UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

JANEZ BRANK

STROJNO UČENJE NA SLIKOVNIH PODATKIH

DIPLOMSKA NALOGA

Mentor:
prof. dr. Ivan Bratko

Ljubljana, julij 2001

Kazalo

Povzetek

Naloga predstavlja nekatere možnosti uporabe tehnik strojnega učenja na slikovnih podatkih. Cilj je rešiti problem nadzorovanega učenja, pri katerem imamo dano neko množico slik, ki jih je uporabnik oz. človeški ekspert razdelil v več razredov, računalnik pa naj bi se naučil klasificirati nadaljnje slike po analogiji z učnimi primeri.

Da lahko na slikah uporabimo katerega od postopkov strojnega učenja, moramo zanje pripraviti neke opise, s katerimi bo znal postopek strojnega učenja ravnati. Preizkusil sem več takšnih tehnik opisovanja slik: globalne vektorje značilk (po en vektor za celotno sliko: histogrami, koherenčni vektorji, avtokorelogrami, zgoščeni avtokorelogrami) in segmentacijo (sliko razdelimo na regije po barvi in teksturi ter opišemo vsako regijo posebej). Za učenje sem pri globalnih opisih slik uporabil metodo najbližjih sosedov in metodo podpornih vektorjev (pri slednji sem preizkusil linearna, kubična in radialna jedra), pri opisih na podlagi segmentacije pa le metodo najbližjih sosedov, ker metoda podpornih vektorjev predpostavlja, da je opis vsakega učnega primera en sam vektor, s segmentacijo pa bi dobili za vsako sliko po več opisov (po enega za vsako regijo).

Naloga najprej podaja pregled literature in predstavi uporabljene postopke, nato pa opiše rezultate poskusov na zbirki 1172 slik, ki sem jih ročno razvrstil v štirinajst razredov. Med pričakovanimi ugotovitvami, ki sledijo iz poskusov, sta: metoda podpornih vektorjev daje boljše rezultate kot metoda najbližjih sosedov; avtokorelogrami dajejo boljše rezultate od histogramov in koherenčnih vektorjev. Presenetljiv rezultat poskusov pa je, da klasifikatorji, temelječi na segmentaciji slik (in metodi najbližjih sosedov), niso dajali višje točnosti od klasifikatorjev, temelječih na globalnih opisih in metodi najbližjih sosedov.

1

Uvod

Zbirke slik (*pictorial databases*) postajajo vse večje in vse pogostejše. Lahko gre za digitalizirane fotografske arhive kakšne ustanove, za komercialne zbirke grafike, namenjene založnikom in oblikovalcem, za zbirke slik, pobrane s spleta; vsem je skupno to, da so praviloma velike, slike v njih pa niso opremljene s kakšnimi dodatnimi podatki ali opisi, ki bi olajšali delo z zbirko. Z vprašanjem, kako obvladovati takšne zbirke in kako olajšati uporabniku dostop do njih, se od začetka devetdesetih let naprej ukvarja vse več raziskovalcev. Poudarek v tej literaturi je večinoma v odgovarjanju na uporabnikove poizvedbe: kako jih sploh formulirati, kako določiti, katere slike so relevantne glede na določeno poizvedbo, in kako vse skupaj izvesti čim bolj učinkovito.

Namen te diplomske naloge pa je pogledati na slikovne podatke malo bolj s stališča strojnega učenja: če slike po videzu ali vsebini razvrstimo v neke razrede, se pojavi vprašanje, kako bi lahko računalnik posnemal to razvrstitev in se naučil razvrščati tudi nadaljnje slike v primeren razred. Tak problem je sicer mogoče manj neposredno uporaben kot odgovarjanje na uporabnikove poizvedbe, bi pa bil koristen, če bi hotel uporabnik na primer kategorizirati neko zbirko slik. Ročno bi jih kategoriziral nekaj, nato pa bi mu računalnik predlagal kategorizacijo ostalih slik, uporabnik pa bi le še popravljal njegove napake.

Čeprav se, kot rečeno, večina literature ne ukvarja specifično z opisanim problemom (redek primer je delo Huanga in sodelavcev [24]), pa vseeno obstaja veliko del, na katera se lahko opiramo. Po eni strani je tu literatura s področja dostopanja do slikovnih zbirk (*image retrieval*), v kateri je opisanih veliko tehnik, kako lahko sliko (ki je v osnovi neka matrika števil) predstavimo s kakšnimi bolj obvladljivimi strukturami, na primer vektorji značilk. Po drugi strani pa je koristna literatura s področja strojnega učenja, ki opisuje algoritme, s katerimi se lahko računalnik nauči razvrščati slike v razrede, potem ko jih imamo enkrat opisane z vektorji ali kakšnimi podobnimi opisi. Pri poskusih v okviru te naloge sem se odločil za dve glavni družini algoritmov: metodo najbližjih sosedov in metodo podpornih vektorjev. Obširnejši pregled relevantne literature podajajo naslednja poglavja.

Drugo poglavje je namenjeno pregledu literature s področja dostopa do slikovnih zbirk in med drugim navaja različne tehnike opisovanja slik, tudi tiste, ki sem jih na koncu zares implementiral in uporabil v poskusih. Tretje poglavje predstavlja metodo najbližjih sosedov, podaja nekaj osnovnih teoretičnih lastnosti te metode in opisuje nekatere tehnike, koristne pri njeni praktični implementaciji. Četrto poglavje je zamišljeno kot pregleden uvod v metodo podpornih vektorjev s poudarkom na reševanju klasifikacijskih problemov; obsega razlago osnovne zamisli razmejevanja učnih vektorjev z ravnino, predstavi problem učenja kot optimizacijsko nalogo in opisuje vlogo jeder kot načina, kako priti do nelinearnih modelov. Vsebuje tudi razdelek o uporabi metode podpornih vektorjev na večrazrednih domenah, saj je to glede na naš problem še posebej zanimivo.

Peto poglavje opisuje poskuse, v okviru katerih sem nekatere prej opisane tehnike in algoritme implementiral in praktično preizkusil na neki konkretni zbirki slik. Sklep povzema glavne ugotovitve poskusov in navaja nekaj možnosti za izboljšave in nadaljnje delo na tem področju. Nekaj ugotovitev je pričakovanih, na primer ta, da daje metoda podpornih vektorjev višje klasifikacijske točnosti od metode najbližjih sosedov, in ta, da so med opisi slik avtokorelogrami uspešnejši od histogramov. Nekatere ugotovitve pa so presenetljive, predvsem razmeroma slabi uspeh metod, ki temeljijo na segmentaciji.

Glavni prispevek te naloge je v sistematični obravnavi in primerjavi različnih tehnik opisovanja slik, različnih mer podobnosti in različnih algoritmov strojnega učenja. Stranska rezultata sta tudi program, s katerim sem poskuse izvajal, ter zbirka 1172 slik, ročno razvrščenih v 14 razredov, ki bi lahko prišla prav še pri kakšnih drugih poskusih te vrste.

2

Obvladovanje slikovnih zbirk

2.1 Uvod

Sodeč po literaturi so bili slikovni podatki doslej na področju strojnega učenja deležni razmeroma omejene količine pozornosti. S slikovnimi zbirkami se ukvarja predvsem presek področij, kot sta dostopanje do informacij (*information retrieval*) in računalniški vid. Osnovni problem, ki ga ta literatura obravnava, je, kako olajšati uporabniku pregledovanje in dostopanje do slikovnih zbirk. V ta namen je potrebno odgovoriti na vprašanja, kot so:

1. Kako naj uporabnik opiše, kaj ga zanima?
2. Kako izraziti neko mero podobnosti, po kateri lahko računalnik ovrednoti in razvrsti slike iz baze kot bolj ali manj relevantne glede na poizvedbo?
3. Kako podatke o slikah organizirati in poindeksirati, tako da bo mogoče na poizvedbe tudi pri velikih slikovnih zbirkah odgovoriti v razmeroma kratkem času (npr. nekaj sekund)?

Seveda teh vprašanj ne moremo obravnavati povsem ločeno, saj se ti problemi in odgovori nanje v precejšnji meri prepletajo. Način, kako naj bi uporabnik izrazil svojo poizvedbo, je pač prilagojen predstavitvi slik, ki jo uporablja program pri izvajanju poizvedbe. Potrebam procesiranja med izvajanjem poizvedbe pa naj bi bila prilagojena tudi organizacija podatkov o slikah v bazi.

Glede prve točke lahko še dodamo, da je seveda zaželjeno, če je način poizvedovanja z uporabnikovega vidika čim preprostejši in čim bolj intuitiven. Pogosto je pomembno tudi, da je priprava podatkovne zbirke čim bolj avtomatizirana in ne zahteva velikih količin ročnega označevanja.

Po zgledu [21] bi lahko postopke za preiskovanje slikovnih zbirk razdelili v tri skupine: (1) tiste, ki temeljijo na besedilu (*text-based image retrieval*); (2) tiste, ki temeljijo na vsebini (se pravi: na videzu; *content-based i. r.*); (3) in tiste, ki temeljijo na semantiki (*semantic-based retrieval*).

2.1.1 Pristopi na podlagi tekstovnih opisov

Nekateri postopki se torej opirajo na tekstovni opis, ki mora biti pripravljen za vsako sliko iz baze. Uporabnik potem v poizvedbi navede pogoje, ki jim mora tak opis ustrezati (na primer: zahteva prisotnost določenih ključnih besed). Ta pristop je razmeroma preprost in ne prinaša posebnih tehničnih problemov, njegova slabost pa je predvsem v tem, da mora nek strokovnjak ali skrbnik slikovne zbirke za vsako sliko pripraviti primeren opis. Pri velikih zbirkah slik si takšnega označevanja preprosto ne moremo privoščiti, problematično pa je tudi pri zbirkah, ki se neprenehoma dopolnjujejo ali spreminjajo (na primer pri zajemanju slik za potrebe spletnega iskalnika). Druga velika pomanjkljivost tega pristopa je, da je slike pogosto težko dobro opisati; različni ljudje bi lahko pri opisu iste slike uporabili tudi čisto druge besede. Zato je mogoče, da uporabnik želenih slik, čeprav so v bazi, ne najde, ker se njegov besednjak razlikuje od opisovalčevega.

Tekstovni ali celo deloma strukturirani opisi so lahko seveda koristno dopolnilo kateremu od ostalih pristopov, še posebej v primeru, če imamo takšne opise za uporabljeno slikovno zbirko že pripravljene. Primer je na primer sistem Chabot [38, 13]. V fotografskem arhivu kalifornijskega Urada za vodne vire je bila vsaka slika opremljena z nekaj (pogosto nepopolnimi) podatki, kot so kraj in datum posnetka, nekaj ključnih besed, podatki o filmu ipd. Vsaki sliki so pripisali še preprost opis, dobljen z analizo slike same: sliko so poenostavili na trinajst osnovnih barv in ugotovili, katerih barv je na sliki največ. Poiskali so tudi „pike“ (*dots*), omejena območja ene barve, obkrožena z drugimi barvami. Tako dobljene podatke o sliki lahko predstavimo simbolično in kvalitativno s preprostim nizom; pri sliki travnika z rumenimi cvetovi lahko na primer dobimo opis „veliko zelene, veliko majhnih rumenih pik“. Takšne nize so dodali v relacijsko bazo k ostalim podatkom o slikah in uporabnik je lahko pri poizvedbah postavljal pogoje tudi glede tega opisa. Ker so bili ostali podatki o slikah skopi, je lahko že tak preprost dodatni opis olajšal iskanje po zbirki; tako je na primer poizvedba, ki je zahtevala ključni besedi „rumena“ in „cvetje“, vrnila manj uporabnih slik od poizvedbe, ki je zahtevala le ključno besedo „cvetje“ in omembo rumene barve v simboličnem opisu. Simbolični opisi sami bi bili za iskanje verjetno premalo zanesljivi, v kombinaciji s ključnimi besedami, pripravljenimi od prej, pa so se dobro obnesli.

2.1.2 Pristopi na podlagi vsebine

Program naj bi iz slike sam izračunal oz. določil nekakšen opis, s katerim si bo lahko kasneje pomagal pri pregledovanju zbirke. Uporabnik po tako urejeni zbirki običajno pregleduje tako, da določi „poizvedovalno sliko“, računalnik pa naj bi mu pokazal tiste slike iz zbirke, ki so poizvedovalni sliki po videzu podobne. Včasih lahko uporabnik poizvedovalno sliko tudi nariše (sistem MUSEUM [22]; Jacobs in sodelavci [28]); v tem primeru je treba upoštevati, da je to le skica, na kateri marsikaj manjka in ki se zato precej razlikuje celo od tiste slike v zbirki, ki jo je imel uporabnik pri risanju mogoče v mislih.

Sliko ali del slike pri takšnih pristopih običajno opišemo z vektorjem številskih vrednosti. V poštev pridejo na primer histogrami, statistični momenti, glavni koeficienti valčnih transformirank. Razdaljo med temi vektorji lahko merimo s katero od običajnih norm, na primer manhattansko (L_1-norma) ali evklidsko razdaljo (L_2-norma). Večja ko je razdalja med vektorjema, manj podobni naj bi si bili sliki, ki ju ta dva vektorja opisujeta.

Če se naš vektor nanaša le na sliko kot celoto, je tak opis seveda precej grob, kar lahko pri poizvedovanju predstavlja težave. Tako je lahko na primer nek predmet prisoten tako na sliki iz zbirke kot na poizvedovalni sliki, vendar na drugem koncu slike in sta zato opisa slik čisto drugačna; uporabniku se sliki sicer zdita podobni, programi pa ne.

Zato je koristno, če sliko razdelimo na več delov in z vektorjem opišemo vsakega posebej. Tu nastopi problem segmentacije: kako razdeliti sliko na več delov? Pogosto bi si želeli, da bi vsak del ustrezal nekemu predmetu (ali pa značilnemu elementu ozadja). S tehnikami, kot je odkrivanje robov, je mogoče to sicer do neke mere avtomatizirati, vendar se sistemi, ki temeljijo na opisovanju posameznih predmetov, večinoma zanašajo na pomoč človeka pri pripravi slikovne zbirke. Tak primer je QBIC [16, 18], kjer skrbnik pri pripravi zbirke očrta na sliki zanimive predmete ali pa vsaj pokaže na ozadje, program pa iz barve ozadja sklepa na to, kje je ozadje in kje predmeti.

Druga možnost je, da skušamo segmentacijo povsem avtomatizirati in razdeliti sliko na področja s homogeno barvo ali teksturo. Posamezno tako področje sicer ne predstavlja nujno nekega koncepta iz uporabnikovega pojmovnega sveta (ni predmet, ampak le del predmeta), tako da npr. poizvedbe po obliki predmetov v tako opremljeni zbirki niso možne. Čim boljša segmentacija na podlagi teksture je še vedno predmet številnih raziskav.

Še ena težava, ki jo moramo rešiti, če se odločimo slike razkosati in opisovati z vektorji vsak del posebej, je ta, da lahko zdaj z razdaljo med vektorji pravzaprav ugotavljamo podobnost delov slik, ne pa še slik kot celot. Podobnost med slikama moramo nekako izračunati iz podobnosti med njunimi deli. Če je nek del prve slike podoben več delom druge slike, ali naj na drugi sliki vsi prispevajo k meri podobnosti ali le eden od njih? Če je nek majhen del prve slike podoben velikemu delu druge slike, koliko naj to prispeva k podobnosti med slikama? Popolnih odgovorov na taka vprašanja najbrž sploh ni.

2.1.3 Pristopi na podlagi semantičnih opisov

Primer takega pristopa opisuje Quintana [44]. Poskusne osebe so si ogledale slike iz njegove zbirke, nato pa poskušale po spominu navesti čim več slik in podrobnosti z njih. Takšni opisi naj bi odražali dejstvo, da si ljudje nekatere reči zapomnijo bolje kot druge. V drugem poskusu pa je ljudi naprosil, naj slike, za katere se jim zdi, da nekako sodijo skupaj, združijo v skupine. Tako dobljene podatke je analiziral in sestavil bazo znanja v obliki semantičnih okvirov (po en okvir za vsako sliko). Pojme (koncepte), ki se pojavljajo v okvirih (in na slikah), je organiziral v hierarhijo (predikat *is-a*). Tudi uporabnikova poizvedba naj bi bila podana v obliki (deloma izpolnjenega) okvira.

Kot kaže že ta opis, je bilo s pripravo razmeroma majhne zbirke (32 slik) ogromno dela, potrebno pa je bilo sodelovanje več poskusnih oseb in še človeka, ki se spozna na pripravo baze znanja in tvorbo semantičnih okvirov (*knowledge engineer*). Ker odkrivanja semantičnih pojmov na slikah še ne znamo zadovoljivo avtomatizirati, je semantični pristop za večje zbirke slik zaenkrat nepraktičen.

Z vidika uporabnika je prednost zbirke slik, opremljenih s semantičnimi opisi, tudi v tem, da je mogoče zasnovati preprost in razmeroma intuitiven uporabniški vmesnik za iskanje po zbirki. Pri sistemu SCORE [3] lahko uporabnik na primer prek ikon zahteva, naj bodo na sliki prisotni določeni koncepti, ter določi njihove lastnosti ali atribute (na primer: plesalec = človek + atribut ples).

2.2 Tehnike opisovanja vsebine slik

Sliko si predstavljajmo kot matriko I, sestavljeno iz H vrstic in W stolpcev. Slikovni element (*pixel*) na preseku stolpca x in vrstice y označimo z $I[x, y]$.

2.2.1 Histogrami

Histogrami so eden od najstarejših načinov, kako nekaj informacije o videzu slike zajeti v vektor realnih števil. Da dobimo histogram, barvni prostor, ki mu pripadajo elementi naše slike, najprej razdelimo v končno število (na primer C) celic (*bins*) ter za vsak slikovni element ugotovimo, kateri celici pripada njegova barva. Če si v mislih iz vsake celice izberemo še neko barvo, ki predstavlja célo celico, lahko torej rečemo, da naša slika zdaj temelji na C-barvni paleti. $I[x, y] \in \{1, \ldots, C\}$ je zdaj indeks, na katerem se v paleti nahaja barva slikovnega elementa (x, y).

Barvni histogram je vektor, ki za vsako barvo v paleti hrani število slikovnih elementov, ki so te barve:

$$H_c = |\{(x, y) : I[x, y] = c\}| \qquad \text{za} \qquad 1 \leq c \leq C$$

Pogosto, sploh če imamo opravka s slikami različnih velikosti, je koristno tak histogram normalizirati: če vse komponente delimo z $W \times H$, bo hranil histogram deleže posameznih barv na sliki. Dober vir o rabi histogramov pri ugotavljanju podobnosti med slikami sta Swain in Ballard [46].

Lepe lastnosti histogramov so, da so razmeroma robustni (histogram se ne bo močno spremenil, tudi če se predmet na sliki premakne ali pa malce zasuka, če se malo spremeni zorni kot kamere in če je del predmeta zakrit z nečim drugim), neobčutljivi na ločljivost slike in da jih je preprosto računati [46, str. 14b]. Občutljivi pa so na primer na spremembe osvetlitve, saj se lahko ob tem močno spremeni barva večine slikovnih elementov; to lahko poskušamo omiliti s predhodno obdelavo slike. Če znamo ločiti predmet od ozadja ali pa vsaj vemo, katere barve pripadajo ozadju (in jih lahko zato mogoče ignoriramo), bo histogram odporen tudi na spremembe v ozadju.

Pomembna slabost histogramov pa je seveda ta, da zanemarijo vse podatke o razporeditvi barv po sliki. Dve sliki sta lahko čisto različni, čeprav vsebujeta iste barve in celo v podobnih razmerjih. Tisoč pik rdeče barve, razsutih po celi sliki, bo imelo na histogram približno enak vpliv kot rdeč kvadrat velikosti 32×32 ali rdeč pas oblike 5×200. Takšne težave so še posebej izrazite pri velikih slikovnih zbirkah, kjer je še toliko bolj verjetno, da se bosta našli dve različni sliki s podobnima histogramoma.

Zaradi naštetih pomanjkljivosti so se pojavile različne izboljšane vrste histogramov in podobnih struktur. Videti je, da se histograme v zadnjih letih uporablja nekaj manj kot včasih.

Če imamo histogram nekega predmeta H^P in neke slike H^I, lahko podobnost med njima opišemo s formulo:

$$HI(H^P, H^I) = \frac{\sum_{c=1}^{C} \min\{H_c^P, H_c^I\}}{\sum_{c=1}^{C} H_c^P}.$$

Izrazu v števcu pravita *presek histogramov* (*histogram intersection*). Če je neka barva na sliki zelo pogosta, na predmetu pa ne, je najbrž barva ozadja in ji zato na mero podobnosti ne dovolimo toliko vplivati. Tu smo seveda predpostavili, da temeljita oba histograma na isti paleti barv in da smo šteli v histogram predmeta le slikovne elemente, ki so res pripadali predmetu (ne pa na primer kakšnemu ozadju, na katerem je bil narisan). Če bi histograma normalizirali, bi bila opisana mera pravzaprav enakovredna L_1-normi:

$$1 - HI(H^P, H^I) = \frac{1}{2} \sum_{c=1}^{C} |H_c^P - H_c^I|.$$

L_1-norma seveda meri razdaljo med vektorjema in s tem različnost med slikama, HI pa podobnost med slikama, zato sta različno predznačeni.

Drugačno mero razdalje med histogrami uporablja sistem QBIC [16]. Skušali so upoštevati dejstvo, da so si različne barve iz palete vendarle lahko podobne med sabo: rdeča in oranžna slika sta si podobni, zgoraj opisana mera HI pa bi ocenila njuno podobnost kot 0.

Naj bo torej a_{ij} neka ocena podobnosti med barvama i in j. Potem definirajmo razdaljo med histogramoma H^1 in H^2 kot

$$d(H^1, H^2) = (H^1 - H^2)^T A (H^1 - H^2) = \sum_{i=1}^{C} \sum_{j=1}^{C} (H_i^1 - H_i^2) a_{ij} (H_j^1 - H_j^2).$$

Dejstvo, da je na enem histogramu neke barve več kot na drugem, se torej lahko kompenzira, če je na drugem neke podobne barve več kot na prvem. Če rečemo, da si niso barve med sabo čisto nič podobne, imamo $A = I$ in opisana mera razdalje postane navadna evklidska razdalja (L_2-norma).

Za matriko A predlagajo, naj bi izhajali kar iz barv kot trorazsežnih vektorjev (npr. v prostoru RGB); če je d_{ij} evklidska razdalja med vektorjema barv i in j, lahko vzamemo $a_{ij} = 1 - d_{ij}/d_{max}$ ali pa $a_{ij} = \exp[-\sigma(d_{ij}/d_{max})]$ (za $d_{max} \equiv \max_{i,j} d_{ij}$) [27, str. 731a].

Slabost opisane formule je, da moramo pri njej sešteti kar $O(C^2)$ členov, in da metriko, ki nam jo v prostoru histogramov definira, težko zajamemo v prostorske indeksne strukture, s katerimi si sicer pogosto pomagamo pri indeksiranju takšnih vektorjev. Pokazali pa so [27], da lahko, če definiramo $d_a(H^1, H^2)$ kot evklidsko razdaljo med povprečnima barvama histogramov H^1 in H^2, poiščemo konstanto λ, za katero je $d(H^1, H^2) \geq \lambda d_a(H^1, H^2)$. Torej, če nas za nek H^1 zanimajo histogrami H^2, ki ustrezajo pogoju $d(H^1, H^2) \leq D$, lahko za začetek poiščemo take z $d_a(H^1, H^2) \leq D/\lambda$, nato pa le med njimi preverjamo pogoj $d(H^1, H^2) \leq D$. Prednost te strategije je, da je d_a veliko lažje računati, omogoča pa nam tudi uporabo prostorskih podatkovnih struktur (npr. k-d-drevesa, R-drevesa, itd.), da indeksiramo histograme po povprečni barvi in učinkovito poiščemo tiste, ki so dovolj blizu določeni drugi povprečni barvi.

2.2.2 Razčlenjevanje histogramov

Histogram si lahko predstavljamo kot skupino košaric, po eno za vsako barvo. Slikovni elementi določene barve gredo vsi v isto košarico, na koncu pa za vsako košarico preštejemo, koliko elementov vsebuje. Zamisel razčlenjevanja histogramov (*histogram refinement*) pa je, da elemente znotraj vsake košarice razdelimo še naprej (na podlagi kakšnega drugega kriterija, ne le barve) in štejemo vsako skupino posebej [41, 43].

Preprost primer bi bil, da bi ločili slikovne elemente glede na to, ali se pojavljajo bliže osrednjemu delu slike ali pa bolj na robovih; tako bi za vsako barvo dobili dve števili (*centering refinement*). Običajno so namreč zanimivejši prizori na sredini slike, robni deli pa so manj pomembni.

Pomembnejša zamisel te vrste pa so koherenčni vektorji, ki jih opisuje naslednji razdelek.

Seveda je mogoče različne kriterije razlikovanja slikovnih elementov tudi kombinirati (*successive refinement*). Pass in Zadih [42] predlagata še več kriterijev, med drugim prisotnost robov v okolici slikovnega elementa in hevristiko, ki naj bi se odzivala na prisotnost teksture. Na ta način dobljenemu sestavljenemu histogramu pravita *joint histogram.*

2.2.3 Koherenčni vektorji

Koherenčni vektorji, ki so jih uvedli Pass in sodelavci [43], poskušajo omiliti eno od glavnih pomanjkljivosti histogramov: njihovo popolno zanemarjanje prostorske razporeditve barv. Pri koherenčnih vektorjih skušamo razlikovati med primeri, ko se neka barva pojavlja razpršena med veliko majhnih peg ali pik, ter primeri, ko tvori neko večjo, enotno površino.

Slikovne elemente neke slike si lahko predstavljamo kot točke grafa. Točki (x_1, y_1) in (x_2, y_2) naj bosta povezani, če sta ustrezna slikovna elementa enake barve in se dotikata vsaj z ogliščem, če že ne s stranico. Povezane komponente takega grafa predstavljajo sklenjene skupine slikovnih elementov enake barve. Takih povezanih komponent ni težko določiti. Izberimo si nek prag T in slikovni element razglasimo za koherentnega, če pripada povezani komponenti, ki pokriva vsaj T slikovnih elementov. (Za T so izbrali na primer 1 % površine slike [41, str. 97b].)

Za vsako barvo c preštejmo koherentne in nekoherentne slikovne elemente tiste barve. Če je prvih na primer α_c, drugih pa β_c, lahko iz teh števil tvorimo barvni koherenčni vektor (*color coherence vector*)

$$\langle(\alpha_1, \beta_1), \ldots, (\alpha_C, \beta_C)\rangle.$$

Če bi te vrednosti paroma sešteli, bi dobili v $\alpha_c + \beta_c$ ravno število slikovnih elementov barve c, torej bi bil vektor vsot ravno histogram opazovane slike.

Razliko med koherenčnima vektorjema dveh slik, $G = \langle(\alpha_c, \beta_c)\rangle_{c=1}^C$ in $G' = \langle(\alpha'_c, \beta'_c)\rangle_{c=1}^C$, lahko merimo kar z L_1-normo:

$$d(G, G') = \sum_{c=1}^{C} |\alpha_c - \alpha'_c| + |\beta_c - \beta'_c|.$$

Če bi z L_1-normo primerjali dva histograma, bi pri barvi c dobili člen $|(\alpha_c+\beta_c)-(\alpha'_c-\beta'_c)|$, kar je očitno vedno manjše ali enako členu $|\alpha_c-\alpha'_c|+|\beta_c-\beta'_c|$. Torej je razdalja med koherenčnima vektorjema vsaj tolikšna kot med pripadajočima histogramoma; koherenčni vektorji so strožji pri ocenjevanju podobnosti slik.

Slabost koherenčnih vektorjev je na primer v tem, da lahko zaradi teksture na kakšni sliki zaznajo veliko samotnih (nekoherentnih) slikovnih elementov določene barve. Tudi pri koherentnih slikovnih elementih ne moremo ločiti primerov, ko pripadajo več majhnim povezanim komponentam (ki mogoče komaj segajo čez prag T), od tistih, ko so vsi v eni sami veliki komponenti.

2.2.4 Korelogrami

Pri histogramih je bil poudarek na štetju slikovnih elementov določene barve po celi sliki; zajeli so torej globalno prisotnost barve na sliki, zanemarili pa lokalne značilnosti njene razporeditve. Korelogram [23] pa daje poudarek prav na lokalnih lastnostih, slabše pa predstavi globalne.

Kot pri histogramu tudi tu začnemo s tem, da barvni prostor kvantiziramo in sliko v bistvu poenostavimo na C različnih barv. Osnovni podatki, ki jih hrani korelogram, so odgovori na vprašanja oblike: „Kakšna je verjetnost, da bomo v bližini slikovnega elementa barve i naleteli na slikovni element barve j?“ Korelogram slike I lahko torej definiramo kot

$$\gamma_{ij}^{(k)} \equiv P[I(p_2)=j \mid I(p_1)=i \wedge |p_1-p_2|=k].$$

Pri tem sta $p_1=(x_1,y_1)$ in $p_2=(x_2,y_2)$ slikovna elementa slike I, razdaljo med njima pa zaradi preprostosti merimo z normo L_∞. Razdalja $|p_1-p_2|$ nam torej pomeni $\max\{|x_1-x_2|,|y_1-y_2|\}$.

Teh verjetnosti ni težko izračunati. Če zanemarimo anomalije, do katerih bo prišlo na robovih slike (spomnimo se, da na robovih slike praviloma tako ali tako ni vizualno pomembnih delov prizora), lahko rečemo, da za vsak slikovni element obstaja $8k$ takih, ki so na razdalji k od njega (tvorijo namreč kvadrat s stranico $2k+1$). Potem je

$$\gamma_{ij}^{(k)} = \frac{|\{\langle p_1,p_2\rangle : I(p_1)=i, I(p_2)=j, |p_1-p_2|=k\}|}{|\{p_1 : I(p_1)=i\}|\cdot 8k}.$$

Za k-je običajno vzamemo nekaj različnih majhnih števil, na primer $k\in D\equiv\{1,3,5,7\}$. Če imamo $d\equiv|D|$ različnih vrednosti k in C različnih barv, je korelogram pravzaprav tabela velikosti $C\times C\times d$. Ker je taka tabela že kar velika in okorna, pogosto raje delamo z *avtokorelogrami:*

$$\alpha_c^{(k)} \equiv \gamma_{cc}^{(k)}.$$

Avtokorelogram torej podaja le verjetnosti, da se v bližini slikovnega elementa določene barve znajdejo še drugi slikovni elementi iste barve.

Avtokorelogram si lahko predstavljamo kot vektor s $C\times d$ komponentami. Razdaljo med dvema avtokorelogramoma lahko, podobno kot pri histogramih, merimo z L_1-normo, lahko pa posamezne komponente pred tem še obtežimo.

Še varčnejša različica avtokorelograma je zgoščeni avtokorelogram (*banded autocorrelogram*) [24]. Pri tem seštejemo istoležne komponente avtokorelograma za vse opazovane razdalje k:

$$\beta_c \equiv \sum_{k \in D} \alpha_c^{(k)}.$$

Komponenta β_c torej pove verjetnost, da je naključno izbran slikovni element v okolici nekega slikovnega elementa barve c tudi sam barve c. Zgoščeni avtokorelogram je tako le C-razsežni vektor, zato glede pomnilniških in računskih potreb ni nič zahtevnejši od histograma.

2.2.5 Lokalni histogrami

Ker histogram določene slike ne pove ničesar o tem, kako so posamezne barve razporejene po njej, je več avtorjev poskušalo takšne podatke v opis slike pripeljati prek uporabe več histogramov, od katerih vsak predstavlja nek manjši kos slike.

Neugodno pa je, da ti predlogi večinoma temeljijo na povsem fiksnem razrezu slike; ta se torej nikakor ne prilagaja vsebini slike. Namesto enega histograma jih imamo zdaj za vsako sliko več in razdaljo med slikama merimo tako, da seštejemo razdalje med istoležnimi histogrami. Sliko naj bi na primer razrezali na 8×8 enakih celic, ali pa na osrednji del in štiri robne, ali pa tudi uporabili več razrezov hkrati (sliko razrežemo na 3×3 celice, pa na 5×5 celic in si zapomnimo histograme za teh 34 celic in še enega za celo sliko).

Druga težava tega pristopa je, da moramo za vsako sliko hraniti kar veliko podatkov, npr. 35 ali 64 histogramov. Te histograme si lahko predstavljamo kot matriko: $H = (h_{ic})$, pri čemer h_{ic} pove, kolikšen delež slike predstavljajo slikovni elementi, ki ležijo v celici i in so barve c. Stehling in sodelavci [49] so predlagali naslednjo izboljšavo: namesto histogramov $H_i = (h_{i1}, \ldots, h_{iC})$, ki so pravzaprav vrstice matrike H, hranimo njene stolpce. Tak stolpec $H_c = (h_{1c}, \ldots, h_{nc})^T$ (če imamo n celic) za neko barvo c podaja zastopanost barve c po posameznih celicah. Prednost je v tem, da na določeni sliki mnoge barve izmed C barv naše palete najbrž sploh niso prisotne in tistih stolpcev ni treba hraniti, ker vsebujejo same ničle. Opisane stolpce imenujejo „histogrami barvnih oblik“ (*color shape histograms*). Zdaj ne primerjamo histogramov istoležnih celic, ampak histograma iste barve:

$$D(H_c^{(1)}, H_c^{(2)}) = \frac{\sum_{i=1}^{n} |h_{ic}^{(1)} - h_{ic}^{(2)}|}{A_c^{(1)} + A_c^{(2)}}, \qquad S(H^{(1)}, H^{(2)}) = \sum_{c=1}^{C} w_c \cdot (1 - D(H_c^{(1)}, H_c^{(2)}).$$

D torej meri različnost med stolpcema $H_c^{(1)}$ in $H_c^{(2)}$ po L_1-normi, vrednosti $A_c^{(1)}$ oz. $A_c^{(2)}$ v imenovalcu pa predstavljata delež barve c na prvi oz. drugi sliki. S meri podobnost med slikama; temelji na zamisli, da je podobnost nasprotje razdalje oz. različnosti, prispevke posameznih barv pa lahko različno obteži (predlagajo $w_c = \min\{A_c^{(1)}, A_c^{(2)}\}$, tako da imajo redkejše barve manjšo težo).

2.2.6 Opisovanje oblik

Obliko predmetov se v sistemih za pregledovanje slikovnih zbirk na podlagi podobnosti uporablja redkeje. Razlog je verjetno predvsem v tem, da je za zanesljivo odkrivanje predmetov in njihovih oblik potrebna pri pripravi zbirke pomoč človeka, pa tudi v tem, da se lahko oblika določenega predmeta na sliki močno spremeni, če kamera nanj pogleda iz drugačnega zornega kota. Pri avtomatskem odkrivanju predmetov ali regij na sliki preveč zlahka pride do prevelikih popačenj, da bi se bilo mogoče zanašati na podatke o obliki.

QBIC [16] na primer omogoča deloma avtomatizirano definiranje predmetov na slikah, nato pa obliko vsakega predmeta predstavi z vektorjem 20 številskih vrednosti, med katerimi so površina, mere okroglosti in ekscentričnosti, orientacija glavne osi lika ter nekaj statističnih momentov. Uporabnik pri poizvedbi skicira (kot mnogokotnik) silhueto predmeta, ki ga zanima, program pa izračuna pripadajoči vektor in išče njemu bližnje vektorje iz baze. Ugodna lastnost takšne predstavitve je, da lahko npr. zanemarimo tisto komponento vektorja, ki predstavlja orientacijo, pa bomo našli predmete določene oblike, tudi če so zavrteni. Vendar pa takšna razmeroma groba predstavitev ne more razlikovati med manjšimi podrobnostmi posameznih oblik.

Oblike lahko opišemo tudi s pomočjo Fourierjeve transformacije [7]. Obod opazovanega lika si predstavljajmo kot krivuljo in jo parametrizirajmo:

$$t \mapsto (x(t), y(t)) \qquad \text{za} \qquad 0 \leq t \leq T.$$

Funkciji $x(t)$ in $y(t)$ diskretizirajmo in obdržimo nekaj glavnih koeficientov njunih Fourierjevih transformirank. Če je lik konveksne ali vsaj zvezdaste oblike, bi lahko obod izrazili tudi v polarnih koordinatah kot funkcijo $\phi \mapsto r(\phi)$, $0 \leq \phi \leq 2\pi$, in transformirali to.

2.2.7 Opisovanje tekstur

Teksturo različni avtorji definirajo na različne načine, ugotavljajo pa tudi, da splošno sprejete definicije ni. Običajno rečemo, da gre za neke bolj ali manj redno ponavljajoče se vzorce na sliki. Obstajajo posebne zbirke fotografij različnih tekstur; najbolj znan je verjetno Brodatzov album [8].

Če naj si s teksturo kako pomagamo pri opisovanju slik, moramo znati na sliki prepoznati območja, ki jih prekriva določena tekstura. Ta problem se imenuje segmentacija slik po teksturi (*texture segmentation*); že dolgo časa ga preučujejo področja, kot sta računalniški vid in prepoznavanje vzorcev.

Ker naj bi bila tekstura nekaj, kar se približno periodično ponavlja, ni presenetljivo, da jih pogosto poskušajo prepoznavati s pomočjo frekvenčne analize. Pogosto si pomagajo z Gaborjevimi funkcijami (imenujejo se po Dennisu Gaborju, ki jih je prvi opisal leta 1946; v dve dimenziji in na področje analize tekstur pa so jih prenesli John G. Daugman in mnogi drugi v osemdesetih letih). Te funkcije so oblike

$$G(x, y) = e^{i\omega(x\cos\theta + y\sin\theta)} \cdot e^{-(x^2+y^2)/(2\sigma^2)}.$$

Prvi faktor tega produkta je pravzaprav sinusno valovanje. Njegov realni del je oblike $\cos\omega(x\cos\theta + y\sin\theta)$; to so torej neskončni valovi, pri čemer poteka vrh vsakega vala pod kotom $\theta - \pi/2$ (valovi so vzporedni premici $x\cos\theta + y\sin\theta = 0$); ω pa je frekvenca tega valovanja (razmik med vrhovoma dveh sosednjih valov je $\omega/2\pi$). Imaginarna komponenta je po obliki enaka, le da ima sin namesto cos in je torej po fazi zamaknjena za $\pi/2$. Drugi faktor pa je Gaussova (zvonasta) funkcija; σ vpliva na to, kako hitro bo padala proti 0. Ta Gaussova funkcija bo povsod razen v neki okolici koordinatnega izhodišča enaka 0 in bo zato sinusoido zadušila. Od slednje ostanejo le deli nekaj sosednjih valov.

Gaborjevo funkcijo uporabimo kot filter; če ima nek del slike visoko korelacijo z njo, nam to nakazuje prisotnost valovanja pod kotom približno θ in s frekvenco blizu ω. Zato je pametno uporabiti več različnih vrednosti θ in ω (banka Gaborjevih filtrov).

Gaborjeve funkcije imajo več lepih lastnosti. Iz načela nedoločenosti v frekvenčni analizi sledi, da imajo funkcije z ožjim časovnim spektrom širši frekvenčni spekter in obratno. Če neko funkcijo uporabimo kot filter, pa bi si želeli čim ožjega časovnega spektra (kar bi dalo boljšo lokalizacijo pojavov v času), bomo morali za to plačati s širšim frekvenčnim spektrom (kar pomeni slabšo lokalizacijo pojavov v frekvenčnem prostoru) in obratno. Najboljše razmerje med širino obeh spektrov pa imajo Gaussove funkcije. Ta dobra lastnost se prenese tudi na Gaborjeve, saj so to le sinusoide, ki nihajo znotraj Gaussovih ovojnic.

Druga lepa lastnost Gaborjevih funkcij je, naj bi tudi človekov zaznavni sistem teksture prepoznaval podobno kot banka Gaborjevih filtrov [52]. Posamezne skupine nevronov v možganski skorji naj bi delovale kot Gaborjevi filtri z različnimi ω in θ. Temu naj bi se še približali, če bi drugi faktor (Gaussovo funkcijo) v smeri valovanja raztegnili za približno dve tretjini (nivojnice take Gaussove funkcije torej ne bi bile več krožnice, ampak elipse).

Teksturo na nekem delu slike lahko torej opišemo z vektorjem vrednosti, ki jih dobimo ob korelaciji tistega dela slike z različnimi Gaborjevimi filtri. Alternativa takim opisom so na primer formule, ki so jih predlagali Tamura in sodelavci [51] in ki naj bi izražale grobost, kontrast, usmerjenost (ali so vzorci v teksturi vsi usmerjeni v približno isto smer), regularnost v zgradbi, to, ali je tekstura sestavljena iz črt, itd. Te mere naj bi se približno ujemale s človekovim dojemanjem teksture. Uporabili so jih tudi pri projektu QBIC [16].

V mnogih slikovnih zbirkah so teksture zanimive predvsem na „teksturnih slikah“ — takih, ki so v celoti prekrite z neko teksturo. Na ostalih slikah, npr. takih, ki prikazujejo nek heterogen prizor, je tekstura predmetov zaradi njihove oblike in zornega kota pogosto preveč popačena. Zato je koristno ločiti teksturne slike od ostalih. Preprosto hevristiko v ta namen so predlagali Li in sod. [34]: z grobim in preprostim postopkom segmentacije (gl. razd. 2.3.4) razdelijo sliko na več regij; regija je lahko sestavljena iz več nepovezanih kosov, glavno pa je, da imajo podobno barvo in teksturo. Pri teksturnih slikah bo vsaka regija približno enakomerno razpršena po celi sliki (ker je prisotna v vsaki pojavitvi tistega vzorca, iz katerega je tekstura sestavljena), pri neteksturnih pa bodo regije praviloma bolj zaokrožene in omejene na nek del slike. Zato sliko preprosto razdelijo na 16 enako velikih delov in pogledajo, kolikšen delež določene regije je na posameznih delih. Bolj ko so te porazdelitve enakomerne, bolj je verjetno, da je slika teksturna.

2.3 Valčna transformacija

2.3.1 Uvod

Valčna transformacija (*wavelet transform*) je priljubljeno in uspešno matematično orodje za frekvenčno analizo slik, pa tudi drugih signalov. Tudi teoretično je dobro raziskana.

Naslednji neformalni argument v podporo uvedbi valčne transformacije je povzet po [30, pogl. 1–3]. Če funkcijo $f(t)$, $f : \mathbb{R} \to \mathbb{C}$, analiziramo z običajno Fourierjevo transformacijo, dobimo transformiranko

$$\hat{f}(\omega) \equiv \int f(t) e^{-2\pi i \omega t} dt.$$

To je izraz oblike $\int f(t)\bar{g}(t)dt$, pri čemer predstavlja znak $\bar{\cdot}$ kompleksno konjugacijo. Takšen izraz je pravzaprav skalarni produkt funkcij f in g, če si ju predstavljamo kot vektorja v prostoru (kvadratno integrabilnih) funkcij. Če vzamemo dva vektorja določene velikosti, bo njun skalarni produkt največji takrat, kadar kažeta v isto smer; enak 0, če sta pravokotna; in najmanjši tedaj, kadar kažeta v nasprotni smeri. S tem torej pravzaprav merimo, koliko se vektorja ujemata v smeri. Tako si lahko predstavljamo tudi integral $\int f(t)\bar{g}(t)dt$: z njim merimo ujemanje med funkcijama f in g. V tem smislu nam $\hat{f}(\omega)$ podaja ujemanje med signalom $f(t)$ in kompleksno „sinusoido“ $e_\omega(t) \equiv e^{2\pi i \omega t}$; vendar pa meri to ujemanje po celem intervalu, po katerem integriramo (na primer: po celi realni osi). Če je f na primer glasbeni posnetek, bodo vse note določene frekvence prispevale k ustrezni vrednosti transformiranke $\hat{f}$, iz transformiranke pa ne bomo mogli na preprost način razbrati, kje v času je prišlo do posameznih pojavitev teh not in kako dolgo so trajale.

Zato bi lahko razmišljali o uvedbi okenske Fourierjeve transformacije, pri kateri na naš signal pogledamo skozi „okno“, ki je različno od 0 le na nekem omejenem intervalu. Naj bo torej $g(u)$ naša okenska funkcija, ki je različna od 0 le v neki okolici točke $u = 0$. Okno lahko zamaknemo v času, da bo osredotočeno okrog točke $u = t$, in z njim pomnožimo funkcijo f:

$$f_t(u) \equiv \bar{g}(u - t) f(u).$$

Potem izvedemo navadno Fourierjevo transformacijo:

$$\tilde{f}(\omega, t) \equiv \hat{f}_t(\omega) = \int e^{-2\pi i \omega u} \bar{g}(u - t) f(u) du.$$

Vendar pa bi zdaj, ko na signal f pogledamo skozi okno omejene širine, opazili v transformiranki $\hat{f}_t$ le tiste frekvenčne pojave, ki imajo dovolj visoko frekvenco, da naredijo znotraj enega okna vsaj kak nihaj. Počasnejša nihanja pa bi se v funkciji $\tilde{f}(\omega, t)$ izrazila na časovni osi, torej pri spreminjanju parametra t — z drugimi besedami, na funkciji f bi jih opazili šele, ko bi se z oknom premikali po njej.

Do valčne transformacije pridemo, če uvedemo okna različne širine. Začnimo z osnovno okensko funkcijo $\psi(t)$ (reče se ji tudi matični valček, *mother wavelet*). Nato jo skrčimo ali raztegnimo za nek faktor s ter zamaknimo po času za t. Tako dobimo družino valčkov

$$\psi_{s,t}(u) \equiv |s|^{-p} \psi\left(\frac{u - t}{s}\right).$$

Valčna transformiranka potem preprosto podaja ujemanje med signalom f in posameznimi valčki:

$$\tilde{f}(s,t) \equiv \int f(u)\psi_{s,t}(u)du.$$

Pogosto rečemo, da nam $\tilde{f}(s,t)$ izraža „detajl“ funkcije f v času t pri velikostnem redu (*scale*) s.

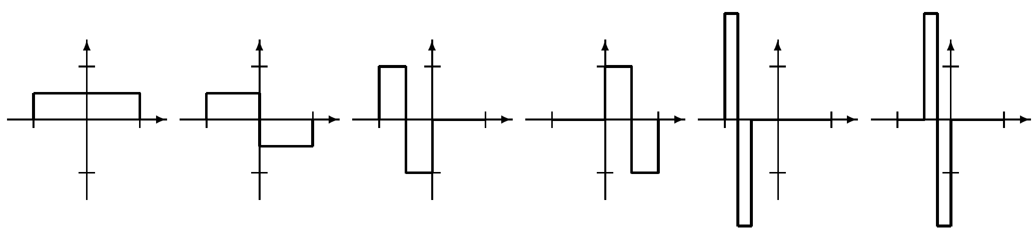
Za parameter p si običajno izberejo kakšno nenegativno število; $p = 1$ na primer zagotavlja, da bo $\int |\psi_{s,t}(u)|du$ enak pri vseh s; $p = 1/2$ pa, da bo pri vseh s enak integral $\int |\psi_{s,t}(u)|^2 du$. S tem torej zagotovimo, da se ohranja L_1- ali L_2-norma funkcije ψ. Za teorijo ni zelo pomembno, kakšen p si izberemo, v praksi pa lahko poskušamo z njim v transformiranki zadušiti podatke o majhnih podrobnostih; ko s raste, se $\psi_{s,t}$ duši s faktorjem $1/|s|^p$ in temu primerno manjše so tudi vrednosti funkcij $\tilde{f}(s,t)$.

Izraz *valčki* je menda nastal zato, ker je ψ praviloma gladka, lokalizirana in nihajoča.

2.3.2 Diskretne valčne transformacije

Za razliko od signalov iz prejšnjega razdelka so slike diskretni in dvorazsežni signali. Barvne slike so pravzaprav tudi večkanalni signali, saj je barva opisana kot vektor (običajno s tremi komponentami); vendar pa to praviloma ne zahteva posebne obravnave, saj kanale običajno obdelujemo ločeno, na primer rdeče komponente posebej, zelene posebej in modre posebej, kot da bi pač imeli tri enokanalne signale.

Diskretni signali potrebujejo tudi diskretne valčne funkcije. Primer takšne diskretne družine valčkov so Haarove funkcije, ki jih prikazuje slika 2.1. Vidimo, da je bil pri tej družini parameter p postavljen na 1: ko se okno zoža na polovico svoje širine ($s = 1/2$), se mu amplituda podvoji ($|s|^{-p} = 2$). S tem smo dosegli, da je $\int |\psi_{s,t}(u)|du$ pri vseh enak 1.



Slika 2.1: Haarovi valčki.

Zaradi svoje strukture so še posebej primerni za delo z diskretnimi signali dolžine 2^n. Pri signalu s štirimi vzorci, $a = \langle a_0, a_1, a_2, a_3 \rangle$, bi na primer vzeli prve štiri Haarove funkcije:

$$H^0 = \frac{1}{4}\langle 1,1,1,1\rangle,\ H^1 = \frac{1}{4}\langle 1,1,-1,-1\rangle,\ H^2 = \frac{1}{2}\langle 1,-1,0,0\rangle,\ H^3 = \frac{1}{2}\langle 0,0,1,-1\rangle,$$

in računali skalarne produkte: transformiranka je $A = \langle A_0, A_1, A_2, A_3 \rangle$ za $A_i = \sum_{j=0}^{3} H^i_j a_j$. Tako je $A_0 = (a_0 + a_1 + a_2 + a_3)/4$, $A_1 = (a_0 + a_1 - a_2 - a_3)/4$, $A_2 = (a_0 - a_1)/2$, $A_3 = (a_2 - a_3)/2$. Če bi imel a osem komponent, bi morali tudi vse H^i raztegniti na osem komponent ter vzeti še naslednje štiri Haarove funkcije (od katerih sta prvi dve prikazani na gornji sliki).

Opazimo lahko, da je A_2 ravno odstopanje komponente a_0 od povprečja prvih dveh komponent: $(a_0 - a_1)/2 = a_0 - (a_0 + a_1)/2$. Podobno je A_3 odstopanje komponente a_2 od povprečja a_2 in a_3. Zapomnimo si obe povprečji: $a_{01} = (a_0 + a_1)/2$, $a_{23} = (a_2 + a_3)/2$, pa vidimo, da je A_0 zdaj povprečje teh dveh povprečij, A_1 pa odstopanje vrednosti a_{01} od teh dveh povprečij.

Isti vzorec bi se ponavljal tudi, če bi bilo zaporedje a daljše. Najprej moramo vzeti člene a-ja po dva in dva in izračunati povprečja ter odstopanje prvega člena v paru od povprečja. Odstopanja tvorijo drugo polovico transformiranega zaporedja A, prvo polovico pa dobimo tako, da isti postopek ponovimo na zaporedju povprečij. Tako smo dobili naslednji zelo preprosti postopek za računanje Haarove transformacije:

```
HAAR(⟨a_0, a_1, ..., a_{2^n - 1}⟩)
    če je n = 0, vrni a_0;
    sicer:
        za i od 0 do 2^{n-1} - 1:
            b_i ← (a_{2i} + a_{2i+1})/2;
            c_i ← a_{2i} - b_i;
        B ← HAAR(b);
        vrni ⟨B_0, ..., B_{2^{n-1}-1}, c_0, ..., c_{2^{n-1}-1}⟩;
```

Ravno zaradi svoje preprostosti so Haarove funkcije zelo priljubljena izbira, kadar je treba izvajati valčno transformacijo na veliki količini slikovnih podatkov. Koeficienti dobljene transformiranke imajo tudi zelo jasen intuitivni pomen; prvi je povprečje celega signala, drugi pove odstopanje povprečja leve polovice signala od skupnega povprečja, itd. Iz transformiranke lahko v celoti rekonstruiramo potek prvotnega signala, če pa obdržimo le prvih nekaj koeficientov, bi lahko še vedno rekonstruirali njegov približni potek. Zato kot približen opis prvotnega signala pogosto uporabijo prvih nekaj koeficientov njegove transformiranke.

Ena od slabosti Haarovih valčkov je na primer ta, da ob rekonstrukciji slike iz glavnih koeficientov njene Haarove transformiranke dobimo sliko z bolj ali manj velikimi kvadratnimi področji enake barve; v takih primerih je bolje uporabiti kakšno drugo družino valčnih funkcij, na primer valčke Daubechiesove. Vendar pa za potrebe indeksiranja slikovnih zbirk ta slabost pravzaprav ni pomembna.

2.3.3 Valčna transformacija na dvorazsežnih signalih

Imejmo nek dvorazsežni signal — matriko $I[x, y]$ z W stolpci in H vrsticami. Vrednosti $I[x, y]$ so lahko na primer intenzitete posameznih slikovnih elementov neke sivinske slike. Valčno transformacijo izvedimo najprej na posameznih vrsticah te matrike; sestavimo novo matriko, katere vrstice tvorijo ravno dobljene transformiranke; nad to novo matriko izvedemo valčno transformacijo po stolpcih. Transformiranke stolpcev združimo v novo matriko in to je naš končni rezultat.

```
DVORAZSEŽNA VALČNA TRANSFORMACIJA(I)
    za y od 0 do H - 1:
```

$\langle J[0,y], \ldots, J[W-1,y] \rangle \leftarrow \text{DWT}(\langle I[0,y], \ldots, I[W-1,y] \rangle)$;
za x od 0 do $W-1$:
$\langle \hat{I}[x,0], \ldots, \hat{I}[x,H-1] \rangle \leftarrow \text{DWT}(\langle J[x,0], \ldots, J[x,H-1] \rangle)$;
vrni $\hat{I}$;

Pri tem kot podprogram DWT kličemo diskretno valčno transformacijo enorazsežnega signala. Če hočemo tu na primer uporabiti Haarovo transformacijo, je koristno, če sta W in H potenci števila 2. Največkrat se vzame kar kvadratno I velikosti $2^n \times 2^n$. Če bi se vprašali, kako se posamezni elementi transformiranke $\hat{I}$ izražajo z elementi prvotne matrike I, bi dobili družino okenskih funkcij dveh spremenljivk, ki so po smislu podobne enorazsežnim Haarovim funkcijam. Ko se na primer konča prva zanka gornjega podprograma, je v $J[0,y]$ povprečje vrstice y matrike I. V drugi zanki na stolpcu, ki ga v J tvorijo ta povprečja, izvedemo Haarovo transformacijo; tako v $\hat{I}[0,0]$ dobimo povprečje prvega stolpca J-ja, to pa je povprečje povprečij posameznih vrstic matrike I. Torej je $\hat{I}[0,0]$ ravno povprečje cele matrike I. Podobno v $\hat{I}[0,1]$ dobimo razliko med povprečjem gornje polovice stolpca $J[0,\cdot]$ in povprečjem celega stolpca; tako nam $\hat{I}[0,1]$ pove, za koliko povprečje zgornje polovice slike odstopa od povprečja cele slike. Ostali elementi transformiranke $\hat{I}$ so kombinacije povprečij vedno manjših delov matrike I.

Druga možnost za vpeljavo dvorazsežne valčne transformacije pa je naslednja: sliko I velikosti $2^n \times 2^n$ si mislimo sestavljeno iz $2^{n-1} \times 2^{n-1}$ okenc velikosti 2×2. Vsako od teh okenc pretransformiramo po zgornjem postopku; za Haarove valčke to na primer pomeni:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \to \frac{1}{2} \begin{bmatrix} a+b & a-b \\ c+d & c-d \end{bmatrix} \to \frac{1}{4} \begin{bmatrix} a+b+c+d & a-b+c-d \\ a+b-c-d & a-b-c+d \end{bmatrix} \tag{2.1}$$

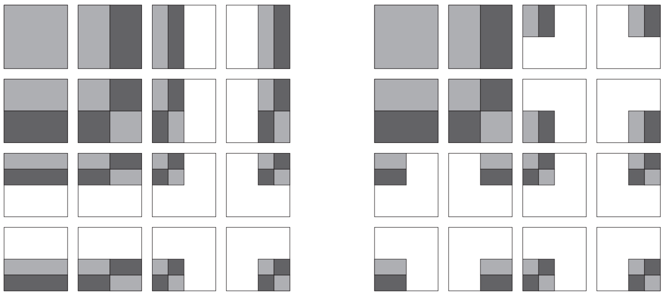
Zgornje leve elemente vseh transformiranih okenc lahko združimo v novo matriko, recimo ji A, velikosti $2^{n-1} \times 2^{n-1}$; podobno lahko zgornje desne združimo v neko matriko B, spodnje leve in spodnje desne pa v C in D. Transformiranka cele slike I naj bo matrika

$$\hat{I} \equiv \begin{bmatrix} \hat{A} & B \\ C & D \end{bmatrix}.$$

Da dobimo $\hat{A}$, moramo matriko povprečij A obdelati na enak način (tudi njo razrežemo na okenca velikosti 2×2 in tako naprej; vsakič nam ostane matrika s pol krajšimi stranicami, tako da se postopek ustavi, ko pridemo do matrike velikosti 2×2 in jo pretransformirano po formuli (2.1)).

Tej možnosti pravijo včasih „nestandardna dekompozicija“, prej opisanemu pristopu (uporabi enorazsežne transformacije na vrsticah in nato na stolpcih) pa „standardna dekompozicija“. Dobljeni transformiranki nista enaki, čeprav je v obeh prisotna vsa informacija s prvotne slike. Družini Haarovih funkcij, na katerih temeljita obe vrsti dekompozicije, sta namreč različni; pri nestandardni dekompoziciji je območje, na katerem je posamezna funkcija neničelna, vedno kvadratne oblike, pri standardni pa je pogosto tudi raztegnjeno. (Razliko na primeru transformiranja matrik velikosti 4×4 prikazuje slika 2.2.) Pri delu s slikovnimi zbirkami se včasih bolje obnese eden, včasih pa drugi pristop.

Ugodno pri nestandardni dekompoziciji je tudi to, da se lahko, če hočemo, po nekaj korakih ustavimo in v določenem trenutku matrike A ne pretransformiramo



Slika 2.2: Primerjava dvorazsežnih Haarovih funkcij.
Leva skupina funkcij ponazarja, kako se pri standardni dekompoziciji izražajo koeficienti transformiranke $\hat{A}$ z elementi prvotne matrike A, če je le-ta reda 4×4. Na svetlo osenčenih delih so funkcije enake $+C$, na temno osenčenih $-C$, drugod pa 0. Konstanta C je odvisna od tega, za kakšno obliko normalizacije se odločimo. Če je na primer $C = 1$, nam prva funkcija iz prve vrstice izračuna vsoto vseh komponent matrike A, pri $C = 1/16$ pa njihovo povprečje. Podobno nam prva funkcija iz druge vrstice sešteje vrednosti iz zgornje polovice matrike A in odšteje vsoto vrednosti iz spodnje polovice. Desna skupina funkcij na enak način ponazarja delovanje nestandardne dekompozicije.

naprej v $\hat{A}$, ampak jo pustimo nespremenjeno. Če se ustavimo po k korakih, nam v zgornjem levem kotu transformiranke ostane matrika velikosti $2^{n-k} \times 2^{n-k}$, ki je pravzaprav pomanjšana verzija prvotne slike.

Barvne slike so praviloma sestavljene iz več barvnih kanalov ali ravnin (na primer rdeče, zelene in modre). V takem primeru gledamo vsak kanal kot samostojno matriko in izračunamo njegovo transformiranko ločeno od ostalih kanalov.

2.3.4 Uporaba valčne transformacije na slikovnih podatkih

Običajno se jo uporablja za pripravljanje vektorjev, ki na jednrnat način v grobem opisujejo neko sliko ali del slike. Če obdržimo le glavnih nekaj koeficientov transformiranke, je to že grob opis prvotne slike, ostali koeficienti pa navajajo le še podrobnosti. Če sta si dva takšna vektorja blizu, sta si tudi sliki, ki ju predstavljata, v grobem podobni.

Soroden pogled na uporabo valčne transformacije pri delu s slikami je tudi takšen: transformiranka pravzaprav podaja frekvenčni spekter slike. Različni deli transformiranke predstavljajo različne dele frekvenčnega prostora. Komponente, ki pripadajo nizkofrekvenčnim nihanjem (npr. povprečje cele slike), izražajo temeljne barve velikih delov slike; komponente, ki pripadajo visokofrekvenčnim nihanjem, pa kažejo na morebitno prisotnost vzorcev, ki se po sliki hitro spreminjajo, torej nekakšnih tekstur. Zato si velja zapomniti, če že ne samih komponent transformiranke, pa vsaj skupno energijo komponent različnih frekvenčnih pasov.

Jacobs in sodelavci [28] na primer opisujejo postopek, pri katerem uporabnik nariše oz. skicira videz slike, ki ga zanima. Tako poizvedovalno sliko kot slike iz baze se raztegne ali skrči na velikost 128×128 in obdela s Haarovo transformacijo. Zapomnimo si le glavni koeficient (povprečje po celi sliki), za ostale koeficiente pa predznak in še to le v primeru, če so po absolutni vrednosti dovolj veliki (drugače pa si mislimo, da so enaki 0). K meri različnosti med dvema slikama tako po eni strani prispeva razdalja med glavnima koeficientoma, po drugi pa razlike v ostalih koeficientih. Pri tem imajo koeficienti iz različnih frekvenčnih pasov v tej vsoti različno težo (uteži so določili eksperimentalno). Prednost te metrike je, da je zelo preprosta in se jo da hitro računati.

Sistem WBIIS (*Wavelet-Based Image Indexing and Searching*) [57] uporablja valčke Daubechiesove. Slike normalizira na velikost 128×128 in izvede štiri korake transformacije; zgornja leva podmatrika velikosti 16×16 tako dobljene transformiranke je sestavljena iz štirih delov, velikih 8×8; od tega je zgornji levi kar pomanjšana oblika prvotne slike (vsak od njegovih elementov je povprečje po nekem oknu velikosti 16×16 prvotne slike), ostali trije pa povedo nekaj o odstopanjih od teh povprečij. WBIIS si zapomni teh 16×16 koeficientov; ker imamo tri barvne kanale, tvorijo vsi dobljeni koeficienti nek 768-razsežni vektor. Nato naredi še peti korak valčne transformacije in si zapomni zgornjih levih 8×8 koeficientov transformiranke (kar ima enako vlogo kot zgornjih levih 16×16 po štirih korakih transformacije, le da so podatki zdaj še bolj zgoščeni); tako dobimo še nek 192-razsežni vektor. Pri izvajanju poizvedb primerja slike iz baze s poizvedovalno sliko najprej po 192-razsežnih vektorjih (evklidska razdalja) in če so ti dovolj blizu, še po 768-razsežnih vektorjih. Slike, ki prestanejo oba preizkusa, tvorijo odgovor na uporabnikovo poizvedbo.

WALRUS (Natsev in sodelavci, *WAveLet-based Retrieval of User-specified Scenes*) [36, 37] izhaja iz opažanja, da dajejo sistemi, kot je WBIIS, pogosto slabe rezultate, ker izračunajo transformiranko le za celo sliko. Če bi si ločeno shranjevali podatke o različnih delih slike, bi lažje zaznali primere, ko obstaja podobnost le med nekim delom poizvedovalne slike in nekim (po možnosti ne istoležnim) delom določene slike iz baze. Transformiranka celotne slike se namreč kar precej spremeni, če se na sliki na primer premakne nek večji predmet; če bi imeli transformiranke različnih (in prekrivajočih se) manjših delov slike, pa bi kakšna od njih lahko še vedno opisovala tisti predmet na njegovem novem položaju. Natsev in sodelavci so pokazali, da lahko pri sliki velikosti $W \times H$ na precej učinkovit način izračunamo transformiranke vseh oken velikosti $2^n \times 2^n$ (teh oken je $(W - 2^n + 1) \times (H - 2^n + 1)$) z neke vrste dinamičnim programiranjem; vsako tako transformiranko lahko namreč sestavimo s kombiniranjem transformirank štirih oken velikosti $2^{n-1} \times 2^{n-1}$, na katera lahko razrežemo vsako okno velikosti $2^n \times 2^n$. Zato lahko začnemo pri transformirankah oken velikosti 2×2, z njihovo pomočjo izračunamo transformiranke oken velikosti 4×4 in tako naprej.

Od vsake transformiranke si zapomnimo na primer njeno zgornjo levo podmatriko velikosti 2×2; ker imamo tri barvne kanale (uporabili so prostor YCC), dobimo s tem nek dvanajstrazsežni vektor. Iz ene slike s tem dobimo več tisoč takšnih vektorjev; bilo bi nepraktično, če bi hoteli vse shraniti ali pa vse upoštevati pri primerjanju s poizvedovalno sliko. Zato jih združimo v skupine ali gruče (*clusters*); pri vsaki skupini si zapomnimo le centroid. Ker pa ta ne predstavlja več nekega preprostega kvadratnega okna, pač pa za njim stoji več vektorjev, ki predstavljajo različna okna, si shranimo tudi približen opis področja na sliki, ki ga pokrivajo vektorji iz te skupine (to je unija oken, ki jih predstavljajo ti vektorji).

V tem postopku lahko vidimo prizadevanja po neke vrste preprosti segmentaciji slike glede na barvo in teksturo. V isto skupino naj bi prišla okna s podobnimi glavnimi koeficienti valčnih transformirank, takšna okna pa naj bi si bila tudi po videzu približno podobna. Več o meri podobnosti med tako segmentiranimi slikami je v razdelku 2.4.

Tudi SIMPLIcity (*Semantics-sensitive Integrated Matching for Picture LIbraries*) [53, 54] uporablja valčno transformacijo pri segmentaciji slik. Sliko (v barvnem prostoru LUV) razdeli na okenca velikosti 4×4 in na vsakem oknu izvede Haarovo transformacijo. Zapomnimo si povprečje celega okna pri vsakem od treh barvnih kanalov, pri kanalu L (svetlost) pa še energije visokofrekvenčnih pasov (transformiranka okna 4×4 je namreč sestavljena iz štirih podmatrik velikosti 2×2; zgornja desna in obe spodnji se nanašajo na višje frekvenčne komponente kot zgornja leva; pri vsaki od teh treh podmatrik si zapomnimo koren povprečja kvadratov vseh štirih elementov). Tako vsako okno opišemo s šestrazsežnim vektorjem. Te vektorje združimo v skupine, na primer s postopkom k-povprečij (k-*means,* znan tudi kot „metoda voditeljev“). Okenca, ki so pristala v isti skupini, tvorijo eno regijo (regija je torej lahko sestavljena iz več nepovezanih delov). Opis regije je kar centroid vseh vektorjev v njej.

2.4 Mere podobnosti ob segmentaciji slik

Ko sliko, običajno z neko vrsto segmentacije, razdelimo na več regij in pripravimo nek opis za vsako od njih, je običajno dokaj jasno, kako bomo lahko ugotavljali podobnost med dvema regijama. Opisi regij so običajno vektorji z različnimi številskimi parametri, na primer koeficienti valčnih transformirank (razd. 2.3.4), in za mero razdalje (različnosti) med regijami lahko uporabimo evklidsko ali kakšno sorodno razdaljo med njihovimi vektorji. Manj očitno pa je, kako naj bi na podlagi podobnosti med regijami določili podobnost med slikami.

WALRUS šteje dve regiji za podobni, če se njuna vektorja po max-normi razlikujeta za manj kot nek ε (slednji je parameter, podan skupaj s poizvedbo). Recimo, da smo za določeno sliko B iz baze in poizvedovalno sliko Q našli naslednje pare $\langle$regija poizvedovalne slike, regija slike iz baze$\rangle$: $\langle Q_1, B_1\rangle, \ldots, \langle Q_n, B_n\rangle$. Za mero podobnosti med slikama naj bi vzeli delež površine, ki jo na njiju skupaj pokrivajo te regije:

$$\textit{podobnost}\,(Q, B) = \frac{A(\cup_i Q_i) + A(\cup_i B_i)}{A(Q) + A(B)}. \tag{2.2}$$

(Pri tem pomeni $A(\cdot)$ površino območja v oklepajih.) Neugodno je, da se lahko ista regija ene slike pojavlja v paru z več regijami druge slike; lahko bi poskušali izbrati tako neko podmnožico teh parov, pri kateri se nobena regija ne pojavlja v več kot enem paru, vendar se izkaže, da je problem, kako izbrati takšno podmnožico in pri tem maksimizirati mero (2.2), NP-težak. Natsev in sodelavci [36] so zato predlagali še neko požrešno hevristiko za izbor podmnožice parov, na koncu pa so uporabili kar vse pare in se niso menili za morebitno večkratno nastopanje iste regije v različnih parih.

V vsakem primeru pa je slabost tako zastavljene mere podobnosti ta, da lahko neka majhna regija z ene slike nastopa v paru s kakšno veliko regijo druge slike (ali pa z več takimi regijami) in s tem zelo močno prispeva k meri podobnosti. To utegne biti sicer včasih celo koristno, npr. če je nek predmet na eni sliki majhen, na drugi pa prikazan od blizu in zato povečan čez celo sliko; vendar pa je verjetno pogosteje bolj škodljivo kot koristno.

Takim težavam se skuša izogniti postopek „integriranega primerjanja regij“ (*integrated region matching,* IRM) [35], ki so ga uporabili v sistemu SIMPLIcity(gl. razd.2.3.4). Naj bo d_{ij} razdalja med i-to regijo prve slike in j-to regijo druge slike. Za razdaljo med slikama kot celotama bi uporabili obteženo vsoto:

$$d(Q,B) = \sum_i \sum_j s_{ij} d_{ij}, \tag{2.3}$$

naloga postopka IRM pa je pripraviti uteži s_{ij}. Predpostavimo, da si znamo za vsako regijo vnaprej izbrati njen skupni prispevek k ujemanju med slikama, torej števila $q_i = \sum_j s_{ij}$ in $p_j = \sum_i s_{ij}$. Uteži q_i in p_j so lahko na primer sorazmerne s površino regij, lahko pa imajo regije na robovih slike manjšo težo kot ostale, ker so zanimivi prizori ponavadi bolj pri sredini slike. Te uteži naj bi normalizirali, tako da bi bilo $\sum_i q_i = \sum_j p_j = 1$.

IRM pravzaprav na požrešen način prenaša težo regij na uteži s_{ij}. Najprej postavi vse s_{ij} na 0. Nato pa v vsakem koraku izbere med vsemi pari (I,J), pri katerih je

$$\textstyle\sum_j s_{Ij} < q_I \qquad \text{in} \qquad \sum_i s_{iJ} < p_J, \tag{2.4}$$

tistega z najmanjšo d_{IJ}. Uteži s_{IJ} zdaj dodamo toliko teže, kolikor je mogoče, preden se katera od neenačb (2.4) spremeni v enačbo: torej s_{IJ} povečamo za $\min\{q_I - \sum_j s_{Ij}, p_J - \sum_i s_{iJ}\}$. To ponavljamo, dokler je še kje kak tak par regij, ki še nista porabili vse svoje teže.

To torej pomeni, da imajo prednost pri dodeljevanju uteži s_{ij} tisti pari regij (i,j), ki so si bolj podobni, ko pa so take regije razdale že vso težo, kar so je mogle (ko je torej $\sum_j s_{ij} = q_i$ ali pa $\sum_i s_{ij} = p_j$), pridejo na vrsto tudi ostale.

Tak pristop k obteževanju podobnosti med regijami omogoča, da se pojavlja ista regija v paru z več drugimi, vendar bodo lahko ti pari na podobnost med slikama vplivali le, če je ta regija dovolj velika. Ne bo pa se moglo zgoditi, da bi neka majhna regija zaradi svoje podobnosti z neko veliko regijo druge slike močno vplivala na mero podobnosti med slikama.

2.5 Barvni prostori

Ko imamo opravka z barvnimi slikami, predstavimo vsako barvo običajno s tremi števili. Barve torej preslikamo v nek trorazsežen prostor, ki mu zato pravimo tudi barvni prostor (*color space*) [15, 13. pogl.] [6, razd. 2.1]. V rabi je več različnih barvnih prostorov, od katerih ima vsak kakšne prednosti in pomanjkljivosti. Tudi za potrebe pregledovanja slikovnih zbirk je lahko en prostor primernejši od drugega; če hočemo na primer s številom opisati, koliko sta si dve barvi različni, bi lahko uporabili razdaljo med njunima vektorjema, ta pa je lahko v različnih barvnih prostorih zelo različna.

Barvni prostor RGB opiše vsako barvo kot vsoto rdeče, zelene in modre komponente (vsaka je z intervala $[0,1]$); to se dobro ujema s tvorbo barv pri zaslonih s katodno cevjo, je pa pogosto neugodno za uporabnika, saj si mnogi ljudje težko predstavljajo „koordinate“ določene barve v tem prostoru. Prijaznejši v tem smislu naj bi bil prostor HSV, v katerem je vsaka barva opisana s svojim temeljnim odtenkom, zasičenostjo in svetlostjo. Temeljni odtenki (*hues*) so razporejeni v barvni krog in navedemo v bistvu kót željenega odtenka. Zasičenost (*saturation*) je 0 pri odtenkih sive, 1 pa pri povsem živih barvah. Svetlost ali „vrednost“ (*lightness, value*) pa je 0 pri črni barvi in 1 pri povsem svetli barvi (kolikor ji pač dopuščata temeljni odtenek in zasičenost).

Če hočemo z razdaljami med barvnimi vektorji meriti različnost med barvami, sta oba prostora do neke mere neugodna. Pri HSV imata na primer dve temni barvi lahko zelo različna temeljna odtenka (različni vrednosti H), torej bi bila tudi evklidska razdalja med tema vektorjema velika, čeprav sta si po videzu takšni barvi precej podobni; poleg tega vrednosti H tvorijo krog in bi morali upoštevati, da sta si odtenka pod kotoma 0° in 359° zelo podobna, čeprav se številki tako razlikujeta. Morali bi torej definirati razmeroma zapletene mere razdalje med vektorji (česar pa v literaturi običajno ne počnejo). Kljub temu se tako RGB kot HSV pogosto uporablja, prvega na primer v sistemu QBIC [16, 27], HSV pa v VisualSEEk [47]. Huang in sodelavci so pri svojih poskusih s korelogrami ugotovili, da jim dajeta oba prostora približno enake rezultate [24, str. 222b].

Obstaja več poskusov, da bi s preprosto transformacijo prostora RGB prišli do prostora, v katerem se razdalja med vektorji bolje ujema s človekovim občutkom za različnost barv. Take so na primer „nasprotne barvne osi“ (*opponent color axes*), ki naj bi se dobro ujemale tudi z delovanjem človekovega vidnega sistema [46, str. 16b]:

$$\begin{aligned} rg &\leftarrow r - g \\ by &\leftarrow 2b - r - g \\ wb &\leftarrow r + g + b \end{aligned}$$

Zadnja komponenta naj bi izražala svetlost barve, ostali dve pa kontraste oz. temeljni odtenek. Podoben barvni prostor so uporabili v sistemu WBIIS [57]:

$$\begin{aligned} c_1 &\leftarrow (r + g + b)/3 \\ c_2 &\leftarrow (r + (1 - b))/2 \\ c_3 &\leftarrow (r + 2(1 - g) + b)/4 \end{aligned}$$

S spremembo koordinatnih osi lahko iz prostora RGB preidemo tudi v prostor YIQ, ki se uporablja tudi pri prenosu televizijskih signalov (NTSC). Tega so uporabili na primer Jacobs in sodelavci [28].

Primeri prostorov, v katerih naj bi se razdalje med barvami kot vektorji še bolje ujemale z različnostjo med barvami, kot jo zaznava človek, so YCC, $L^*u^*v^*$ in $L^*a^*b^*$ („perceptualno uniformni“ barvni prostori). Preslikava iz RGB v te prostore je nelinearna. YCC so na primer uporabili Natsev in sod. [36] (in ugotovili, da je za njihove potrebe veliko primernejši od RGB), $L^*u^*v^*$ pa Wang in sod. [54]. Ker pa se tudi tu pojavljajo odstopanja od perceptualne uniformnosti, obstajajo tudi kompleksnejše metrike za primerjanje barv, npr. CMC in CIE94.

2.6 Vrednotenje postopkov preiskovanja slikovnih zbirk

Pri poizvedovanju po slikovnih zbirkah se pričakuje predvsem, da program v odgovor na določeno poizvedovalno sliko poišče v bazi nekaj slik, ki so ji podobne. Pri tem je sicer zaželjeno, ne pa nujno potrebno, da vrne poleg njih še čim manj nepodobnih in da se podobne pojavijo višje v seznamu zadetkov kot nepodobne: uporabnik lahko razmeroma hitro pregleda na primer nekaj deset slik in med njimi opazi tiste, ki ga zanimajo; glavno je, da mu ni treba pregledovati cele zbirke, ki ima lahko več tisoč slik. Zato si mnogi postopki s tega področja prizadevajo predvsem, da bi našli vse relevantne slike, ne moti pa jih včasih precejšnje število lažnih zadetkov [19, str. 78b].

V literaturi vrednotijo takšne postopke s kriteriji, kot jih poznamo na področju dostopanja do informacij (*information retrieval*). To sta predvsem natančnost (*precision;* kolikšen delež dokumentov, ki jih je program označil kot relevantne glede na uporabnikovo poizvedbo, je res relevantnih) in priklic (*recall;* kolikšen delež vseh relevantnih dokumentov je program označil kot relevantne). Priklic je pri slikovnih zbirkah težko določiti, saj bi morali ročno pregledati vse slike in ugotoviti, katere so zares relevantne glede na uporabnikovo poizvedbo; zato priklica pri delu s slikovnimi zbirkami običajno ne ugotavljajo.

Večina algoritmov za iskanje po slikovnih zbirkah v nekem smislu oceni vse slike iz zbirke in vsaki pripiše neko oceno relevantnosti oz. podobnosti s poizvedovalno sliko. Za izračun natančnosti je treba potegniti nek prag, gledati le prvih N najbolje ocenjenih slik in ugotoviti, koliko je med njimi res relevantnih. Dobljeni delež je seveda odvisen od tega, kakšen N vzamemo; ker je relevantnih le nekaj slik, bomo z večanjem N-ja nujno prej ali slej začeli dobivati vse manjše natančnosti. Za N običajno vzamejo število slik, ki naj bi jih bil uporabnik ob poizvedbi še pripravljen pregledati; to je običajno nekaj deset, največ nekaj sto slik.

Še en priljubljen način ocenjevanja rezultatov poizvedbe temelji na povprečni uvrstitvi [16]: ročno se poišče v bazi nekaj (recimo K) slik, ki naj bi bile najbolj podobne poizvedovalni; ko program razvrsti slike po relevantnosti, bi bilo idealno, če bi se tistih K pojavilo na prvih mestih; njihova povprečna uvrstitev v tem vrstnem redu bi bila v tem primeru $(0 + 1 + \ldots + (K - 1))/K$ (vrstni red štejejo od 0 naprej); dejanska povprečna uvrstitev (*average rank*) relevantnih slik v vrstnem redu pa je pač večja ali enaka tej idealni vrednosti. Na koncu vzamemo razmerje med dejanskim in idealnim vrstnim redom; manjše ko je, bolje je program zadel relevantne slike.

Poseben primer tega pristopa je tudi, da si izberejo le eno sliko, ki jo imajo za najpodobnejšo poizvedovalni, in gledajo, na katerem položaju v vrstnem redu se pojavi [42].

3

Metoda najbližjih sosedov

3.1 Uvod

Metoda najbližjih sosedov je ena od starejših metod na področjih strojnega učenja in prepoznavanja vzorcev. Uporabljala sta jo že E. Fix in J. L. Hodges v začetku petdesetih let. Pomemben primer zgodnje analize te metode je članek Coverja in Harta [11].

V osnovi deluje metoda najbližjih sosedov tako, da si zapomni vse učne primerke, ko pa mora klasificirati nek nov primerek, pregleda nekaj (recimo k) njemu najbližjih učnih primerkov (to so njegovi „najbližji sosedje“) in novi primerek uvrsti v tisti razred, kateremu pripada največ izmed teh najbližjih sosedov.

V ta namen je seveda potrebno definirati mero razdalje med primerki; šele potem vemo, kateri so si najbližji sosedje. Če so primerki opisani z nekimi vektorji, lahko uporabimo kar katero od vektorskih norm, npr. evklidsko razdaljo. Je pa pri tej metodi ugodno ravno to, da lahko deluje tudi s kakšnimi drugačnimi merami podobnosti in tudi v primerih, ko naši primerki sploh niso opisani z vektorji.

Če pregledujemo več najbližjih sosedov, jih lahko pri glasovanju o razredu tudi obtežimo: glas primerka i ima na primer težo $1/d_i$, če je d_i razdalja med učnim primerkom i in novim primerkom, ki bi ga radi klasificirali.

Na področju strojnega učenja pravijo tem metodam pogosto „učenje na podlagi primerkov“ (*instance-based learning*). Algoritem IB1, ki ga opisujejo Aha in sodelavci [1], je pravzaprav enak običajnemu postopku najbližjih sosedov, le da zaloge vrednosti atributov najprej normalizira in da definira mero razdalje tudi za nominalne in diskretne atribute.

3.2 Lastnosti

V nadaljevanju predpostavimo, da imamo primerke vendarle predstavljene z vektorji in da pare $\langle$vektor, oznaka razreda$\rangle$ zajemamo iz neodvisnih in enako porazdeljenih verjetnostnih spremenljivk z gostoto $p(\mathbf{x}, w_r)$. Razredi so označeni z $w_1, \ldots, w_c$. Robna gostota pri vektorju $\mathbf{x}$ je tako $p(\mathbf{x}) = \sum_{r=1}^{c} p(\mathbf{x}, w_r)$, verjetnost r-tega razreda pa $P(w_r) = \int p(\mathbf{x}, w_r) d\mathbf{x}$.

Konvergenca najbližjega soseda. Naj bodo $\mathbf{x}$ in $\mathbf{x}_i$, $i = 1, 2, \ldots$, neodvisne in enako porazdeljene verjetnostne spremenljivke. Naj bo $\mathbf{x}'_n$ najbližji sosed vrednosti $\mathbf{x}$ med vrednostmi $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Potem je mogoče pokazati, da konvergira $\mathbf{x}'_n$ proti $\mathbf{x}$ z verjetnostjo 1, ko gre k proti ∞.

O tem se lahko prepričamo z naslednjim neformalnim razmislekom: naj bo $p(\mathbf{x})$ gostota porazdelitve spremenljivk $\mathbf{x}$ in $\mathbf{x}_i$. Verjetnost, da je nek $\mathbf{x}_i$ vsaj za δ oddaljen od $\mathbf{x}$, je potem $1-q(\mathbf{x})$, pri čemer je $q(\mathbf{x})$ integral gostote $p(\mathbf{x})$ po krogli s središčem v $\mathbf{x}$ in polmerom δ. Potem je verjetnost, da ni nobeden od prvih k učnih vektorjev oddaljen od $\mathbf{x}$ za manj kot δ, enaka $(1 - q(\mathbf{x}))^k$. Ta verjetnost gre proti 0, ko gre k proti ∞, razen če imamo smolo in je $q(\mathbf{x}) = 0$; torej če je gostota verjetnosti v neki okolici točke $\mathbf{x}$ enaka 0. Toda prav ta lastnost pomeni, da je verjetnost, da ima naključno izbrani $\mathbf{x}$ to lastnost, enaka 0. Torej lahko rečemo, da gre (ko gre k proti ∞) verjetnost, da je najbližji sosed od $\mathbf{x}$ oddaljen za več kot δ, proti 0 z verjetnostjo 1. Formalen dokaz je v [11, razd. 5].

Verjetnost napake pri metodi najbližjega soseda. Gornji razmislek je pokazal, da je pri dovolj veliki učni množici najbližji sosed (recimo mu $\mathbf{x}'$) točke $\mathbf{x}$ gotovo zelo blizu te točke. Predpostavimo zato, da so verjetnosti $P(w_r|\mathbf{x}')$, da je najbližji sosed $\mathbf{x}'$ v določenem razredu r, enake verjetnostim $P(w_r|\mathbf{x})$, da je $\mathbf{x}$ sam v tem razredu.

Kakšna je verjetnost, da z napovedjo na podlagi najbližjega soseda zadenemo napačen razred? To je verjetnost, da je najbližji sosed v drugem razredu kot $\mathbf{x}$ sam:

$$e_{\mathit{1\text{-}NN}} = \sum_{r=1}^{c} P(w_r|\mathbf{x})[1 - P(w_r|\mathbf{x}')],$$

kar je v skladu s predpostavko enako

$$= \sum_{r=1}^{c} P(w_r|\mathbf{x})[1 - P(w_r|\mathbf{x})] = 1 - \sum_{c=1}^{k} P^2(w_r|\mathbf{x})$$

Po drugi strani pa bi na primer Bayesov klasifikator ocenil vse verjetnosti $P(w_r|\mathbf{x})$ in vedno predlagal razred r z največjo verjetnostjo v točki $\mathbf{x}$. Zmoti se torej le v primeru, če $\mathbf{x}$ ni iz tega razreda:

$$e_{Bayes} = 1 - \max_{r\in\{1,\ldots,c\}} P(w_r|\mathbf{x}).$$

Odnos med tema verjetnostma napake si lahko preprosto predstavljamo, če imamo le dva razreda. Označimo tedaj $p = P(w_1|\mathbf{x})$ in $1-p = P(w_2|\mathbf{x})$ ter predpostavimo, da je prvi razred verjetnejši (torej $p \geq 1/2$). Potem je

$$e_{\mathit{1\text{-}NN}} = 1 - p^2 - (1-p)^2 = 2p(1-p) \qquad \text{in} \qquad e_{Bayes} = 1 - p.$$

Očitno je torej $\qquad e_{Bayes} \leq e_{\mathit{1\text{-}NN}} \leq 2e_{Bayes}(1 - e_{Bayes}).$

Klasifikator z enim najbližjim sosedom ima vsaj tolikšno verjetnost napake kot Bayesov klasifikator, vendar ne več kot dvakrat tolikšno. Podobno je mogoče za c razredov pokazati, da je

$$e_{Bayes} \leq e_{\mathit{1\text{-}NN}} \leq \left(2 - \frac{c}{c-1} e_{Bayes}\right) e_{Bayes}.$$

Verjetnost napake pri več najbližjih sosedih. Verjetnost napake pri klasifikatorju, ki pregleda $k > 1$ najbližjih sosedov, lahko zapišemo na podoben način; če imamo c razredov z verjetnostmi $p_r \equiv P(w_r|\mathbf{x})$ za $r = 1, \ldots, p_c$, je

$$e_{k\text{-}NN} = \sum_{r=1}^{c} p_r \sum_{l=k/2}^{k} \binom{k}{l} p_r^{k-l}(1-p_r)^l;$$

pri dveh razredih z verjetnostma p in $1-p$ bi na primer klasifikator, ki gleda tri najbližje sosede, dosegel verjetnost napake

$$\begin{aligned} e_{3\text{-}NN} = p[3p(1-p)^2 + (1-p)^3] + (1-p)[3(1-p)p^2 + p^3] = \\ = p(1-p)[4p(1-p)+1]; \end{aligned}$$

pri tem doseže $4p(1-p)+1$ svojo največjo vrednost, 2, pri $p = 1/2$, drugod pa je manjši. Torej imamo

$$e_{3\text{-}NN} \le e_{1\text{-}NN}$$

in enakost velja le v primeru, če sta razreda enako verjetna. Tudi v splošnem je $e_{k\text{-}NN} \le e_{m\text{-}NN}$ za $k > m$.

Še ena primerjava z Bayesovim klasifikatorjem. Bayesov klasifikator poskuša za dani vektor $\mathbf{x}$ oceniti verjetnosti $P(w_r|\mathbf{x})$ po Bayesovi formuli $P(w_r|\mathbf{x}) = p(\mathbf{x}|w_r)P(w_r)/p(\mathbf{x})$ in nato napove razred $\arg\max_r P(w_r|\mathbf{x})$. Klasifikator, ki dela z najbližjimi sosedi, pa pravzaprav ravno tako napove razred $\arg\max_r P(w_r|\mathbf{x})$, le verjetnost $P(w_r|\mathbf{x})$ oceni drugače: pogleda k najbližjih sosedov vektorja $\mathbf{x}$ in za $P(w_r|\mathbf{x})$ vzame delež, koliko izmed teh sosedov je v r-tem razredu.

3.3 Izvedba v praksi

Pomembna lastnost metode najbližjih sosedov je, da se večina računske zahtevnosti zdaj s časa učenja (ki pravzaprav zajema predvsem shranjevanje učnih podatkov) prenese na čas klasifikacije novih primerkov (ko je treba novi primerek primerjati z vsemi učnimi, da poiščemo najbližje sosede). (Od tod tudi naziv „leno učenje“ — *lazy learning.*)

Zato so se že zgodaj pojavili razni predlogi, ki naj bi zmanjšali prostorsko in časovno zahtevnost klasifikacije z najbližjimi sosedi. Hart [20] je na primer predlagal zgoščeno pravilo najbližjih sosedov (*condensed nearest neighbor rule*): naj množica S na začetku vsebuje le prvi učni primerek; potem pregledujemo ostale učne primerke in pri vsakem pogledamo, če bi z uporabo navadne metode najbližjih sosedov na S kot učni množici pravilno napovedali razred tega primerka; če to ni res, ga dodamo v množico S. Ustavimo se, ko S bodisi vsebuje vse učne primerke ali pa ko opravimo nek prehod skozi vse preostale primerke, ne da bi katerega dodali v S. Pri klasificiranju novih primerkov uporabljamo le tako dobljeno množico S. Zamisel je torej v tem, da bo za uspešno klasifikacijo mogoče zadostovala že neka majhna množica S, ki ne vsebuje vseh učnih primerkov; tako bo klasifikator porabil manj pomnilnika, pa tudi manj časa, ker bo manj kandidatov za najbližje sosede.

Na isti zamisli temelji tudi postopek IB2 [1]. Aha in sod. ugotavljajo, da postane učenje zaradi krčenja množice shranjenih primerkov bolj občutljivo na šum v podatkih. Šumne primerke se na podlagi najbližjih sosedov pogosto klasificira narobe, zato jih večina pristane v množici S; le-ta pri zašumljenih učnih podatkih zato ostane precej velika in vsebuje nesorazmerno velik delež šumnih podatkov. Opisali so tudi robustnejši postopek IB3, ki za vsak primerek iz S vzdržuje podatek o tem, kolikokrat je glasoval za pravilni razred. Če se nek primerek prepogosto moti, ga odstranimo iz S; poleg tega pa za napoved razreda naslednjega učnega primerka (ko se odločamo, ali ga je treba dodati v S) upoštevamo le glasove najboljših primerkov iz S.

Če so naši primerki opisani z vektorji, jih lahko poskušamo organizirati tudi v kakšno prostorsko indeksno strukturo, ki olajša iskanje najbližjih sosedov. Take strukture so na primer k-d-drevo, R-drevo, R^*-drevo, *SS*-drevo, *SR*-drevo, M-drevo in tako naprej. Vsem je skupna naslednja lastnost: vektorji so shranjeni v listih drevesa, v notranjih vozliščih pa so kazalci na otroke in ob vsakem kazalcu še podatek o ovojnici vseh vektorjev v poddrevesu, ki se začenja pri tistem otroku. Ovojnica je lahko hiperkvader (R-drevo, R^*-drevo), hiperkrogla (*SS*-drevo), presek kvadra in krogle (*SR*-drevo), ipd.; v vsakem primeru naj bi šlo za čim manjše geometrijsko telo, ki vsebuje vse vektorje v tistem poddrevesu.

Iskanje najbližjih sosedov neke točke $\mathbf{x}$ lahko zdaj poteka po načelu „najprej najboljši“ [45, 26, 5]. Spodnji postopek predpostavlja, da znamo za vsako vozlišče V določiti spodnjo in zgornjo mejo za razdaljo med $\mathbf{x}$ in vektorji iz poddrevesa, ki se začenja pri V. Recimo tema mejama MINDIST$(\mathbf{x}, V)$ in MAXDIST$(\mathbf{x}, V)$; če imamo za V znano kvadrasto ovojnico (*bounding box*), sta dobra kandidata za ti dve oceni razdalji med $\mathbf{x}$ in njemu najbližjim oz. najbolj oddaljenim ogliščem te ovojnice.

ISKANJE NAJBLIŽJIH SOSEDOV(točka $\mathbf{x}$, drevo T)
  Ustanovimo prioritetno vrsto Q in vanjo dodamo koren drevesa T.
  Vzdržujemo podatek o kandidatu $\hat{\mathbf{x}}$ za najbližjega soseda točke $\mathbf{x}$.
    Na začetku kandidata še nimamo in si mislimo $d(\mathbf{x}, \hat{\mathbf{x}}) = \infty$.
  Ponavljamo, dokler Q ni prazna:
    Vzamemo iz Q vozlišče V z najmanjšo MINDIST$(\mathbf{x}, V)$.
    Če je V list, pregledamo vektorje, ki so shranjeni v njem;
      med njimi je mogoče kak dober kandidat za najbližjega soseda,
        točke $\mathbf{x}$, torej nek $\mathbf{y}$ z $d(\mathbf{x}, \mathbf{y}) < d(\mathbf{x}, \hat{\mathbf{x}})$.
      Če smo našli tak $\mathbf{y}$, naj bo $\hat{\mathbf{x}} \leftarrow \mathbf{y}$ in
        odstranimo iz Q vsa vozlišča U z MINDIST$(\mathbf{x}, U) > d(\mathbf{x}, \hat{\mathbf{x}})$. (†)
    Če je V notranje vozlišče, dodamo vse njegove otroke U v vrsto Q,
        razen tistih, za katere je MINDIST$(\mathbf{x}, U) > d(\mathbf{x}, \hat{\mathbf{x}})$. (†)
      Naj bo D največja vrednost MAXDIST$(\mathbf{x}, U)$ med dodanimi otroki.
      Nato iz vrste Q odstranimo vsa tista vozlišča W,
        za katera je MINDIST$(\mathbf{x}, W) > D$. (‡)

Opisani postopek torej uporablja MINDIST kot oceno za to, kako bližnje sosede utegne najti v nekem poddrevesu, in si prizadeva najprej obiskati obetavnejša vozlišča drevesa. Tiste, za katere je jasno, da v njih ne bo mogoče najti ničesar boljšega od doslej najboljšega kandidata (†), ali pa ničesar boljšega od tega, kar bomo zagotovo še našli v nekem drugem poddrevesu (‡), pa čim prej odstranimo iz vrste, da jih ne bi po nepotrebnem pregledovali.

Berchtold in sodelavci [5] so pokazali, da ta algoritem ne obišče nobenega vozlišča V, za katerega je $\textsc{mindist}(\mathbf{x}, V)$ večja od razdalje med $\mathbf{x}$ in njemu najbližjim sosedom (lahko bi torej rekli, da je v nekem smislu optimalen, saj o vozliščih z manjšo $\textsc{mindist}$ res ne moremo sklepati, da ne vsebujejo kakšnega še bližjega soseda, in je torej razumno, da jih obiščemo in pregledamo); predpostavimo le, da je, če je U otrok vozlišča V, zagotovo $\textsc{mindist}(\mathbf{x}, V) \leq \textsc{mindist}(\mathbf{x}, U)$.

Najprej opazimo, da glavna zanka opazovanega algoritma obravnava vozlišča V po nepadajoči $\textsc{mindist}(\mathbf{x}, V)$. Da to dokažemo, moramo le zagotoviti, da v vrsti, ko iz nje vzamemo V, ni in ne bo nobenega U z $\textsc{mindist}(\mathbf{x}, U) < \textsc{mindist}(\mathbf{x}, V)$. Res, če bi bil tak U v vrsti že od prej, bi zdaj vzeli iz vrste njega, ne pa V. Če pa bi hoteli nek tak U kasneje dodati v vrsto, bi moral biti to potomec enega od vozlišč, ki so v vrsti zdaj, recimo nekega W; toda $\textsc{mindist}(\mathbf{x}, V) \leq \textsc{mindist}(\mathbf{x}, W)$ (ker smo zdajle vzeli iz vrste V, ne pa W) in $\textsc{mindist}(\mathbf{x}, W) \leq \textsc{mindist}(\mathbf{x}, U)$ (ker je U potomec vozlišča W, zato razdalja od $\mathbf{x}$ do najbližjega vektorja iz U gotovo ni manjša kot do najbližjega vektorja iz W). Prišli smo v protislovje, torej takega U-ja sploh ni.

Zdaj pa naj bo D razdalja med $\mathbf{x}$ in njemu najbližjim sosedom $\mathbf{y}$ in recimo, da naš algoritem v nekem trenutku obišče (torej: vzame iz vrste) neko vozlišče V z $\textsc{mindist}(\mathbf{x}, V) > D$. Vozlišče (recimo mu W), v katerem je $\mathbf{y}$, mora imeti $\textsc{mindist}(\mathbf{x}, W) \leq D$, torej smo ga morali vzeti iz vrste že pred V; toda to pomeni, da smo doslej že naleteli na $\mathbf{y}$ in tedaj smo morali V (ali pa kakšnega njegovega prednika) odstraniti iz vrste ali pa ga kasneje vsaj nismo dodali vanjo, ker ima pač $\textsc{mindist}(\mathbf{x}, V)$ večji od $d(\mathbf{x}, \mathbf{y})$. (Da smo na W morali naleteti, sledi iz dejstva, da imajo tako on kot vsi njegovi predniki $\textsc{mindist}(\mathbf{x}, W) \leq D$ in torej ni mogoče, da bi se na podlagi vrstic (†) ali (‡) kateremu od teh vozlišč izognili.) Torej ni mogoče, da bi obiskali kak tak V.

Isti algoritem lahko posplošimo tudi za iskanje k najbližjih sosedov.

4

Metoda podpornih vektorjev

4.1 Uvod

Metoda podpornih vektorjev (*support vector machine,* SVM) je pravzaprav bolj paradigma ali skupek metod kot pa en sam konkreten postopek. Te metode temeljijo na teoretični podlagi, ki so jo že od šestdesetih let naprej razvijali Vladimir Vapnik in drugi (statistična teorija učenja), pravi razcvet pa doživljajo od srede devetdesetih let naprej. Dobro so se obnesle pri številnih realnih problemih. Omogočajo delo z zelo bogatimi prostori hipotez, hkrati pa se lahko razmeroma dobro izogibajo pretiranemu prilagajanju. V tem poglavju se bomo osredotočili na uporabo metode podpornih vektorjev pri klasifikaciji, drugače pa jo je moč uporabiti tudi za regresijo in še za nekatere druge probleme.

4.1.1 Razmejevanje razredov z ravnino

Predpostavimo, da so naši učni podatki pari $(\mathbf{x}_i, y_i)$, $i \in \{1, \ldots, l\}$, pri čemer je $\mathbf{x}_i \in \mathbb{R}^d$ nek d-dimenzionalni realni vektor, $y_i \in \{-1, +1\}$ pa oznaka razreda, v katerega ta vektor spada. (Pozneje bomo videli, kako lahko to metodo uporabimo tudi za primere, ko imamo več kot dva razreda.) Prostor $\mathbb{R}^d$ bomo poskušali razdeliti z neko hiperravnino in si pri tem prizadevali, da bi bili predstavniki enega razreda na eni, predstavniki drugega pa na drugi strani te hiperravnine. (V nadaljevanju recimo namesto hiperravnina raje kar ravnina.) Takšna ravnina je enolično določena s svojo normalo $\mathbf{w} = (w_1, \ldots, w_d)$ in neko konstanto ali pragom b (*bias*); njena enačba je potem

$$w_1 x_1 + \ldots + w_d x_d + b = 0 \qquad \text{ali} \qquad \langle \mathbf{w}, \mathbf{x} \rangle + b = 0,$$

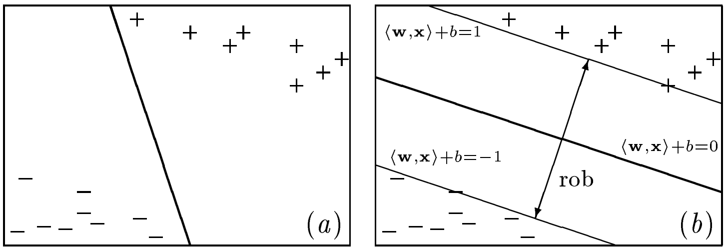
pri čemer je $\langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{w}^T \mathbf{x} = \sum_{i=1}^d w_i x_i$ skalarni produkt vektorjev $\mathbf{w}$ in $\mathbf{x}$.

Ko bi takšno ravnino uporabljali kot klasifikator, bi vektorje, ki ležijo na eni strani te ravnine, uvrstili v razred $+1$, tiste z nasprotne pa v razred -1. Primeren klasifikator bi bil torej

$$f(\mathbf{x}) = \operatorname{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b).$$

Vektorji, ki ležijo na tisti strani ravnine, proti kateri kaže normala $\mathbf{w}$, dajejo pozitivne vrednosti $\langle \mathbf{w}, \mathbf{x} \rangle + b$, tisti z nasprotne strani pa negativne.

Problem učenja smo s tem prevedli na vprašanje, kako za dano učno množico določiti primerno normalo $\mathbf{w}$ in konstanto b, da bo imel dobljeni klasifikator čim višjo točnost. Neformalno si lahko poskušamo na to vprašanje odgovoriti s takšnim razmislekom: načeloma si ne želimo, da bi kateri od učnih (ali kasneje testnih, a na to težje vplivamo) vektorjev ležal na ravnini, saj bi tedaj dobili $\langle \mathbf{w}, \mathbf{x}_i \rangle + b = 0$ in ne bi vedeli, kam ga uvrstiti. Nasprotno, želimo si, da bi učni vektorji ležali na pravi strani ravnine in čim dlje od nje, saj nam to zagotavlja, da bodo vrednosti $\langle \mathbf{w}, \mathbf{x}_i \rangle + b$ po absolutni vrednosti velike in se ne bo treba spraševati, ali ni predznak $\operatorname{sgn}(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$ le posledica šuma ali naključne napake. Tak razmislek nam na primer pravi, da je ravnina (pravzaprav premica, ker je $d = 2$) s slike (4.1a) gotovo manj posrečena izbira kot tista s slike (4.1b).



Slika 4.1: Razmejevanje učnih vektorjev z ravnino.

Poleg tega pa za takšnim razmislekom stojijo tudi ugotovitve statistične teorije učenja. Tam obstaja več izrekov, ki podajajo zgornje meje verjetnosti napake klasifikatorja, in pri katerih so te zgornje meje tem nižje, čim dlje stran od ravnine ležijo učni vektorji.

Na obeh straneh ravnine si torej lahko predstavljamo nekakšno „nikogaršnje ozemlje“, ki sega na vsaki strani do najbližjega učnega vektorja. Temu ozemlju (ali pa njegovi širini) pravimo „rob“ (*margin*). V prejšnjih odstavkih smo torej ugotovili, da želimo najti ravnino s čim širšim robom.

Smiselno je zahtevati, naj teče ravnina $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ po sredi roba. V tem primeru bi, če bi jo zamaknili v eno ali drugo smer do najbližjega učnega vektorja, dobili ravnini $\langle \mathbf{w}, \mathbf{x} \rangle + b = \gamma$ in $\langle \mathbf{w}, \mathbf{x} \rangle + b = -\gamma$ za neko konstanto γ. Dosedanjo zahtevo, naj za pozitivne učne vektorje $\mathbf{x}$ velja $\langle \mathbf{w}, \mathbf{x} \rangle + b > 0$, za negativne pa $\langle \mathbf{w}, \mathbf{x} \rangle + b < 0$, moramo zdaj popraviti: za pozitivne naj velja $\langle \mathbf{w}, \mathbf{x} \rangle + b > \gamma$, za negativne pa $\langle \mathbf{w}, \mathbf{x} \rangle + b < -\gamma$.

Parameter γ je v tesni zvezi s širino roba. Naj bo $\mathbf{x}_1$ neka točka na ravnini $\langle \mathbf{w}, \mathbf{x} \rangle + b = -\gamma$. Širina roba je enaka razdalji od $\mathbf{x}_1$ do njej najbližje točke (recimo ji $\mathbf{x}_2$) na ravnini $\langle \mathbf{w}, \mathbf{x} \rangle + b = \gamma$. Najbližjo točko bomo našli, če gremo v smeri pravokotno na ti dve ravnini, torej v smeri $\mathbf{w}$. Tako lahko zapišemo $\mathbf{x}_2 = \mathbf{x}_1 + \lambda \mathbf{w}$. Ker je $\mathbf{x}_1$ na prvi ravnini, $\mathbf{x}_2$ pa na drugi, imamo še pogoja $\langle \mathbf{w}, \mathbf{x}_1 \rangle + b = -\gamma$ in $\langle \mathbf{w}, \mathbf{x}_2 \rangle + b = \gamma$. Če ju odštejemo, dobimo $\langle \mathbf{w}, \mathbf{x}_2 - \mathbf{x}_1 \rangle = 2\gamma$, torej $\lambda \langle \mathbf{w}, \mathbf{w} \rangle = 2\gamma$ in $\lambda = 2\gamma / ||\mathbf{w}||^2$. Širina roba je tako $||\mathbf{x}_2 - \mathbf{x}_1|| = ||\lambda \mathbf{w}|| = 2\gamma / ||\mathbf{w}||$. Lahko bi rekli, da je γ „algebraična“ širina roba, ker se meri z vrednostjo funkcije $\langle \mathbf{w}, \mathbf{x} \rangle + b$; $2\gamma / ||\mathbf{w}||$ pa je „geometrijska“ ali „evklidska“ širina roba, ker je to res evklidska razdalja med ravninama na eni in drugi strani roba.

Vendar pa je v dosedanji formulaciji pravzaprav ena spremenljivka preveč. Če enačbo ravnine $\sum_{i=1}^d w_i x_i + b = 0$ delimo s poljubno pozitivno konstanto c, ostane to enačba iste ravnine, le da ima zdaj normalo $(1/c)\mathbf{w}$ in prag b/c (pri negativnem c bi se spremenila njena orientacija — normala bi kazala v nasprotno smer). Če bi enačbo naše ravnine na primer delili z γ, bi se gornje neenačbe poenostavile v

$$\begin{array}{rcll} \langle \mathbf{w}', \mathbf{x}_i \rangle + b' & \geq & 1, & \text{če je } y_i = +1; \\ \langle \mathbf{w}', \mathbf{x}_i \rangle + b' & \leq & -1, & \text{če je } y_i = -1. \end{array} \tag{4.1}$$

Pri tem smo pisali $\mathbf{w}' = (1/\gamma)\mathbf{w}$ in $b' = b/\gamma$. Če obstaja neka ravnina $(\mathbf{w}, b)$ z robom $2\gamma > 0$, bo torej obstajala tudi neka ravnina (namreč $(\gamma^{-1}\mathbf{w}, b/\gamma)$) z algebraičnim robom 1. Zato se lahko omejimo le na iskanje ravnin z algebraičnim robom $\gamma = 1$. Geometrijski rob take ravnine je $2\gamma/||\mathbf{w}|| = 2/||\mathbf{w}||$ in je torej tem širši, čim krajša je normala ravnine.

Obe vrsti pogojev iz (4.1) lahko združimo v eno samo obliko, če ju pomnožimo z y_i. Zdaj lahko iskanje najprimernejše ravnine zastavimo kot naslednji optimizacijski problem:

$$\begin{array}{rl} \text{minimiziraj} & P(\mathbf{w}, b) = \frac{1}{2}||\mathbf{w}||^2 = \langle \mathbf{w}, \mathbf{w} \rangle = \sum_{j=1}^{d} w_j^2 \\ \text{pri pogojih} & \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}, \\ & \forall i \in \{1, \ldots, l\} : y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1. \end{array}$$

Minimiziramo torej dolžino normale, ker vemo, da nam bo to (ob upoštevanju neenačb) dalo najširši rob. Namesto $||\mathbf{w}||$ gledamo raje $||\mathbf{w}||^2$, ker je problem tako matematično bolj obvladljiv. Koeficient $\frac{1}{2}$ je tu le zaradi lepšega nadaljevanja izpeljave. Pogojev $\mathbf{w} \in \mathbb{R}^d$ ter $b \in \mathbb{R}$ v nadaljevanju ne bomo več posebej navajali.

4.1.2 Prehod na dualni problem

Dobljeni problem lahko predelamo v obvladljivejšo obliko s pomočjo Lagrangeovih multiplikatorjev, dobro znanega prijema iz teorije optimizacije [4]. Pri nalogi oblike „minimiziraj $P(\mathbf{x})$ pri pogojih $\forall i : P_i(\mathbf{x}) \geq 0$“ uvedemo po en nenegativni multiplikator α_i za vsak i in definiramo Lagrangeovo funkcijo $L(\mathbf{x}, \boldsymbol{\alpha}) \equiv P(\mathbf{x}) - \sum_i \alpha_i P_i(\mathbf{x})$. Naloga „minimiziraj $P^*(\mathbf{x}) \equiv \max_{\boldsymbol{\alpha} \geq \mathbf{0}} L(\mathbf{x}, \boldsymbol{\alpha})$ pri pogojih $\forall i : \alpha_i \geq 0$“ je potem enakovredna prvotni nalogi, naloga „maksimiziraj $Q(\boldsymbol{\alpha}) \equiv \min_{\mathbf{x} \in \mathbb{R}^d} L(\mathbf{x}, \boldsymbol{\alpha})$ pri pogojih $\forall i : \alpha_i \geq 0$“ pa ji je pod določenimi pogoji dualna (torej ima Q pri optimalni $\boldsymbol{\alpha}$ enako vrednost kot P pri optimalnem $\mathbf{x}$; L pa ima za ta $\mathbf{x}$ in $\boldsymbol{\alpha}$ v točki $(\mathbf{x}, \boldsymbol{\alpha})$ sedlo). Korist takšnega pristopa je v tem, da je dualna naloga pogosto preprostejša od prvotne.

Lagrangeova funkcija našega problema bi bila:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \tfrac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle - \sum_{i=1}^{l} \alpha_i (y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1).$$

Pri tem je $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_l)$ vektor vseh l Lagrangeovih multiplikatorjev. Dualna naloga bo oblike

$$\begin{array}{rl} \text{maksimiziraj} & Q(\alpha) \equiv \min_{\mathbf{w}, b}\{L(\mathbf{w}, b, \boldsymbol{\alpha})\} \\ \text{pri pogojih} & \forall i \in \{1, \ldots, l\} : \alpha_i \geq 0. \end{array}$$

Da določimo Q, moramo poiskati minimum funkcije L po spremenljivkah $\mathbf{w}$ in b. Ta nastopi v točki, kjer so parcialni odvodi L-ja po vseh w_j in po b enaki 0.

$$\partial L / \partial w_j = w_j - \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_{ij} = 0, \qquad \partial L / \partial b = -\sum_{i=1}^{l} \alpha_i y_i = 0.$$

V minimumu torej velja

$$\mathbf{w} = \textstyle\sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \quad \text{in} \quad \sum_{i=1}^{l} \alpha_i y_i = 0. \tag{4.2}$$

Če to vstavimo v funkcijo L, dobimo

$$\begin{aligned} Q(\boldsymbol{\alpha}) &= \frac{1}{2}\left\langle \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i, \sum_{j=1}^{l} \alpha_j y_j \mathbf{x}_j \right\rangle - \sum_{i=1}^{l} \alpha_i \left(y_i \left(\left\langle \sum_{j=1}^{l} \alpha_j y_j \mathbf{x}_j, \mathbf{x}_i \right\rangle + b \right) - 1 \right) = \\ &= -\frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \sum_{i=1}^{l} \alpha_i = -\frac{1}{2} \boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha} + \mathbf{1}^T \boldsymbol{\alpha}, \end{aligned}$$

če definiramo $\mathbf{Q}$ kot matriko z elementi $q_{ij} = y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$, vektor $\mathbf{1}$ pa kot vektor samih enic $(1, 1, \ldots, 1)$. Namesto da maksimiziramo $Q(\boldsymbol{\alpha})$, lahko tudi minimiziramo $-Q(\boldsymbol{\alpha})$. S tem smo prišli do naslednjega problema:

$$\begin{array}{ll} \text{minimiziraj} & Q(\boldsymbol{\alpha}) = \frac{1}{2} \alpha^T \mathbf{Q} \alpha - \mathbf{1}^T \boldsymbol{\alpha} \\ \text{pri pogojih} & \forall i \in \{1, \ldots, l\} : \alpha_i \geq 0, \\ & \sum_{i=1}^{l} \alpha_i y_i = 0. \end{array}$$

Ker je bil začetni problem konveksen, nam teorija optimizacije zagotavlja, da je novi problem dualen prvotnemu. To pomeni, da je vrednost funkcije Q (pravzaprav $-Q$, ker smo jo vmes negirali) pri optimalnem α enaka vrednosti P pri optimalnih $\mathbf{w}$ in b, slednja pa lahko tudi izračunamo iz optimalne α po formulah (4.2). Če torej rešimo dualni problem, bomo iz njegove rešitve brez težav dobili tudi optimalno rešitev prvotnega problema.

Dobljeni dualni problem je v matematiki v splošnem znan kot kvadratno programiranje: kriterijska funkcija je kvadratni polinom l spremenljivk, množica dopustnih rešitev pa je določena s skupino linearnih neenačb. Za numerično reševanje takšnih problemov obstaja več postopkov, na primer različne oblike gradientnega spuščanja. Te postopke lahko uporabimo tudi pri učenju klasifikatorjev za metodo podpornih vektorjev.

4.1.3 Prehod na mehki rob

Doslej smo od našega klasifikatorja zahtevali, naj vse učne primere klasificira pravilno: za vsak i je moral $\mathbf{x}_i$ ležati na pravi strani ravnine $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$. To zahtevo je pametno omiliti, saj včasih pozitivnih in negativnih primerkov z ravnino sploh ni mogoče razmejiti (učna množica ni linearno separabilna — ta primer prepoznamo po tem, da za primarni problem sploh ni dopustnih rešitev, pri dualnem problemu pa je mogoče vrednost kriterijske funkcije neomejeno povečevati), včasih pa je to sicer mogoče, vendar se zaradi šuma in napak v učnih vektorjih mogoče odpovemo kakšni ravnini, ki bi imela veliko širši rob, če bi ignorirali šumne podatke, namesto nje pa uporabimo kakšno neugodnejšo ravnino, ki pa pravilno klasificira vse učne vektorje. S tem torej tvegamo, da klasifikatorja sploh ne bomo našli, ali pa, da bomo našli takega, ki se je pretirano prilagodil naši učni množici.

Takšni formulaciji našega optimizacijskega problema pravimo, da zahteva „trdi rob“ (*hard margin*): noben učni vektor ne sme ležati na robu (pravzaprav v robu) ali pobegniti skozenj na napačno stran ravnine. Bolj fleksibilen pristop pa je „mehki rob“ (*soft margin*): ravnini dovolimo, da kakšen učni vektor tudi narobe klasificira, vendar to upoštevamo v kriterijski funkciji in takšno ravnino zaradi napačnih klasifikacij malce slabše ocenimo. Zdaj torej ne zahtevamo več, naj za vsak učni podatek $(\mathbf{x}_i, y_i)$ velja

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1,$$

pač pa dovolimo

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \qquad \text{za neko } \xi_i \geq 0, \tag{4.3}$$

kriterijsko funkcijo pa v ta namen dopolnimo v

$$P(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{l} \xi_i.$$

Spremenljivkam ξ_i pravimo „kazenske spremenljivke“. C je neka konstanta, s katero lahko vplivamo na to, kaj nam je bolj pomembno: širok rob, pa čeprav je zato nekaj vektorjev narobe klasificiranih; ali pa zahtevamo čim višjo točnost na učni množici, četudi to pomeni ožji rob in tveganje pretiranega prilagajanja. Če bi postavili $C = \infty$, bi dobili spet različico s trdim robom, kjer ne sme biti noben učni vektor klasificiran narobe.

S premajhno C torej tvegamo, da bo klasifikator naredil veliko napak, ker le-te na vrednost kriterijske funkcije ne bodo dovolj vplivale; s preveliko C pa tvegamo, da bomo zašli v pretirano prilagajanje učnim podatkom. Primeren C je zato pametno izbrati s prečnim preverjanjem; preizkusimo več možnih C-jev in izberemo najboljšega.

Dopolnjeni optimizacijski problem:

$$\begin{array}{ll} \text{minimiziraj} & P(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{l} \xi_i \\ \text{pri pogojih } (\forall i) & y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 + \xi_i \;\; \geq \;\; 0, \\ & \qquad\qquad\qquad\qquad \xi_i \;\; \geq \;\; 0. \end{array} \tag{4.4}$$

lahko spet predelamo v dualnega, le izpeljava je nekoliko daljša. Zdaj moramo uvesti Lagrangeove multiplikatorje (recimo jim μ_i) tudi za neenačbe $\xi_i \geq 0$. Lagrangeova funkcija je zdaj

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2} \sum_{j=1}^{d} w_i^2 + C \sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \alpha_i (y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i) - \sum_{i=1}^{l} \mu_i \xi_i.$$

Multiplikatorji μ_i in α_i morajo biti seveda vsi nenegativni. Parcialno odvajanje po $\mathbf{w}$, b in zdaj tudi po $\boldsymbol{\xi}$ nam da pogoje

$$\begin{array}{c} \mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i, \\ \sum_{i=1}^{l} \alpha_i y_i = 0, \qquad \alpha_i + \mu_i = C \text{ za vsak } i. \end{array} \tag{4.5}$$

Ko jih vstavimo v L, lahko izpeljemo naslednji dualni problem:

$$\begin{array}{ll} \text{minimiziraj} & Q(\boldsymbol{\alpha}) = \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j - \sum_{i=1}^{l} \alpha_i \\ \text{pri pogojih} & \forall i \in 1..l : 0 \leq \alpha_i \leq C \\ & \sum_{i=1}^{l} \alpha_i y_i = 0 \end{array} \tag{4.6}$$

Kriterijska funkcija je torej popolnoma enaka kot pri formulaciji s trdim robom, le vrednosti α_i so zdaj vse omejene na interval $[0, C]$; vektor $\boldsymbol{\alpha}$ mora torej ležati v hiperkocki $[0, C]^l$. Zato se ne more več zgoditi, da bi lahko vrednost $Q(\alpha)$ povečevali prek vseh meja; prvotni problem pa ima vedno veliko dopustnih rešitev — karkšna koli $\mathbf{w}$ in b si že izberemo, vedno bo moč najti take ξ_i, da bodo veljali pogoji (4.3). Multiplikatorjev μ_i smo se znebili, ker so enolično določeni z α_i.

Opisani način sicer ni edini, kako lahko v učenje takšnega klasifikatorja vpeljemo zamisel o mehkem robu, je pa daleč najpogostejši. Alternativa je na primer ta, da v kriterijski funkciji namesto L_1-norme vektorja $\boldsymbol{\xi}$ uporabimo L_2-normo. Tako dobimo $P(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{l}\xi_i^2$. Če iz tega spet izpeljemo dualni problem, se izkaže, da je tudi zelo podoben dualnemu problemu za trdi rob (torej nima omejitev $\alpha_i \leq C$), pač pa se diagonalnim elementom matrike $\mathbf{Q}$ prišteje vrednost $1/(2C)$. Nekateri avtorji, na primer Mangasarian, pa v kriterijsko funkcijo namesto $\frac{1}{2}||\mathbf{w}||^2$ vključijo tudi $\frac{1}{2}\sqrt{||\mathbf{w}||^2 + b^2}$, kar v dualnem problemu spet le spremeni matriko $\mathbf{Q}$.

4.1.4 Podporni vektorji

Še eno koristno orodje za razumevanje dosedanjih izpeljav je Karush-Kuhn-Tuckerjev izrek iz teorije optimizacije. V našem primeru nam ta izrek za optimalno rešitev $\boldsymbol{\alpha}$ problema (4.6) in pripadajoča $\mathbf{w}$ ter b zagotavlja, da za vsak i velja

$$\alpha_i(y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 + \xi_i) = 0 \tag{4.7}$$

in še

$$\mu_i \xi_i = 0 \quad \text{oziroma} \quad (C - \alpha_i)\xi_i = 0.$$

(Spomnimo se, da za vsak i velja $\mu_i = C - \alpha_i$.) Pri vsakem i lahko razlikujemo med naslednjimi tremi možnostmi:

1. Če je $\alpha_i = 0$, mora biti $\mu_i = C$; in, da bo $\mu_i\xi_i$ še vedno nič, mora biti $\xi_i = 0$. To pa pomeni, da je neenačba $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \geq 1 - \xi_i$ pravzaprav izpolnjena brez kazenske spremenljivke; vektor $\mathbf{x}_i$ torej zadošča enačbi $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \geq 1$, kar pomeni, da bi ga naša ravnina klasificirala pravilno.

2. Če je $0 < \alpha_i < C$, je tudi $0 < \mu_i < C$, zato mora biti $\xi_i = 0$. Hkrati pa mora zaradi pogoja (4.7) veljati $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 + \xi_i = 0$ oziroma $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) = 1$. Ta vektor torej tudi leži na pravi strani naše ravnine, hkrati pa leži ravno na meji roba (na eni od ravnin $\langle \mathbf{w}, \mathbf{x} \rangle + b = \pm 1$).

3. Če je $\alpha_i = C$, je $\mu_i = 0$, zato je mogoče, da je $\xi_i > 0$. Podobno kot v prejšnjem odstavku mora biti tudi tu $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) = 1 - \xi_i$. Če je $0 \leq \xi_i < 1$, bo leva stran še vedno pozitivna in ravnina bi ta vektor pravilno klasificirala; vendar pa bi že ležal na (oz. v) robu. Pri $\xi_i = 1$ bi ležal ravno na ravnini $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$. Pri $1 < \xi_i < 2$ bi ležal že na napačni strani ravnine, vendar še vedno znotraj roba. Vektorje s $\xi_i > 1$ naša ravnina klasificira v napačni razred.

Vektorjem iz druge in tretje skupine pravimo „podporni vektorji“ (*support vectors*). Vektorji iz druge skupine so tisti, ki ležijo prav na ravninah $\langle \mathbf{w}, \mathbf{x} \rangle + b = \pm 1$, naša klasifikacijska ravnina $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ pa je na sredi med njima. Lahko si torej predstavljamo, da jo ti vektorji obdajajo z obeh strani in jo podpirajo, da se ne bi kam premaknila.

Drugi pomen podpornih vektorjev izvira iz formule (4.5), s katero lahko izrazimo normalo $\mathbf{w}$ iz multiplikatorjev $\boldsymbol{\alpha}$ in učnih podatkov: $\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i$. V tem izrazu prispevajo vektorji iz prve skupine (ker imajo $\alpha_i = 0$) same ničelne člene in nanj sploh ne vplivajo; v bistvu je torej dovolj, če gre ta vsota le po podpornih vektorjih.

Izkaže se celo naslednje: če bi izmed vseh učnih podatkov obdržali le podporne vektorje in za tako zmanjšano učno množico spet poiskali optimalno ravnino, bi dobili isto ravnino kot prej na celotni učni množici.

Podpornim vektorjem iz tretje skupine pravijo tudi „mejni“ podporni vektorji (*bounded support vectors*), ker so njihovi multiplikatorji na svoji zgornji meji ($\alpha_i = C$, višje ne morejo iti).

4.1.5 Določanje b-ja

Pogoji (4.5), ki izhajajo iz parcialnih odvodov funkcije L, nam omogočajo določiti $\mathbf{w}$, ne pa tudi praga b, saj ga sploh ne omenjajo. Zato pa lahko b določimo iz pogoja (4.7), ki ga zagotavlja Karush-Kuhn-Tuckerjev izrek. Za ne-mejne podporne vektorje, torej tiste iz druge od zgoraj naštetih treh skupin, je $\alpha_i > 0$ in $\xi_i = 0$, zato dobi izraz (4.7) obliko

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 = 0,$$

torej

$$b = 1/y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle = y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle.$$

Pri tem smo upoštevali, da je $y_i \in \{+1, -1\}$ in zato $y_i = 1/y_i$. Zaradi morebitnih numeričnih nenatančnosti pri računanju pa je še bolje, če vzamemo za b povprečje vrednosti $y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle$ po vseh ne-mejnih podpornih vektorjih i.

Enakovreden pristop k določanju b pa je naslednji [10, str. 29]: med vektorji iz prve in druge skupine, torej tistimi, ki ne ležijo znotraj roba ali celo na napačni strani ravnine $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$, poiščemo tiste, ki so ravnini najbližji. Med pozitivnimi so to tisti $\mathbf{x}_i$ z najmanjšo vrednostjo $\langle \mathbf{w}, \mathbf{x}_i \rangle + b$, med negativnimi pa tisti z največjo; klasifikacijska ravnina leži kot smo ugotovili v razdelku 4.1.1, ravno na pol poti med njimi. Toda opisani vrstni red se nič ne spremeni, četudi gledamo le vrednosti $\langle \mathbf{w}, \mathbf{x}_i \rangle$. Naj bo $c_\oplus \equiv \min_{i:y_i=+1} \langle \mathbf{w}, \mathbf{x}_i \rangle$ in $c_\ominus \equiv \max_{i:y_i=-1} \langle \mathbf{w}, \mathbf{x}_i \rangle$, pri čemer gre i vsakič le po vektorjih iz prve in druge skupine. Torej leži ravnina $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ na sredi med ravninama $\langle \mathbf{w}, \mathbf{x} \rangle = c_\oplus$ in $\langle \mathbf{w}, \mathbf{x} \rangle = c_\ominus$; torej je $b = -(c_\oplus + c_\ominus)/2$. Dobili smo formulo:

$$b = -\frac{1}{2} \left(\min_{i:y_i=+1, \xi_i=0} \langle \mathbf{w}, \mathbf{x}_i \rangle + \max_{i:y_i=-1, \xi_i=0} \langle \mathbf{w}, \mathbf{x}_i \rangle \right).$$

4.2 Nelinearni modeli

4.2.1 Prehod na nelinearne modele

Doslej so bili klasifikatorji, ki smo jih dobili z metodo podpornih vektorjev, navadne hiperravnine v d-razsežnem prostoru. Izkaže se, da je že to (v kombinaciji z mehkim robom) dovolj za uspešno reševanje marsikaterega praktičnega problema. Vendar pa metoda podpornih vektorjev omogoča tudi, da na eleganten način pridemo do nelinearnih modelov, torej klasifikatorjev, pri katerih so pozitivni in negativni vektorji razmejeni s kakšno kompleksnejšo ploskvijo, ne pa z ravnino.

Mislimo si, da bi vektorje (učne, kasneje pa tudi testne) preslikali iz prostora $\mathbb{R}^d$ v nek drug prostor $\mathcal{F}$. Ta preslikava, označimo jo s ϕ, je lahko nelinearna, prostor $\mathcal{F}$ pa ima lahko zelo veliko (mogoče celo neskončno mnogo) dimenzij. Oznako $\mathcal{F}$ ponavadi uporabljamo, ker gre za „prostor oblik“ (*feature space*) — preslikava ϕ, ki vektorju $\mathbf{x} \in \mathbb{R}^d$ priredi nek vektor $\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots) \in \mathcal{F}$, naj bi v $\mathbf{x}$-u odkrila njegove oblike ali značilke $\phi_k(\mathbf{x})$. V nadaljevanju bomo videli, da se s podrobnejšo definicijo prostora $\mathcal{F}$ običajno ni treba ukvarjati, ker zadostuje, če znamo v njem računati skalarne produkte slik vektorjev iz $\mathbb{R}^d$, to pa lahko pogosto počnemo, ne da bi eksplicitno računali same slike.

Če bi se zdaj v skladu s postopki iz prejšnjega razdelka učili na vektorjih $\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_l)$ iz prostora $\mathcal{F}$, bi kot klasifikator dobili neko ravnino v prostoru $\mathcal{F}$:

$$\langle \hat{\mathbf{w}}, \hat{\mathbf{x}} \rangle_{\mathcal{F}} + b = 0,$$

pri čemer je normala $\hat{\mathbf{w}}$ zdaj vektor iz $\mathcal{F}$, z oznako $\langle \cdot\,, \cdot \rangle_{\mathcal{F}}$ pa smo poudarili, da gre za skalarni produkt v prostoru $\mathcal{F}$. Vektorje $\mathbf{x}$ iz $\mathbb{R}^d$ bi klasificirali tako, da bi jih preslikali v $\mathcal{F}$ in potem pogledali, kje leži slika $\phi(\mathbf{x})$ glede na dobljeno ravnino:

$$f(\mathbf{x}) = \operatorname{sgn}(\langle \hat{\mathbf{w}}, \phi(\mathbf{x}) \rangle_{\mathcal{F}} + b).$$

Lahko bi torej rekli, da smo prostor $\mathbb{R}^d$ zdaj razmejili s ploskvijo

$$\langle \hat{\mathbf{w}}, \phi(\mathbf{x}) \rangle_{\mathcal{F}} + b = 0,$$

ki ni nujno ravnina, saj ϕ ni nujno linearna preslikava. Še več, ta ploskev lahko prostor $\mathbb{R}^d$ celo razdeli na več nepovezanih delov; nekateri bi pripadli razredu $+1$, drugi pa razredu -1.

Jedra nam torej dajo na razpolago širšo družino krivulj (oz. ploskev), s katerimi lahko razmejujemo pozitivne in negativne primerke v prostoru $\mathbb{R}^d$. Tako imamo več možnosti, da se prilagodimo morebitnim nelinearnostim v porazdelitvi pozitivnih in negativnih primerkov. Bogatejši prostor hipotez sicer prinaša tudi določeno tveganje pretiranega prilagajanja (*overfitting*), vendar se metoda podpornih vektorjev letemu precej dobro izogiba, ker se trudi najti klasifikator s čim širšim robom (ki naj torej ne bi šel preblizu učnim vektorjem in naj se ne bi prilagajal vsemu šumu in nepomembnim podrobnostim v učnih podatih) in je pripravljena v ta namen dopustiti tudi nekaj napak pri klasifikaciji na učni množici.

Normala $\hat{\mathbf{w}} \in \mathcal{F}$ se še vedno izraža po formuli (4.5):

$$\hat{\mathbf{w}} = \textstyle\sum_{i=1}^{l} \alpha_i y_i \phi(\mathbf{x}_i),$$

tako da naš klasifikator pravzaprav računa naslednji izraz:

$$f(\mathbf{x}) = \operatorname{sgn}(\langle \hat{\mathbf{w}}, \phi(\mathbf{x}) \rangle_{\mathcal{F}} + b) = \operatorname{sgn}\left(b + \textstyle\sum_{i=1}^{l} \alpha_i y_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle_{\mathcal{F}}\right).$$

Kriterijsko funkcijo Q optimizacijskega problema (4.6) pa lahko zdaj zapišemo kot

$$Q(\boldsymbol{\alpha}) = \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{F}} - \sum_{i=1}^{l} \alpha_i.$$

Definicija matrike $\mathbf{Q}$ se je torej spremenila v $q_{ij} = y_i y_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{F}}$. Opazimo lahko, da niti pri učenju niti pri uporabi klasifikatorja ni treba eksplicitno računati slik $\phi(\mathbf{x})$; dovolj je, če znamo za dana vektorja $\mathbf{x}$ in $\mathbf{z}$ izračunati skalarni produkt njunih slik,

$$K(\mathbf{x}, \mathbf{z}) \equiv \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle_{\mathcal{F}}. \tag{4.8}$$

Tako definirani funkciji K pravimo „jedro“ (*kernel function*). Namesto da bi eksplicitno izračunali sliki $\phi(\mathbf{x})$ in $\phi(\mathbf{z})$ in ju skalarno zmnožili, lahko skalarni produkt (za primerne ϕ) pogosto izračunamo preprosteje in ceneje kar neposredno iz vektorjev $\mathbf{x}$ in $\mathbf{z}$. V splošnem torej sploh ni potrebno posebej poznati preslikave ϕ; dovolj je že, če znamo računati njej pripadajočo jedrno funkcijo K. (Določeni funkciji K lahko ustreza tudi več preslikav ϕ, vsaka v drugačen prostor $\mathcal{F}$. Trivialen primer je na primer $K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{z} + 1$, ki ustreza na primer preslikavi $\phi(\mathbf{x}) = [\mathbf{x}^T 1]^T$ v prostor $\mathbb{R}^{d+1}$ s standardnim skalarnim produktom, pa tudi preslikavi $\phi(\mathbf{x}) = [\mathbf{x}^T \frac{1}{2}\frac{1}{2}\frac{1}{2}\frac{1}{2}]^T$ v prostor $\mathbb{R}^{d+4}$ s standardnim skalarnim produktom.)

Normale $\hat{\mathbf{w}} \in \mathcal{F}$ zdaj ne bomo več hranili v eksplicitni obliki, pač pa bomo shranili uteži α_i, y_i in tiste od vektorjev $\mathbf{x}_i$, ki imajo neničelno α_i (torej podporne vektorje) in za klasifikacijo uporabljali formulo

$$f(\mathbf{x}) = \operatorname{sgn}\left(b + \textstyle\sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x})\right). \tag{4.9}$$

Do funkcije K bi lahko prišli tako, da bi si najprej izbrali preslikavo ϕ in nato poskušali najti neposredno izražavo skalarnega produkta $\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle_{\mathcal{F}}$ z vektorjema $\mathbf{x}$ in $\mathbf{z}$. Pogosto pa naredimo obratno: najprej si izberemo jedrno funkcijo K, potem pa moramo le še preveriti, če res obstajata nek prostor $\mathcal{F}$ in neka preslikava $\phi : \mathbb{R}^d \to \mathcal{F}$, za katera velja (4.8) pri vseh $\mathbf{x}$ in $\mathbf{z}$.

V funkcionalni analizi obstaja Mercerjev izrek, ki je lahko koristen pri preverjanju te lastnosti; podaja namreč naslednji zadostni pogoj za obstoj takega prostora $\mathcal{F}$ in preslikave ϕ: naj bo $\mathcal{C}$ neka kompaktna podmnožica prostora $\mathbb{R}^d$, $K(\mathbf{x}, \mathbf{z})$ pa zvezna in simetrična na $\mathcal{C} \times \mathcal{C}$ in za vsako funkcijo f, ki je kvadratno integrabilna na $\mathcal{C}$, naj velja

$$\iint_{\mathcal{C}\times\mathcal{C}} f(\mathbf{x}) K(\mathbf{x}, \mathbf{z}) f(\mathbf{z})\, d\mathbf{x}\, d\mathbf{z} \geq 0.$$

Potem obstajata primerna $\mathcal{F}$ in ϕ, da velja (4.8). Z drugimi besedami, jedro K mora biti simetrično in pozitivno semidefinitno. Enakovreden pogoj je tudi naslednji: za vsako končno množico vektorjev $\mathbf{x}_1, \ldots, \mathbf{x}_l \in \mathbb{R}^d$ in njihovih oznak $y_1, \ldots, y_l \in \{+1, -1\}$ mora biti pripadajoča matrika $\mathbf{Q}$, definirana s pravilom $q_{ij} \equiv y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, pozitivno semidefinitna.

4.2.2 Jedra

V rabi je več različnih družin jeder. Znotraj vsake družine imajo vsa jedra $K(\mathbf{x}, \mathbf{z})$ podobno obliko, razlikujejo pa se po vrednosti kakšnega parametra.

Polinomska jedra. Ta so v splošnem oblike

$$K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T\mathbf{z} + 1)^p = (x_1 z_1 + \ldots + x_d z_d + 1)^p.$$

Če bi ta izraz razvili na posamezne člene, bi dobili

$$K(\mathbf{x}, \mathbf{z}) = \sum \binom{p}{p_1, p_2, \ldots, p_d} (x_1 z_1)^{p_1} \cdots (x_d z_d)^{p_d}, \tag{4.10}$$

pri čemer ima ta vsota po en člen za vsako d-terico eksponentov $(p_1, \ldots, p_d)$, za katere je $0 \leq \sum_{i=1}^{d} p_i \leq p$. Teh členov je $\binom{d+p}{p}$. Na primer: pri $p = 2$ bi dobili d členov oblike $c \cdot (x_i z_i)^2$, $d(d-1)/2$ členov oblike $c \cdot (x_i z_i)(x_j z_j)$ za $i \neq j$, d členov oblike $c \cdot x_i z_i$ in še konstantni člen, skupaj torej ravno $\binom{d+2}{2} = (d+2)(d+1)/2$. Naj funkcija ϕ preslika vektor $\mathbf{x}$ v vektor z $\binom{d+p}{p}$ komponentami oblike $\sqrt{\binom{p}{p_1, p_2, \ldots, p_d}} x_1^{p_1} \cdots x_d^{p_d}$; očitno je potem skalarni produkt dveh takih slik, $\phi(\mathbf{x})$ in $\phi(\mathbf{z})$, ravno vsota (4.10).

Včasih konstantni člen $+1$ v definiciji jedra tudi izpustijo; takšno jedro ustreza podobni preslikavi ϕ v nek $\binom{d+p-1}{p}$-razsežni prostor.

Pri $p = 1$ nam dá polinomsko jedro spet navaden linearni klasifikator, poleg tega pa se uporabljajo še polinomska jedra stopnje do 5 ali 6. Pri $p = 2$ govorimo tudi o kvadratnem, pri $p = 3$ pa o kubičnem jedru.

Za primer, kaj pomeni uporaba takšnega jedra pri učenju z metodo podpornih vektorjev, si lahko na primer ogledamo kvadratno jedro ($p = 2$) nad točkami na navadni dvorazsežni realni ravnini $\mathbb{R}^2$ ($d = 2$). Jedro je tedaj oblike

$$K(\mathbf{x}, \mathbf{z}) = x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 x_2 z_1 z_2 + 2 x_1 z_1 + 2 x_2 z_2 + 1.$$

Funkcija, ki bi jo računal klasifikator, bi bila zato oblike

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) \right) = \operatorname{sgn}(A x_1^2 + B x_2^2 + C x_1 x_2 + D x_1 + E x_2 + F)$$

za neke konstante $A, B, \ldots, F$, ki so odvisne le od $\mathbf{x}_i$, y_i, α_i, ne pa od vektorja $\mathbf{x} = (x_1, x_2)^T$. To torej pomeni, da smo ravnino $\mathbb{R}^2$ razdelili z neko stožernico (krivuljo drugega reda), ne pa s premico (kar bi bila hiperravnina v $\mathbb{R}^2$) kot pri linearnem klasifikatorju.

Radialna jedra so oblike

$$K(\mathbf{x}, \mathbf{z}) = e^{-||\mathbf{x} - \mathbf{z}||^2 / 2\sigma^2}.$$

Lahko si jih predstavljamo takole: v formuli 4.9 prispeva vsak podporni vektor x_i nek glas za svoj razred y_i; njegova teža pri tem glasovanju pa je $\alpha_i K(\mathbf{x}_i, \mathbf{x})$ in je torej tem večja, čim bližje temu vektorju je opazovani vektor $\mathbf{x}$. Ta teža pada z oddaljenostjo od $\mathbf{x}_i$ v obliki zvonaste krivulje (širino zvona določa σ). Z zadosti veliko σ bi lahko pravilno klasificirali vse vektorje iz poljubne učne množice (če le ne vsebuje primerkov $\mathbf{x}_i = \mathbf{x}_j$, $y_i \neq y_j$), ker bi se vpliv učnega vektorja $\mathbf{x}_i$ v formuli 4.9 res poznal le v zelo majhni okolici točke $\mathbf{x}_i$. Vendar pa bi bili tedaj tudi skoraj vsi učni vektorji podporni; to bi skoraj zagotovo pomenilo pretirano prilagajanje učnim podatkom. Po drugi strani premajhna σ pomeni, da so zvonovi preširoki in oddaljeni učni vektorji bi imeli prevelik vpliv na klasifikacijo; to bi pomenilo, da se učimo zelo okornega klasifikatorja, ki se povsod odloča približno enako.

Sigmoidna jedra:

$$K(\mathbf{x}, \mathbf{z}) = \operatorname{th}(\kappa \, \mathbf{x}^T \mathbf{z} + \Theta).$$

Klasifikator, ki temelji na takem jedru, deluje kot dvonivojska nevronska mreža. Izračun pri klasifikaciji $\sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x})$ si lahko namreč predstavljamo tako, da imamo na prvem nivoju po en nevron za vsak podporni vektor; vhod $\mathbf{x}$ se prek uteži $\mathbf{x}_i$ privede na i-ti nevron, ki ga obdela še s faktorjem κ, pragom Θ in sigmoidno funkcijo th (hiperbolični tangens). Izhod i-tega nevrona prvega nivoja (ki znaša $\kappa\, \mathbf{x}_i^T \mathbf{x} + \Theta$) gre z utežjo $\alpha_i y_i$ v edini nevron drugega nivoja, ki vse svoje vhode sešteje in vrne predznak dobljene vsote. (Enako si lahko predstavljamo vsak klasifikator, dobljen z metodo podpornih vektorjev; nevroni prvega nivoja pač računajo $K(\mathbf{x}_i, \mathbf{x})$, kakršno koli jedro že vzamemo.)

Jedra χ^2 sta predlagala Teytaud in Jalam [50] za klasifikacijo besedil. Lahko si jih predstavljamo kot različico radialnih jeder: pri slednjih je eksponent sorazmeren z $||\mathbf{x} - \mathbf{y}||^2 = \sum_{i=1}^{d} (x_i - y_i)^2$. To torej pomeni, da lahko različne komponente k tej vsotid prispevajo zelo različne količine: razlika med $x_i = 1000$ in $y_i = 1010$ bi prispevala veliko več kot med $x_i = 100$ in $y_i = 101$, čeprav bi lahko rekli, da sta v relativnem smislu obe razliki enako veliki glede na red velikosti komponent x_i in y_i. Tako naj bi poskušali komponente „normalizirati“:

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{d} \frac{(x_i - y_i)^2}{(x_i + y_i)/2} \right).$$

Vendar pa vektorje pred učenjem z metodo podpornih vektorjev pogosto v vsakem primeru normalizirajo, na primer tako, da ležijo vsi učni vektorji po normalizaciji v notranjosti enotske hiperkocke $[0, 1]^l$. Zato je uporabnost jedra χ^2 do neke mere vprašljiva.

Jedra za nize: tudi ta so mišljena za delo z besedili (ali kakšnimi drugi zaporedji nekih osnovnih simbolov), le da tu besedila sploh ne predelamo v vektor. Predstavljajmo si ga kar kot dolg niz znakov. Niz $u = a_1 a_2 \cdots a_k$ (a_i so posamezni znaki) se pojavlja v nizu x, če lahko slednjega zapišemo kot $x = x_1 a_1 x_2 a_2 \cdots x_k a_k x_{k+1}$ (x_i so lahko nizi nič ali več znakov). Ta pojavitev naj ima za nas vrednost $\gamma^{|a_1 x_2 a_2 x_3 \cdots x_k a_k|}$ (za neko $\gamma \in (0, 1)$; tako zadušimo vpliv zelo nestrnjenih pojavitev, ki najbrž res nič ne pomenijo). Označimo s $\phi_u(x)$ vsoto vrednosti vseh pojavitev niza u v nizu x. Zdaj lahko naše jedro definiramo kot

$$K(x, y) = \sum_{u \in \Sigma^k} \phi_u(x) \phi_u(y).$$

Ta vsota gre torej po vseh nizih u dolžine k nad opazovano abecedo Σ. Že iz definicije je očitno, da je to res jedro, saj smo najprej definirali funkcijo ϕ (prostor $\mathcal{H}$ je v tem primeru $\mathbb{R}^{\Sigma^k}$) in šele iz nje K. Vsota v definiciji K-ja ima sicer lahko zelo veliko členov, a na srečo se da K računati z dinamičnim programiranjem v času $O(k\,|x|\,|y|)$.

To jedro [33] se menda pri klasifikaciji besedil obnese približno tako dobro kot tradicionalni pristopi (vektorji TF-IDF ter polinomska ali radialna jedra) — včasih je boljše, včasih slabše — tako da pravzaprav ni zelo privlačno. Intuitivno zamisel, da bi lahko razpršena skupina črk, ki po naključju pač tvorijo nek določen podniz, povedala o besedilu kaj posebej pametnega — razen če ne gre ravno za prvih nekaj črk neke besede ali pa mogoče dveh zaporednih besed — ni posebej prepričljiva. Tako naj bi to jedro delovalo bolj po naključju kot namenoma.

Drug pogled na jedra za nize je, da besedilo gledamo kot zaporedje besed (mogoče normaliziranih) in potem gledamo podzaporedja besed.

4.3 Reševanje v praksi

Da pridemo do klasifikatorja po metodi podpornih vektorjev, moramo rešiti optimizacijsko nalogo (4.6), torej minimizirati $Q(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{Q}\boldsymbol{\alpha} - \mathbf{1}^T\boldsymbol{\alpha}$ pri pogojih $\boldsymbol{\alpha} \in [0, C]^l$ in $\mathbf{y}^T\boldsymbol{\alpha} = 0$. To je poseben primer problema, ki je v matematiki znan kot *kvadratno programiranje* in za njegovo numerično reševanje obstajajo različni postopki. Lahko bi ga na primer reševali z gradientnim spuščanjem; iz točke α se premaknemo v smeri gradienta $\nabla_{\boldsymbol{\alpha}} Q(\boldsymbol{\alpha}) = \mathbf{Q}\boldsymbol{\alpha} - \mathbf{1}$ do minimuma funkcije v tej smeri (dolžino premika lahko določimo analitično). Slabost tega pristopa je v tem, da je naslednji korak pod temi pogoji gotovo pravokoten glede na prejšnjega, zato se minimumu včasih približujemo počasi in z veliko kratkimi cikcakastimi premiki. Ta problem je manjši pri *metodi konjugiranih gradientov* [48], ki smer premika izbira tako, da je pravokotna glede na smeri vseh prejšnjih premikov, in pri tem zagotavlja, da vsakič najdemo minimum po vsem afinem podprostoru, ki je dosegljiv iz začetne točke z gibanjem v smereh dosedanjih premikov. Zato po največ l premikih najdemo minimum po celem l-razsežnem prostoru. Pri metodi konjugiranih gradientov pa je težje upoštevati zahtevo, naj ostane $\boldsymbol{\alpha}$ znotraj kocke $[0, C]^l$; zato na primer Burges [9] opisuje kombiniran pristop, pri katerem se uporablja gradientno spuščanje, ko je $\boldsymbol{\alpha}$ na robu kocke, in metoda konjugiranih gradientov, ko je $\boldsymbol{\alpha}$ v notranjosti.

Vendar pa običajni postopki za reševanje kvadratnih programov večinoma zahtevajo, da je ves čas pri roki vsebina matrike $\mathbf{Q}$; bilo bi časovno preveč potratno, če bi hoteli njene elemente generirati sproti ob vsaki uporabi. (Spomnimo se, da je $q_{ij} \equiv y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$; določitev elementa q_{ij} torej zahteva en klic jedrne funkcije, kar v splošnem ni poceni.) Matrika $\mathbf{Q}$ je pri učenju po metodi podpornih vektorjev velika $l \times l$, pri čemer je l število učnih vektorjev. Že pri ne tako velikih učnih množicah (npr. nekaj tisoč učnih vektorjev) torej postane matrika $\mathbf{Q}$ prevelika za pomnilnik večine računalnikov.

Zato so se za potrebe učenja po metodi podpornih vektorjev pojavili različni postopki, ki se posvečajo le po nekaj spremenljivkam α_i naenkrat. Najprej si izberemo nekaj izmed spremenljivk α_i, gledamo $Q(\boldsymbol{\alpha})$ kot funkcijo le teh (ostale naj bodo začasno konstantne) in poiščemo njen minimum. Potem izberemo neko drugo skupino spremenljivk α_i, poiščimo minimum glede na te spremenljivke in tako naprej.

S pomočjo Karush-Kuhn-Tuckerjevega pogoja (4.7) lahko občasno preverimo, ali smo dosegli globalni minimum: zadosten pogoj za to je namreč to, (a) da je α dopustna rešitev: veljati morajo neenačbe iz (4.4)

$$y_i(\langle \hat{\mathbf{w}}, \phi(\mathbf{x}_i)\rangle_{\mathcal{F}} + b) \geq 1 - \xi_i;$$

in (b) da obstajajo take $\xi_i \geq 0$, da za vsak i velja

$$\alpha_i(y_i(\langle \hat{\mathbf{w}}, \phi(\mathbf{x}_i)\rangle_{\mathcal{F}} + b) - 1 + \xi_i) = 0 \qquad \text{in} \qquad (C - \alpha_i)\xi_i = 0.$$

Pri tem lahko skalarni produkt $\langle \hat{\mathbf{w}}, \phi(\mathbf{x}_i)\rangle_{\mathcal{F}}$ razvijemo v $\sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_i)$. Če ta dva pogoja združimo, vidimo, da mora za vsak i veljati:

$$\begin{array}{rcl} \alpha_i = 0 & \Longrightarrow & y_i(\langle \hat{\mathbf{w}}, \phi(\mathbf{x}_i)\rangle_{\mathcal{F}} + b) \geq 1 \\ 0 < \alpha_i < C & \Longrightarrow & y_i(\langle \hat{\mathbf{w}}, \phi(\mathbf{x}_i)\rangle_{\mathcal{F}} + b) = 1 \\ \alpha_i = C & \Longrightarrow & y_i(\langle \hat{\mathbf{w}}, \phi(\mathbf{x}_i)\rangle_{\mathcal{F}} + b) \leq 1 \end{array} \tag{4.11}$$

Obstajajo pa različne hevristike in strategije glede izbire spremenljivk, ki jih bomo optimizirali. Običajno se je pametno najprej posvetiti tistim spremenljivkam, ki pogoje (4.11) najhuje kršijo.

Vapnik je na primer predlagal obdelavo po kosih (*chunking*). Izberemo si nekaj spremenljivk in minimiziramo Q glede na njih; nato obdržimo od njih le tiste z $\alpha_i > 0$, dodamo pa v skupino še nekaj izmed doslej neobdelanih spremenljivk. Spet minimiziramo Q po tako popravljeni skupini spremenljivk; završemo iz nje tiste z $\alpha = 0$ in dodamo nekaj takih, ki jih v tem koraku nismo optimizirali. To ponavljamo, dokler ne pridemo do globalnega minimuma. Ta postopek temelji na dejstvu, da je prvotni problem učenja enakovreden tistemu, pri katerem se učimo le na podpornih vektorjih (razdelek 4.1.4). Iz obdelovane skupine spremenljivk ves čas odstranjujemo tiste, katerih vektorji niso podporni ($\alpha_i = 0$) in na koncu se bomo učili ravno na vseh podpornih vektorjih. Zdaj torej ni več treba hraniti v pomnilniku cele matrike $\mathbf{Q}$ velikosti $l \times l$, pač pa le še matriko, veliko $k \times k$, če imamo k podpornih vektorjev.

Osuna in sodelavci so opisali različico tega pristopa (*decomposition*), pri katerem v vsakem koraku dodamo v skupino enako število spremenljivk, kolikor smo jih prej vzeli iz nje. Tako je število spremenljivk, ki jih optimiziramo v vsakem koraku, konstantno in lahko bomo obdelovali probleme s poljubno veliko učnimi (in podpornimi) vektorji. Pokazati je mogoče, da tudi ta postopek zagotovo doseže optimalno rešitev, važno je le, da v vsakem koraku dodamo med spremenljivke vsaj eno tako, ki je prej kršila svoj KKT pogoj (4.11).

Podoben pristop opisuje Joachims [29]; s preprosto hevristiko vsakič izberemo neko majhno število spremenljivk (npr. deset), minimiziramo Q glede na te in to ponavljamo, dokler ne dosežemo globalnega optimuma.

Dovolj pa je celo, če si vsakič izberemo samo dve spremenljivki. Zagotoviti moramo namreč veljavnost pogoja $\mathbf{y}^T\boldsymbol{\alpha} = 0$ iz (4.6), tako da moramo, čim spreminjamo neko α_i, spreminjati vsaj še neko drugo α_j, da bo lahko vsota $\sum_{i=1}^{l} y_i\alpha_i$ ostala enaka 0. Če fiksiramo vse ostale spremenljivke, je Q zdaj preprosta kvadratna funkcija dveh spremenljivk, ti dve pa sta za povrhu povezani s pogojem $y_i\alpha_i + y_j\alpha_j + c = 0$ za konstanto $c = \sum_{k\neq i, k\neq j} y_k\alpha_k$ (torej ležita na premici $\alpha_i = y_i(c - y_j\alpha_j)$). Tako okleščeni problem lahko rešimo analitično in poiščemo optimalni α_i in α_j. Ta pristop se imenuje „zaporedno minimalno optimiziranje“ (SMO, *sequential minimal optimization*) [40]. α_i in α_j spet izbiramo z neko hevristiko (α_i naj bo najhujši kršitelj pogojev (4.11), α_j pa taka spremenljivka, da v paru z α_i obeta največji premik proti minimumu).

4.4 Posplošitev na več razredov

V svoji osnovni obliki nam dá metoda podpornih vektorjev klasifikator (ravnino), ki zna ločevati med dvema razredoma. Obstaja pa tudi več načinov, kako sestaviti klasifikatorje, ki bodo razlikovali med več (recimo k) razredi.

Eden proti ostalim (*one-against-all*): za vsakega od k razredov sestavimo po en klasifikator: pri učenju klasifikatorja za razred m uporabimo predstavnike tega razreda kot pozitivne učne primerke, predstavnike vseh ostalih razredov pa kot negativne. Ko je treba klasificirati nek nov vektor $\mathbf{x}$, ga pokažemo vsem klasifikatorjem. Seveda je mogoče, da jih po več reče, da sodi $\mathbf{x}$ v njihov razred, ali pa ga vsi zavrnejo. Včasih ni nič narobe, če nek primerek uvrstimo v več razredov hkrati; če pa hočemo natanko en razred, izberemo pač tistega, čigar klasifikator je $\mathbf{x}$-u pripisal najvišjo oceno.

Slabost te strategije je, da je v razredu m običajno veliko manj primerkov kot v vseh ostalih skupaj, zato imajo ti ostali neugodno velik vpliv na potek dobljene ravnine; to lahko omilimo, če napačno klasificirane primerke razreda m kaznujemo huje kot napačno klasificirane primerke ostalih razredov (namesto ene same konstante C ločimo med $C_\oplus$ in $C_\ominus$). Poleg tega je lahko meja med enim razredom in ostalimi razredi precej zapletena in se je zanjo težje naučiti dobrega klasifikatorja, kot če bi na primer iskali le mejo med dvema razredoma (in se ne zmenili za predstavnike ostalih razredov) [31, str. 257–8].

Eden proti enemu (*one-against-one, all-pairs*): za vsak par razredov, npr. m_1 in m_2, sestavimo po en klasifikator, ki bo ločeval med primerki teh dveh razredov [17, 31]. Pri učenju tega klasifikatorja veljajo predstavniki razreda m_1 recimo za pozitivne učne primere, predstavniki m_2 za negativne, ostalih pa sploh ne gledamo. Tako moramo sicer pripraviti $k(k-1)/2$ klasifikatorjev, vendar pa je z vsakim od njih veliko manj dela kot pri prejšnjem pristopu, ker so učne množice manjše. Pri metodi podpornih vektorjev je časovna zahtevnost učenja približno $O(l^2)$, če se učimo na l učnih primerkih. Recimo, da pristop „eden proti vsem“ za učenje vsakega klasifikatorja porabi približno $c \cdot l^2$ časovnih enot (za neko konstanto c); naučiti pa se jih mora k, torej je skupna zahtevnost ckl^2 enot. Pristop „eden proti enemu“ pa vsakič gleda le dva razreda, torej povprečno $l \cdot 2/k$ primerkov, če so razredi približno enako veliki; takih klasifikatorjev pa imamo $k(k-1)/2$, kar pomeni časovno zahtevnost $ck(k-1)/2 \cdot (2l/k)^2 \approx 2cl^2$ — velika izboljšava glede na ckl^2, predvsem pa časovna zahtevnost ni več odvisna od števila razredov k.

Da klasificiramo nek nov vektor, ga pokažemo vsem klasifikatorjem in se odločimo za tisti razred, ki ga je predlagalo največ klasifikatorjev. Tudi po tej plati se lahko ta pristop kosa s pristopom „eden proti vsem“, saj glavnino časa porabimo za izračunavanje jedra $K(\mathbf{x}, \mathbf{x}_i)$ med danim vektorjem $\mathbf{x}$ in vsemi podpornimi vektorji $\mathbf{x}_i$. Isti $\mathbf{x}_i$ se lahko pojavi kot podporni vektor pri več klasifikatorjih, tako da lahko prihranimo čas, če izračunamo $K(\mathbf{x}, \mathbf{x}_i)$ le enkrat in si to vrednost zapomnimo. Poleg tega so problemi, na katerih se uči pristop „eden proti enemu“, preprostejši in imajo zato praviloma manj podpornih vektorjev.

Postopek DAGSVM: to izboljšavo postopka „eden proti enemu“ so predlagali Platt in sod. [39]. Enako kot prej se tudi tu naučimo $k(k-1)/2$ klasifikatorjev, ki naj bi znali ločevati pripadnike po dveh razredov; drugače pa jih uporabljamo pri klasifikaciji. Mislimo si klasifikator (hiperravnino), ki ločuje med razredoma m_1 in m_2. Če ta klasifikator nek vektor $\mathbf{x}$ uvrsti v m_2, ne pa v m_1, to pomeni, da leži $\mathbf{x}$ na nasprotni strani te ravnine kot večina predstavnikov razreda m_1. Ker so pripadniki dveh razredov v večrazrednem problemu lahko drug od drugega kar precej oddaljeni, ima takšna ravnina pogosto tudi precej širok rob. Iz tega sklepamo, da leži tak $\mathbf{x}$ verjetno precej daleč od predstavnikov razreda m_1, tako da verjetno res ne spada v ta razred. Zato na $\mathbf{x}$ v bodoče ne bomo poganjali tistih klasifikatorjev, ki ločujejo med m_1 in drugimi razredi, saj si mislimo, da bi tudi oni ugotovili, da $\mathbf{x}$ ni v razredu m_1. Po vsaki klasifikaciji z enim od binarnih klasifikatorjev na ta način izločimo enega od razredov iz nadaljnje obravnave; tako bo po $k-1$ klasifikacijah ostal le še en razred in vektor $\mathbf{x}$ uvrstimo vanj. S tem smo do odgovora prišli že po uporabi samo $k-1$ izmed $k(k-1)/2$ binarnih klasifikatorjev.

Sodeč po eksperimentalnih podatkih v [39] ta postopek po točnosti nič ne zaostaja za postopkom „eden proti enemu", klasifikacija z njim pa je bila v praksi (za $k = 10$ razredov) vsaj dvakrat hitrejša. Ime DAGSVM izvira iz dejstva, da si lahko opisani način uporabe posameznih binarnih klasifikatorjev predstavljamo kot sprehod (od korena do enega od listov) po nekem usmerjenem acikličnem grafu — torej kot bi namesto odločitvenega drevesa uporabili odločitveni DAG.

4.4.1 Kodiranje izhoda

Kodiranje izhoda (*output coding*) je posplošitev tehnik, kot sta „eden proti ostalim" in „eden proti enemu". Tako kot tidve tudi ta pristop ni vezan na metodo podpornih vektorjev, ampak ga lahko uporabljamo tudi pri drugih vrstah klasifikatorjev.

Naj bo $M \in \{-1, 0, +1\}^{k \times p}$ „kodna matrika" reda $k \times p$. Sestavimo p binarnih klasifikatorjev; s-ti stolpec matrike M določa, kakšno učno množico bomo uporabili pri učenju s-tega klasifikatorja: predstavniki razreda r bodo zanj pozitivni učni primerki, če je $M_{rs} = +1$, in negativni učni primerki, če je $M_{rs} = -1$; če pa je $M_{rs} = 0$, jih pri učenju s-tega klasifikatorja sploh ne bomo uporabili. Tako naj bi torej klasifikator s predstavnike razreda r označeval z oznako M_{rs} (če je ta neničelna; če pa je $M_{rs} = 0$, pač ne vemo, kako bo označeval predstavnike tega razreda, saj mu jih tudi v učni množici nismo pokazali). Lahko bi rekli, da smo razred r z matriko M „zakodirali" v vektor $(M_{r1}, \ldots, M_{rp})$.

Da klasificiramo nek vektor $\mathbf{x}$, si ogledamo napovedi vseh p klasifikatorjev: $(f_1(x), \ldots, f_p(x))$. Ta vektor moramo „dekodirati" v eno od oznak razredov. Če je $\mathbf{x}$ iz razreda r, bi pričakovali, da se bodo te napovedi po predznakih dobro ujemale z r-to vrstico matrike M. Vektor $\mathbf{x}$ bomo torej uvrstili v tisti razred, pri katerem je to ujemanje največje:

$$f(\mathbf{x}) = \underset{r \in \{1,\ldots,k\}}{\arg\min} \left\{ \textstyle\sum_{s=1}^{p} L(M_{rs} f_s(\mathbf{x})) \right\}.$$

Pri tem je $M_{rs} f_s(\mathbf{x})$ pozitivna, če se napoved $f_s(\mathbf{x})$ po predznaku ujema z M_{rs}, in negativna, če se ne; absolutna vrednost pa izraža zaupanje, ki ga ima klasifikator f_s v svojo napoved. Če je torej $M_{rs} f_s(\mathbf{x})$ negativna in po absolutni vrednosti velika, pomeni to hudo napako klasifikatorja. L naj bi bila neka funkcija, ki ima pri negativnih argumentih velike vrednosti, pri pozitivnih pa majhne; na primer $L(z) = e^{-z}$.

Strategijo „eden proti ostalim" lahko v ta okvir vključimo z matriko velikosti $k \times k$, za katero je $M_{rr} = +1$ in $M_{rm} = -1$ za $r \neq m$. Strategija „eden proti enemu" pa ustreza matriki s $k(k-1)/2$ stolpci; če jih namesto z indeksi $1, \ldots, \binom{k}{k-1}$ označimo z množicami $\{r, m\}$, $r < m$, je primerna matrika takšna: $M_{r,\{r,m\}} = +1$, $M_{m,\{r,m\}} = -1$, drugod pa so ničle.

Allwein in sodelavci [2] so preizkusili ti dve in še nekaj drugih kodnih matrik, med drugim „polno matriko" (ki ima po en stolpec za vsako razbitje razredov $\{1, \ldots, k\}$

na dve neprazni podmnožici; torej $2^k - 2$ stolpcev) ter dve matriki z naključno izbranimi vrednostmi (v eni je bilo veliko ničel, v eni pa malo). Matrika strategije „eden proti vsem" se je obnesla slabše od ostalih, ostale pa približno enako dobro. V splošnem velja, da je za potrebe dekodiranja ugodneje, če so si vrstice matrike M čim dlje druga od druge (po Hammingovi razdalji), po drugi strani pa je ugodno, če je v M čim več ničel, saj imamo tako pri gradnji klasifikatorjev manjše in preprostejše učne množice. V tem smislu je videti strategija „eden proti enemu" dober kompromis. Do podobnih ugotovitev sta prišla tudi Hsu in Lin [25].

4.4.2 Razširitve optimizacijskega problema

Weston in Watkins [55, 56] sta predlagala dve razširitvi prvotnega optimizacijskega problema, s katerima lahko sestavimo klasifikator za več razredov. Tako lahko iščemo skupino k ravnin $(\mathbf{w}_m, b_m)$, $m = 1, \ldots, k$, ki jih bomo potem uporabljali po načelu: novi vektor pripišemo tistemu razredu, čigar ravnina mu da najvišjo oceno.

$$f(\mathbf{x}) = \arg\max_{1 \leq m \leq k} \{\langle \mathbf{w}_m, \mathbf{x} \rangle + b_m\}$$

Kriterijska funkcija je v tem primeru

$$\begin{gathered} P(\mathbf{w}, \boldsymbol{\xi}) = \tfrac{1}{2} \textstyle\sum_{m=1}^k ||\mathbf{w}_m||^2 + C \sum_{i=1}^l \sum_{m \neq y_i} \xi_i^m \\ \text{pri pogojih } \langle \mathbf{w}_{y_i}, \mathbf{x}_i \rangle + b_{y_i} \geq \langle \mathbf{w}_m, \mathbf{x}_i \rangle + b_m + 2 - \xi_i^m. \end{gathered}$$

Če je tak pogoj za nek i izpolnjen (in so kazenske spremenljivke ξ_i^m enake 0), to pomeni, da bi dobljeni klasifikator pravilno razvrstil učni vektor $\mathbf{x}_i$ (v razred y_i). Kazenske spremenljivke morajo biti nenegativne in imajo podobno vlogo kot pri tvorbi dvorazrednega klasifikatorja.

Iz opisanega optimizacijskega problema je mogoče z uporabo Lagrangeovih multiplikatorjev preiti na dualni problem, ki je spet podoben (4.6), vendar pa je spremenljivk pri kvadratnem programiranju zdaj $(k-1) \cdot l$, ne samo l.

Ista avtorja sta predlagala tudi naslednji pristop k tvorbi večrazrednih klasifikatorjev: dvorazredni klasifikator, kot nam ga da osnovna metoda podpornih vektorjev, je oblike

$$f(x) = \operatorname{sgn}\left(\textstyle\sum_{i=1}^l \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b\right),$$

kar lahko (pri $b_{+1} = \max\{b, 0\}$ in $b_{-1} = \max\{-b, 0\}$) zapišemo kot

$$f(x) = \operatorname{sgn}\left(\left(\textstyle\sum_{i: y_i = +1} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b_{+1}\right) - \left(\textstyle\sum_{i: y_i = -1} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b_{-1}\right)\right),$$

to pa je enakovredno $f(x) = \arg\max_{m \in \{+1, -1\}} \left\{\sum_{i: y_i = m} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle\right\}$.

Pri k razredih bi torej lahko uporabili klasifikator

$$f(x) = \arg\max_{m \in \{1, \ldots, k\}} \{f_m(\mathbf{x})\} \quad \text{za} \quad f_m(x) = \left\{\textstyle\sum_{i: y_i = m} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b_m\right\}.$$

Koeficienti α_i naj bi ustrezali pogojem

$$f_{y_i}(x_i) \geq f_m(x_i) + 2 - \xi_i^m \quad \text{za vse } i, \text{ za vse } m \neq y_i.$$

Tukaj torej spet poskušamo zagotoviti, da bi klasifikator pravilno razvrstil učne primerke, če pa to ne gre z dovolj širokim robom, se aktivirajo kazenske spremenljivke. Kriterijska fukcija naj bi bila lahko kar

$$\sum_{i=1}^{l} \alpha_i + C \sum_{i=1}^{l} \sum_{m \neq y_i} \xi_i^m .$$

V tem primeru imamo problem linearnega, ne kvadratnega, programiranja.

Avtorja ugotavljata, da niso klasifikatorji, dobljeni s tema pristopoma, po klasifikacijski točnosti nič boljši od klasifikatorjev tipa „eden proti vsem“, pač pa imajo manj podpornih vektorjev.

5

Poskusi

5.1 Metodologija

5.1.1 Načrt poskusov

Poskuse, ki jih opisuje to poglavje, lahko razdelimo v dve glavni skupini glede na to, kakšne opise slik uporabljamo. Pri prvi skupini opišemo celotno sliko z enim samim vektorjem (npr. histogramom); za razvrščanje dobljenih vektorjev sem uporabil metodo najbližjih sosedov in metodo podpornih vektorjev. Pri drugi skupini poskusov pa razdelimo sliko na več delov (regij) in opišemo vsakega posebej. Med takšnimi opisi slik je mogoče definirati različne mere podobnosti, preizkusil pa sem dve (WALRUS in IRM; gl. razd. 2.4). Na podlagi takšne mere podobnosti lahko slike spet razvrščamo z metodo najbližjih sosedov.

Zbirko slik, na kateri sem poskuse izvajal, opisuje naslednji razdelek. Pri vseh poskusih sem uporabil desetkratno prečno preverjanje: slike sem naključno razdelil v deset skupin (in pri tem pazil le na to, da je bil delež določenega razreda v vseh skupinah čim bolj podoben; temu pravimo „stratificirano prečno preverjanje“) in izvedel deset poskusov (ena od skupin postane testna množica, ostale pa tvorijo učno množico). Klasifikacijske točnosti, navedene v kasnejših razdelkih, so vedno povprečja po desetkratnem prečnem preverjanju.

5.1.2 Uporabljena zbirka slik

Videti je, da v literaturi s področja pregledovanja slikovnih zbirk ne obstaja nobena standardizirana zbirka slik, ki bi jo vsi uporabljali za lažje primerjanje rezultatov. Različni avtorji uporabljajo različne zbirke; priljubljene so na primer zbirke podjetja Corel in podobnih. Jaz sem se odločil za zbirko *misc,* ki je prosto dostopna (*http://www-db.stanford.edu/IMAGE/*) in se je že uporabljala v literaturi, na primer pri sistemih WBIIS [57] in WALRUS [36, 37]. Prvotni vir te zbirke je menda podjetje VIRAGE [37, razd. 6.2]. Sestavlja jo 9907 slik, ki so vse velikosti 85×128, 96×128 ali pa 128×85 slikovnih elementov. Skoraj vse so fotografski posnetki, tematsko pa je zbirka precej raznolika. Ugodna pri tej zbirki je tudi njena obvladljivost (slike so razmeroma majhne in ni jih tako zelo veliko) in pa to, da je v nekem smislu vendarle dosti bolj homogena, kot bi lahko bila, če bi slike brez posebne dodatne obdelave zbrali z veliko različnih virov (npr. s spletnih strani). S tem odpade ena od težav pri že tako dovolj kompleksnem problemu.

Tako kot v drugih podobnih zbirkah tudi v *misc* slike niso nikakor označene ali kategorizirane, niti nimajo kakšnih besednih ali semantičnih opisov. Slike z isto tematiko se ponekod pojavljajo skupaj, ena za drugo, pogosto pa so tudi premešane brez pravega reda. Zato sem vse slike ročno pregledal in izbral za nadaljnje poskuse 1172 slik, ki sem jih razvrstil v štirinajst razredov: metulji, ameriška zastava, sončni zahodi, jesen, cvetje, planeti, Zemlja iz vesolja, avtomobili, gore, oblaki, morje, jadralne deske, jadrnice, živali na travniku. Slika 5.1 prikazuje nekaj predstavnikov teh razredov. Podoben pristop so uporabili na primer Huang in sodelavci [24].

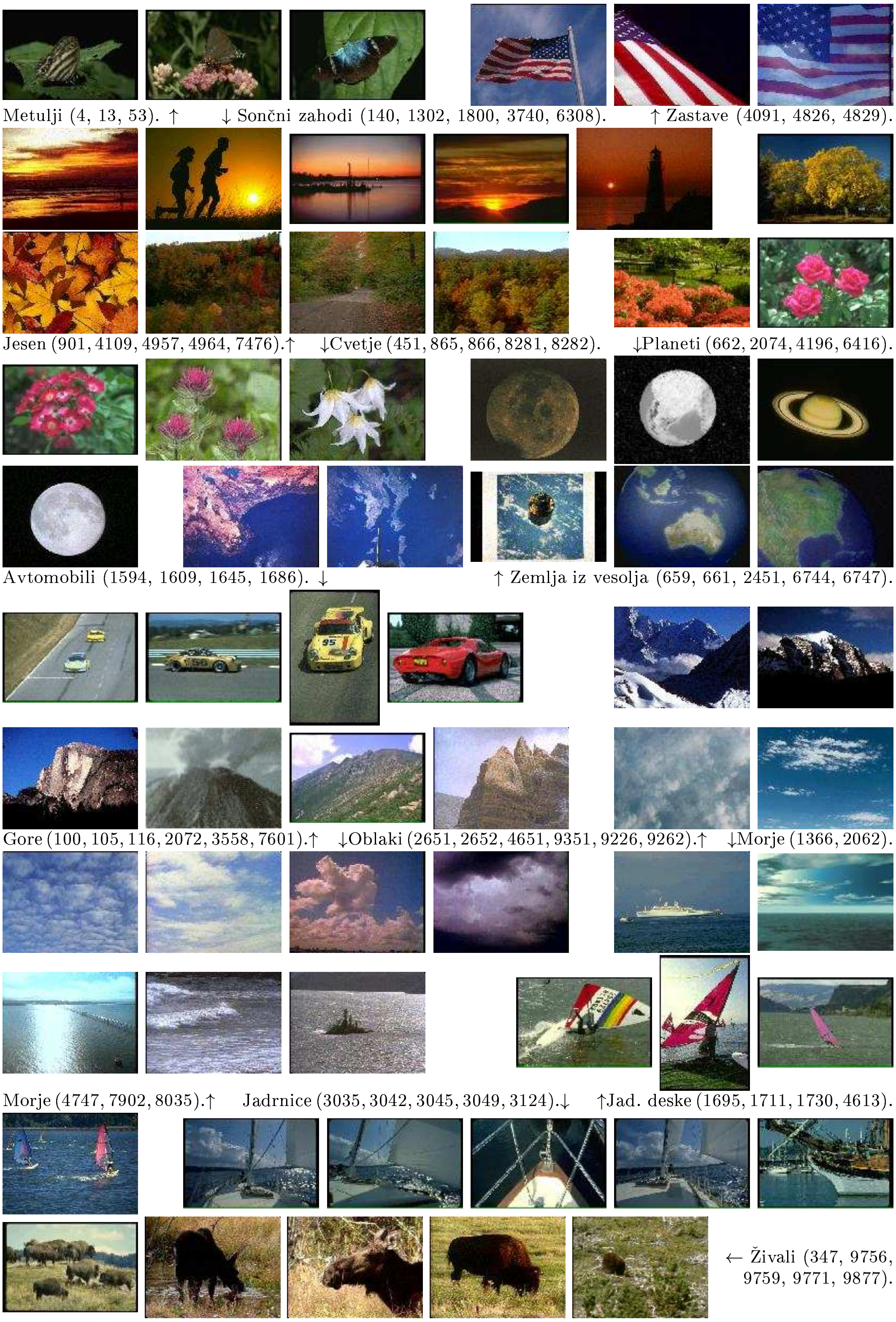
Pri izbiranju in razvrščanju slik sem si prizadeval, da bi predstavljala dobljena klasifikacija dober izziv preizkušanim tehnikam in algoritmom. Nekateri od teh razredov imajo značilne barve in splošni videz, po katerem naj bi se precej dobro ločili od ostalih. Živali v zadnjem razredu se na primer večinoma pojavljajo kot precejšnje črne lise na zelenkastem ozadju; značilne barve imajo tudi slike sončnih zahodov ali slike ameriške zastave; slike cvetja pa večinoma prikazujejo rdeče ali rožnate cvetove na zelenem ozadju. Nekateri razredi pa so si po barvah tudi precej podobni in zato pričakujemo, da jih bodo algoritmi slabše razlikovali med sabo. Slike metuljev imajo na primer pogosto zelo podobno ozadje kot slike cvetja, barve metuljev pa so med seboj različne in pogosto neizrazite, tako da klasifikatorju tudi ne bodo nujno dosti koristile. Za razrede, kot so gore, oblaki, jadralne deske in jadrnice je značilna velika količina modre in bele barve, zaradi katerih jih je pogosto ravno tako težko ločiti med sabo; jadralne deske in jadrnice so nekoliko manjši problem, ker so ti predmeti običajno precej drugačne barve kot morje okoli njih. Razred jesen vsebuje slike gozdov in dreves v jesenskih barvah; te rdečkaste in rjavkaste odtenke algoritmi včasih zamešajo s sončnimi zahodi ali cvetjem. V razredu avtomobili je skupna točka predvsem veliko sive barve (asfalt), avtomobili sami pa so precej različnih barv.

5.1.3 Mikropovprečja in makropovprečja

Pri problemih, kjer razvrščamo primerke v enega od več (disjunktnih) razredov, obstajata dve možnosti, kako izraziti točnost nekega klasifikatorja. Recimo, da klasifikator za vsak razred r ($1 \leq r \leq k$) pravilno razvrsti p_r primerkov tega razreda, ostalih n_r pa napačno (torej jih uvrsti v kakšen drug razred). Klasifikacijska točnost pri razredu r je torej $a_r \equiv p_r/(p_r + n_r)$, točnost po celotni množici testnih primerkov pa $a \equiv (\sum_{r=1}^k p_r)/\sum_{r=1}^k (p_r + n_r)$. Ta točnost je torej v bistvu uteženo povprečje vrednosti a_r, pri čemer je a_r obtežena z $(p_r + n_r)/\sum_{s=1}^k (p_s + n_s)$. Razredi z več predstavniki imajo na skupno točnost a večji vpliv od tistih z manj predstavniki. Tako dobljeni točnosti a pravimo „mikropovprečje" (*microaverage*), ker jo lahko izračunamo tako, da najprej seštejemo pare (p_r, n_r) po vseh razredih r; dobimo par (p, n) za $p = \sum_{r=1}^k p_r$, $n = \sum_{r=1}^k n_r$ in iz njega izračunamo klasifikacijsko točnost $a = p/(p + n)$.

Druga možnost pa je „makropovprečje" (*macroaverage*) [32], pri katerem najprej izračunamo točnost pri vsakem razredu, torej pri vsakem paru (p_r, n_r), za skupno točnost pa vzamemo povprečje točnosti razredov: $A \equiv \frac{1}{k}\sum_{r=1}^k a_r$. Na vrednost A vplivajo vsi razredi enako, ne glede na to, koliko primerkov je v vsakem.

V uporabljeni množici slik imajo nekateri razredi znatno več predstavnikov od



Slika 5.1: Primeri slik iz zbirke *misc*, skupaj z imeni razredov in številkami slik.

ostalih, ker je bilo v zbirki *misc*, iz katere sem te slike izbiral, slik nekaterih vrst pač bistveno več kot nekaterih ostalih. Da bi zmanjšal vpliv teh (v bistvu nezaželjenih) razlik v velikosti razredov, podajam v nadaljevanju tega poglavja pri opisovanju dobljenih klasifikatorjev vedno makropovprečja, ne mikropovprečij. Druga možnost bi bila, da bi nekatere razrede pač umetno zmanjšal, torej izbral zanje precej manj slik, kot jih je bilo mogoče na voljo, vendar se mi je zdelo primerneje ostati pri večji učni in testni množici.

Izkaže se, da so večji razredi pri uporabljeni množici slik večinoma taki, ki jih je lažje klasificirati, na primer cvetje in sončni zahodi (prim. tabelo 5.3). To je deloma posledica značilnega videza in barvne strukture slik v teh razredih, deloma pa tudi posledica dejstva, da je, če ima razred več predstavnikov, tudi bolj verjetno, da za neko testno sliko iz tega razreda obstaja tudi med učnimi kakšna, ki ji je zelo podobna, zato bo imel klasifikator več možnosti, da jo pravilno razvrsti. Iz navedenih razlogov so pri poskusih iz tega poglavja mikropovprečja (v katerih imajo večji razredi večji vpliv na točnost) praviloma za nekaj odstotkov večja od makropovprečij. Pomembno pa je, da so makropovprečja ravno tako primerna za medsebojno primerjanje različnih klasifikatorjev kot mikropovprečja; če je bil en klasifikator boljši od drugega po makropovprečju, je bil tudi po mikropovprečju in obratno.

5.2 Poskusi z globalnimi opisi slik

Pri tej skupini poskusov opišemo celotno sliko z enim samim vektorjem. Uporabil sem štiri vrste vektorskih opisov: histograme, koherenčne vektorje, avtokorelograme in zgoščene avtokorelograme (vsi so opisani v razdelku 2.2). Vsi ti pristopi predpostavljajo, da smo barvni prostor razdelili na neko razmeroma majhno število področij in da barv znotraj istega področja med seboj ne razlikujejo; uporabil sem štiri vrste takšne kvantizacije: RGB-64 (barvni prostor RGB, vsako od osi razdelimo na štiri enake dele; tako imamo paleto s 64 barvami) RGB-216 (vsako od osi razdelimo na šest enakih delov), RGB-332 (rdečo in zeleno os razdelimo na osem, modro pa na štiri dele; dobimo 256 barv) in HSV-422 (barvni prostor HSV, os z barvnimi odtenki razdelimo na 16 delov, ostali dve pa na štiri). Takšne oblike kvantizacije so uporabljali že tudi drugi, RGB-64 na primer v [43, 23, 49]; kvantizacija RGB-332 temelji na zamisli, da so modri odtenki temnejši in se zato manj pozna, če jih predstavimo manj natančno. Podobno izhajamo pri HSV-422 iz predpostavke, da je temeljni odtenek pomembnejši od zasičenosti in svetlosti.

Opisane možnosti nam dajo skupaj šestnajst vrst globalnih vektorskih opisov slik. Na teh vektorjih sem gradil klasifikatorje po metodi najbližjih sosedov (pri tem je treba določiti, kako se meri razdalja med vektorji; preizkusil sem L_1-normo in L_2-normo) in po metodi podpornih vektorjev (tu sem preizkusil več vrst jeder).

5.2.1 Metoda najbližjih sosedov

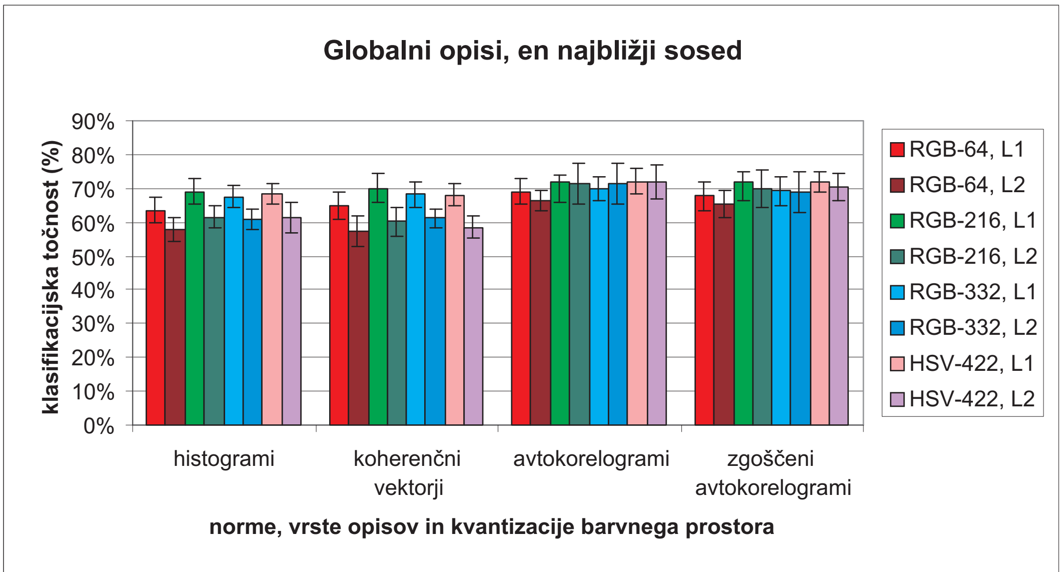
Točnosti klasifikatorjev, dobljenih z metodo enega najbližjega soseda, prikazujeta histogram na sliki 5.2 in tabela 5.1. Opazimo lahko, da so klasifikatorji, ki temeljijo na manhattanski razdalji (norma L_1), v skoraj vseh primerih boljši od tistih, ki temeljijo na evklidski razdalji (L_2). (Pri histogramih in koherenčnih vektorjih so te razlike tudi statistično signifikantne (stopnja zaupanja: $p = 0{,}95$), pri (navadnih in zgoščenih) avtokorelogramih pa ne, razen pri navadnih avtokorelogramih v RGB-64.) Podobno so opazili že Huang in sodelavci [23, str. 328a]; razlog naj bi bil v tem, da se večja razlika v kakšni barvi (ki sama po sebi še ne pomeni, da sta si sliki po videzu močno različni) pri L_1-normi ne pozna toliko kot pri L_2-normi.

Pri koherenčnih vektorjih, ki so v nekem smislu izboljšava histogramov, bi pričakovali, da bodo dajali tudi nekaj boljše klasifikatorje; to je sicer res, vendar so razlike v točnosti neznatne (nikjer več kot 2 %), čeprav signifikantne (razen pri HSV-422).

Avtokorelogrami, navadni in zgoščeni, dajejo boljše rezultate kot histogrami (le pri HSV-422 razlika ni signifikantna); razlike so večje pri uporabi bolj grobe kvantizacije (npr. RGB-64). L_2-norma daje tudi tu slabše rezultate od L_1-norme, vendar so razlike manjše. Oboje pomeni, da so avtokorelogrami robustnejši opisi slik.

Po pričakovanjih dajejo zgoščeni avtokorelogrami malo nižje točnosti kot navadni, kar je razumljivo, saj vsebujejo manj informacije o slikah. So pa razlike tudi tu majhne, približno 1 %. Navadni avtokorelogrami so signifikantno boljši od zgoščenih le pri kvantizaciji RGB-64, tako da so drugod zgoščeni avtokorelogrami pravzaprav boljši, saj dosegajo primerljive rezultate, čeprav zahtevajo precej manjše vektorje in so zato varčnejši s pomnilnikom in časom.

Rezultati pri kvantizaciji HSV-422 so signifikantno boljši od tistih pri RGB-216 in RGB-332, ti pa od tistih pri RGB-64; med RGB-216 in RGB-332 pa ni signifikantnih razlik.



Slika 5.2: Primerjava klasifikacijske točnosti klasifikatorjev po metodi najbližjega soseda v odvisnosti od vrste globalnih opisov slik. Stolpci so povprečja točnosti po prečnem preverjanju, črte pa se raztezajo za eno standardno deviacijo od povprečja v obe smeri.

Vrsta opisov	Kvantizacija in norma							
	RGB-64		RGB-216		RGB-332		HSV-422	
	L_1	L_2	L_1	L_2	L_1	L_2	L_1	L_2
histogrami	63,56	57,70	68,63	61,58	67,58	60,82	68,50	61,28
koherenčni vektorji	65,00	57,26	69,96	60,13	68,27	61,10	68,07	58,57
avtokorelogrami	69,12	66,17	71,67	71,34	69,84	71,30	71,97	71,77
zgoščeni avtokorelogrami	67,66	65,41	71,73	70,05	69,15	68,80	71,80	70,45

Tabela 5.1: Klasifikacijske točnosti (v odstotkih; povprečje po desetkratnem prečnem preverjanju), dosežene z metodo enega najbližjega soseda pri različnih vrstah globalnih opisov slik.

5.2.2 Metoda podpornih vektorjev

Vektorje, ki opisujejo slike, lahko uporabimo tudi za učenje po metodi podpornih vektorjev. Preizkusil sem klasifikatorje, ki temeljijo na treh različnih vrstah jeder: linearnih, kubičnih in radialnih (prim. razd. 4.2.2). Za učenje in klasifikacijo po metodi podpornih vektorjev sem uporabil program LIBSVM [12], ki so ga razvili na tajvanski univerzi. Ima več ugodnih lastnosti: je hiter, prosto dostopen, kompatibilen z znanim starejšim programom SVM*light*, za naš problem pa je pomembno predvsem, da podpira tudi večrazredne probleme (uporablja pristop „eden proti enemu"; gl. razd. 4.4). Pri radialnih jedrih sem za parameter σ uporabil kar privzeto vrednost programa LIBSVM ($1/\sigma^2 = 1/d$), ki se je vseskozi dobro obnesla. Drugih vrednosti σ nisem sistematično preizkušal, sem pa dobil občutek, da je mogoče σ razmeroma precej spreminjati, ne da bi se rezultati bistveno izboljšali ali poslabšati.

Pri delu z metodo podpornih vektorjev je potrebno določiti parameter C, ki v kriterijski funkciji (4.4) določa, kaj bo pri optimizaciji pomembnejše: širok rob (če je C majhen) ali malo napak na učni množici (če je C velik). Ker mora posredovati med števili ξ_i, ki so v bistvu razdalje narobe klasificiranih vektorjev od ravnine, merjene z dolžino normale $\hat{\mathbf{w}}$, in členom $||\hat{\mathbf{w}}||^2$ (ki je kvadrat dolžine normale), je C močno odvisen od reda velikosti dolžine normale; ker pa leži normala $\hat{\mathbf{w}}$ v prostoru $\mathcal{F}$, to pomeni, da je C odvisen od izbire jedra (saj je od jedra odvisno, kakšna sploh sta prostor $\mathcal{F}$ in preslikava vanj).

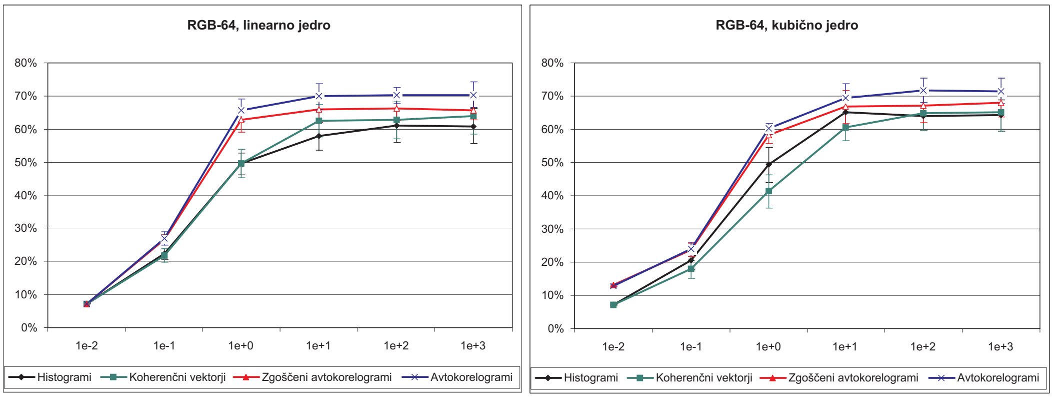
Iz teoretičnih razlogov je možen kandidat za parameter C vrednost R^{-2}, če je R radij krogle, ki vsebuje vse vektorje $\phi(\mathbf{x}_i)$ [14, str. 104]; to namreč v nekem izreku, ki podaja zgornjo mejo verjetnosti napake v odvisnosti od R in C, zagotovi najtesnejšo zgornjo mejo. Ker bi bilo najmanjšo tako kroglo težko poiskati, predlaga program SVM*light* [29] naslednjo vrednost:

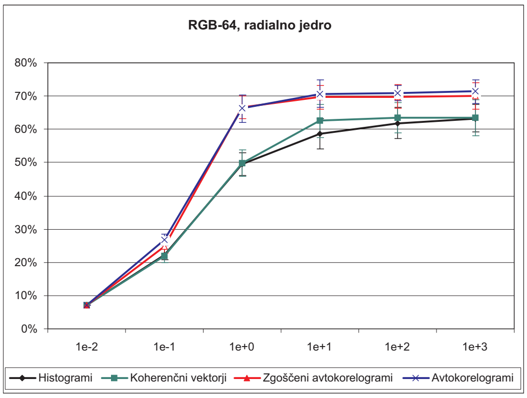
$$C_0 \equiv \left[\frac{1}{l} \sum_{i=1}^{l} ||\phi(\mathbf{x}_i) - \phi(\mathbf{0})||_{\mathcal{F}}^2 \right]^{-2}, \tag{5.1}$$

mislimo si torej, da bi središče krogle postavili v $\phi(\mathbf{0})$ in vzeli za radij povprečno razdaljo do slik $\phi(\mathbf{x}_i)$.

V splošnem pa velja, da je dobro v praksi preizkusiti več različnih C; zato sem vzel kot izhodišče vrednost C_0 po predlogu (5.1), poleg tega pa preizkusil še $C = C_0 \cdot 10^e$ za $e \in \{-2, -1, 0, 1, 2, 3\}$. Izkazalo se je, da je $C = C_0$ običajno premajhna in nam dá klasifikatorje s širokim robom, toda preveč klasifikacijskimi napakami; $C_0/10$ in $C_0/100$ sta bili v tem pogledu seveda še slabši. Najboljši vrednosti sta bili običajno $10C$ in $100C$, pri $1000C$ pa se točnost na učni množici praviloma ne izboljša več (pač pa je učenje pogosto počasnejše, ker se algoritem trudi, da bi se čim bolj prilagodil učni množici in se tako izognil visokim kaznim za napačno klasificirane učne vektorje).

Točnosti pri različnih vrednostih 10^e in različnih vrstah jeder (vse za kvantizacijo RGB-64) prikazujejo grafi na sliki 5.3. Ti grafi se nanašajo na učenje na vektorjih, dobljenih pri kvantizaciji RGB-64, vendar pa so jim grafi pri drugih kvantizacijah po obliki zelo podobni.



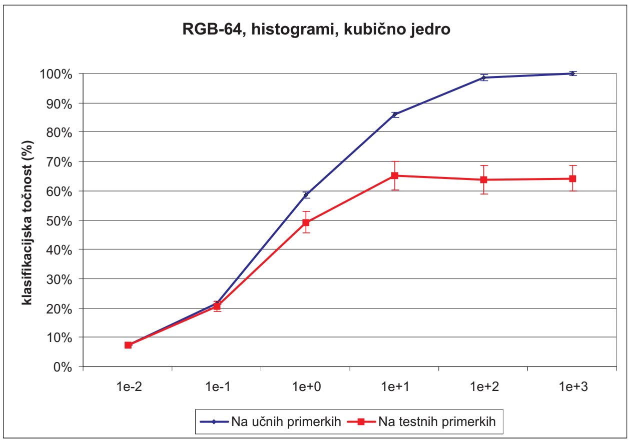


Grafi prikazujejo klasifikacijske točnosti klasifikatorjev po metodi podpornih vektorjev (pri različnih vrstah jeder in opisov slik) pri različnih vrednostih parametra C. Os x je logaritemska in pri vrednosti 10^e so podani rezultati klasifikatorja s parametrom $C = C_0 \cdot 10^e$. V vseh primerih je bilo uporabljeno desetkratno prečno preverjanje.

Slika 5.3: Vpliv parametra C.

Odnos med točnostjo na testni in na učni množici pri različnih vrednostih parametra C pa je prikazan na grafu 5.4. Opazimo lahko, da sta pri majhnih vrednostih C točnost na učni in na testni množici podobni (in nizki), kar pomeni, da se klasifikator takrat niti testni množici še ni primerno prilagodil (ker ga klasifikacijske napake premalo stanejo in raje išče čim širši rob). Pri višjih vrednostih C pa točnost na učni množici vse bolj narašča (s tem pa tudi trajanje učenja), na testni množici pa ne več (in celo malo pade), kar dokazuje, da prihaja do pretiranega prilagajanja učni množici.

Tabela 5.2 in graf 5.5 pa prikazujeta točnosti vseh preizkušenih vrst jeder in opisov pri najboljši izmed preizkušenih vrednosti parametra C. Standardne deviacije v točnosti po prečnem preverjanju znašajo od 2 do 5 %, tako da, če se dva



Graf prikazuje klasifikacijsko točnost na učni in na testni množici pri različnih vrednostih parametra C. Os x je kot pri grafih na sliki 5.3. Tudi ta graf prikazuje povprečja točnosti po desetkratnem prečnem preverjanju. Standardne deviacije so prikazane tako za učne kot za testne podatke, vendar so tiste za učne podatke zelo majhne.

Slika 5.4: Vpliv parametra C.

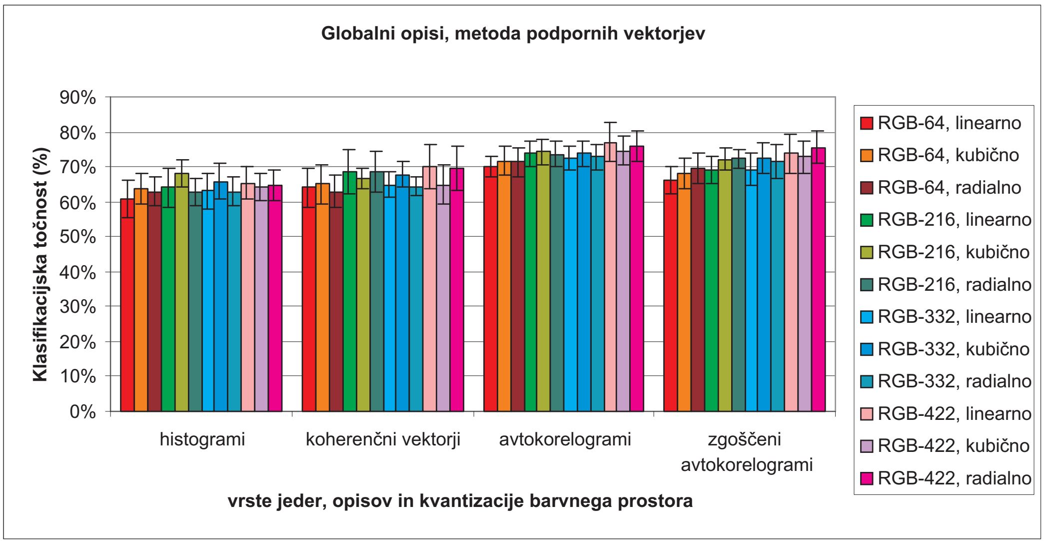
klasifikatorja po točnosti razlikujeta le za okoli 1 % (ali še manj), ta razlika običajno ni statistično signifikantna, razlika dveh ali več odstotkov pa običajno je.

Kvantizacija	Jedro	Vrsta opisov: histogrami	koherenčni vektorji	avtokorelogrami	zgoščeni avtokor.
RGB-64	linearno	61,07 %	63,99 %	70,37 %	66,13 %
	kubično	65,19 %	65,16 %	71,68 %	68,08 %
	radialno	63,32 %	63,42 %	71,36 %	70,06 %
RGB-216	linearno	64,00 %	68,53 %	73,75 %	68,90 %
	kubično	68,19 %	66,79 %	74,04 %	72,13 %
	radialno	63,92 %	69,17 %	74,05 %	72,73 %
RGB-332	linearno	63,95 %	65,94 %	73,97 %	71,86 %
	kubično	65,97 %	67,64 %	73,65 %	72,56 %
	radialno	63,88 %	66,24 %	73,94 %	72,65 %
HSV-422	linearno	65,43 %	69,94 %	76,94 %	73,69 %
	kubično	64,11 %	64,65 %	74,44 %	72,68 %
	radialno	65,50 %	70,02 %	77,03 %	76,81 %

Tabela 5.2: Klasifikacijske točnosti pri najboljši vrednosti parametra C za različne vrste opisov in jeder. Vse točnosti so povprečja po desetkratnem prečnem preverjanju.

- Opazimo lahko, da so odnosi med različnimi vrstami opisov zelo podobni tistim pri metodi najbližjih sosedov: koherenčni vektorji so malenkost boljši od histogramov (predvsem pri linearnih jedrih), avtokorelogrami (navadni in zgoščeni) so občutno boljši od obojih; zgoščeni avtokorelogrami so malo slabši od navadnih (predvsem pri linearnih in kubičnih jedrih).

- Tudi glede kvantizacije so opažanja podobna: kvantizacije z več barvami so v splošnem boljše. RGB-64 je značilno slabša od vseh ostalih; razlike med RGB-216 in RGB-332 večinoma niso signifikantne. Razlike med kvantizacijami so



Slika 5.5: Primerjava klasifikacijske točnosti klasifikatorjev po metodi podpornih vektorjev v odvisnosti od vrste globalnih opisov slik. Stolpci so povprečja točnosti po prečnem preverjanju, črte pa se raztezajo za eno standardno deviacijo od povprečja v obe smeri.

večje pri linearnih jedrih, drugod manjše. Pač pa ima tukaj HSV-422 opaznejšo prednost pred kvantizacijami, temelječimi na prostoru RGB, česar pri metodi najbližjih sosedov ni bilo opaziti.

- Kubična in radialna jedra so pri večini opisov boljša od linearnih, vendar ne povsod (zlasti ne pri histogramih in navadnih avtokorelogramih); med kubičnimi in radialnimi jedri pa ni signifikantnih razlik.
- Klasifikatorji, ki jih dobimo z linearnim jedrom, so približno enako dobri kot tisti pri metodi najbližjih sosedov, z drugimi jedri pa lahko dosežemo do pet ali sedem odstotkov višje točnosti.

5.2.3 Primer kontingenčne tabele

Kontingenčna tabela je tabela, ki prikazuje, kam razvršča nek klasifikator predstavnike posameznih razredov. Za vsak par razredov (i, j) je v njej podatek o tem, koliko predstavnikov razreda i je klasifikator uvrstil v razred j. Izraz „kontingenčna tabela“ je pogost v literaturi s področja dostopanja do informacij (*information retrieval*), vendar imajo tam običajno opraviti le z dvema razredoma in je ta tabela zato manjša. Huang in sodelavci [24] pravijo takšni tabeli „matrika zamenjav“ (*confusion matrix*).

Tabela 5.3 prikazuje kontingenčno tabelo najboljšega dobljenega klasifikatorja. Idealna kontingenčna tabela bi imela povsod razen na diagonali same ničle, neničelni elementi pa kažejo na značilne oblike pomot pri klasifikaciji. Opazimo lahko, da se struktura te tabele precej dobro ujema s tem, kar bi človek glede na podobnost slik iz različnih razredov pričakoval. Klasifikator pogosto pomeša razrede: metulji in cvetje; gore in oblaki; morje, jadrnice, jadralne deske. Najbolj neizrazit razred je morje; slike iz tega razreda se zaradi veliko modre in bele barve pogosto uvrstijo v razreda gore ali oblaki. Podobne kontingenčne tabele imajo tudi drugi klasifikatorji.

				met	zas	son	jes	cve	pla	zem	avt	gor	obl	mor	sur	jad	živ
met	84	17	83,17 %	84	1	1	1	9	1	0	0	0	0	0	2	0	2
zas	26	6	81,25 %	0	26	2	0	1	1	0	0	1	0	1	0	0	0
son	194	30	86,61 %	7	0	194	6	8	4	0	3	1	0	0	1	0	0
jes	51	17	75,00 %	0	0	4	51	9	1	0	0	1	0	0	1	0	1
cve	172	29	85,57 %	8	1	6	5	172	3	1	2	0	0	0	1	0	2
pla	38	18	67,86 %	2	0	6	1	3	38	2	0	1	1	2	0	0	0
zem	48	13	78,69 %	1	0	0	0	0	0	48	1	5	2	2	2	0	0
avt	89	10	89,90 %	1	1	1	0	1	1	1	89	0	0	0	2	2	0
gor	39	21	65,00 %	0	0	0	2	0	2	3	2	39	4	3	3	2	0
obl	44	21	67,69 %	0	0	2	0	0	2	2	0	7	44	6	1	0	1
mor	21	14	60,00 %	2	0	0	0	0	0	0	0	4	6	21	0	1	1
sur	54	17	76,06 %	2	0	1	1	2	0	2	5	0	0	1	54	3	0
jad	27	10	72,97 %	0	0	0	0	0	0	0	3	2	0	0	5	27	0
živ	55	7	88,71 %	3	0	0	1	1	0	1	0	0	0	0	1	0	55

Tabela 5.3: Kontingenčna tabela najboljšega klasifikatorja (SVM, radialno jedro, avtokorelogrami pri kvantizaciji HSV-422). Makropovprečje točnosti po razredih je 77,03 %, mikropovprečje pa 80,37 %. Na preseku vrstice i in stolpca j je število predstavnikov razreda i, ki jih je klasifikator uvrstil v razred j. Drugi in tretji stolpec podajata število pravilno in napačno razvrščenih predstavnikov posameznih razredov, četrti pa točnost pri tem razredu. O imenih razredov gl. razd. 5.1.2; „jad“ = jadrnice, „sur“ = jadralne deske (surfi).

5.2.4 Primerjava vseh klasifikatorjev tega razdelka

Različne klasifikatorje lahko primerjamo po klasifikacijski točnosti, so pa še drugi načini. Pri desetkratnem prečnem preverjanju lahko na primer za vsako razbitje podatkov na učno in testno množico uredimo klasifikatorje po točnosti, nato pa za vsak klasifikator izračunamo povprečje njegovih uvrstitev v teh desetih vrstnih redih.

Na oba načina sem razvrstil vse klasifikatorje, ki jih omenja ta razdelek: tiste po metodi najbližjega soseda (norma L_1 ali L_2; skupaj 32 klasifikatorjev) in tiste po metodi podpornih vektorjev (tri vrste jeder, vsak pri najboljšem C; skupaj 48 klasifikatorjev). Vsak klasifikator torej dobi neko povprečno uvrstitev med 1 in 80.

Oba vrstna reda sta precej podobna, čeprav ne čisto enaka. V obeh primerih so na vrhu naslednji trije klasifikatorji z metodo podpornih vektorjev: (1) radialno jedro, kvantizacija HSV-422, zgoščeni korelogrami; (2) enako, a navadni korelogrami; (3) avtokorelogrami, HSV-422, linearno jedro. Njihove povprečne uvrstitve so 8,1, 8,4 in 9,4. Poučno pri tem je, da torej ni klasifikatorja, ki bi zmagoval pri vseh desetih razbitjih; če je najboljša povprečna uvrstitev okoli 8, to pomeni, da je vsak klasifikator pri kakšnem razbitju slab in da se vrstni red na vrhu precej spreminja. Najslabši klasifikator je bil v obeh razvrstitvah tisti po metodi enega najbližjega soseda na histogramih (norma L_2) pri kvantizaciji RGB-64. Pogled na celotni vrstni red pokaže, da so avtokorelogrami (navadni ali zgoščeni) pri vrhu; norma L_2 pri metodi najbližjih sosedov daje najslabše rezultate; kvantizacija HSV-422 daje najboljše rezultate, RGB-64 najslabše. V srednjem delu razvrstitve so precej pomešani različni klasifikatorji po metodi najbližjega soseda in po metodi podpornih vektorjev; zato ne moremo reči, da je ena metoda v splošnem in vedno bistveno boljša od druge.

5.3 Poskusi v zvezi s segmentacijo slik

Za segmentacijo sem uporabljal algoritem WALRUS, pri katerem pa je mogoče nastavljati več parametrov, ki vplivajo na segmentacijo. Sliko pretvorimo v barvni prostor YCC; na vsakem oknu velikosti $w \times w$, pri katerem sta koordinati zgornjega levega kota večkratnika števila t, izračunamo Haarovo transformacijo in si zapomnimo njene glavne štiri koeficiente; ker storimo to za vsakega od treh barvnih kanalov, dobimo kot opis okna na ta način vektor z 12 komponentami. Vektorje združimo v skupine (*clustering*), za kar sem po zgledu avtorjev WALRUSa uporabil postopek BIRCH [59, 58], ki je hiter, dokaj preprost in primeren za probleme, pri katerih imamo opravka z veliko vektorji. Od vsake skupine si zapomnimo centroid (povprečje vseh vektorjev v njej) in še področje, ki ga ta centroid zdaj predstavlja: to je unija vseh oken, katerih vektorji so prišli v to skupino. Centroid nam torej rabi kot opis videza določene regije.

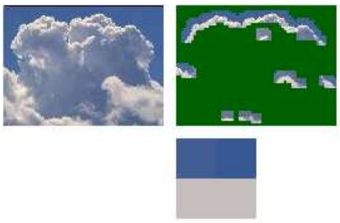
Za primerjavo tako segmentiranih slik sem uporabil dva pristopa: tistega iz postopka WALRUS in pa algoritem IRM. Oba sta opisana v razdelku 2.4. Iz primerjave postopkov pričakujemo, da bo dajal IRM boljše ocene podobnosti slik, vendar pa je tudi počasnejši. Razlika med njima je tudi v tem, da WALRUS dopušča, da se regije prekrivajo (torej $t < w$), IRM pa je bil zamišljen za disjunktne regije in se je kasneje izkazalo, da se pri prekrivajočih se regijah obnese slabše.

Mera podobnosti pri postopku WALRUS temelji na tem, da poišče vse pare regij (ena iz prve slike, ena iz druge slike), katerih opisi so si dovolj blizu (L_∞-razdalja manj kot ε). Potem pogleda, kolikšen delež površine obeh slik pokrivajo regije v teh parih; ta delež je mera podobnosti med slikama. Zato so rezultati pri uporabi te mere podobnosti odvisni ne le od segmentacije, pač pa tudi od parametra ε. IRM nima posebnih parametrov te vrste.

Obe meri podobnosti lahko uporabimo kot osnovo za razvrščanje slik po metodi najbližjih sosedov. Metode podpornih vektorjev pa tu nisem uporabljal, ker ta predpostavlja, da je vsak učni primerek en sam vektor, tu pa imamo zdaj za vsako sliko po več vektorjev (po enega za vsako regijo).

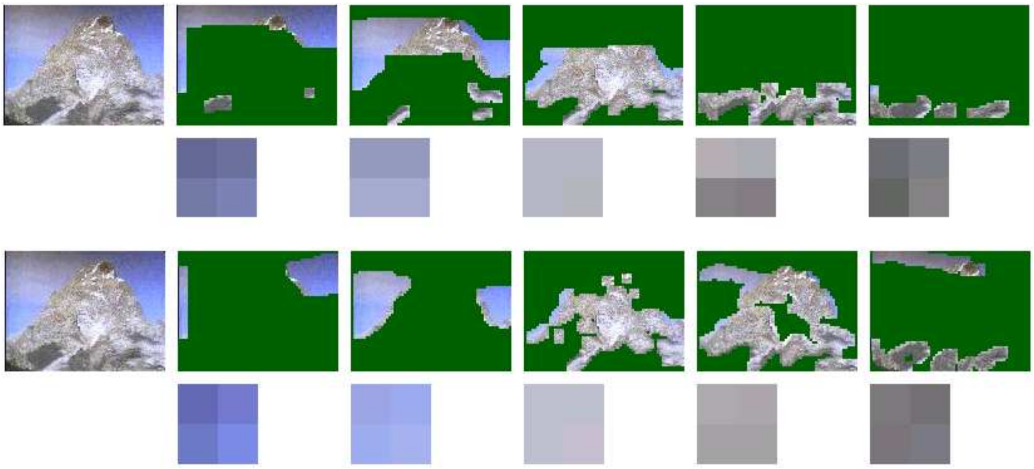
5.3.1 Komentarji k segmentaciji

Sliki (5.6a) in (5.6b) prikazujeta dva problematična vidika postopkov segmentacije, kakršne uporablja postopek WALRUS. Za opis vsakega okna se uporablja vektor, ki podaja povprečje celega okna in odstopanja povprečij manjših delov od tega povprečja. Zato imajo okna ob robovih med predmeti (ali pa med predmeti in ozadjem) običajno zelo značilne in izrazite vektorje, ki se dobro ločijo od ostalih in zato pogosto ustvarijo samostojne regije. Pri segmentaciji slike oblaka lahko na primer poleg regij za oblak in regij za nebo nastane tudi regija, ki sledi meji med oblakom in nebom (slika 5.6a).



Prva slika v vsaki skupini je slika, ki jo segmentiramo, nadaljnje slike pa kažejo nekaj regij. Področje, ki določeni regiji ne pripada, je pobarvano temno zeleno. Pod sliko regije je prikazana 2×2 slikovna elementa velika sličica, kot bi jo rekonstruirali iz vektorja, ki opisuje regijo.

(a) 4271.jpg.



(b) 7964.jpg.

Slika 5.6: Težave pri segmentaciji. Postopek WALRUS, $w = 8$, $t = 2$.

Druga slabost je potencialno neugodna segmentacija, če se posveti premalo pozornosti postopku za združevanje vektorjev v skupine. Natsev in sodelavci so menda uporabili le prvo fazo postopka BIRCH, ki tvori skupine in jih povezuje v drevesasto strukturo, s katero si pomaga, da čim hitreje najde skupino, ki je najbližja določenemu novemu elementu in v katero ga je zato pametno dodati. Ta strategija je sicer praktična in pri delu z veliko podatki zelo koristna, vendar pa pomeni, da marsikateri vektor ne bo prišel v zares najbližjo skupino, ker bo algoritem zaradi na videz ugodnejših statistik v višjih delih drevesa zablodil v kak drug del drevesa.

Da bi take težave omilili, lahko opravimo še enega ali več dodatnih prehodov skozi podatke; od vsake skupine si zapomnimo le centroid, nato pa vse vektorje na novo razporedimo (vsakega dodelimo skupini z najbližjim centroidom). Ob tem se lahko število skupin tudi zmanjša (če h kakšnemu centroidu ob prerazporejaju ne pride noben vektor). Slika (5.6b) prikazuje v prvi vrstici nekaj regij, dobljenih ob prvotni tvorbi skupin, v drugi vrstici pa iste regije po šestih dodatnih prehodih skozi podatke. Opazimo lahko, da so nove regije bolj homogene in se tudi bolje ujemajo s pojmi, ki bi jih na sliki opazil človek (gora, nebo). Vendar pa se je pri poskusih pokazalo, da na kakovost dobljenih klasifikatorjev takšna izboljšava segmentacije nima vedno opaznega učinka.

5.3.2 Vpliv parametrov segmentacije

Preizkusil sem naslednje vrste segmentacije: $w = 8, t \in \{2, 4, 8\}$; in $w = 4, t \in \{2, 4\}$. Pri vsaki sem poskusil začetno segmentacijo tudi izboljšati z dodatnimi prehodi skozi podatke, kot je to opisano v prejšnjem razdelku. Poskusov z mero podobnosti po postopku IRM nisem poganjal pri vseh segmentacijah, ker so začetni poskusi pokazali, da se IRM pri prekrivajočih se regijah obnese zelo slabo, in ker so poskusi z IRM precej zamudni.

Rezultate (za klasifikator po metodi enega najbližjega soseda) prikazujeta tabela 5.4 in graf 5.7. Iz njiju lahko potegnemo naslednje ugotovitve:

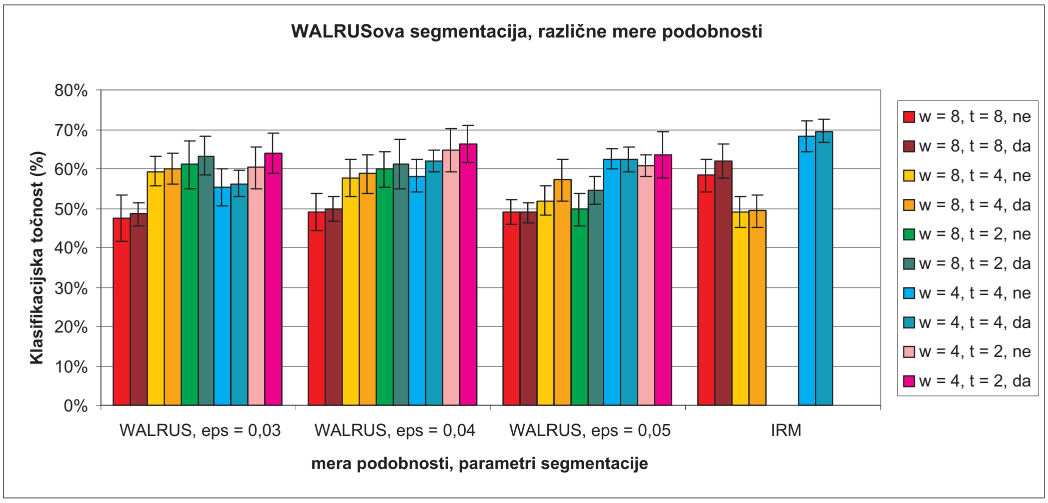
Parametri segmentacije			Povprečno	Mera podobnosti			
		Dodatni	število regij	WALRUS, $\varepsilon =$			IRM
w	t	prehodi	na sliki	0,03	0,04	0,05	
8	8	ne	10,2150	47,75	49,12	48,93	58,46
8	8	da	10,2116	48,54	49,75	48,82	61,93
8	4	ne	16,9198	59,19	57,44	51,70	48,86
8	4	da	16,9138	60,21	58,63	56,86	49,09
8	2	ne	27,2159	61,13	59,72	49,50	
8	2	da	27,2039	63,24	61,20	54,43	
4	4	ne	20,3865	55,33	58,07	62,57	68,30
4	4	da	20,3746	56,30	62,03	62,45	69,55
4	2	ne	37,1732	60,11	64,58	60,80	
4	2	da	37,1561	64,17	66,21	63,39	

Tabela 5.4: Klasifikacijske točnosti (v odstotkih; povprečje po desetkratnem prečnem preverjanju), dosežene z metodo enega najbližjega soseda pri različnih segmentacijah in merah podobnosti.

- Segmentacije z večjim številom regij dajejo boljše rezultate kot tiste z manjšim (vendar pa je v tem primeru tudi klasifikacija počasnejša; pri IRM je časovna zahevnost sorazmerna s kvadratom števila regij).

- Če uporabljamo mero podobnosti iz postopka WALRUS, je dobro uporabljati prekrivajoča se okna ($t < w$). To je precej razumljivo, saj gosteje posuta okna omogočajo več svobode pri obliki regij, tako da se lahko regije bolje prilagajajo oblikam pojavov na sliki. $w = 4, t = 2$ daje na primer signifikantno boljše rezultate od $w = 4, t = 4$; $w = 8, t \in \{2, 4\}$ pa boljše od $w = 8, t = 8$; pač pa razlike med $w = 8, t = 2$ in $w = 8, t = 4$ niso statistično signifikantne.

- Za parameter ε pri WALRUSu je v splošnem težko priporočiti najprimernejšo vrednost. V splošnem velja, da premajhen ε povzroči, da tudi tistih slik, ki so podobne določeni sliki, ne prepoznamo kot take, prevelik ε pa, da tudi tiste, ki ji v resnici niso podobne, razglasimo za zelo podobne. Pri tem poskusu začne $\varepsilon = 0{,}05$ v večini primerov že kvariti rezultate, med ostalima dvema pa je $\varepsilon = 0{,}04$ mogoče malo boljši, vendar so razlike zelo majhne in statistično največkrat nesignifikantne.



Slika 5.7: Primerjava klasifikacijske točnosti klasifikatorjev po metodi enega najbližjega soseda v odvisnosti od parametrov segmentacije in uporabljene mere podobnosti. Stolpci so povprečja točnosti po prečnem preverjanju, črte pa se raztezajo za eno standardno deviacijo od povprečja v obe smeri. Parametri segmentacije so kot v tabeli 5.4.

- Dodatni prehodi skozi vektorje pri segmentaciji sicer malenkostno izboljšajo klasifikacijsko točnost dobljenih klasifikatorjev, vendar so te izboljšave neznatne in pogosto statistično nesignifikantne. Res pa je tudi, da ti dodatni prehodi nič ne škodujejo in nas zelo malo stanejo (čas, ki ga porabimo zanje, je razmeroma majhen v primerjavi s časom, ki ga porabimo za prvo fazo segmentacije in kasneje za klasifikacijo). Tudi pri IRM dodatni prehodi ne prinesejo signifikantnih sprememb v točnosti, razen pri $w = t = 8$.

- Mera podobnosti IRM daje bistveno boljše rezultate (več kot 10 % višjo točnost) kot WALRUSova mera podobnosti. Kot že rečeno, pa je njena slabost v tem, da je klasifikacija z IRM veliko počasnejša; poleg tega daje zelo slabe rezultate, če regije niso disjunktne (do tega pride pri $t < w$). Opazimo lahko tudi, da se IRM bolje obnese pri segmentaciji z manjšimi okni ($w = 4$). To je po svoje spodbudno, ker so tudi avtorji postopka IRM delali z okni velikosti 4×4 [35].

Pomembna splošna ugotovitev pri tem poskusu je tudi ta, da je klasifikacijska točnost dobljenih klasifikatorjev precej močno odvisna od segmentacije. S spreminjanjem parametrov segmentacije smo lahko pri isti meri podobnosti (WALRUS, $\varepsilon = 0{,}04$) prišli od 49 % do 66 %. Mogoče bi se dalo z nadaljnjim prilagajanjem parametrov segmentacije dobljeno točnost še izboljšati.

Druga pomembna ugotovitev je ta, da se najboljši klasifikatorji, dobljeni pri tem poskusu, sicer lahko kosajo s tistimi, dobljenimi pri uporabi globalnih opisov (in metode najbližjih sosedov), vendar pa niso nič boljši od njih, čeprav bi glede na naprednejši pristop to pričakovali; in so slabši od tistih, ki smo jih pri uporabi globalnih opisov dobili z metodo podpornih vektorjev.

5.3.3 Različni opisi regij pri isti segmentaciji

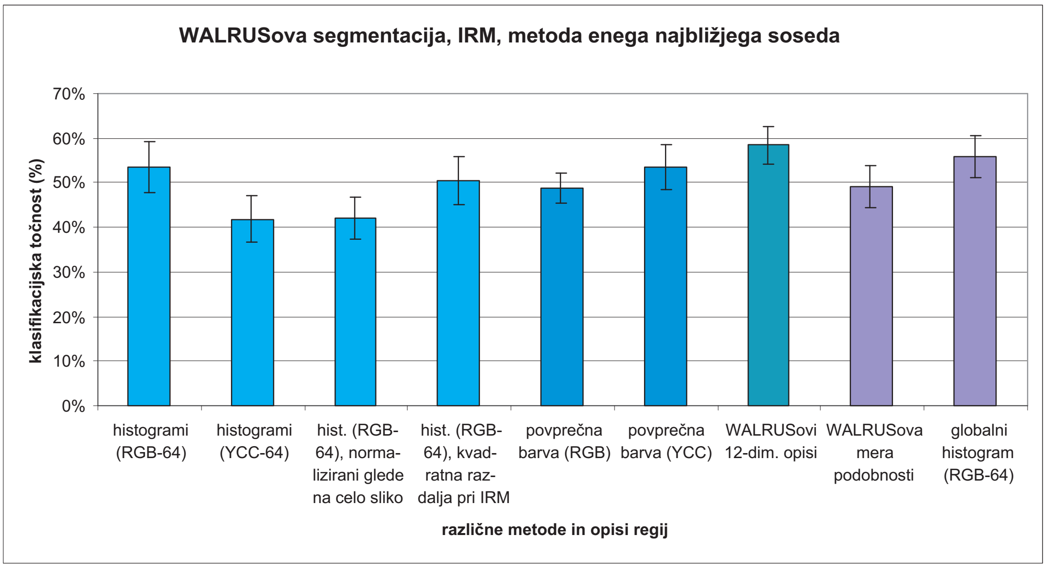
Namen tega poskusa je preveriti, kakšen je vpliv različnih tehnik opisovanja regij na klasifikacijsko točnost, če je postopek segmentacije slik ves čas enak. Uporabil sem segmentacijo iz postopka WALRUS, ki razreže sliko na okna velikosti 8×8, vsako okno opiše z glavnimi štirimi koeficienti njegove valčne transformiranke (nastane dvanajst koeficientov, ker imamo tri barvne kanale) ter nato dobljene vektorje združi v skupine. Nastalo je povprečno 10,2 regij na sliko. Za mero podobnosti med slikami sem uporabil postopek IRM, nad tako dobljeno mero podobnosti pa sem gradil klasifikatorje po metodi najbližjih sosedov.

Tudi če so vse naštete reči določene, pa lahko spreminjamo način opisovanja posameznih regij. V vseh primerih opišemo regije z nekakšnimi vektorji, IRM pa potem računa (evklidske) razdalje med njimi in jih kombinira v razdaljo med slikama kot celotama. Preizkusil sem naslednje vrste opisov regij: opisi iz postopka WALRUS (regija je pri segmentaciji nastala iz neke skupine 12-razsežnih vektorjev in za opis regije vzamemo kar njihov centroid); povprečna barva regije (kot urejena trojica, izražena v barvnem prostoru YCC ali RGB); in različne vrste histogramov: eni temeljijo na barvnem prostoru RGB, ki ga kvantiziramo v 64 barv (vsako os razdelimo na štiri dele), eni pa na enaki kvantizaciji prostora YCC. Histogrami so bili načeloma normalizirani glede na regijo, ki so jo predstavljali (torej se komponente vsakega histograma seštejejo v 1); pri histogramih iz prostora RGB sem preizkusil tudi možnost, da so normalizirani glede na celotno sliko (tako se v 1 seštejejo šele vse komponente histogramov vseh regij določene slike). Pomislil sem tudi, da bi lahko v formuli (2.3) namesto razdalje d_{ij} vzel njen kvadrat, d_{ij}^2, kar naj bi program odvračalo od tega, da slike prerad spoznava za podobne.

Histograme sem v ta poskus vključil zato, ker se v literaturi kot ena od možnosti, kako izboljšati postopke, ki temeljijo na histogramih, navaja zamisel, da bi sliko razdelili na več delov in izračunali histograme posameznih delov. Vendar pa so tako opisani pristopi večinoma statični (razdelitev slike na dele ni odvisna od vsebine slike, ampak je določena vnaprej in enaka za vse slike). To jim omogoča primerjanje istoležnih histogramov med sabo, torej lahko vse histograme v mislih staknemo skupaj v en sam dolg vektor, vendar je lahko to tudi problematično, saj otežkoči prepoznavanje podobnosti v primerih, ko se nek predmet premakne na drug del slike. Če hočemo histograme kombinirati z dinamično segmentacijo, pa moramo uporabiti kak postopek, kot je IRM, ki iz podobnosti med regijami izračuna oceno podobnosti med slikama.

Rezultate tega poskusa prikazuje graf na sliki 5.8. Od histogramskih opisov so se še najbolje obnesli navadni histogrami v prostoru RGB, normalizirani glede na svojo regijo. Zamisli, da bi se histogrami nanašali na barvni prostor YCC, ali pa, da bi bili normalizirani glede na površino cele slike, sta dali klasifikatorje z 11 ali 12 % nižjo klasifikacijsko točnostjo. Ravno tako se ni obnesla zamisel o uporabi kvadrata razdalje pri postopku IRM, čeprav se je točnost tu poslabšala le za kake 3 %.

Zanimiva je primerjava s klasifikatorji, ki temeljijo na globalnem histogramu. Pri njih torej slike ne delimo na regije, ampak izračunamo histogram za celotno sliko in za mero razdalje med slikama uporabimo kar razdaljo med njunima histogramoma. Več o teh klasifikatorjih je v razdelku 5.2.1, na grafu 5.8 pa je za primerjavo prikazan tisti, ki je najbolj primerljiv z ostalimi histogramskimi metodami na tem grafu:



Slika 5.8: Primerjava klasifikacijske točnosti pri različnih vrstah opisov regij; mera podobnosti med slikami je IRM. Vse temeljijo na isti razdelitvi slik na regije ($w = 8$, $t = 8$, brez dodatnih prehodov skozi podatke). Za primerjavo je podana še točnost klasifikatorja, ki temelji na globalnem histogramu (RGB-64, norma L_2), in tistega, ki temelji na meri podobnosti postopka WALRUS ($\varepsilon = 0{,}04$).

uporablja kvantizacijo prostora RGB v 64 barv, za razdaljo med histogramoma pa uporablja L_2-normo. Tako dobljeni klasifikatorji so imeli za približno 4 % višjo točnost od najboljših klasifikatorjev, temelječih na segmentaciji in histogramskih opisih.

Klasifikatorji, pri katerih je regija opisana samo s povprečno barvo vseh slikovnih elementov v njej, kažejo, da je prostor YCC za te namene vendarle primernejši od RGB (razlika v klasifikacijski točnosti je približno 5 %); to je tudi pričakovano, ker se v YCC evklidske razdalje med barvnimi vektorji bolje ujemajo s tem, koliko različni se zdita ustrezni barvi človeškemu opazovalcu. Pri tem dajejo opisi s povprečno barvo v YCC približno enako dobre klasifikatorje kot opisi s histogrami (pri kvantizaciji prostora RGB na 64 barv); zato so pravzaprav veliko boljši način opisovanja regij, saj so veliko bolj ekonomični od histogramov (povprečno barvo opišemo z urejeno trojico, histogram pa ima 64 komponent).

Še ena možnost je, da regijo opišemo s povprečjem vseh vektorjev, ki opisujejo okna, ki so se pri segmentaciji (po postopku WALRUS) združila v to regijo. Ker se okna pri tem poskusu niso prekrivala, podajajo prve tri komponente tega povprečja ravno povprečno barvo slikovnih elementov po celi regiji (v prostoru YCC; ostalih devet komponent pa kaže, kako v povprečju posamezni deli oken odstopajo od povprečja celega okna). Tako je takšen opis pravzaprav dopolnitev opisa s povprečno barvo v prostoru YCC; dejstvo, da daje tudi za približno 5 % višjo klasifikacijsko točnost (kar je vsekakor statistično signifikantna razlika), kaže, da dajejo dodatne komponente vendarle koristne podatke o regijah. Mogoče je to tudi posledica dejstva, da so se dodatne komponente vektorjev upoštevale pri združevanju v skupine in s tem pri tvorbi regij; zato ni čudno, da tako dobljene regije bolje opišemo, če si zapomnimo tudi povprečje teh ostalih komponent vektorjev. Klasifikatorji, ki temeljijo na takšnih celotnih WALRUSovih opisih, so približno enako dobri (oz. celo malo boljši) kot klasifikatorji z globalnim histogramom.

Za primerjavo prikazuje graf tudi obnašanje klasifikatorjev, ki mero podobnosti namesto s postopkom IRM računajo po pristopu iz sistema WALRUS. Regije so v tem primeru opisane z enakimi 12-razsežnimi vektorji kot v prejšnjem odstavku; razlika je le v tem, kako se te vektorje uporabi za določanje podobnosti med dvema slikama. Vidimo, da nam daje WALRUSova mera podobnosti približno 7 ali 8 % nižjo klasifikacijsko točnost. To je po svoje pričakovano, saj je iz primerjave mer podobnosti pri WALRUS in IRM (razdelek 2.4) videti, da IRM preudarneje sestavi podobnosti med regijami v podobnost med slikama.

Splošni vtis pri rezultatih tega poskusa je, da je rezultate segmentacije (vsaj takšne, kot jo je uporabil ta poskus) težko koristno uporabiti pri gradnji klasifikatorjev. Dejstvo, da se posamičnih regij ni splačalo predstavljati s celimi histogrami, si lahko razlagamo s tem, da je barva znotraj vsake regije razmeroma homogena (to je konec koncev ena od stvari, ki jih pričakujemo od regij po segmentaciji) in histogram zato ne pove kaj dosti koristnega — že če si zapomnimo le povprečno barvo vsake regije, so rezultati enako dobri kot pri najboljši vrsti histogramov.

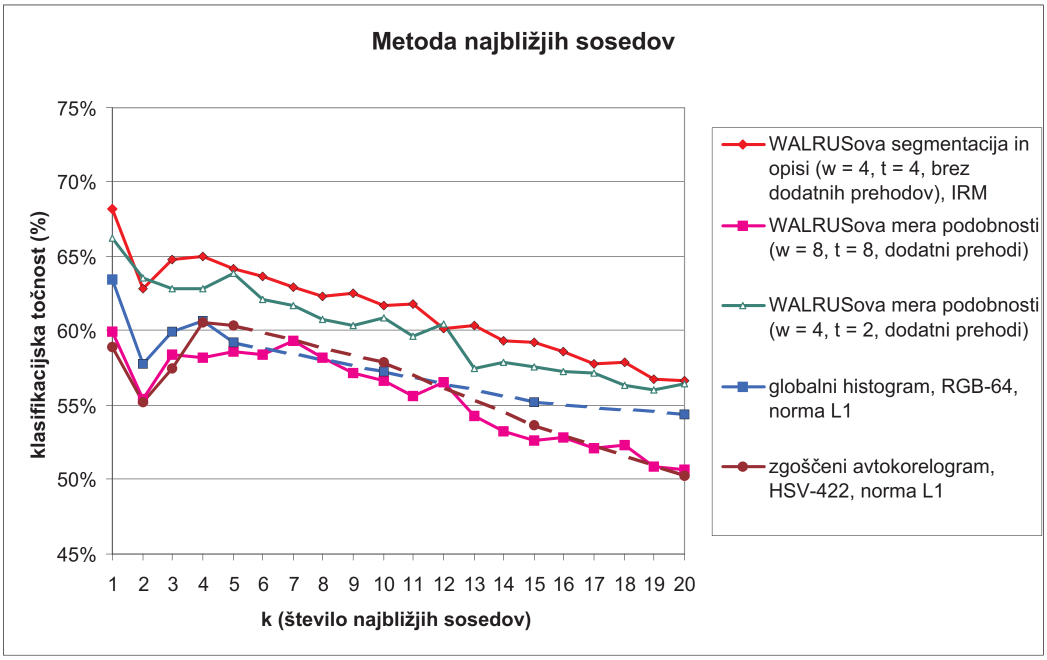
Razlog za slabe rezultate klasifikatorjev, temelječih na segmentaciji, je mogoče tudi v postopku IRM, s katerim določamo podobnost med slikama na podlagi podobnosti med njunimi regijami. Če neka regija ne uspe porabiti svoje teže (q_i ali p_j) za tiste s_{ij}, ki ustrezajo podobnim regijam druge slike (majhni d_{ij}), se bo ta teža pač prenesla na tiste uteži, ki ustrezajo parom z manj podobnimi regijam druge slike; tako lahko neka regija, že npr. zaradi svoje malo večje velikosti, hitro poveča razdaljo med slikama.

5.4 Vpliv števila najbližjih sosedov

Pri poskusih z metodo najbližjih sosedov smo doslej vedno klasificirali na podlagi enega samega soseda. Poskusi z več najbližjimi sosedi (v tem primeru uvrstimo primerek v tisti razred, ki mu pripada največ izmed k najbližjih sosedov) so pokazali, da je tako še najbolje, saj pri $k > 1$ klasifikacijska točnost le še upada. Graf na sliki 5.9 prikazuje odvisnost točnosti od k za nekaj zelo različnih skupin klasifikatorjev: IRM, WALRUS pri dveh različnih vrstah segmentacije, globalni histogram ob kvantizaciji RGB-64 ter zgoščeni avtokorelogram s kvantizacijo HSV-422. Tudi druge vrste opisov, segmentacij, kvantizacij ipd. dajo podoben potek grafa.

Opazimo torej lahko, da v vseh primerih točnost ob prehodu s $k = 1$ na $k = 2$ močno upade, pri $k = 3$ sicer spet naraste, vendar je še vedno manjša kot pri $k = 1$, pri večjih k pa večinoma le še pada. Nenadni padec pri $k = 2$ si lahko razložimo s tem, da je uporaba dveh najbližjih sosedov tako ali tako nespametna, saj se je v primerih, ko glasujeta različno, težko odločiti, katerega bi upoštevali. Očitno se pogosto zgodi, da najbližji sosed glasuje za pravilen razred, drugi najbližji sosed pa za napačnega.

Malo robustnejše v smislu odvisnosti točnosti od k so le tiste skupine klasifikatorjev, ki temeljijo na WALRUSovi meri podobnosti; tu začne točnost zares upadati šele



Slika 5.9: Primerjava klasifikacijske točnosti pri metodi k najbližjih sosedov za različne k ter različne mere podobnosti med slikami. Prikazana so povprečja po desetkratnem prečnem preverjanju, standardne deviacije pa ne, ker postane graf v tem primeru preveč nepregleden.

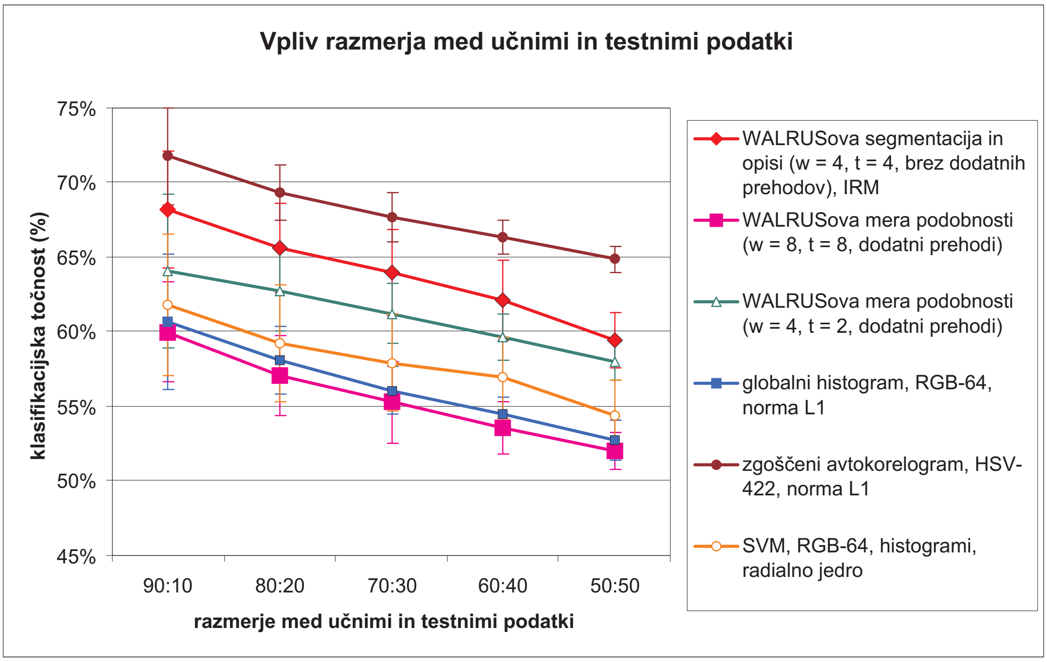
pri $k > 7$, pogosto pa pred tem doseže še večje vrednosti kot pri $k = 1$. Praktično povsod drugod pa daje $k = 1$ daleč najboljše klasifikatorje.

5.5 Vpliv razmerja med učnimi in testnimi podatki

Razredi, ki jih uporabljamo pri poskusih tega poglavja, so definirani pretežno na „semantičen način“: slike v istem razredu prikazujejo prizore z isto tematiko, na primer sončne vzhode, jadralne deske, metulje ipd. To pa še ne pomeni, da imajo te slike po videzu toliko skupnega, da bi s tehnikami, kot jih uporabljajo tu opisani algoritmi, vedno lahko zaznali takšno podobnost. Tako za določeno sliko sončnega zahoda ni nujno, da bo podobna mnogim ostalim slikam sončnih zahodov v bazi — upamo lahko predvsem, da bo podobna vsaj kakšni (in da nobeni sliki iz kakšnega drugega razreda ne bo podobna še bolj). Če naj ima klasifikator kaj upanja, da bo slike iz testne množice pravilno razvrščal, mora biti za vsako testno sliko v učni množici tudi kakšna, ki ji je dovolj podobna in sodi v isti razred.

Zato pričakujemo, da bo klasifikacijska točnost upadla, če bomo več slik uporabili za testne primerke, manj pa jih bomo pustili v učni množici, saj se v tem primeru zmanjšuje možnost, da bo med učnimi slikami ostala kakšna, ki je zares podobna neki testni sliki.

Graf na sliki 5.10 kaže, da se to res zgodi in da je ta pojav približno enak tudi pri zelo različnih vrstah klasifikatorjev. Pri desetkratnem prečnem preverjanju, kot smo ga uporabljali v preostanku tega poglavja, razdelimo podatke na deset skupin



Slika 5.10: Primerjava klasifikacijske točnosti pri metodi enega najbližjega soseda (za različne mere podobnosti) in metodi podpornih vektorjev pri različnih razmerjih med učnimi in testnimi podatki.

in uporabimo vsakič eno kot testne podatke, ostale kot učne podatke; razmerje med učnimi in testnimi podatki je torej 90 % : 10 %. Če uporabimo po dve skupini za testno množico in ostale za učno, imamo razmerje 80 % : 20 % in tako naprej. Opazimo lahko, da pada točnost približno linearno z deležem učnih podatkov; ko pridemo do razmerja 50 % : 50 % (torej ko je učnih podatkov toliko kot testnih), je že približno za petnajst odstotkov nižja kot pri 90 % : 10 %. Zanimivo je tudi, kako hitro se zmanjšujejo standardne deviacije; s štirih ali petih odstotkov pri 90 % : 10 % na komaj dober odstotek pri 50 % : 50 %.

5.6 Primerjava z rezultati iz literature

Zanimivo je primerjati rezultate dobljenih klasifikatorjev s tistimi, ki jih opisujejo Huang, Kumar in Zabih [24]. Slike so opisali z zgoščenimi avtokorelogrami (kvantizacija: prostor RGB, vsaka os razdeljena na 8 intervalov), ki pa so jih potem s pomočjo razcepa na singularne vrednosti (*singular value decomposition*) predelali v vektorje z manj komponentami. Nad dobljeno množico vektorjev gradijo od zgoraj navzdol neke vrste binarno odločitveno drevo; množico vsakič razcepijo na dva dela tako, da pridejo predstavniki nekaterih razredov v levo poddrevo, predstavniki ostalih razredov pa v desno poddrevo. Če je treba razcepiti neko množico S, naredimo to takole: vsak vektor $\mathbf{x} \in S$ v mislih klasificirajmo v razred, kateremu pripada $\mathbf{x}$-ov najbližji sosed iz množice $S - \{\mathbf{x}\}$; mislimo si graf, v katerem je po ena točka za vsak razred, pretok po povezavi $(i : j)$ pa je število vektorjev razreda i, ki bi jih po zgornjem postopku uvrstili v razred j. Z določenimi hevristikami si prizadevamo razrede razbiti na dve taki skupini, da bo večina pretoka znotraj iste skupine, čim manj pa med skupinama. Nato rekurzivno zgradimo obe poddrevesi za nastali skupini vektorjev; ustavimo se, ko imamo v skupini le še predstavnike enega samega razreda; njemu pripada list drevesa.

Njihova slikovna zbirka je obsegala enajst razredov s po 90 slikami. Pri poskusih so dobili z navadno metodo enega najbližjega soseda točnosti okoli 80 %, s svojim odločitvenim drevesom pa povprečno kakšne tri odstotke višje. Če so namesto zgoščenih avtokorelogramov uporabili histograme, so bile točnosti pri metodi najbližjega soseda za kakšne tri odstotke nižje, pri odločitvenem drevesu pa povprečno 12 % nižje. Očitno je torej njihovo odločitveno drevo precej občutljivo na vrsto opisov slik, pri primerno izbranih opisih slik pa je njegova prednost pred navadno metodo najbližjih sosedov precej majhna.

V splošnem lahko ugotovimo, da so njihovi rezultati primerljivi z rezultati poskusov, opisanih v prejšnjih razdelkih. Njihove nekaj odstotkov višje klasifikacijske točnosti si lahko razlagamo z dejstvom, da so imeli manj razredov; mogoče so bili podatki tudi manj šumni; nekaj pa gotovo prispeva tudi predobdelava vektorjev z razcepom na singularne vrednosti. Korist slednjega je po poročanju avtorjev tudi v tem, da lahko dobimo vektorje, veliko krajše od prvotnih, rezultati pa so še vseeno dobri. Izvedli so trikratno prečno preverjanje in tudi pri njih znašajo standardne deviacije točnosti od dva do štiri odstotke.

6

Zaključek

V okviru te diplomske naloge sem preizkusil več strategij klasifikacije slik v skupino disjunktnih razredov. Ena skupina poskusov je obsegala tehnike, pri katerih celotno sliko opišemo z nekim vektorjem, te vektorje pa razvrščamo po metodi najbližjih sosedov ali pa po metodi podpornih vektorjev. Preizkusil sem štiri vrste globalnih opisov slik: histograme, koherenčne vektorje, avtokorelograme in zgoščene avtokorelograme. Ti poskusi so pokazali (kar je pravzaprav tudi pričakovano), da dajejo koherenčni vektorji boljše rezultate od histogramov, avtokorelogrami (navadni in zgoščeni) pa boljše od obojih, pri čemer so zgoščeni malo slabši od navadnih. Od teh štirih vrst opisov je zato videti zgoščeni avtokorelogram še najboljši, saj je veliko manjši in zato bolj obvladljiv od navadnega avtokorelograma, daje pa tudi dobre rezultate. Za mero razdalje med vektorji (pri metodi najbližjih sosedov) se je izkazala manhattanska razdalja primernejša od evklidske, kar se tudi ujema z ugotovitvami iz literature [23, str. 328a]. Metoda podpornih vektorjev daje boljše rezultate od metode najbližjih sosedov; med preizkušenimi vrstami jeder so se linearna obnesla malo slabše od kubičnih in radialnih, med slednjimi pa ni bistvenih razlik.

Druga skupina poskusov je temeljila na segmentaciji slik na regije z enotno barvo ali teksturo. Izkazalo se je, da so parametri segmentacije precej pomembni, ker lahko močno vplivajo na točnost dobljenih klasifikatorjev. Od preizkušenih mer podobnosti med segmentiranimi slikami je bila mera IRM boljša od mere iz postopka WALRUS, je pa klasifikacija z IRM veliko počasnejša; oboje je glede na algoritma, na katerih ti dve meri temeljita, tudi pričakovano.

Najbolj presenetljiv rezultat opisanih poskusov je po mojem mnenju ugotovitev, da strategije, ki temeljijo na segmentaciji, niso uspele dati signifikantno boljših klasifikatorjev od strategij, ki temeljijo na globalnem opisu slik. Človek bi intuitivno pričakoval, da s segmentacijo vendarle pridemo do koristnih dodatnih informacij o vsebini slike; teh informacij v globalnem opisu slike (npr. histogramu) nimamo (vsaj ne v eksplicitni obliki) in bi zato pričakovali, da bo mogoče na podlagi segmentacije priti do boljših mer podobnosti in tudi do boljših klasifikatorjev. Opravljeni poskusi seveda ne dokazujejo, da se tega ne da, kažejo pa, da to ni preprosto in da segmentacija sama po sebi še zdaleč ne zagotavlja, da bomo dobili dobre klasifikatorje.

Še en nepričakovan rezultat je dejstvo, da so dajale metode k najbližjih sosedov za $k > 1$ skoraj povsod slabše rezultate kot pri $k = 1$. To kaže na določeno občutljivost uporabljenih mer podobnosti (od katerih je odvisno, kaj sploh obvelja za najbližjega soseda); če je že najbližji sosed v pravem razredu, se očitno dostikrat zgodi, da bosta drugi ali tretji v napačnih razredih.

Možnosti za nadaljnje delo je na tem področju precej, vidim pa jih predvsem v treh smereh.

1. Obstaja še precej tehnik opisovanja slik in merjenja podobnosti med njimi, ki jih tu nisem preizkusil, pa bi bilo zanimivo videti, kako se primerjajo z ostalimi. Tu je na primer kvadratna mera podobnosti med histogrami iz sistema QBIC [27]; pa različni postopki segmentacije slik, npr. [54]; in različni barvni prostori ter različne možnosti kvantizacije le-teh. S kakšno kombinacijo takšnih tehnik bi se dalo mogoče priti do še boljših klasifikatorjev. Glede metode podpornih vektorjev bi lahko preizkusili tudi druge tehnike za delo z večrazrednimi problemi (poleg sedaj uporabljene „eden proti enemu"), vendar sodeč po rezultatih iz literature dvomim, da bi lahko na ta način dosegli kakšne bistvene izboljšave.

2. Skoraj gotovo je, da je mogoče iz slik potegniti še kaj več informacije, kot je dobijo sedanje tehnike opisovanja slik in sedanje mere podobnosti. Z novimi vrstami opisov slik in merami podobnosti bi se verjetno dalo izboljšati tako rezultate pri odgovarjanju na poizvedbe kot pri gradnji klasifikatorjev.

3. Poskusi, opravljeni v okviru te naloge, so pokazali na več značilnih odnosov med različnimi vrstami opisov slik ali različnimi merami podobnosti; bilo bi zanimivo, če bi se dalo nekako razumeti vzroke teh odnosov. Najbolj očitno vprašanje je seveda to, zakaj dajejo klasifikatorji, temelječi na segmentaciji, razmeroma slabe rezultate (v primerjavi s tistimi, ki izhajajo iz globalnih opisov); drugo zanimivo vprašanje je, zakaj se rezultati poslabšajo, če opazujemo več kot enega najbližjega soseda; zakaj koherenčni vektorji skorajda niso uspešnejši od histogramov, čeprav se intuitivno zdi, da bi morali biti; in tako naprej.

Še ena zamisel, ki sem jo sprva nameraval preizkusiti, vendar so me neobetavni rezultati poskusov s segmentacijo odvrnili od tega, je naslednja: če sliko segmentiramo in opišemo vsako regijo posebej (na primer z WALRUSovim 12-razsežnim vektorjem), je zdaj vsaka slika opisana z več vektorji, ne le z enim, tako da za klasifikacijo ne moremo neposredno uporabiti metode podpornih vektorjev. (Zato so tudi vsi tu opisani poskusi pri segmentiranih slikah temeljili na metodi najbližjih sosedov.) Vsaka slika iz baze prispeva nekaj vektorjev (kolikor regij pač ima) in na koncu dobimo nekaj tisoč, mogoče nekaj deset tisoč vektorjev. Združimo jih v skupine, tako da blizu ležeči vektorji pridejo v isto skupino; od vsake skupine si zapomnimo le centroid. Recimo, da bi nastalo n (reda velikosti nekaj sto) skupin. Določeno sliko lahko zdaj predstavimo z n-razsežnim vektorjem, v katerem i-ta komponenta pove, kolikšen delež površine te slike pokrivajo regije, katerih vektorji sodijo v i-to izmed n skupin.

Tako bi imeli slike opisane z neke vrste redkimi vektorji (*sparse vectors;* večina komponent bi bila enakih 0), vsako sliko z natanko enim vektorjem. Na teh redkih vektorjih bi lahko gradili klasifikatorje z metodo podpornih vektorjev. Podoben pristop se veliko uporablja pri delu z besedili, kjer besedilo pogosto predstavimo z redkim vektorjem: za vsako besedo v slovarju je po ena komponenta, ki pove, kolikokrat se ta beseda pojavlja v opazovanem besedilu; tam daje metoda podpornih vektorjev zelo dobre rezultate. Tu imamo namesto besed opise posameznih regij, tako da je prostor teh opisov zvezen in ga z združevanjem opisov regij v skupine v nekem smislu nasilno poenostavimo v neko diskretno množico (lahko si mislimo, da naš „slovar“ tvorijo centroidi dobljenih skupin).

Opisani pristop nam omogoča kombinirati segmentacijo slik in uporabo metode podpornih vektorjev. Kolikor mi je znano, takšnega pristopa v literaturi še niso omenjali, vendar pa se zaradi neobetavnih rezultatov poskusov v zvezi s segmentacijo in metodo najbližjih sosedov na koncu nisem odločil, da bi opisano zamisel res preizkusil.

Zahvala

Na tem mestu bi se rad zahvalil: prof. Ivanu Bratku za mentorstvo pri diplomski nalogi; Marku Grobelniku, Dunji Mladenić in Urošu Jovanoviču za njihove nasvete in spodbude; še posebej pa Aleksu Jakulinu, ki je prebral nalogo in podal veliko koristnih pripomb in predlogov.

Hvaležen sem tudi vsem avtorjem, ki so se prijazno odzivali na mojo elektronsko pošto s prošnjami za pojasnila in kopije člankov.

Literatura

[1] David W. Aha, Dennis Kibler, Marc K. Albert: *Instance-based learning algorithms.* Machine Learning, 6:37–66, 1991.

[2] Erin L. Allwein, Robert E. Schapire, Yoram Singer: *Reducing multiclass to binary: A unifying approach for margin classifiers.* Journal of Machine Learning Research, 1:113–141, December 2000.

[3] Y. Alp Aslandogan, Chuck Their, Clement Yu: *A system for effective content based image retrieval.* Proceedings of the fourth ACM international conference on Multimedia, November 18–22, 1996, Boston, USA, pp. 429–430.

[4] Vladimir Batagelj: *Optimizacijske metode: skripta v pripravi.* Ljubljana, oktober 1998.

[5] Stefan Berchtold, Christian Böhm, Daniel A. Keim, Hans-Peter Kriegel: *A cost model for nearest-neighbour search in high-dimensional data space.* Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), Tucson, AZ, USA, May 11–15, 1997, pp. 78–86.

[6] Alberto del Bimbo: *Visual Information Retrieval.* Morgan Kaufmann, 1999.

[7] Marinette Bouet, Ali Khenchaf, Henri Briand: *Shape representation for image retrieval.* Proceedings of the seventh ACM international conference (part 2) on Multimedia, October 30–November 5, 1999, Orlando, FL, USA, pp. 1–4.

[8] Phil Brodatz: *Textures: A photographic album for artists and designers.* New York: Dover, 1966. *http://www.ux.his.no/~tranden/brodatz.html*

[9] Christopher J. C. Burges: *A tutorial on support vector machines for pattern recognition.* Data Mining and Knowledge Discovery, 2(2):121–167, June 1998.

[10] Colin Campbell: *Algorithmic approaches to training support vector machines: A survey.* Proceedings of the eighth European Symposium on Artificial Neural Networks (ESANN 2000), Bruges, Belgium, April 26–28, 2000, pp. 27–36.

[11] T. M. Cover, Peter E. Hart: *Nearest neighbor pattern classification.* IEEE Transactions on Information Theory, 13(1):21–27, January 1967.

[12] Chih-Chung Chang, Chih-Jen Lin: LIBSVM*: A library for support vector machines* (version 2.3). Department of Computer Science and Information Engineering, National Taiwan University, April 2001.

[13] Chad Carson, Virginia E. Ogle: *Storage and retrieval of feature data for a very large online image collection.* IEEE Computer Society Bulletin of the Technical Committee on Data Engineering, 19(4):19–27, December 1996.

[14] Nello Christianini, John Shawe-Taylor: *An introduction to support vector machines and other kernel-based learning methods.* Cambridge University Press, 2000.

[15] James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes: *Computer Graphics: Principles and Practice.* Addison-Wesley, 1990.

[16] Christos Faloutsos, Will Equitz, Myron Flickner, Wayne Niblack, Dragutin Petkovic, Ronald Barber: *Efficient and ef-*

fective querying by image content. Journal of Intelligent Information Systems, 3:231–262, 1994.

[17] Jerome H. Friedman: *Another approach to polychotomous classification.* Technical report, Department of Statistics, Stanford University, October 1996.

[18] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, Peter Yanker: *Query by image and video content: The QBIC system.* IEEE Computer, 28(9):23–32, September 1995.

[19] Amarnath Gupta, Ramesh Jain: *Visual information retrieval.* Communications of the ACM, 40(5):69–79, 1997.

[20] Peter E. Hart: *The condensed nearest neighbor rule.* IEEE Transactions on Information Theory, 14(3):515–6, May 1968.

[21] Wynne Hsu, Tat-Seng Chua, H. H. Pung: *An integrated color-spatial approach to content-based image retrieval.* Proceedings of the third ACM international conference on Multimedia, November 5–9, 1995, San Francisco, CA, USA, pp. 305–313.

[22] Kyoji Hirata, Toshikazu Kato: *Query by visual example: Content based image retrieval.* Advances in Database Technology, Proceedings of the third international conference on Extending Database Technology (EDBT '92), Vienna, Austria, March 23–27, 1992, pp. 56–71. LNCS, vol. 580.

[23] Jing Huang, S. Ravi Kumar, Mandar Mitra: *Combining supervised learning with color correlograms for content-based image retrieval.* Proceedings of the fifth ACM international conference on Multimedia, November 9–13, 1997, Seattle, WA, USA, pp. 325–334.

[24] Jing Huang, S. Ravi Kumar, Ramin Zabih: *An automatic hierarchical image classification scheme.* Proceedings of the sixth ACM international conference on Multimedia, September 13–16, 1998, Bristol, United Kingdom, pp. 219–228.

[25] Chih-Wei Hsu, Chih-Jen Lin: *A comparison on methods for multi-class support vector machines.* Department of Computer Science and Information Engineering, National Taiwan University, April 2001.

[26] Gísli R. Hjaltason, Hanan Samet: *Ranking in spatial databases.* Advances in Spatial Databases, Proceedings of the 4th International Symposium (SSD), Portland, Maine, USA, August 6–9, 1995. LNCS, vol. 951, pp. 83–95.

[27] James L. Hafner, Harpreeet Singh Sawney, Will Equitz, Myron Flickner, Wayne Niblack: *Efficient color histogram indexing for quadratic form distance functions.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(7):729–736, July 1995.

[28] Charles E. Jacobs, Adam Finkelstein, David H. Salesin: *Fast multiresolution image querying.* Proceedings of the 22nd ACM conference on Computer Graphics (SIGGRAPH '95), Los Angeles, CA, USA, August 6–11, 1995, pp. 277–286.

[29] Thorsten Joachims: *Making large-scale support vector machine learning practical.* In: Bernard Schölkopf, Christopher J. C. Burges, Alexander J. Smola, eds., *Advances in kernel methods: Support vector learning,* The MIT Press, 1999, pp. 169–184.

[30] Gerald Kaiser: *A friendly guide to wavelets.* Birkhäuser, Boston, 1994.

[31] Ulrich Hans-Georg Kreßel: *Pairwise classification and support vector machines.* In: Bernard Schölkopf, Christopher J. C. Burges, Alexander J. Smola, eds., *Advances in kernel methods: Support vector learning,* The MIT Press, 1999, pp. 255–268.

[32] David Dolan Lewis: *Evaluating text categorization.* Proceedings of the third

DARPA Speech and Natural Language Workshop, Asilomar, CA, USA, February 1991, pp. 312–318.

[33] Huma Lodhi, John Shawe-Taylor, Nello Christianini, Chris Watkins: *Text classification using string kernels.* NeuroCOLT Technical Report NC-TR-00-079, June 2000.

[34] Jia Li, James Ze Wang, Gio Wiederhold: *Classification of textured and non-textured images using region segmentation.* Proceedings of the seventh IEEE International Conference on Image Processing (ICIP), Vancouver, BC, Canada, September 2000, pp. 754–757.

[35] Jia Li, James Ze Wang, Gio Wiederhold: *IRM: Integrated region matching for image retrieval.* Proceedings of the eighth ACM international conference on multimedia, October 30–November 3, 2000, Los Angeles, CA, USA, pp. 147–156.

[36] Apostol Natsev, Rajeev Rastogi, Kyuseok Shim: *WALRUS: A similarity retrieval algorithm for image databases.* Technical Report P009, Bell Laboratories, Murray Hill, 1998.

[37] Apostol Natsev, Rajeev Rastogi, Kyuseok Shim: *WALRUS: A similarity retrieval algorithm for image databases.* Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, Philadelphia, PA, USA, May 31–June 3, 1999, pp. 395–406.

[38] Virginia E. Ogle, Michael Stonebraker: *Chabot: Retrieval from a relational database of images.* IEEE Computer, 28(9):40–48, September 1995.

[39] John C. Platt, Nello Christianini, John Shawe-Taylor: *Large margin DAGs for multiclass classification.* Advances in Neural Information Processing Systems 12 (Proceedings of NIPS'99, Denver, CO, USA, November 29–December 4, 1999), pp. 547–553. The MIT press, 2000.

[40] John C. Platt: *Fast training of support vector machines using sequential minimal optimization.* In: Bernard Schölkopf, Christopher J. C. Burges, Alexander J. Smola, eds., *Advances in kernel methods: Support vector learning*, The MIT Press, 1999, pp. 185–208.

[41] Greg Pass, Ramin Zabih: *Histogram refinement for content-based image retrieval.* Proceedings of the IEEE Workshop on Applications of Computer Vision, Sarasota, FL, USA, December 1996, pp. 96–102.

[42] Greg Pass, Ramin Zabih: *Comparing images using joint histograms.* Journal of Multimedia Systems, 7(3):234–240, May 1999.

[43] Greg Pass, Ramin Zabih, Justin Miller: *Comparing images using color coherence vectors.* Proceedings of the fourth ACM international conference on Multimedia, November 18–22, 1996, Boston, USA, pp. 65–73.

[44] Yuri Quintana: *Organization and retrieval in a pictorial digital library.* Proceedings of the 2nd ACM international conference on Digital libraries, July 23–26, 1997, Philadelphia, PA, USA, pp. 13–20.

[45] Nick Roussopoulos, Stephen Kelley, Frédéric Vincent: *Nearest neighbor queries.* Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, CA, USA, May 22–25, 1995, pp. 71–79.

[46] Michael J. Swain, Dana H. Ballard: *Color indexing.* International Journal of Computer Vision, 7(1):11–32, November 1991.

[47] John R. Smith, Shih-Fu Chang: *VisualSEEk: A fully automated content-based image query system.* Proceedings of the fourth ACM international conference on Multimedia, November 18–22, 1996, Boston, USA, pp. 87–98.

[48] Jonathan Richard Shewchuk: *An introduction to the conjugate gradient me-*

thod without the agonizing pain, Edition $1\frac{1}{4}$. School of Computer Science, Carnegie Mellon University, August 4, 1994.

[49] Renato O. Stehling, Mario A. Nascimento, Alexandre X. Falcão: *On "shapes" of colors for content-based image retrieval.* Proceedings of the ACM Multimedia Workshop, Marina del Rey, CA, USA, October 30–November 3, 2000, pp. 171–174.

[50] Olivier Teytaud, Radwan Jalam: *Kernel-based text categorization.* Technical Report RR-0007, ERIC, Université Lumière Lyon 2, July 2000.

[51] Hideyuki Tamura, Shunji Mori, Takashi Yamawaki: *Texture features corresponding to visual perception.* IEEE Transactions on Systems, Man, and Cybernetics, 8(6):460–473, June 1978.

[52] Mark R. Turner: *Texture discrimination by Gabor functions.* Biological Cybernetics, 55(2/3):71–82, 1986.

[53] James Ze Wang, Jia Li, Desmond Chan, Gio Wiederhold: *Semantics-sensitive retrieval for digital picture libraries.* D-Lib Magazine, 5(11), DOI.1045/*november99-wang,* November 1999.

[54] James Ze Wang, Jia Li, Gio Wiederhold: SIMPLI*city: Semantics-sensitive integrated matching for picture libraries.* Advances in Visual Information Systems, Fourth International Conference, VISUAL 2000, Lyon, France, November 2–4, 2000, pp. 360–371. LNCS, vol. 1929.

[55] Jason Weston, Chris Watkins: *Multi-class support vector machines.* Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway University of London, May 20, 1998.

[56] Jason Weston, Chris Watkins: *Support vector machines for multi-class pattern recognition.* Proceedings of the seventh European Symposium on Artificial Neural Networks (ESANN'99), Bruges, Belgium, April 21–23, 1999.

[57] James Ze Wang, Gio Wiederhold, Oscar Firschein, Sha Xin Wei: *Content-based image indexing and searching using Daubechies' wavelets.* International Journal of Digital Libraries (IJODL), 1(4):311–328, December 1997.

[58] Tian Zhang: *Data clustering for very large data sets plus applications.* Ph. D. Thesis, Computer Sciences Department, University of Wisconsin at Madison, 1997.

[59] Tian Zhang, Raghu Ramakrishan, Miron Livny: BIRCH: *An efficient data clustering method for very large databases.* Proceedings of the 1996 ACM SIGMOD Internationl Conference on Management of Data, Montreal, Canada, June 3–6, 1996, pp. 103–114.

Izjava o samostojnosti dela

Izjavljam, da sem diplomsko nalogo izdelal samostojno pod mentorstvom prof. dr. Ivana Bratka. Sodelavce, ki so mi pri delu pomagali, sem navedel v zahvali.

Janez Brank