretko pogrešno naziva dekonvolucijom (što je prosti drugi pojam). Za razliku od standardne konvolucije pri kojoj se podmatrici slike pridružuju skalarni proizvod te podmatrice sa filterom, čime se gubi na dimenziji slike, transponovana konvolucija počiva na množenju filtera vrednošću slike i dodavanju tako dobijene matrice okolnim pikselima nove slike (koja je na početku inicijalizovana na nule). Ovo je ilustrovano na slici 15.6. Kada se filter centririra iznad polja sa crnom pozadinom, daje doprinose u izlaznoj slici svim poljima koja se nalaze oko polja koje u izlaznoj



Slika 15.6: Ilustracija transponovane konvolucije.

slici odgovara polju sa crnom pozadinom. Očigledno, polje koje je u gornjem levom uglu izlazne slike, može dobiti samo jedan doprinos (kada je filter na polju sa crnom pozadinom) i zato ima vrednost 1. Da je u gornjem levom polju filtera recimo broj 2, onda bi i gornje levo polje izlazne slike imalo vrednost 2. Polja koja imaju vrednost 9 u izlaznoj slici odgovaraju podmatrici dimenzija 2×2 u centru ulazne slike. Očigledno, ona dobijaju doprinos sa 9 polja (odgovarajuće polje u ulaznoj matrici i 8 koja ga okružuju). Ako razmišljamo odakle dolaze doprinosi za neko izlazno polje, vidimo da se njegova vrednost može zapisati kao skalarni proizvod odgovarajućih polja u ulaznoj slici sa nekim koeficijentima, što znači da i ova operacija ima formu konvolucije! Ispostavlja se da postoji matrični zapis slike i filtera (koji nismo diskutovali i nećemo ga diskutovati) u kojem se konvolucija vrši standardnim matričnim množenjem, a razlika između konvolucije i transponovane konvolucije se očekivano sastoji u transponovanju jedne od matrica. Otud i naziv.

Glava 16

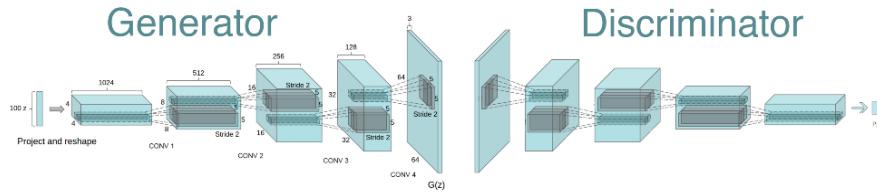
Generativni modeli

Kao što je već rečeno, generativni modeli su modeli koji modeluju zajedničku raspodelu promenljivih $p(x)$. Modelovanje zajedničke raspodele tipično zahteva veću količinu podataka, podložnije je prokletstvu dimenzionalnosti i neretko nosi i druge izazove kao što će se videti u nastavku. S druge strane, prednosti su jasne – zajednička raspodela nosi potpunu informaciju o podacima (naravno, ako je dobro modelovana) i omogućava predviđanje nekih promenljivih na osnovu drugih, daje intervale pouzdanosti i omogućava generisanje podataka.

U slučaju podataka kao što su slike, problem dimenzionalnosti je drastično izražen. Metode koje počivaju na modelovanju gustine raspodele, a potom na uzorkovanju iz nje posebnim metodama, nisu se pokazale dobro. Takve metode su i računski zahtevne u vreme generisanja. Stoga, moderni generativni modeli poput generativnih suparničkih mreža i varijacionih autoenkodera, funkcionišu na drugom principu. Oni direktno modeluju preslikavanje iz nekog niskodimenzionalnog prostora u kojem se vrši uzorkovanje jednostavnim metodama (na primer iz normalne ili uniformne raspodele) u visokodimenzionalni prostor podataka, tako da vrednosti funkcije imaju željenu zajedničku raspodelu ako se argumenti uzimaju iz prepostavljene jednostavne raspodele. Odnonsno, uči se funkcija koja preslikava datu jednostavnu niskodimenzionalnu raspodelu u složeniju visokodimenzionalnu raspodelu. Ovo drastično popravlja računske performanse u vreme generisanja, ali može značajno otežati obučavanje i matematički zakomplikovati modele i algoritme. U nastavku ćemo se upoznati sa jednim, trenutno najpopularnijim pristupom generativnom modelovanju visokodimenzionalnih podataka.

16.1 Generativne suparničke mreže

Generativne suparničke mreže (eng. *generative adversarial networks*), ili skraćeno GAN-ovi, zasnovaju se na zanimljivoj i originalnoj ideji obučavanja. Ta ideja je posebno problematična za pouzdano i stabilno sprovođenje obučavanja,



Slika 16.1: Arhitektura DCGAN-a.

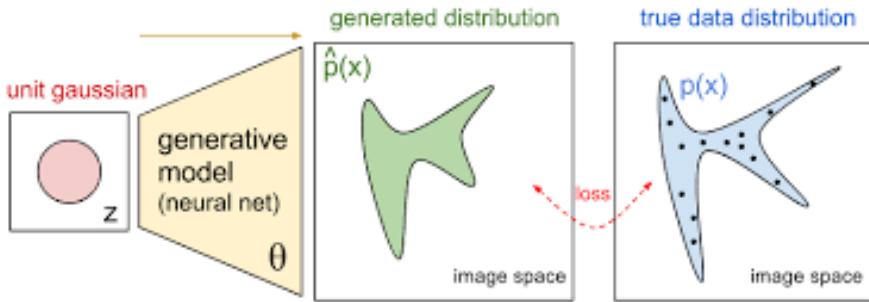
ali daje impresivne rezultate kada obučavanje uspe. Postoji veliki broj trikova i modifikacija, ali mi ćemo se zadržati samo na osnovnoj varijanti, kako bismo opisali ključni princip.

Osnovna ideja suparničkog pristupa obučavanju generativnih modela je da se koriste dva modela – *generator*, koji generiše podatke (u nastavkućemo govoriti slike, iako ovakvi modeli nisu ograničeni samo na slike) i *diskriminatator* koji klasificiše slike u prave i generisane. Diskriminatator se obučava tako da što bolje razlikuje generisane slike od pravih, a generator se obučava da što bolje generiše slike koje diskriminatator neće razlikovati od pravih. Očito, kako se generator menja, mora i diskriminatator da se menja i obrnuto, tako da se ova dva modela obučavaju naizmenično. Kao analogija ovom vidu obučavanja često se navodi nadmetanje falsifikatora i policije, pri čemu obe strane postaju bolje u tome što rade.

Teorijski okvir podrazumeva samo to da su diskriminatator $D(x)$ i generator $G(z)$ diferencijabilne funkcije, ali su to u praksi uvek neuronske mreže. Generator transformiše neki latentni prostor odabrane dimenzije u sliku određene rezolucije, recimo koristeći transponovane konvolucije, dok diskriminatator predstavlja standardnu konvolutivnu mrežu. Ovakva arhitektura je ilustrovana slikom 16.1. Generator implicitno definiše raspodelu podataka. Kao što je prikazano na slici 16.2, cilj je da raspodela koju definiše generator što bolje odgovara pravoj.

Vrednost z iz latentnog prostora se bira iz normalne raspodele $\mathcal{N}(0, \sigma^2 I)$ određene dimenzije (na primer 100) i preslikava generatorom u prostor slika. Dimenzije latentnog prostora mogu imati i intuitivan smisao, koji nije zadat unapred, već naučen, mada to nije garantovano, baš kao i kod autoenkodera. Ipak, primetimo da je za razliku od standardnih autoenkodera latentni prostor po definiciji kompaktan. Ilustracija veze između latentnog prostora i prostora slika ilustrovana je na slici 16.3. U datom primeru, variranjem po jednoj dimenziji menja se izraz lica, a po drugoj orientacija. Slučajnim izborom koordinata u ovom prostoru, nasumice se bira lice.

Funkcija greške može se formulisati iz perspektive teorije igara, konkretno igara sa nultom sumom, u kojoj jedan igrač pokušava da poveća svoju dobit, a drugi da smanji maksimalnu dobit koju prvi igrač može da ostvari. Umesto dobiti, kao što smo navikli, pričamo o gubitku, odnosno grešci. Kada formulišemo



Slika 16.2: Ilustracija aproksimacije raspodele pomoću generativne suparničke mreže.

funkciju koja meri grešku diskriminatora, generator se bira tako da maksimišu je grešku najboljeg diskriminatora – onog sa minimalnom greškom za dati generator. Za dati generator G i diskriminator D ima smisla formulisati grešku diskriminatora kao

$$\mathbb{E}_x[-\log D(x)] + \mathbb{E}_z[-\log(1 - D(G(z)))]$$

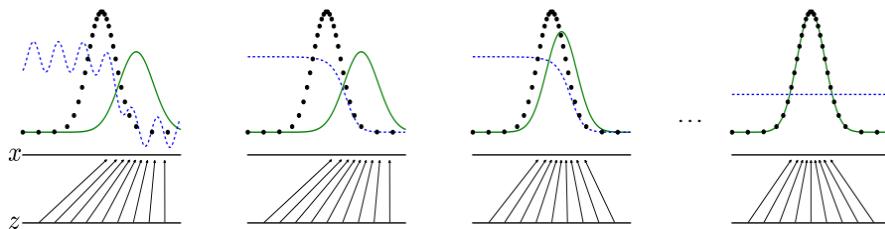
pri čemu je prvo očekivanje po raspodeli stvarnih podataka $p(x)$, a drugo po raspodeli u latentnom prostoru. Smisao prvog dela je da diskriminator mora davati vrednost blisku 1 pravim slikama. Zaista, ako je $D(x)$ blisko 1, očekivanje je blisko nuli, odnosno greška je mala. Kako se $D(x)$ približava 0, greška teži pozitivnoj beskonačnosti. Smisao drugog dela je da diskriminator slikama koje je generisao generator daje vrednost blisku 0. Zaista, ako je $D(G(z))$ vrednost bliska 0, $1 - D(G(z))$ je vrednost bliska 1 i greška je mala. U suprotnom, ako je $D(G(z))$ vrednost bliska 1, greška je velika. Odnosno, vidimo da ova greška dobro izražava kvalitete diskriminatora. Imajući to u vidu, kao što smo rekli, problem koji treba rešiti je

$$\max_{w_G} \min_{w_D} \mathbb{E}_x[-\log D_{w_D}(x)] + \mathbb{E}_z[-\log(1 - D_{w_D}(G_{w_G}(z)))]$$

Pri rešavanju ovog problema, očekivanja se naravno aproksimiraju prosećima, a samo obučavanje se tipično sprovodi stohastički, kao i inače, ali smerujući minimizaciju po w_D i maksimizaciju po w_G . Promene diskriminatora i generatora prilikom ovog procesa ilustrovane su slikom 16.4. Primetimo da je skicirana raspodela podataka (tačkastom linijom) i raspodela koja se dobija preslikavanjem raspodele latentnog prostora generatorom u ciljni prostor. Na početku, diskriminator ne pridružuje stabilno visoke vrednosti pravim podacima i niske generisanim, a raspodele stvarnih i generisanih podataka se razlikuju. Nakon obučavanja diskriminatora, on može dobro razlikovati prave od generisanih slika, kao što pokazuje druga slika. Treća slika pokazuje stanje nakon što



Slika 16.3: Ilustracija veze između latentnog prostora i prostora slika. Po horizontalnoj dimenziji se menja orijentacija, a po vertikalnoj izraz lica.

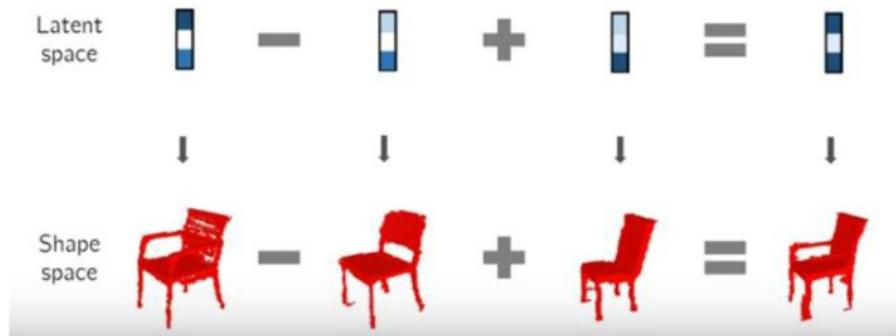


Slika 16.4: Ilustracija promena generatora i diskriminatara u toku obučavanja.

se u odnosu na taj korak poprave i generator i diskriminat, a poslednja raspodelu idealnog generatora i nemoć diskriminatora da u tom slučaju različiti između dve raspodele.

Za broj koraka minimizacije koji se vrši pre prelaska na maksimizaciju i obrnuto postoje smernice, ali one ne predstavljaju garanciju efikasnog, pa ni uspešnog obučavanja. Ključni problem je da je ovaj proces optimizacije prilično nestabilan. Moguće je da se od generatora koji radi relativno dobro, daljom optimizacijom dođe do generatora koji radi vrlo loše. Lako se dešava da proces optimizacije ne napreduje usled nestajućih gradijenata. Ovo se tipično dešava na početku obučavanja. Tada je lako obučiti diskriminator da vrlo dobro razlikuje generisane od pravih slika. On tada daje vrednosti bliske 0 i 1, odnosno nalazi se u režimu u kojem je sigmoidna funkcija praktično ravna, a samim tim joj je izvod blizak 0, što dovodi do nestajućih gradijenata. Ovo jasno govori da nije dobra praksa obučavati diskriminator do konvergencije, pa preći na generator i obrnuto, već se mora alternirati nakon malog broja koraka optimizacije. Postoje i drugačije definicije funkcije greške koje ublažavaju ovaj problem.

Primećeno je da generativne suparničke mreže omogućavaju aritmetiku u latentnom prostoru. Ovo je ilustrovano slikom 16.5. Ukoliko su poznate latentne reprezentacije stolice sa naslonom za ruke i slične stolice bez naslona za ruke, dodavanjem njihove razlike na reprezentaciju drugačije stolice bez naslona, može se dobiti reprezentacija takve stolice sa naslonom za ruke, a onda i njena slika upotrebom generatora. Ovo svojstvo nije karakteristično isključivo za generativne suparničke modele, već se javlja često u metodima koji koriste latentne prostore ili uče reprezentaciju podataka pomoću neuronskih mreža.



Slika 16.5: Ilustracija aritmetike u latentnom prostoru.

Deo IV

Dodatak

Glava 17

Matematičko predznanje

Mašinsko učenje je u velikoj meri matematička disciplina. Kako po prirodi teorija koje se bave proučavanjem generalizacije, tako i po sveprisutnosti matematičkog aparata u metodama i primenama, čak i kada su one prvenstveno inženjerske prirode. Oblasti matematike koje su od najvećeg značaja za mašinsko učenje su linearna algebra, analitička geometrija, matematička analiza, verovatnoća i statistika. Pored njih, primenu nalaze i funkcionalna analiza, topologija, hiperbolička geometrija i druge. U nastavku je dat pregled matematičkih tema koje su najznačajnije za mašinsko učenje. Poznavanje nekih osnovnih pojmoveva, poput matrične algebre, determinantni, limesa, izvoda i drugih se podrazumeva. Neke od definicija koje slede mogu biti i opštije, ali je izlaganje namerno prilagođeno potrebama razumevanja mašinskog učenja.

17.1 Sopstvene vrednosti i sopstveni vektori

Kvadratna matrica $A \in \mathbb{R}^{n \times n}$ ima *sopstveni vektor* $x \in \mathbb{R}^n$ i odgovarajuću *sopstvenu vrednost* $\lambda \in \mathbb{R}$, ukoliko važi

$$Ax = \lambda x \quad x \neq 0$$

Drugim rečima, matrica A ne menja pravac sopstvenog vektora (iako mu možda menja smer). Poznato je da različitim sopstvenim vrednostima odgovaraju linearno nezavisni sopstveni vektori, kao i da su sopstvene vrednosti jednake nulama karakterističnog polinoma

$$\det(A - \lambda I)$$

Svakoj sopstvenoj vrednosti λ_i , $i = 1, \dots, m$ odgovara sopstveni potprostor V_i . Ukoliko se dimenzije ovih sopstvenih potprostora sabiraju na dimenziju matrice A , matrica se može svesti na dijagonalnu. Ako je D dijagonalna matrica sopstvenih vrednosti, u kojoj je svaka sopstena vrednost λ_i ponovljena $\dim(V_i)$ puta i ako je X matrica čije su kolone odgovarajući sopstveni vektori, važi

$$AX = XD$$

Onda važi i

$$A = XDX^{-1} \quad D = X^{-1}AX$$

Drugim rečima, linearno preslikavanje indukovano matricom A se može opisati faktorima izduživanja (sopstvenim vrednostima) duž određenih pravaca (sopstvenih vektora). Ukoliko je neka od sopstvenih vrednosti negativna, radi se o promeni smera. Ukoliko pomenuuti uslov vezan za dimenzije prostora V_i nije ispunjen, matrica X neće biti invertibilna.

U slučaju relanih simetričnih matrica, uvek je moguće konstruisati odgovarajuću dijagonalnu matricu. Dodatno, tada važi $X^{-1} = X^T$, odnosno $XX^T = X^TX = I$. Drugim rečima, matrica sopstvenih vektora je ortogonalna.

17.2 Definitnost

Kvadratna matrica $A \in \mathbb{R}^{n \times n}$ je *pozitivno semi definitna* ukoliko za svaki vektor $x \in \mathbb{R}^n$ važi

$$x^T Ax \geq 0$$

Analogno se definiše *negativna semi definitnost*. Kvadratna matrica A je *strog pozitivno definitna* ili krace *pozitivno definitna* ukoliko važi

$$x^T Ax > 0$$

Analogno se definiše *negativna definitnost*. Pojam pozitivne definitnosti predstavlja uopštenje pojma pozitivnosti brojeva na matrice. Slično je sa ostalim pojmovima.

Prema Silvesterovom kriterijumu, matrica A je pozitivno definitna ako i samo ako su determinante svih kvadratnih podmatrica koje uključuju element a_{11} pozitivne. Kriterijumi za pozitivnu semidefinitnost i za negativnu definitnost su nešto komplikovani i ne diskutujemo ih.

Ukoliko je simetrična matrica A dijagonalno dominantna, odnosno važi

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}| \quad \text{za svako } i$$

i ako su dijagonalni elementi nenegativni, matrica A je pozitivno semidefinitna. Obrnuto ne mora da važi.

Kvadratna simetrična matrica A je pozitivno definitna ako i samo ako ima pozitivne sopstvene vrednosti. Ako je X matrica čije su kolone sopstveni vektori matrice A , znamo da važi $A = XDX^{-1}$. Zbog simetričnosti matrice, sopstveni vektori su ortogonalni, odnosno važi $X^{-1} = X^T$, što znači da važi $x^T Ax = x^T XDX^T x = (X^T x)D(X^T x)$, gde je matrica D dijagonalna. Transformacija $X^T x = y$ predstavlja promenu koordinata i nadalje pišemo $y^T Dy$. Ovaj izraz je pozitivan ako važi

$$\sum_{i=1}^n \lambda_i y_i^2 > 0$$

što je tačno za svako y ako i samo ako su sve sopstvene vrednosti λ_i duž dijagonale matrice D pozitivne. Time smo se uverili u tvrdnju sa početka paragrafa.

Kvadratna matrica A je pozitivno definitna ako i samo ako postoji donje-trougaona matrica L , takva da važi $A = LL^T$. Ovakva faktorizacija matrice A naziva se *Čoleski dekompozicijom*. Pokušaj Čoleski dekompozicije predstavlja računski efikasan način da se proveri pozitivna definitnost matrice.

17.3 Norma i skalarni proizvod

Neka je $X \subseteq \mathbb{R}^n$. *Norma* je funkcija $\|\cdot\| : X \rightarrow \mathbb{R}$ takva da za svako $\alpha \in \mathbb{R}$ i $x, y \in X$ važi:

- $\|\alpha x\| = |\alpha| \|x\|$
- $\|x + y\| \leq \|x\| + \|y\|$
- Ako važi $\|x\| = 0$, onda važi $x = 0$.

Intuitivno, norma vektora se povezuje sa njegovim intenzitetom, odnosno dužinom. Ipak, norme se mogu definisati na različite načine.

Takozvane p norme nad vektorima iz \mathbb{R}^n se definišu na sledeći način:

$$\|x\|_p = \sqrt[p]{\sum_{i=1}^n x_i^p}$$

Za svake dve p norme $\|\cdot\|_a$ i $\|\cdot\|_b$ postoje konstante $0 < c_1 \leq c_2$, takve da za svako $x \in X$ važi

$$c_1 \|x\|_b \leq \|x\|_a \leq c_2 \|x\|_b$$

Ovo svojstvo se naziva *ekvivalentnošću* p normi. Jedna od implikacija je da konvergencija u jednoj p normi znači konvergenciju u bilo kojoj drugoj p normi.

Norme se mogu lako definisati i nad matricama. U slučaju p normi, definicija se oslanja na p norme nad vektorima:

$$\|A\|_p = \max_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}$$

Još jedna često korišćena matrična norma, ekvivalentna matričnim p normama je Frobenijusova norma:

$$\|A\|_F^2 = \sum_{i=1}^m \sum_{j=1}^n a_{ij}^2$$

Ukoliko za neki vektor x važi $\|x\| = 1$ kažemo da je taj vektor *normiran*. Postupak *normiranja* se izvodi deljenjem vektora njegovom normom. Očito, vektor može biti normiran u odnosu na jednu normu, a da nije normiran u odnosu na drugu.

Neka je $X \subseteq \mathbb{R}^n$. *Skalarni proizvod* je funkcija $\|\cdot\| : X \times X \rightarrow \mathbb{R}$ takva da za svako $\alpha \in \mathbb{R}$ i $x, y, z \in X$ važi:

- $x \cdot y = y \cdot x$
- $x \cdot (y + z) = x \cdot y + x \cdot z$
- $x \cdot (\alpha y) = \alpha(x \cdot y)$
- $x \cdot x \geq 0$
- $x \cdot x = 0 \Leftrightarrow x = \mathbf{0}$

Skalarni proizvod prirodno indukuje normu $x \cdot x = \|x\|$. Takođe, postoji veza između skalarnog proizvoda i ugla između vektora. Naime, važi $x \cdot y = \|x\|\|y\| \cos \angle(x, y)$, odnosno, u slučaju normiranih vektora, skalarni proizvod je kosinus ugla između dva vektora. Ukoliko je kosinus između dva vektora 1, ti vektori su kolinearni. U slučaju normiranih, oni su jednaki. Ukoliko je kosinus 0, vektori su ortogonalni, a ukoliko je -1 , vektori su suprotnih smerova. Otud, skalarni proizvod se može uzeti za meru sličnosti vektora sa rasponom $[-1, 1]$ i u tom svojstvu se često koristi u mašinskom učenju.

17.4 Izvod, parcijalni izvod i gradijent

Za funkciju $f : \mathbb{R} \rightarrow \mathbb{R}$ jedne promenljive, *izvod* $f' : \mathbb{R} \rightarrow \mathbb{R}$ se definiše na sledeći način:

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}$$

Očito, izvod predstavlja odnos priraštaja funkcije na granicama nekog intervala i dužine tog intervala kada dužina tog intervala teži nuli. Intuitivno, izvod funkcije u nekoj tački predstavlja nagib funkcije u toj tački ili formalnije koeficijent pravca njene tangente u toj tački.

Za funkciju $f : \mathbb{R}^n \rightarrow \mathbb{R}$ više promenljivih, *parcijalni izvod* $\frac{\partial f}{\partial x_i} : \mathbb{R} \rightarrow \mathbb{R}$ se definiše kao

$$\frac{\partial f(x_1, \dots, x_n)}{\partial x_i} = \lim_{h \rightarrow 0} \frac{f(x_1, \dots, x_i + h, \dots, x_n) - f(x_1, \dots, x_i, \dots, x_n)}{h}$$

Uместo notacije $\frac{\partial}{\partial x_i}$, nekad ćemo koristiti notaciju ∂_i . Parcijalni izvod kvantificuje nagib duž samo jednog koordinatnog pravca.

Izvodi i parcijalni izvodi se umesto izračunavanjem limesa, obično izračunavaju prema poznatim pravilima za izračunavanje izvoda nekih poznatih funkcija i pravila izvoda složene funkcije. U svom najjednostavnijem obliku, ovo pravilo glasi

$$\partial_i(f \circ g) = (\partial_i f \circ g)\partial_i g$$

Ukoliko je funkcija g vektorska, odnosno važi $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$, onda pravilo glasi

$$\partial_i(f \circ g) = \sum_{j=1}^m (\partial_j f \circ g)\partial_i g_j$$

gde g_j označava j -tu koordinatu (vektorske) vrednosti funkcije g .

Vektor svih parcijalnih izvoda funkcije naziva se *gradijentom funkcije* i označava ∇f .

Pored prvih parcijalnih izvoda, od značaja su i drugi parcijalni izvodi

$$\frac{\partial^2 f(x_1, \dots, x_n)}{\partial x_i \partial x_j}$$

Umesto $\frac{\partial^2}{\partial x_i \partial x_j}$ nekada ćemo skraćeno pisati ∂_{ij} . Matricu drugih parcijalnih izvoda funkcije ćemo nazivati *hesijanom funkcije* i označavati $\nabla^2 f$.

Primer 15 Neki primer izračunavanja.

17.5 Konveksnost

Neka je X konveksan skup i neka je realna funkcija f definisana na njemu. Tada za funkciju f kažemo da je *konveksna* ukoliko za svako $\alpha \in [0, 1]$ i svako $x_1, x_2 \in X$ važi

$$f(\alpha x_1 + (1 - \alpha)x_2) \leq \alpha f(x_1) + (1 - \alpha)f(x_2)$$

Ukoliko funkcija ispunjava dati uslov sa strogom nejednakosću, funkcija je *strogo konveksna*. Funkcija f je (*strogo*) *konkavna* ukoliko je funkcija $-f$ (strogo) konveksna. Intuitivno, funkcija je konveksna, ako je svaka njena sečica u svakoj tački između tačaka preseka iznad grafika funkcije. Sva naredna razmatranja su data za konveksne funkcije. Obično važe analogna svojstva za strogo konveksne i za konkavne funkcije.

Ukoliko je funkcija f diferencijabilna u tački x , onda je konveksna u tački x ako i samo ako postoji okolina tačke x takva da za svako y iz te okoline važi

$$f(y) \geq f(x) + \nabla f(x)^T(y - x)$$

ili u slučaju jedne promenljive

$$f(y) \geq f(x) + f'(x)(y - x)$$

Drugim rečima, funkcija je konveksna ukoliko je njen grafik iznad tangentne ravni.

Dva puta diferencijabilna funkcija f je strogo konveksna u nekoj tački ukoliko je hesijan $\nabla^2 f(x)$ pozitivno definitan u nekoj okolini te tačke. Hesijan funkcije koja je konveksna u nekoj tački u nekoj okolini te tačke ima pozitivne sopstvene vrednosti.

Konveksne funkcije imaju naredna svojstva koja često mogu olakšati prepoznavanje da je neka funkcija konveksna:

- Ako su f_1, \dots, f_m konveksne funkcije i važi $w_1 \geq 0, \dots, w_m \geq 0$, onda je i funkcija

$$w_1 f_1(x) + \dots + w_m f_m(x)$$

konveksna funkcija.

- Ako je f konveksna funkcija, A matrica i b vektor odgovarajućih dimenzija, onda je i $f(Ax + b)$ konveksna funkcija.
- Ako su f_1, \dots, f_m konveksne funkcije, onda je i funkcija

$$\max\{f_1(x), \dots, f_m(x)\}$$

konveksna funkcija. Isto važi i za supremum nad beskonačnim skupom konveksnih funkcija.

- Kompozicija $f \circ g$ je konveksna ako je funkcija f konveksna i neopadajuća po svim argumentima, a funkcija g konveksna ili ako je funkcija f konveksna i nerastuća po svim argumentima, a g konkavna.

Funkcija diferencijabilna u tački x se naziva *jako konveksnom* ukoliko za svako y iz neke okoline tačke x važi

$$f(y) \geq f(x) + \nabla f(x)^T(y - x) + \frac{m}{2} \|y - x\|^2$$

za neko $m > 0$. Jaka konveksnost označava stroži uslov od konveksnosti. Ne treba je mešati sa strogom konveksnošću. Lako se pokazuje da je dva puta diferencijabilna funkcija jako konveksna u tački x ukoliko je matrica

$$\nabla^2 f(x) - mI$$

pozitivno semidefinitna za neko $m > 0$.

17.6 Lokalni optimumi

Funkcija u nekoj tački ima (*strogji*) *lokalni minimum* ukoliko u nekoj okolini te tačke uzmišta (strogo) veće vrednosti nego u toj tački. Analogno se definiše (*strogji*) *lokalni maksimum* funkcije. Lokalni minimumi i lokalni maksimumi se skupa nazivaju *lokalnim optimumima* funkcije. U nastavku će biti reči o lokalnim minimumima, ali analogni zaključci važe i za stroge lokalne minimume i za lokalne maksimume.

Ako je x optimum funkcije i ako je ona u njemu diferencijabilna, onda važi $\nabla f(x) = 0$. Da bi funkcija imala minimum u tački x mora da važi $\nabla^2 f(x)$ mora biti pozitivno semidefinitan, odnosno sopstvene vrednosti hesijana moraju biti nenegativne. Ukoliko bi postojala negativna sopstvena vrednost, pravac odgovarajućeg sopstvenog vektora bi bio pravac u kojem vrednost funkcije opada u odnosu na tačku x , pa ona ne bi bila optimum.

17.7 Integral

U ovom odeljku integral nećemo formalno definisati, već će akcenat biti na njegovom smislu. Od ključnog značaja je intuicija uopštenja sume. Sume se formulišu oslanjajući se na diskretne promenljive kao indekse. Ovakva operacija se često vrši nad nizovima – funkcijama diskretne promenljive:

$$\sum_{i=1}^n a_i$$

Ipak, nekad se prirodno nameće potreba sumiranja vrednosti funkcije po kontinualnoj promenljivoj. Ovaj posao se vrši integracijom, koja se u jednodimenzionom slučaju može razumeti kao suma označenih površina pravougaonika između grafika funkcije i koordinatne x ose, kada dužine stranica pravougaonika koje leže na x osi teže nuli.

Pored sumiranja, integral često služi težinskom uprosečavanju funkcija. Neka je potrebno izračunati prosečnu vrednost rude u nekom rudnom ležištu. Ukoliko bi se ležište moglo podeliti na n homogenih celina od kojih svaka ima vrednost v_i po toni i obuhvata količinu rude q_i izraženu u tonama, prirodno bi bilo uprosečiti vrednosti v_i , ali pridajući veću težinu onoj vrednosti koja se javlja u većoj količini. Takva prosečna vrednost data je izrazom:

$$\sum_{i=1}^n \frac{q_i}{\sum_{j=1}^n q_j} v_i$$

Težina koja se pridaje svakoj vrednosti je jednaka udelu količine takve rude u ležištu. U slučaju da ne postoji homogene celine, već kvalitet i gustina željenog materijala približno neprekidno variraju od tačke do tačke ležišta, prirodno je zamjeniti sumu integralom u kojem su diskretne vrednosti i količine zamjenjene funkcijama $v(x)$ i $q(x)$ koje zavise od lokacije:

$$\int_{\mathcal{D}} v(x)q(x)dx$$

pri čemu je \mathcal{D} oblast koju zauzima ležište. Kako bi izvedena zamena sume integralom imala smisla, potrebno je da funkcija $q(x)$ zadovoljava ključno svojstvo koje zadovoljavaju težine pridružene vrednostima u sumiranju, a to je da se sabiraju na 1:

$$\sum_{i=1}^n \frac{q_i}{\sum_{j=1}^n q_j} = 1$$

Analogno tome, mora važiti

$$\int_{\mathcal{D}} q(x)dx = 1$$

Navedena interpretacija integrala kao uprosečavanja je sveprisutna u mašinskom učenju, pre svega zbog oslanjanja na verovatnoću i pojам matematičkog očekivanja.

17.8 Verovatnoća

Verovatnoća se bavi kvantifikovanjem izglednosti različitih događaja. Postoje dve ključne interpretacije pojma verovatnoće u okviru dve različite škole statistike. *Frekventistička statistika* interpretira verovatnoću nekog događaja kao dugoročnu frekvenciju opažanja tog događaja. Stoga, verovatnoća pisma ili glave pri bacanju novčića jednaka je po 0.5 zato što se u ponovljenim eksperimentima primećuje da se udeo eksperimenata u kojima je ishod pismo bliži 0.5 kako broj ponavljanja eksperimenta raste. S druge strane, *bajesovska statistika* interpretira verovatnoću kao nivo uverenja. Pošto novčić ima dve strane i pošto ne vidimo nikav razlog da jedna strana pada češće nego druga, podjednako očekujemo glavu koliko i pismo. Otud verovatnoća 0.5 za svaki od ishoda. Iako uverenja mogu biti proizvoljna, jednom kada su uverenja za elementarne ishode eksperimenta definisana, verovatnoće komplikovanih događaja se računaju u skladu sa pravilima verovatnoće koja garantuju saglasnost izračunavanja. Konkretno, ako je verovatnoća padanja glave 0.5, verovatnoća padanja tri glave mora biti 0.125. Pravila računanja verovatnoće složenijih događaja (koji se definišu nad elementarnim ishodima) definisana su aksiomama verovatnoće i važe u obe interpretacije. Zapravo, ova pravila definišu (ali ne jedinstveno) pojam *verovatnosne mere*, koju označavamo sa p .

Pored verovatnoće događaja, definiše se i *uslovna verovatnoća* događaja kao

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

Ukoliko događaj Y predstavlja parne ishode bacanja kockice, a događaj X ishode 4, 5 i 6, ukoliko znamo da je bacanjem kockice dobijen paran broj, verovatnoća da je to neki od brojeva 4, 5 i 6 mora biti $2/3$ pošto to ne može biti 5, a 4 i 6 su dva ishoda od mogućih 2, 4 i 6. Presek događaja $X \cap Y$ označavaćemo i kao XY . Koristeći definiciju uslovne verovatnoće dva puta, izvodimo sledeću jednakost:

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

poznatu pod nazivom *Bajesova formula*, a koja povezuje uslovne verovatnoće $P(X|Y)$ i $P(Y|X)$ i koja se često koristi u mašinskom učenju.

Važan koncept je koncept nezavisnosti događaja. Intuitivno, dva događaja su nezavisna ukoliko informacija o događanju jednog ne daje nikakvu informaciju od događanju drugog, ili formalno:

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

Na osnovu definicije uslovne verovatnoće, ova relacija se može predstaviti i drugačije

$$P(XY) = P(X)P(Y)$$

Nezavisnost može biti i uslovna:

$$P(XY|Z) = P(X|Z)P(Y|Z)$$

U tom slučaju informacija o događaju X nam ne govori ništa o događaju Y ako imamo informaciju da se desio događaj Z . Primera radi, veličina vokabulara kojim vlada jedna osoba nam ne govori ništa o njenoj visini ako su nam poznate njene godine. U suprotnom, bogat vokabular sugerije veću visinu, pošto su deca i niža i imaju manji vokabular od odraslih.

Važne pojmove gustine raspodele i kumulativne funkcije raspodele uvešćemo naopakim redom – onim koji odgovara značaju tih pojmovima u mašinskom učenju i verovatno primenjenoj matematici uopšte, iako ne i opštosti pojmovima. Nenegativna funkcija $p : \mathbb{R}^n \rightarrow \mathbb{R}$ za koju važi

$$\int_{\mathbb{R}^n} p(x)dx$$

naziva se *gustina raspodele*. Intuitivno, ova funkcija verovatnjim događajima pridružuje više vrednosti, a manje verovatnim niže. Međutim upravo navedena formulacija je problematična jer strogo gledano, svaka pojedinačna tačka je mera nula, pa ima i verovatnoću nula. Ipak, ova intuicija važi na proizvoljno malim intervalima, što dozvoljava da o gustini raspodele razmišljamo na navedeni način. Odgovarajuća *kumulativna funkcija raspodele*¹ je:

$$F(x_1, \dots, x_n) = \int_{-\infty}^{x_1} \dots \int_{-\infty}^{x_n} p(x_1, \dots, x_n) dx_1 \dots dx_n$$

Ona u mašinskom učenju ne igra važnu ulogu, ali daje vezu sa pojmom verovatnosne mere. Naime, može se pokazati da za datu kumulativnu funkciju raspodele F postoji tačno jedna verovatnosna mera P za koju važi

$$F(x_1, \dots, x_n) = P((-\infty, x_1] \times \dots \times (-\infty, x_n])$$

Moguće je definisati i diskretnu funkciju raspodele nad nizom realnih brojeva, ali se u mašinskom učenju prvenstveno oslanjam na pojam gustine raspodele nad neprekidnim domenom.

Važan pojam vezan za raspodelu promenljive je pojam *marginalne raspodele*. Ukoliko je poznata zajednička raspodela slučajnih vektora X i Y , marginalna raspodela vektora X je raspodela vektora X dobijena sumiranjem ili integracijom (zavisnosti od toga da li je diskretni ili neprekidni slučaj) po Y . Važi:

$$p(x) = \int_y p(x, y)dy$$

Ova formula ima intuitivnu interpretaciju. Ako nas zanima koliko često se opaža neka vrednost vektora X i znamo koliko se često ona javlja u kombinacijama sa

¹Pojam kumulativne funkcije raspodele je opštiji od ove definicije. Postoje kumulativne funkcije raspodele za koje ne postoji odgovarajuća gustina raspodele.

svim različitim vrednostima vektora Y , intuitivno je da učestalost svih pojavljivanja te vrednosti vektora X dobijamo sumirajući učestalosti tih kombinacija.

Slučajna promenljiva je funkcija koja preslikava elementarne ishode u \mathbb{R} . Na primer visina slučajno odabrane osobe iz neke populacije je slučajna promenljiva. Njena svojstva su opisana pridruženom raspodelom verovatnoće njenih vrednosti. Verovatnoća da neprekidna slučajna promenljiva uzme vrednost iz nekog intervala je verovatnoća ishoda za koje slučajna promenljiva daje vrednosti iz tog intervala. Na osnovu ove veze sa pojmom verovatnosne mere nad nekim prostorom dogadaja, pojmovi gustine raspodele i kumulativne funkcije raspodele se prirodno definišu za slučajne promenljive. Intuitivniji način razumevanja gustine raspodele slučajne promenljive, bio bi sledeći. Neka je dat uzorak vrednosti slučajne promenljive, recimo uzorak visna različitih ljudi. Sve visine imaju svoje mesto na realnoj pravoj koju možemo izdeliti na podintervale jednakе širine. Lažne slikovitosti radi, zovimo te intervale korpicama (eng. bin). U svaku korpicu upada određeni broj visina. Ukoliko za svaku korpicu nacrtamo pravougaonik visine proporcionalne broju (gle!) visina u njoj, dobijamo *histogram* visina. Smanjujući širinu korpice, a povećavajući broj izmerenih ljudi, histogram postaje sve uglačaniji i teži gustini raspodele. Pažljiv čitalac bi trebalo da je primetio da je primer zasnovan na visinama problematičan jer ljudi nema neprebrojivo mnogo, dok je gustina raspodele definisana nad neprebrojivim domenom. Takav čitalac je slobodan da razmišlja o temperaturama, osim ako se dovoljno razume u fiziku da se zapita o kvantima energije i počne da sumnja u to da matematički pojam neprekidnosti može da se savršeno primeni bilo gde u stvarnom svetu.

Pomenuti pojam histograma je važan za upotrebu mačinskog učenja u praktici kako bi se sumarno vizualizovale vrednosti različitih promenljivih. Ključni problem u njegovoj upotrebi je izbor širine korpica. Ukoliko su korpice vrlo široke, dobija se vrlo gruba slika raspodele vrednosti promenljive. Ukoliko su korpice vrlo uske, ne očekuje se da se u bilo kojoj nađe veliki broj tačaka, možda nijedna. Takav histogram ne daje nikakvu korisnu informaciju, jer se iz njega obično ne može razaznati nikakav karakterističan oblik, a i to što se vidi može značajno zavisiti od konkretnog (slučajno generisanog) uzorka. Stoga histogram uvek treba pogledati za veći broj različitih širina korpica.

17.9 Sredina i rasipanje slučajne promenljive

Slučajne promenljive su opisane svojom raspodelom. Ipak, raspodele je često korisno sumirati nekim jednostavnim veličinama, kako bi se bolje razumele. Neke od osnovnih informacija o raspodelama, za koje smo obično zainteresovani, su srednja vrednost i rasipanje slučajne promenljive. Ovi intuitivni pojmovi se formalizuju pojmovima *matematičkog očekivanja* i *varijanse* slučajne promenljive.

Matematičko očekivanje je prosek vrednosti slučajne promenljive otežan njihovom učestalošću, odnosno verovatnoćom. Ako je $p(x)$ gustina raspodele

promenljive X , onda se, imajući u vidu diskusiju upotrebe integrala u svrhe uprosečavanja, matematičko očekivanje ove promenljive može definisati kao:

$$\mathbb{E}[X] = \int_{\mathcal{D}} xp(x)dx$$

gde je \mathcal{D} skup vrednosti koje uzima slučajna promenljiva X . Očekivanje se u praksi aproksimira uzoračkom sredinom:

$$\mu_n = \frac{1}{n} \sum_{i=1}^n x_i$$

Prosek nije jedini način da se definiše ideja sredine. Otud ni matematičko očekivanje nije jedini način definisanja srednje vrednosti. Jedan, često pogodniji, parametar raspodele slučajne promenljive je *medijana*, odnosno vrednost od koje slučajna promenljiva u pola slučajeve va uzima manju, a u pola veću vrednost. Formalnije, to je vrednost m , takva da važi

$$\int_{-\infty}^m xp(x)dx = \frac{1}{2}$$

Primetimo da ova definicija pretpotavlja jednodimenzionali slučaj. Postoje višedimenziona uopštenja, ali o njima neće biti reči. Medijana se u praksi aproksimira *uzoračkom medijanom*, koja se može definisati kao element na sredini sortiranog niza vrednosti uzorka ukoliko je broj elemenata u uzorku neparan ili kao aritmetička sredina dve susedne vrednosti na sredini sortiranog niza ukoliko je broj elemenata u uzorku paran. Osnovni kvalitet medijane je njena robusnost, odnosno neosetljivost na određenu količinu šuma u podacima. Razmotrimo uzorak brojeva $\{1, 3, 5, 7, 9\}$. Ukoliko dođe do pogrešnog očitavanja i 1 bude očitano kao 3, dobija se uzorak $\{3, 3, 5, 7, 9\}$. Prosek novodobijenog uzorka je drugi, ali medijana je ista kao kod prvog uzorka. Potrebno je da neki od podataka bude pogrešno očitan kao podatak sa druge strane medijane da bi se ona promenila. Takođe, medijana je u praksi često realističnija slika srednje vrednosti od proseka, ali to naravno zavisi od tačnog smisla pojma sredine koji nam je važan. Na primer, u slučaju analize bogatstva građana jedne zemlje, medijana daje dosta važniju informaciju. Prosek plata je direktno proporcionalan sumi plata (za faktor broja građana čija plata se analizira). Što je ukupno bogatstvo veće, veća je i prosečna vrednost. Ipak, prosek ne govori ništa o raspodeli bogatstva. Na primer, ukoliko 99% građana raspolaže primanjima od 10.000 dinara, dok 1% građana raspolaže primanjima od 9.010.000 dinara mesečno, prosečno primanje je 100.000 dinara, što ostavlja utisak da građani imaju dobar životni standard, što je pogrešno u 99% slučajeva. S druge strane, vrednost medijane je 10.000 dinara, što daje vernu sliku materijalnog stanja građana, osim u 1% slučajeva.

Varijansa se definiše kao prosečno odstupanje od proseka, odnosno

$$\text{var}[X] = \mathbb{E}[X - \mathbb{E}X]^2$$

i aproksimira se uzoračkom varijansom:

$$\sigma_n^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \mu_n)^2$$

Kao što prosek nije jedina formalizacija pojma sredine, tako ni varijansa nije jedina formalizacija pojma rasipanja slučajne promenljive. Ako se sa m označi medijana, onda bi jedna alternativa mogla biti

$$m[X - m[X]]$$

17.10 Statističke ocene i njihova svojstva

Funkcije uzoraka koje zadovoljavaju određene pretpostavke (koje su u mašinskom učenju tipično zadovoljene) nazivaju se *statistikama*. Prosek, uzoračka varijansa, maksimum i minimum uzorka predstavljaju primere statistika. Statistike često služe za empirijsku aproksimaciju, odnosno *ocenu*, nekih teorijskih veličina. Na primer, rastojanje između dve udaljene tačke, koje predstavlja objektivnu veličinu kojoj nemamo direktni pristup, može se oceniti prosekom većeg broja geodetskih merenja. Svako od tih merenja sadrži grešku o kojoj razmišljamo kao o slučajnoj (nekad je pozitivna, nekad negativna, nekada mala, a nekad velika), ali se njihovim uprosecavanjem greška smanjuje. I dalje, na osnovu konačnog uzorka ne možemo biti sigurni da je ocena precizna, pa čak i uopste upotrebljiva, pa zbog toga obično ulazimo u analizu svojstava takve ocene. Ocena je *konzistentna* (eng. consistent) ukoliko teži veličini koju ocenjuje kako veličina uzorka teži beskonačnosti, odnosno:

$$\lim_{n \rightarrow \infty} \hat{\theta}_n = \theta^*$$

gde je $\hat{\theta}_n$ ocena na osnovu uzorka veličine n , a θ^* prava vrednost veličine koja se ocenjuje. Ocena je *nepristrasna* ukoliko je očekivanje ocene jednako vrednosti koja se ocenjuje, odnosno:

$$\mathbb{E}[\hat{\theta}] = \theta$$

gde je \mathbb{E} očekivanje u odnosu na raspodelu podataka sa gustom $p_\theta(x)$ od kojih $\hat{\theta}$ zavisi. Eventualno odstupanje tog očekivanja od prave vrednosti

$$\mathbb{E}[\hat{\theta}] - \theta$$

naziva se *sistematskim odstupanjem*. Pristrasnost ne mora isključivati konzistentnost, pošto se sistematsko odstupanje može smanjivati sa povećanjem veličine uzorka. Među ocenama jednakog sistematskog odstupanja, ocena najmanje varijanse je *najbolja*. Intuitivno je preferirati najbolju nepristrasnu ocenu. Ipak, pokazuje se da pristrasne ocene često mogu biti korisnije jer ukupna greška neke ocene ne zavisi samo od sistematskog odstupanja, već i od varijanse, a pristrasne ocene mogu imati osetno manju varijansu od nepristrasnih.

17.11 Statistički modeli

Statistički model je definisan prostorom događaja i familijom verovatnosnih mera nad tim prostorom događaja. Ključni problem matematičke statistike je izbor jedne iz skupa kandidatnih verovatnosnih mera, koja u nekom smislu najbolje opisuje empirijski opažene podatke. Statistički modeli često prepostavljaju određena svojstva pomenutih verovatnosnih mera od kojih neka predstavljaju tehničke pogodnosti (čest je slučaj da se iz ovog razloga prepostavlja normalna raspodela podataka), a neke predstavljaju odluke prilikom modelovanja zavisnosti među različitim promenljivim koje model dovodi u vezu. Prilikom definisanja modela, često se vodi računa o tome da se model definiše tako da je u praksi moguće vršiti izbor verovatnosne mere na osnovu podataka, čak i ako neke od postavljenih prepostavki nisu realistične. Iako nepovoljne, ovakve prepostavke su potrebne, jer model koji ne bi zadovoljio ovaj zahtev ne bi bio koristan u praksi.

17.12 Metod maksimalne verodostojnosti

Kao što je rečeno, ključni problem matematičke statistike je izbor jedne iz skupa kandidatnih verovatnosnih mera, koja u nekom smislu najbolje opisuje empirijski opažene podatke. Specifikacija statističkog modela često uključuje konačan broj parametara čije vrednosti je potrebno izabrati kako bi se izabrala verovatnosna mera, odnosno kako bi se precizirao model. Postavlja se pitanje – na koji način je moguće vršiti izbor vrednosti tih parametara kako bi se dobio smislen rezultat. Jedan od ključnih principa na kojima počivaju metodi izbora vrednosti parametara statističkog modela, prisutan i u drugim oblastima statistike (npr. u testiranju statističkih hipoteza) je da *ne verujemo u neverovatne događaje*, odnosno da odbacujemo vrednosti parametara pri kojima bi opaženi podaci bili malo verovatni. Alternativno, prihvatom vrednosti parametara pri kojima bi opaženi podaci bili visoko verovatni. Naglasimo intuitivnost ovog principa. Negativan bilans na bankovnom računu koji nekada vidimo kroz sistem elektronskog bankarstva, pored ostalih mogućnosti, može biti objašnjen bilo nepažljivim trošenjem novca, bilo promenom stanja memorije bankovnog servera usled pogotka čestice kosmičkog zračenja u neku memorijsku celiju. Iako bismo preferirali da se drugo objašnjenje ispostavi tačnim, racionalan posmatrač bi se ipak opredelio za prvo, upravo zato što deluje verovatnije. Jedan pristup formalizaciji ovakovog principa je *metod maksimalne verodostojnosti* (eng. maximal likelihood). Biće formulisan za neprekidan slučaj, ali je definicija u diskretnom slučaju analogna, samo što se umesto gustine raspodele koristi diskretna funkcija raspodele. Neka je p_θ gustina raspodele određena parametrima θ statističkog modela. Neka je $X = \{x_1, \dots, x_n\}$ skup opažanja. Pod prepostavkom njihove nezavisnosti, gustina raspodele opaženih događaja

je jednaka:

$$\mathcal{L}(\theta) = \prod_{i=1}^n p_\theta(x_i)$$

Funkcija \mathcal{L} se naziva verodostojnošću parametara θ , a princip izbora vrednosti je

$$\theta^* = \max_{\theta} \mathcal{L}(\theta)$$

U praksi je pogodnije baratati sumama nego proizvodima, kako zbog analitički pogodnjeg diferenciranja, tako i zbog izbegavanja prekoračenja (u slučaju množenja brojeva velikih po apsolutnoj vrednosti) i potkoračenja (u slučaju množna brojeva malih po apsolutnoj vrednosti). Takođe, konvencija je da se problemi optimizacije češće predstavljaju kao problemi nalaženja minimuma, nego maksimuma. Stoga se prethodni problem često zamenjuje narednim

$$\theta^* = \max_{\theta} \mathcal{L}(\theta) = \min_{\theta} (-\log \mathcal{L}(\theta)) = \min_{\theta} \left(-\sum_{i=1}^n \log p_\theta(x_i) \right)$$

pri čemu druga jednakost važi zahvaljujući tome što je logaritam monotono rastuća funkcija. Funkcija koja se minimizuje je negativna vrednost logaritma verodostojnosti (eng. negative log likelihood) i često se označava skraćenicom NLL.

Nalaženje minimuma često nije moguće izvršiti analitički, pa se stoga često vrši gradijentnim metodama, koje počivaju na izračunavanju gradijenta funkcije $-\log \mathcal{L}(\theta)$ po parametrima θ .