SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1146

Klasificiranje slika Fisherovim vektorima izgrađenim nad slučajnim distribucijskim šumama značajki slike

Dino Pačandi

Zagreb, srpanj 2015.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

Zahvaljujem se svome mentoru prof.dr.sc. Siniši Šegviću i dr.sc. Josipu Krapcu na pruženoj stručnoj pomoći i savjetima, prije i tijekom izrade ovog diplomskog rada.

Sadržaj

1. Uvod

Klasifikacija slika jedan je od glavnih predmeta proučavanja računalnog vida. Ona se svodi na semantičku interpretaciju sadržaja slike temeljem koje se slici dodjeljuje neka od mogućih oznaka (npr. slika na kojoj se nalazi automobil može biti označena kao *motorno vozilo* ili *automobil*). To je veoma zahtjevan zadatak jer se traženi objekt na slici može pojaviti u raznim scenama promatranim s raznih gledišta pod različitim osvjetljenjem te sami objekti mogu biti djelomično zaklonjeni. Primjerice, scena automobila u gradskom prometu jako se razlikuje od scene automobila na šumskoj stazi. To su samo neki od čimbenika koji otežavaju ispravnu i robusnu klasifikaciju.

Jedan od pristupa klasifikaciji slika temelji se na ideji preuzetoj iz klasifikacije teksta, [1]. Opisnici slike niske razine grupiraju se nekim algoritmom nenadziranog strojnog učenja tako da svaka grupacija predstavlja jednu slikovnu riječ (npr. algoritam k-srednjih vrijednosti). Takve skupine zajedno čine slikovni rječnik te se pomoću njega za neku sliku može izgraditi histogram slikovnih riječi koji se pak može iskoristiti za klasifikaciju slike.

Pristup klasifikaciji Fisherovim vektorima predstavlja proširenje pristupa klasifikaciji pomoću slikovnog rječnika. Slikovni rječnik modelira se nekim ekspresivnijim generativnim modelom (od spomenutih k centroida), primjerice Gaussovom mješavinom. Histogram slikovnih riječi zamjenjuje se Fisherovim vektorom koji mjeri odstupanje opisnika slike od naučenog generativnog modela u njegovom parametarskom prostoru.

Uobičajeno je kod korištenja Fisherovih vektora kao generativni model koristiti Gaussovu mješavinu. U nastavku će biti prikazan pristup koji Gaussovu mješavinu zamjenjuje distribucijskom šumom.

2. Klasifikacija slika Fisherovim vektorima

2.1. Klasifikacija slika histogramima slikovnih riječi

Ideja klasifikacije slika histogramima slikovnih riječi bazirana je na pristupu klasifikacije teksta koji koristi broj pojavljivanja *diskriminantnih riječi* u tekstu kako bi se odredio njegov tip (npr. sport, kultura, crna kronika, itd.), [1].

Za klasifikaciju slika pomoću slikovnog rječnika potrebno je nekoliko algoritama. Prvi korak je opisati značajke slike; potrebno je računalu predstaviti sadržaj slike (npr. informaciju o rubovima, kutevima, distribuciji intenziteta, amplitudama i smjerovima gradijenata, itd.). Zatim se pomoću nekog od algoritama nenadziranog strojnog učenja izgrađuje slikovni rječnik. Pomoću slikovnog rječnika za neku neviđenu značajku može se odrediti kojoj slikovnoj riječi pripada te se tako može izgraditi histogram slikovnih riječi (broj pojava slikovnih riječi na slici). Takav histogram može iskoristiti klasifikator kako bi odredio što se nalazi na slici.

2.1.1. SIFT

Značajke slike potrebno je nekako predstaviti računalu te je u tu svrhu razvijeno više različitih postupaka od kojih svaki rezultira poljem brojeva koje se koristi kao vektor značajki (na razini jednog ili više piksela).

Jedan od popularnih algoritama za opis značajki slike je SIFT (engl. *Scale Invariant Feature Transform*) kojeg je objavio (i patentirao) David Lowe 1999. godine, [2]. Algoritam opisuje piksel slike uzimajući u obzir amplitude i smjerove gradijenata njegove okoline (gradijent predstavlja razliku intenziteta međusobno susjednih piksela). Okolina piksela dijeli se u 16 ćelija (veličina ćelije je varija-

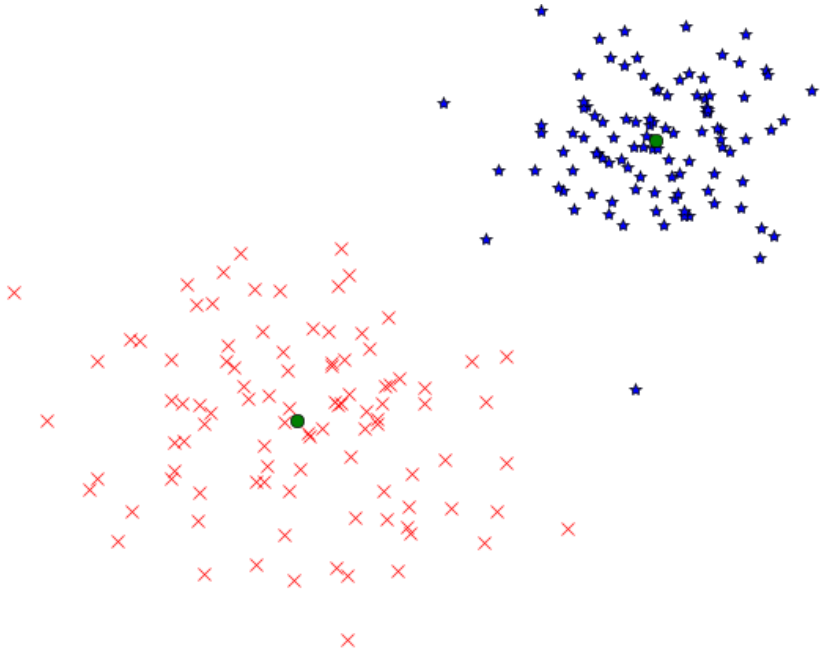
bilna te se može prilagoditi veličini značajke koja se opisuje) te se za svaku od ćelija računa težinski histogram orijentacije gradijenata. Težina s kojom pojedina orijentacija doprinosi histogramu određuje se iz amplitude samog gradijenta. Histogram ćelije sastoji se od 8 komponenata; svaka komponenta predstavlja jednu orijentaciju (međusobno susjedne ćelije predstavljaju orijentacije koje se razlikuju za 45 stupnjeva; analogno s četiri glavne i sporedne strane svijeta). Konačan opisnik dobiva se spajanjem svih 16 histograma u jedan 128 dimenzionalni vektor koji upravo i predstavlja SIFT opisnik.

Važno je spomenuti da se SIFT može koristiti s detektorom interesnih točaka (pronalazak lokacija značajki na slici) ili pak se lokacije (i veličine ćelija) mogu unaprijed definirati - takav pristup naziva se *denseSIFT*. Za klasifikaciju slika uobičajen je koristiti *denseSIFT*.

Glavne karakteristike SIFT značajki su invarijantnost na promjene osvjetljenja, kontrasta i mjerila slike kao i invarijantnost na rotaciju.
Invarijantnost na (linearne) promjene osvjetljenja proizlazi iz samog korištenja gradijenata slike (linearna promijena intenziteta svjetlosti neće promjeniti iznos razlike intenziteta susjednih piksela). Invarijantnost na promjene kontrasta (jači kontrast rezultirati će većim iznosom amplitude gradijenata) postiže se svođenjem izgrađenog SIFT opisnika na jediničnu normu.
Invarijantnost na mjerilo značajke postiže se promatranjem odziva DoG (engl. *Difference of Gaussian*) funkcije (koji predstavlja aproksimaciju LoG (engl. *Laplacian of Gaussian*) funkcije). Pojednostavljeno objašnjenje bilo bi sljedeće: iz početne slike stvara se više novih slika koje su zaglađene Gaussovim filtrom različitih intenziteta (npr. prva slika dobivena je zaglađivanjem s intenzitetom σ, druga slika s intenzitetom 2σ, itd.) te se slike međusobno susjednih intenziteta oduzimaju (razlika Gaussijana). Ako značajka slike iščezne nakon zaglađivanja Gaussovim filtrom intenziteta $n\sigma$, a za intenzitet $(n-1)\sigma$ je još bila prisutna, DoG će dati veliki odziv te se može odrediti mjerilo (i veličina ćelije) u kojem značajka postoji (ili prestaje postojati).
Invarijantnost na rotacije postiže se uzimanjem u obzir dominantne orijentacije gradijenata okoline piksela koji se opisuje prilikom izračuna opisnika.

2.1.2. Algoritam k-srednjih vrijednosti

Algoritam k-srednjih vrijednosti (engl. *k-means*) spada u algoritme nenadziranog strojnog učenja. Svrha takvih algoritama je da pronađu grupaciju neoznačenih



Slika 2.1: Grupiranje podataka u dvije grupe algoritmom k-srednjih vrijednosti

podataka temeljem neke mjere sličnosti. Pridruživanje podataka grupi može biti meko ili čvrsto. Kod mekog pridruživanja, podatak nekoj grupi pripada s određenom mjerom (težinom) dok kod čvrstog grupiranja podatak ili pripada ili ne pripada grupi (težina je 0 ili 1).

Parametar k predstavlja broj grupa u koje se podaci mogu smjestiti. Kod inicijalizacije algoritma, potrebno je odrediti početnu 'srednju vrijednost' za svaku od k grupa (nasumičnim odabirom ili nekom od pametnijih metoda poput *k-means++* koja pokušava maksimizirati međusobnu udaljenost svih grupa). Podaci se zatim raspoređuju po grupama temeljem minimalne euklidske udaljenosti te se izračunavaju nove srednje vrijednosti grupa (iz podataka koji su završili u pojedinoj grupi). Nakon izračuna novih srednjih vrijednosti, podaci se iznova raspoređuju po grupama te se ponovno računaju nove srednje vrijednosti. Cijeli postupak ponavlja se do konvergencije (maksimalni dozvoljeni broj preraspodijela ili ako promjena vrijednosti optimizacijske funkcije između dvije iteracije algoritma padne ispod zadanog praga).
Optimizacijska funkcija algoritma dana je u nastavku:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{N} b_j^i ||x_i - c_j||^2 \tag{2.1}$$

gdje b_j^i predstavlja indikatorsku varijablu pripadnosti primjera x_i centroidu c_j.

2.1.3. Slikovni rječnik i histogrami slikovnih riječi

Prvi korak kod stvaranja slikovnog rječnika je izlučiti značajke slike. One se mogu dodatno predobraditi, npr. smanjivanjem dimenzionalnosti pomoću analize glavnih komponenata (engl. *PCA – Principal component analysis*). Zatim je potrebno odrediti veličinu slikovnog rječnika - broj slikovnih riječi koje će sadržavati (odabir broja riječi je proizvoljan). To upravo predstavlja parametar k algoritma k-srednjih vrijednosti. Nakon provedbe spomenutog algoritma dobiva se slikovni rječnik. On se zatim može iskoristiti za stvaranje histograma slikovnih riječi slike.

Histogram slikovnih riječi sastoji se od pretinaca; po jedan za svaku slikovnu riječ. Za svaki vektor značajki slike (u ovom slučaju SIFT vektor) određuje se kojoj slikovnoj riječi pripada tako da se uzima ona riječ čija je euklidska udaljenost od vektora značajki minimalna (diskretizacija značajki slike). Očekivano je da će ta slikovna riječ biti najsličnija vektoru značajki. Toj riječi uvećava se vrijednost pretinca u histogramu za 1. Takav histogram prosljeđuje se klasifikatoru (npr. SVM) koji će pokušati odrediti što se nalazi na slici.

2.1.4. Analiza glavnih komponenata

Analiza glavnih komponenata (engl. *Principal component analysis*) je matematička metoda koja služi za pronalazak novog koordinatnog sustava prostora u kojem njegove dimenzije neće biti međusobno korelirane. Najčešće se koristi kao postupak preslikavanja iz ulaznog prostora dimenzije d u novi prostor dimenzije d' tako da vrijedi $d' < d$, uz uvjet maksimalnog očuvanja informacije. Metoda se svodi na pronalazak jediničnih, međusobno nekoreliranih (okomitih) vektora koji čine bazu novog koordinatnog sustava.

Kao najjednostavniji primjer, može se promatrati preslikavanje u jednodimenzionalni prostor; prostor koji dobivamo projiciranjem ulaznog prostora na vektor $\mathbf{v}$. Odabir vektora $\mathbf{v}$ obavlja se tako da se maksimizira varijanca varijable dobivene preslikavanjem (tako se osigurava minimalni gubitak informacije). Preostale dimenzije dobivaju se jednakim pristupom uz dodatni uvjet da dimenzije međusobno moraju biti nekorelirane. Prva dimenzija će stoga sadržavati najveću količinu informacije, druga dimenzija će sadržavati manju količinu informacije nego prva, itd.

Pronalazak vektora $v_1,\ v_2, \ldots,\ v_r$ svodi se na pronalazak svojstvenih vektora kovarijacijske matrice podataka, Σ. Kao vektor v_1 odabire se onaj svojstveni vek-

tor čija je pripadna svojstvena vrijednost najveća, kao v_2 onaj svojstveni vektor čija je pripadna svojstvena vrijednost druga po veličini, itd.

Ova metoda izuzetno je korisna u obradi podataka jer nam omogućava da visoko-dimenzionalne podatke predstavimo u nisko-dimenzionalnom obliku bez (prevelikog) gubitka relevantnih informacija što nam omogućava značajno ubrzanje daljnjih izračuna s tim podacima. Također nam i otvara mogućnost jednostavne vizualizacije n–dimenzionalnih podataka koja inače ne bi bila moguća (preslikavanjem u 1D, 2D ili 3D prostor).

2.2. Generativni i diskriminativni modeli

Svrha algoritama strojnog učenja je naučiti nešto iz danih podataka učenjem matematičkog modela te to naučeno znanje i model primijeniti na obradu nekih novih, neviđenih podataka. Takav algoritam ne bi trebao ovisiti o prirodi problema koji rješava nego bi trebao biti općenito upotrebljiv u svrhu u koju ga želimo rabiti. Ako želimo istražiti i naučiti distribuciju (mješavinu distribucija) iz koje su podaci generirani (te to znanje primijeniti na klasifikaciju podataka), koristit ćemo generativne modele. Ako nas ne zanima distribucija kojoj podatci podliježu nego samo klasa kojoj podatak pripada, koristit ćemo diskriminativne modele.

2.2.1. Generativni modeli

Generativni modeli modeliraju vjerojatnost pripadnosti vektora značajki klasi preko zajedničke vjerojatnosti vektora značajki i klase,

$$P(C_i|\mathbf{x}) \propto P(\mathbf{x}, C_i) \tag{2.2}$$

gdje $\mathbf{x}$ predstavlja vektor značajki dimenzije d, a C_i neku klasu.
Pomoću zajedničke vjerojatnosti $P(C_i, \mathbf{x})$ može se pomoću Bayesovog pravila izračunati vjerojatnost da primjer $\mathbf{x}$ pripada klasi C_i, što je upravo $P(C_i|\mathbf{x})$. Bayesovo pravilo glasi

$$P(C_i|\mathbf{x}) = \frac{P(C_i)P(\mathbf{x}|C_i)}{\sum_{j=1}^{|C|} P(C_j)P(\mathbf{x}|C_j)}. \tag{2.3}$$

gdje $P(C_i)$ predstavlja vjerojatnost pojave klase C_i. Nazivnik jednadžbe (2.3) može se i jednostavnije zapisati kao $P(\mathbf{x})$ jer predstavlja vjerojatnost da je vektor

značajki $\mathbf{x}$ generiran iz naučenog generativnog modela. $P(C_i|\mathbf{x})$ naziva se aposteriorna vrijednost klase za vektor značajki, a $P(C_i)$ apriorna vrijednost klase. Aposteriorna vrijednost klase, kao izlaz generativnog modela, ima probabilističku interpretaciju; vrijednost aposteriorne vrijednosti proporcionalna je sigurnosti u točnost klasifikacije (veća aposteriorna vrijednost povlači veću sigurnost).

Takvi naučeni modeli mogu se iskoristiti za generiranje umjetnih podataka iz domene prostora ulaznih podataka (stoga se i nazivaju generativni modeli). Oni su pogodni za uporabu kada je skup za učenje malen (pod uvjetom da se ulazni podaci zaista ravnaju po nekim distribucijama) ili nepotpun (dimenzije koje nedostaju mogu se generirati). Neki od primjera generativnih modela su Bayesovi klasifikatori, mješavine Gaussovih distribucija (engl. *GMM - Gaussian Mixture Model*), itd.

2.2.2. Diskriminativni modeli

Za razliku od generativnih, diskriminativni modeli ne modeliraju zajedničku vjerojatnost vektora značajki i klase $P(C_i, \mathbf{x})$ nego izravno modeliraju pripadnost vektora značajki $\mathbf{x}$ klasi C_i. Diskriminativni modeli mogu se dodatno podijeliti na probabilističke i neprobabilističke modele.

Probabilistički modeli izravno modeliraju aposteriornu vjerojatnost klase za vektor značajki, $P(C_i|\mathbf{x})$ (bez modeliranja zajedničke vjerojatnosti). Izlazi takvih modela imaju probabilističku interpretaciju jednako kao i generativni modeli. Jedan od primjera takvog modela je logistička regresija.

Neprobabilistički modeli modeliraju funkciju $h(\mathbf{x})$ koja izravno (bez modeliranja aposteriorne vjerojatnosti klase) vektoru značajki $\mathbf{x}$ dodjeljuje oznaku pripadnosti klase C_i,

$$h_i(\mathbf{x}) = \begin{cases} 1 & , \textit{ pripada klasi } C_i \\ -1 & , \textit{ ne pripada klasi } C_i \end{cases} \tag{2.4}$$

Takvi modeli nemaju probabilističku interpretaciju, ali se izlaz ipak donekle može iskoristiti za procjenu pouzdanosti klasifikacije. Primjeri takvih modela su stabla odluke, k-najbližih susjeda (engl. *kNN - k-Nearest-Neighbors*), neuronske mreže, stroj s potpornim vektorima (engl. *SVM - support vector machine*).

2.2.3. Linearni i nelinearni modeli

Matematički modeli strojnog učenja dodatno se mogu podijeliti na linearne i nelinearne modele.

Linearni modeli modeliraju linearnu granicu između primjera iz dviju klasa (pravac, ravnina ili općenito hiperravnina) dok nelinearni modeli omogućavaju modeliranje nelinearne granice (polinomi n-tog stupnja).

Nelinearni modeli imaju veću slobodu kod modeliranja granica i veći kapacitet nego linearni modeli. To povlači da će nelinearni modeli moći uspješno razdvojiti veći skup problema nego linearni modeli pa se naizgled može činiti da su oni općenito bolji. Npr. jedan jednostavan zadatak, poput klasifikacije *isključivo ili* (engl. *exclusive or, XOR*) problema ne može se riješiti pomoću linearnog modela dok se pomoću nelinearnog modela može.

Glavni problem nelinearnih modela je upravo njihova mogućnost da se mogu veoma dobro prilagoditi podacima s kojima rade. Kako podaci ne moraju uvijek biti potpuno točni odnosno veoma često sadrže šum, nelinearni model će se također prilagoditi i tom šumu, a to je nepoželjno. Takav model moći će savršeno odvojiti podatke za učenje, ali ne mora davati dobre rezultate na neviđenim podacima; njegova moć generalizacije može biti loša (zbog prilagodbe šumu ili nedovoljnog broja podataka za učenje). Još jedan nedostatak nelinearnih naspram linearnih modela je to što su oni kompliciraniji za učenje i primjenu (uglavnom imaju više parametara, teže je odrediti udaljenost nekog primjera od granice, rade sporije, teško ili nemoguće ih je interpretirati itd.). Odabir linearnog ili nelinearnog modela ovisi o težini problema koji se rješava, ali se općenito preferira jednostavniji pristup, u ovom slučaju linearni model, ako je on dostatan za rješavanje problema.

Primjeri će obično biti linearno razdvojivi ako je ulazni prostor rijetko naseljen, $N < d$, dok u slučaju da je ulazni prostor gusto naseljen, $N > d$, vjerojatnost da će oni biti linearno razdvojivi je malena (N predstavlja veličinu skupa za učenje, a d dimenziju ulaznog prostora). Iz ovoga bi se naivno mogao izvući zaključak da ako želimo koristiti linearne modele, skup podataka za učenje mora biti malen. Međutim, intuitivno je jasno da će se klasifikator bolje ponašati ako mu prilikom učenja predočimo čim veći broj različitih primjera pa je stoga ograničavanje veličine skupa za učenje nepoželjno.

2.2.4. Jezgrene funkcije

Problemu linearne neodvojivosti u ulaznom prostoru značajki može se doskočiti preslikavanjem tog prostora u neki drugi višedimenzionalni prostor značajki. Funkcija koja obavlja preslikavanje iz jednog prostora u drugi može se prikazati kao $f(\mathbf{x}) : \mathbb{R}^n \mapsto \mathbb{R}^m$, gdje n predstavlja dimenziju početnog prostora, a m dimenziju preslikanog prostora te obično vrijedi $m > n$. Vjerojatnost da će primjeri biti linearno razdvojivi u tom novom prostoru značajki veća je nego u ulaznom prostoru.
Općeniti oblik jezgrene funkcije može se zapisati kao

$$K(x_i, x_j) = x_i^\top M x_j. \tag{2.5}$$

Da bi jezgrena funkcija bila valjana , prema Mercerovom teoremu, za matricu M treba vrijediti da je pozitivno semi-definitna, odnosno formalno $\forall x \in \mathbb{R}^n,\ x^\top M x \geq 0$.
Također, za svaku jezgrenu funkciju vrijedi komutativnost, $K(x_i, x_j) = K(x_j, x_i)$.
Uz tako definirane uvjete, možemo izraziti preslikavanje $\phi : \mathbb{R}^n \mapsto \mathbb{R}^m$ tako da vrijedi

$$K(x_i, x_j) = \phi_i^\top \phi_j, \tag{2.6}$$

gdje ϕ_i predstavlja primjer preslikan iz ulaznog prostora u prostor značajki. Preslikavanje se može formalno izraziti kao

$$\phi_i = \sqrt{M} x_i \tag{2.7}$$

što odgovara obliku jednadžbe (2.6). Kada je oblik funkcije preslikavanja ϕ poznat i primjeri se preslikavaju u novi prostor prije treniranja klasifikacijskog modela, govorimo o izravnom oblikovanju.

Iz jednadžbe (2.6) vidljivo je da jezgrena funkcija zapravo predstavlja skalarni produkt dvaju primjera u prostoru značajki. Definiranje skalarnog produkta u prostoru povlači sa sobom mogućnost definiranja udaljenosti između dvaju vektora, odnosno metrike prostora. Pomoću skalarnog produkta možemo definirati euklidsku metriku kao

$$\|\phi_i - \phi_j\|^2 = K(x_i, x_i) - 2K(x_i, x_j) + K(x_j, x_j). \tag{2.8}$$

Pokazano je da jezgrena funkcija predstavlja skalarni produkt dvaju vektora u preslikanom prostoru. Jezgrenu funkciju moguće je definirati i bez poznavanja preslikavanja ϕ; moguće je izračunati skalarni produkt dvaju vektora u nekom prostoru bez poznavanja eksplicitnog preslikavanja u taj prostor. To se naziva

jezgreni trik (engl. *kernel trick*) te je on razlog što su jezgrene funkcije veoma raširene u uporabi. U nastavku neće biti korišten jezgreni trik nego će se koristiti jezgrena funkcija s direktnim oblikovanjem.

2.3. Fisherov vektor kao reprezentacija slike

Svrha klasifikacijskog modela, bio on generativan ili diskriminativan, ispravna je klasifikacija nekog vektora značajki.

Generativni modeli općenito se bolje ponašaju u situacijama kada je skup za učenje relativno malen u odnosu na dimenzionalnost ulaznog prostora, mogu se nositi s nepotpunim ulaznim podacima (komponenta koja nedostaje može se lako generirati) te dopuštaju laku ugradnju nekog apriornog znanja. Također, kao što je spomenuto u prethodnim odjeljcima, mogu se iskoristiti za generiranje novih podataka. S druge strane, diskriminativni modeli bolje se ponašaju kada je dostupan veliki skup za učenje, ne mogu se nositi s nepotpunim podacima te je u njih teško ugraditi apriorno znanje. Iako se na prvi pogled po navedenim svojstvima čini da su generativni modeli superiorniji naspram diskriminativnih modela, potonji je rasprostranjeniji u općenitoj uporabi jer postiže bolje rezultate kod klasifikacije.

U nastavku će biti opisan pristup koji kombinira obje vrste modela u jedan, skupni model. Motivacija takvog pristupa dobiti je model koji će kombinirati snage oba pristupa, a ukloniti nedostatke.

2.3.1. Fisherova jezgra

Većina diskriminativnih modela koristi jezgrene funkcije za usporedbu dvaju vektora značajki (odjeljci 2.2.2 i 2.2.4). Kada bi se jezgrena funkcija nekako izlučila iz generativnog modela, to bi predstavljalo kombinaciju generativnog i diskriminativnog pristupa; diskriminativni model koji radi sa značajkama koje su nekako dobivene pomoću generativnog modela. U nastavku će biti opisan jedan takav pristup kod kojeg će diskriminativni model uspoređivati generativni proces jednog vektora značajki s generativnim procesom drugog vektora značajki.

U odjeljku 2.2.4 opisano je kako odabir jezgrene funkcije definira metriku prostora značajki (udaljenost dvaju primjera u prostoru značajki). Izlučivanjem jezgrene funkcije iz generativnog modela postiže se da sam generativni model, odnosno generativni proces modela $P(\mathbf{x}|\theta)$ (θ predstavlja parametre generativnog

modela), definira metriku prostora značajki.
Usporedba generativnih procesa dvaju različitih vektora značajki svodi se na promatranje kako pojedini parametar generativnog procesa utječe na generiranje pojedinog primjera. U tu svrhu može se iskoristiti gradijent log-izglednosti generativnog modela.

Izglednost (engl. *likelihood*), $\mathcal{L}(\theta|\mathbf{D})$ (θ predstavlja parametre generativnog modela, a $\mathbf{D}$ skup podataka) predstavlja drugačiji pogled na funkciju vjerojatnosti $P(\mathbf{D}|\theta)$. Funkcija vjerojatnosti ima fiksirane parametre modela θ te njen izlaz govori kolika je vjerojatnost da je model s takvim parametrima generirao skup podataka $\mathbf{D}$. S druge strane, izglednost nema fiksirane parametre nego je fiksiran skup podataka $\mathbf{D}$ te kao izlaz daje *izglednost* da su ti podaci generirani iz modela s parametrima θ. Izglednost, za razliku od funkcije gustoće vjerojatnosti, nema strogu probabilističku interpretaciju; površina ispod krivulje njene funkcije ne mora biti jedinična (ali veća izglednost modela obično implicira kvalitetniji model).

Kod učenja modela, cilj je maksimizirati njegovu izglednost; uz fiksirani skup podataka za učenje $\mathbf{D}$ odabrati one parametre modela θ koji maksimiziraju izglednost da su podaci generirani iz takvog modela. Općeniti oblik izglednosti dan je u nastavku:

$$\mathcal{L}(\theta|\mathbf{D}) = P(\mathbf{D}|\theta) = \prod_{i=1}^{N} P(x_i|\theta), \tag{2.9}$$

gdje N predstavlja veličinu skupa za učenje ($N = |D|$). Pretpostavka je da su podaci iz skupa D međusobno nezavisni (što obično nije istina, ali uveliko olakšava matematičko modeliranje problema).

Kako bi se olakšao optimizacijski postupak, obično se promatra logaritmirana vrijednost izglednosti – tzv. log-izglednost (jednadžba (2.10)). Motivacija za time vidljiva je iz jednadžbe (2.9) u kojoj se pojavljuje produkt preko vjerojatnosti svakog pojedinog vektora značajki iz skupa za učenje da je generiran iz modela. Logaritam će taj produkt razbiti na sumu što će olakšati i ubrzati optimizacijski postupak. Pošto je logaritam monotona funkcija, on neće promijenit lokaciju maksimuma funkcije izglednosti.

$$\ln \mathcal{L}(\theta|\mathbf{D}) = \sum_{i=1}^{N} \ln P(x_i|\theta) \tag{2.10}$$

Intuitivno je da gradijent log-izglednosti s obzirom na parametre modela, $\nabla_\theta \ln \mathcal{L}(\theta|\mathbf{D})$, pokazuje u kojem smjeru treba mijenjati parametre modela kako

bi se maksimizirala log-izglednost modela. Time možemo opisati kako pojedini parametar modela utječe na proces generiranja značajki nekog fiksiranog vektora značajki $\mathbf{x}$. To je upravo opis generativnog procesa primjera $\mathbf{x}$ koji se može iskoristiti za izlučivanje jezgrene funkcije iz generativnog modela.

Pristup koji bi koristio gradijent log-izglednosti modela obavljao bi usporedbu značajki u gradijentnom prostoru parametara modela. Takav prostor može se predstaviti pomoću Riemannove mnogostrukosti (engl. *manifold*). Pojednostavljeno, mnogostrukost je prostor u kojem se prilikom izračuna udaljenosti između dvije točke u obzir uzima i reljef (zakrivljenost) prostora. Kao najjednostavniji primjer može se uzeti kružnica; udaljenost neke točke na kružnici od njene nasuprotne točke iznosi $2r$ (r predstavlja polumjer kružnice), ali ako se uzme u obzir činjenica da prilikom kretanja od jedne do druge točke moguće kretati samo po toj kružnici, udaljenost ispada $r\pi$ (polovica opsega kružnice). Riemannova mnogostrukost specifična je po tome što je glatka u svakoj točki što povlači i diferencijabilnost u svakoj točki (važno za gradijent log-izglednosti).

Općeniti oblik modela može se zapisati kao $P(\mathbf{D}|\theta)$, gdje za parametre modela vrijedi $\theta \in \Theta$ (Θ predstavlja skup svih mogućih parametara modela). Riemannova mnogostrukost razreda modela koji parametre crpe iz Θ može se zapisati kao M_Θ. Metrika u M_Θ definirana je pomoću Fisherove informacijske matrice.

Fisherova informacija govori koliko neka slučajna varijabla $\mathbf{x}$ nosi informacija o vrijednosti parametra θ_i nekog modela. Prethodno opisan gradijent log-izglednosti modela naziva se još i *Fisherova vrijednost* (engl. *Fisher score*) te se zapisuje kao

$$U_{\mathbf{x}} = \nabla_\theta \ln \mathcal{L}(\theta|\mathbf{x}). \tag{2.11}$$

Za Fisherovu vrijednost vrijedi da joj je očekivanje (prvi moment) jednako 0,

$$\mathrm{E}[U_D \mid \theta] = \int \frac{\frac{\partial}{\partial\theta}p(x|\theta)}{p(x|\theta)} p(x|\theta)dx = \frac{\partial}{\partial\theta}\int p(x|\theta)dx = \frac{\partial}{\partial\theta}1 = 0 \tag{2.12}$$

Fisherova informacijska matrica (F) koristi se kada θ predstavlja vektor parametara, $\theta = [\theta_1\ \theta_2 \ldots \theta_N]^\top$. Ona odgovara drugom momentu Fisherove vrijednosti,

$$\mathrm{F}(\theta) = \mathrm{E}[U_D U_D^\top \mid \theta] \tag{2.13}$$

U slučaju da je log-izglednost dvostruko diferencijabilna funkcija, Fisherova informacijska matrica može se zapisati i kao

$$\mathrm{F}(\theta) = -\mathrm{E}\left[\frac{\partial^2}{\partial^2\theta} \ln P(\mathbf{D}|\theta) \;\middle|\; \theta\right] \tag{2.14}$$

Fisherova informacija može se interpretirati kao zakrivljenost gradijentnog prostora parametara (M_Θ) u okolici pozicije koja predstavlja Fisherovu vrijednost; velika zakrivljenost nosi više informacija od male zakrivljenosti. To je u skladu s korištenjem druge derivacije u jednadžbi (2.14). Pošto je očekivanje Fisherove vrijednosti jednako nuli, Fisherova informacija predstavlja upravo varijancu Fisherove vrijednosti.

Kao što je prethodno spomenuto, Fisherova informacijska matrica definira metriku gradijentnog prostora parametara modela (M_Θ). Udaljenost između dvaju susjednih modela $P(\mathbf{D}|\theta)$ i $P(\mathbf{D}|\theta+\delta)$ može se definirati kao

$$D(\theta, \theta+\delta) = \|P(\mathbf{D}|\theta) - P(\mathbf{D}|\theta+\delta)\| = \sqrt{\delta^\top \mathrm{F}\delta} \tag{2.15}$$

Kada $\delta \to \mathbf{0}$, ta udaljenost predstavlja informacijsku dobit pomaka za δ u parametarskom prostoru. Ta udaljenost, između dvije funkcije gustoće vjerojatnosti, naziva se Kullback-Leiblerova udaljenost (ili divergencija). U općenitom obliku, Kullback-Leiblerova divergencija je mjera za sličnost dviju distribucija (iako ne predstavlja pravu metriku), npr. P_1 i P_2, a zapisuje se kao $D_{KL}(P_1\|P_2)$. Ona izražava gubitak informacije kada se distribucija P_1 aproksimira pomoću distribucije P_2. To je pogodna mjera za usporedbu slika koje su predstavljene kolekcijom značajki.

Fisherovu vrijednost, $U_\mathbf{D}$, možemo promatrati kao funkciju koja preslikava vektor značajki $\mathbf{x}$ iz ulaznog prostora u točku gradijentnog prostora Riemannove mnogostrukosti parametara modela, tzv. mapiranje Fisherove vrijednosti. Pomoću Fisherove vrijednosti može se definirati "najstrmiji„ pomak u gradijentnom prostoru (pomak koji rezultira najvećom informacijskom dobiti), odnosno smjer δ koji maksimizira log-izglednost parametara modela θ uz minimalni pomak u parametarskom prostoru (M_Θ). To se naziva *prirodni gradijent* (engl. *natural gradient*), a može se izraziti pomoću klasičnog gradijenta (Fisherova vrijednost) i Fisherove informacijske matrice kao $\psi_\mathbf{D} = \mathrm{F}^{-1}U_\mathbf{D}$. Svrha inverza Fisherove informacijske matrice u produktu je normalizacija Fisherove vrijednosti u gradijentnom prostoru parametara (minimalni korak δ za maksimalno povećanje informacijske dobiti).

Jezgrena funkcija koja mjeri sličnost dvaju vektora značajki u M_Θ može se definirati kao

$$K(x_i, x_j) = \psi_{x_i}^\top \mathrm{F}\psi_{x_j}^\top = U_{x_i}^\top \mathrm{F}^{-1}U_{x_j} \tag{2.16}$$

Takva jezgrena funkcija naziva se *Fisherova jezgra*. Često se zbog kompleksnosti njenog izračuna i invertiranja ona zanemaruje te se jednostavno aproksimira jedi-

ničnom matricom. U nastavku će biti prikazan pristup koji aproksimira Fisherovu informacijsku matricu kao dijagonalnu matricu uz određene uvjete.

2.3.2. Fisherov vektor

Fisherova jezgra može se iskoristiti za klasifikaciju slika, kao vrsta proširenja pristupa klasifikacije pomoću histograma slikovnih riječi.

Kao generativni model okana slike koristi se Gaussova mješavina (engl. *GMM - Gaussian Mixture Model*), gdje svaka multivarijantna Gaussova razdioba predstavlja jednu slikovnu riječ. Za razliku od pristupa opisanog u odjeljku 2.1, slikovna riječ ovdje nije predstavljena samo srednjom vrijednosti grupe (centroida) nego dodatno i varijancom i težinom distribucije. Parametri Gaussove mješavine su sljedeći: $\theta = \{w_i,\ \mu_i,\ \Sigma_i \mid i = 1, 2, \ldots, W\}$; w_i predstavlja težinu i-te Gaussove gustoće, μ_i predstavlja srednju vrijednost i-te Gaussove gustoće, Σ_i kovarijacijsku matricu i-te Gaussove gustoće u mješavini, a W broj Gaussovih gustoća u mješavini. U kontekstu slikovnog rječnika, w_i predstavlja učestalost pojave i-te slikovne riječi, μ_i predstavlja srednju vrijednost i-te slikovne riječi, Σ_i predstavlja kovarijancu oko i-te slikovne riječi, a W broj slikovnih riječi. Za Σ_i se pretpostavlja da je dijagonalna matrica kako bi se olakšalo računanje determinante i inverza. Uz tu aproksimaciju, kovarijacijska matrica prelazi u vektor varijanci, $\sigma^2 = diag(\Sigma)$. Uobičajeni način učenja Gaussove mješavine je pomoću algoritma maksimizacije očekivanja (engl. *EM - expectation maximization*).

Težine riječi takvog modela, podliježu uvjetima

$$\forall i,\ w_i \geq 0,\ \sum_{i=1}^{W} w_i = 1 \tag{2.17}$$

Kako bi se izbjeglo baratanje tim uvjetom prilikom izračuna derivacija log-izglednosti, w_i se može prikazati kao normalizirana eksponencijalna funkcija, tzv. *softmax* funkcija koja u sebi inherentno sadrže svojstva navedena pod (2.17), [3]. Oblik *softmax* funkcije dan je u nastavku:

$$w_i = \frac{\exp(\alpha_i)}{\sum_{j=1}^{W} \exp(\alpha_j)} \tag{2.18}$$

U nastavku se pretpostavlja da je slika opisana skupom vektora značajki slike, $\mathbf{D} = \{x_i \mid i = 1, 2 \ldots, N\}$. Dimenzija ulaznog prostora je d.

Sljedeći korak je iz takvog modela izlučiti gradijent log-izglednosti kako bi se mogla izračunati Fisherova vrijednost. Uz pretpostavku da ulazni primjeri podliježu identičnim i nezavisnim distribucijama (engl. *iid - independent and identically distributed*), log-izglednost Gaussove mješavine s parametrima θ za ulazni skup primjera za učenje $\mathbf{D}$ je

$$\mathcal{L}(\theta|\mathbf{D}) = \sum_{i=1}^{N} \ln P(x_i|\theta) \tag{2.19}$$

Ovdje ponovno proizlazi prednost korištenja log-izglednosti naspram obične izglednosti; postignuta je aditivnost doprinosa log-izglednosti pojedinih vektora značajki iz skupa $\mathbf{D}$ ukupnoj log-izglednosti modela. Log-izglednost pojedinog vektora značajki je

$$\mathcal{L}(\theta|\mathbf{x}) = \ln P(\mathbf{x}|\theta) = \ln \sum_{i=1}^{W} w_i p_i(\mathbf{x}|\theta) \tag{2.20}$$

Funkcija $p_i(\mathbf{x}|\theta)$ predstavlja funkciju gustoće vjerojatnosti i-te Gaussove distribucije iz mješavine. U nastavku je dan općenit oblik te funkcije.

$$p_i(\mathbf{x}|\theta) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_i)^\top \Sigma_i^{-1}(\mathbf{x}-\mu_i)\right\} \tag{2.21}$$

Zbog aproksimacije kovarijacijske matrice dijagonalnom matricom, gornja jednadžba prelazi u oblik

$$p_i(\mathbf{x}|\theta) = \frac{1}{\sqrt{(2\pi)^d \prod \sigma_i^2}} \exp\left\{-\frac{1}{2}\left(\frac{\mathbf{x}-\mu_i}{\sigma_i^2}\right)^\top (\mathbf{x}-\mu_i)\right\} \tag{2.22}$$

Za neki podatak $\mathbf{x}$, svaka od komponenata Gaussove mješavine u nekoj je mjeri doprinijela njegovom generiranju. Doprinos pojedine komponente naziva se *odgovornost* (engl. *responsibility, soft-assign*) te se može izraziti kao

$$\gamma_i(\mathbf{x}) = \frac{w_i p_i(\mathbf{x}|\theta)}{\sum_{j=1}^{W} w_j p_j(\mathbf{x}|\theta)} \tag{2.23}$$

Ovo se razlikuje od pristupa opisanog u odjeljku 2.1 gdje samo jedan centroid preuzima odgovornost za neki vektor značajki – čvrsto pridruživanje (engl. *hard asign*), dok u slučaju Gaussove mješavine svaka komponenta je u nekoj mjeri odgovorna – meko pridruživanje (engl. *soft asign*).

Za primjenu Fisherove jezgre potrebno je izraziti gradijent log-izglednosti. Deriviranjem po svim parametrima modela, dobivaju se tri općenita slučaja:

$$\nabla_{\alpha_i} \ln \mathcal{L}(\theta|\mathbf{x}) = \gamma_i(\mathbf{x}) - w_i \tag{2.24}$$

$$\nabla_{\mu_i} \ln \mathcal{L}(\theta|\mathbf{x}) = \gamma_i(\mathbf{x}) \left(\frac{\mathbf{x} - \mu_i}{\sigma_i^2} \right) \tag{2.25}$$

$$\nabla_{\sigma_i} \ln \mathcal{L}(\theta|\mathbf{x}) = \gamma_i(\mathbf{x}) \left[\frac{(\mathbf{x} - \mu_i)^2}{\sigma_i^3} - \frac{1}{\sigma_i} \right] \tag{2.26}$$

U Fisherovoj jezgri, osim gradijenta log-izglednosti, također se pojavljuje i inverz Fisherove informacijske matrice. Kao što je spomenuto u odjeljku 2.3.1, zbog računalne kompleksnosti, ona se često aproksimira jediničnom matricom. U ovom slučaju takva aproksimacija se izbjegava, ali će se ipak pokušati doći do jednostavnije reprezentacije poput dijagonalne matrice. Uz pretpostavku da vrijedi $\forall \mathbf{x} \in D,\ \gamma_{f(\mathbf{x})}(\mathbf{x}) \approx 1 \implies \forall i \neq f(\mathbf{x}),\ \gamma_i(\mathbf{x}) \approx 0$ gdje f predstavlja funkciju koja za vektor značajki $\mathbf{x}$ odabire dominantno odgovornu komponentu Gaussove mješavine (navedena jednadžba zapravo pretpostavlja čvrsto pridruživanje), Fisherova informacijska matrica može se aproksimirati dijagonalnom matricom. Navedena aproksimacija pretpostavlja da će većinu odgovornosti za neki podatak preuzeti jedna distribucija iz mješavine. U nastavku su dane aproksimacije dijagonalnih elemenata:

$$\mathrm{F}_{\alpha_i} = w^\top w - w_i \tag{2.27}$$

$$\mathrm{F}_{\mu_i} = \sigma_i^{-2} w_i \tag{2.28}$$

$$\mathrm{F}_{\sigma_i} = 2\sigma_i^{-2} w_i \tag{2.29}$$

Kao što je prikazano u odjeljku 2.3.1 (jednadžba (2.16)), Fisherova jezgra ima oblik

$$K(x_i, x_j) = \psi_{x_i}^\top \mathrm{F} \psi_{x_j}^\top = U_{x_i}^\top \mathrm{F}^{-1} U_{x_j} \tag{2.30}$$

Fisherova informacijska matrica je pozitivno-semi definitna pa stoga ima inverz i može se rastaviti (pomoću Choleskyjeve dekompozicije) na umnožak $\mathrm{F} = \sqrt{\mathrm{F}}\sqrt{\mathrm{F}}$. Koristeći to svojstvo, vidljivo je da Fisherova jezgra zapravo uspoređuje dva vektora značajki koji se dobivaju kao

$$\phi(\mathbf{x}) = \mathrm{F}^{-0.5} U_{\mathbf{x}} \tag{2.31}$$

te upravo to i predstavlja *Fisherov vektor* (direktno preslikavanje).
Pojedine komponente Fisherovog vektora dane su u nastavku, [4]:

$$\phi_{\alpha_i}(\mathbf{D}) = w_i^{-0.5} \sum_{k=1}^{N} (\gamma_i(\mathbf{x_k}) - w_i) \tag{2.32}$$

$$\phi_{\mu_i}(\mathbf{D}) = w_i^{-0.5} \sum_{k=1}^{N} \gamma_i(\mathbf{x}) \left(\frac{\mathbf{x_k} - \mu_i}{\sigma_i} \right) \tag{2.33}$$

$$\phi_{\sigma_i}(\mathbf{D}) = (2w_i)^{-0.5} \sum_{k=1}^{N} \gamma_i(\mathbf{x}) \left[\frac{(\mathbf{x_k} - \mu_i)^2}{\sigma_i^2} - 1 \right] \tag{2.34}$$

Aditivnosti doprinosa log-izglednosti pojedinih vektora značajki sveukupnoj log-izglednosti modela prenosi se i na Fisherov vektor. Općeniti oblik Fisherovog vektora za skup podataka tada je

$$\phi_\theta(D) = \sum_{i=1}^{N} \phi_\theta(\mathbf{x_i}) \tag{2.35}$$

Kako bi se izbjegao utjecaj veličine skupa podataka na Fisherov vektor, možemo ga normalizirati veličinom skupa podataka

$$\phi_\theta(D) \leftarrow \frac{1}{N} \phi_\theta(D) \tag{2.36}$$

U prethodnim izračunima, radi pojednostavljenja, pretpostavljena je međusobna nezavisnost svih vektora značajki iz skupa D. U stvarnosti, vektori značajki biti će međusobno zavisni te će to rezultirati Fisher vektorima koji će biti neravnomjerno popunjeni; većinski dio težine biti će sadržan u manjem dijelu vektora što negativno utječe na usporedbu skalarnim produktom. Kako bi se umanjio utjecaj te pogrešne pretpostavke, uobičajeno je obaviti *normalizaciju potenciranjem* (engl. *power norm*). Obično se koristi *predznačeno korijenovanje*.

$$\phi_\theta := sign(\phi_\theta)|\phi_\theta|^p, \; p \in \langle 0, \; 1] \tag{2.37}$$

Iz jednadžbi (2.32), (2.33) i (2.34) vidljivo je da Fisherov vektor mjeri odstupanja značajke od generativnog modela. Značajka koja se često pojavljuje (npr. značajke pozadine) imati će malu vrijednost Fisherovog vektora, dok će značajke koje se rijeđe pojavljuju (i time vjerojatno više odstupaju od generativnog modela) imati veliku vrijednost Fisherovog vektora. Ovdje se može pojaviti problem

da su dvije slike koje predstavljaju isti razred, ali sadrže različitu količinu pozadinskog šuma (ili drugačije rečeno, postoci relevantnih vektora značajki u skupu opisnika im se razlikuju) predstavljene veoma različitim Fisherovim vektorima. Kako bi se izbjegao ujecaj broja relevantnih značajki u skupu podataka, Fisherov vektor dodatno se normalizira *l2* normom.

$$\phi_\theta := \frac{\phi_\theta}{\sqrt{\phi_\theta^\top \phi_\theta}} \tag{2.38}$$

2.3.3. Usporedba s histogramima slikovnih riječi

Histogrami slikovnih riječi u obzir uzimaju samo frekvenciju pojave slikovnih riječi. Dimenzionalnost histograma jednaka je broju riječi slikovnoga rječnika.
Fisherov vektor, osim frekvencije pojave slikovnih riječi, u svoj opisnik dodatno uključuje informaciju o srednjim vrijednostima i varijancama komponenata modela. Dimenzija Fisherovog vektora je $W(2d+1)$, što obično predstavlja dostatno veliki vektor da bi se opravdalo korištenje linearnog klasifikatora. Fisherov vektor u sebi inherentno sadrži preslikavanje u gradijentni prostor parametara modela (Fisherova jezgra) pa korištenje linearnog klasifikatora (s linearnom jezgrom) zapravo predstavlja korištenje klasifikatora s Fisherovom jezgrom.

3. Distribucijske šume i Fisherovi vektori

U prethodnom poglavlju opisan je pristup izračuna Fisherovih vektora uz korištenje Gaussove mješavine kao generativnog modela slikovnih okana. U nastavku će biti prikazan pristup koji kao generativni model koristi slučajne distribucijske šume.

3.1. Stabla i šume odluke

3.1.1. Stabla odluke

Stablo, kao podatkovna struktura, sastoji se od unutarnjih čvorova, listova (krajnjih čvorova) te veza između čvorova. Ovisno o broju djece koju unutarnji čvor može imati stabla se dijele na binarna (dva djeteta), ternarna (tri djeteta) ili općenito *k-narna*.

Stablo odluke predstavlja jednu vrstu diskriminativnog klasifikatora. Svaki unutarnji čvor stabla sadrži evaluacijsku funkciju. Svaki podatak koji dolazi u unutarnji čvor podliježe evaluaciji te se na temelju dobivenog rezultata šalje na daljnju obradu lijevom ili desnom djetetu. Takva evaluacijska funkcija obično je veoma primitivna te svoju odluku bazira na manjem skupu atributa vektora značajki. Primjer jedne takve funkcije koja evaluira vektor značajki fiksne duljine v može se izraziti kao

$$AKO\ \boldsymbol{v}[n] > k\ ONDA\ desnoDijete.eval(\boldsymbol{v})\ INAČE\ lijevoDijete.eval(\boldsymbol{v})$$

Takav klasifikator naziva se slabim klasifikatorom jer sam po sebi nije dovoljan za ispravnu klasifikaciju nekog kompleksnog podatka (sam po sebi malo je bolji od nasumičnog pogađanja).

Na putu od korijena stabla do listova, podatak v podliježe većem broju evaluacija slabih klasifikatora (evaluacija svakog unutarnjeg čvora na tom putu).

Takav put, sastavljen od slabih klasifikatora, predstavlja jak klasifikator koji, pod uvjetom da je stablo dobro naučeno, može dobro generalizirati podatke. Evaluacijska funkcija listova stabla određuje konačnu klasifikaciju podataka. Najjednostavniji oblik evaluacijske funkcije lista bio bi da svakom podatku koji dođe u njega pridruži jednaku oznaku (npr.svaki podatak koji završi u određenom listu predstavlja podatak tipa A). Evaluacijska funkcija može biti i kompleksnija, tako da dopusti listu da predstavlja više različitih razreda podataka.

Stabla odluke predstavljaju primjenu „podijeli pa vladaj "pristupa rješavanju kompleksnih problema. Kompleksan problem klasifikacije dijeli se na više manjih i jednostavnijih potproblema čijim rješavanjem dolazimo do rješenja početnog problema.

3.1.2. Učenje stabala odluke

U nastavku se pretpostavlja postojanje skupa za učenje s označenim podacima, $D = \{(x_i, c_i) \mid i = 1,\ 2, \ldots,\ N\}$ gdje x_i predstavlja i-ti vektor značajki, a c_i oznaku i-tog vektora značajki. Za c vrijedi $c \in C$, gdje C predstavlja skup mogućih oznaka podataka.

Učenje unutarnjeg čvora j svodi se na pitanje kako najbolje odijeliti ulazne podatke, tako da vrijedi $D_j = D_j^L \cup D_j^R$. D_j predstavlja skup podataka koji je došao u čvor j, a D_j^L podatke koji se nakon primjene evaluacijske funkcije šalju lijevom djetetu čvora j (analogno za D_j^R). Uvjet koji mora biti zadovoljen je $D_j^L \cap D_j^R = \emptyset$. Problem odjeljivanja podataka može se matematički formalno izraziti koristeći informacijsku dobit i entropiju.

U teoriji informacije, entropija se definira kao mjera za *nered* podataka (ili alternativno kao količina informacije koju nosi neka poruka $\mathbf{x}$). U slučaju podataka koji su nasumično razbacani po prostoru entropija će biti velika, dok u slučaju da su podaci koncentrirani u manjem dijelu prostora entropija će biti malena. Uobičajeno je koristiti tzv. Shannonovu entropiju

$$H(D) = -\sum_{c \in C} p(c) \ln p(c) \tag{3.1}$$

gdje je C skup svih mogućih tipova podataka koje x može poprimiti (skup klasa), a $p(c)$ vjerojatnost da nasumično odabrani primjer $\mathbf{x}$ iz skupa D bude tipa c.

Za neoznačene podatke može se primijeniti diferencijalna entropija, ali o njoj će više biti rečeno kasnije.

Podjelom skupa podataka na dva disjunktna skupa možemo izračunati informacijsku dobit takve podjele. Informacijska dobit biti će tim veća što će entropija dva nova skupa biti manja. Matematički izražena, takva informacijska dobit ima oblik

$$I = H(D) - \sum_{i=L,\ R} \frac{|D^i|}{|D|} H(D^i) \tag{3.2}$$

Evaluacijsku funkciju unutarnjeg čvora j možemo definirati kao

$$\psi_j(\theta_j) : \mathbf{x} \mapsto \{L,\ R\} \tag{3.3}$$

gdje θ_j predstavlja parametre evaluacijske funkcije, a izlaz oznaku lijevog (L) ili desnog (D) djeteta čvora. Parametri θ_j odabiru se tako da podjela podataka inducirana evaluacijskom funkcijom maksimizira informacijsku dobit. Optimizacijsku funkcija unutarnjeg čvora j tada možemo izraziti kao

$$\theta_j^* = \arg\max_{\theta_j} I_j \tag{3.4}$$

Skup parametara θ_j obično je beskonačan pa se pretražuje samo maleni podskup mogućih parametara; τ_j. Optimizacijska funkcija tada poprima oblik

$$\theta_j^* = \arg\max_{\theta_j \in \tau_j} I_j \tag{3.5}$$

Podskup parametara τ_j može biti odabran heuristički ili pak potpuno nasumično. U slučaju nasumičnog odabira, stabla odluke nazivamo slučajna stabla odluke .

3.1.3. Šume odluke

Šuma odluke podatkovna je struktura koja se sastoji od više stabala odluke. U nastavku se pretpostavlja korištenje slučajnih stabala odluke.
Proces učenja šume svodi se na zasebno učenje pojedinog stabla. Procesi treniranja više stabala međusobno su nezavisni pa se mogu trivijalno paralelizirati.

Kada bi proces učenja stabla bio vođen nekom determinističkom heuristikom, pod uvjetom da je svako stablo učeno na istom skupu podataka, sva stabla bi ispala potpuno jednaka. Uvođenjem nedeterminizma u proces učenja to se izbjegava. Pod uvjetom da se parametri skupa τ_j odabiru uniformno iz skupa θ_j, međusobna zavisnost stabala može se izraziti kao $\frac{|\tau_j|}{|\theta_j|}$. Mala vrijednost tog omjera implicira da je svakom čvoru ponuđen malen broj mogućih podjela što

pak implicira da je mala šansa da će dva različita stabla u nekom čvoru dobiti ponuđene jednake podjele (stabla su nezavisna). Velika vrijednost tog omjera implicira suprotno, te je za očekivati da će stabla biti međusobno slična (a time i zavisna).

Šume odluke koriste odluke svih svojih stabala kako bi donijele konačnu odluku o razredu nekog vektora značajki. Jednaki podatak prosljeđuje se svakom od stabala te se uzima u obzir izlaz svakog od listova (npr. većinskim glasanjem).

Šuma kao skupni model, mnogo je robusnija od pojedinog stabla. Razlog tome je što stabla unutar šume mogu biti komplementarna; jedno stablo može se specijalizirati za rješavanje jednog podskupa problema dok se neko drugo stablo može specijalizirati na nekom drugom podskupu.

3.2. Distribucijska stabla i šume

3.2.1. Distribucijska stabla

Distribucijska stabla predstavljaju poseban oblik stabala odluke. Unutarnji čvorovi i dalje predstavljaju slabe klasifikatore (evaluacijske funkcije), ali listovi sada predstavljaju distribucije. U nastavku će biti pretpostavljeno korištenje Gaussovih distribucija.

Svako stablo predstavlja mješavinu distribucija; svaki list predstavlja po jednu distribuciju pa stoga broj distribucija u mješavini odgovara broju listova u stablu. Model se može proširiti i na slučaj gdje pojedini list predstavlja malu mješavinu distribucija, ali to se ovdje ne razmatra.

Ovakav model veoma je sličan Gaussovoj mješavini, ali se ponajviše razlikuje u poimanju odgovornosti pojedine distribucije iz mješavine za generiranje nekog podatka. U slučaju Gaussove mješavine uzimaju se u obzir sve distribucije te se pomoću težina pojedinih distribucija i vrijednosti funkcija gustoće vjerojatnosti računa odgovornost svake od komponenata mješavine. U slučaju distribucijskog stabla, samo jedna distribucija iz mješavine je odgovorna za generiranje podatka – distribucija koju predstavlja list u koji je podatak svrstan. To zapravo predstavlja čvrsto pridruživanje, za razliku od Gaussove mješavine koja koristi meko pridruživanje.

Optimizacijska funkcija distribucijskog stabla jednaka je onoj stabla odluke (3.5). Izračun informacijske dobiti također ostaje nepromijenjen (3.2). Za razliku od stabla odluke gdje su dostupne oznake podataka, ovdje to nije slučaj pa stoga

Shannonova entropija nije primjenjiva. U ovom slučaju može poslužiti diferencijalna entropija, koja se izražava kao

$$H(D) = \frac{1}{2} \ln \left((2\pi e)^d |\Sigma(D)| \right) \tag{3.6}$$

gdje $\Sigma(D)$ predstavlja kovarijacijsku matricu skupa podataka D.
Uz zanemarivanje konstanti, jednadžba informacijske dobiti prelazi u oblik

$$I = \ln |\Sigma(D)| - \sum_{i=\{L,R\}} \frac{|D^i|}{|D|} \ln |\Sigma(D^i)| \tag{3.7}$$

Determinanta matrice može se promatrati kao funkcija koja je proporcionalna s volumenom hiperprostora kojeg opisuju njeni stupci (stupci predstavljaju vektore). Velika kovarijanca elemenata rezultirati će hiperprostorom velikog volumena dok će mala kovarijanca rezultirati manjim i kompaktnijim hiperprostorom. Optimizacijska funkcija će na taj način preferirati podjele koje ulazni prostor dijele na kompaktne potprostore.

Nakon završetka učenja unutarnjih čvorova distribucijskog stabla, svaki list definira se parametrima distribucije (μ_l i Σ_l) i svojom težinom π_l. Parametar μ_l predstavlja srednju vrijednost , a Σ_l kovarijacijsku matricu svih podataka koji su završili u listu l. Težina lista π_l omjer je broja podataka koji su završili u listu i broja podataka u skupu za učenje, $\pi_l = \frac{|D^l|}{D}$.

Funkcija l pridružuje vektoru značajki $\mathbf{x}$ njegov odgovarajući list iz stabla t te se može izraziti kao

$$l(\mathbf{x}) \ : \ \mathbb{R}^d \to \mathbb{L} \in \mathbb{N} \tag{3.8}$$

gdje $\mathbb{L}$ predstavlja skup svih listova stabala (njihove indekse).
Funkcija gustoće razdiobe distribucijskog stabla može se zapisati kao

$$p_t(\mathbf{x}) = \mathcal{N}(\mathbf{x} \mid \mu_{l(\mathbf{x})}, \ \Sigma_{l(\mathbf{x})}) \tag{3.9}$$

gdje $\mathcal{N}$ predstavlja normalnu (Gaussovu) distribuciju.

3.2.2. Odabir slabog klasifikatora

Odabir slabog klasifikatora koji će stablo koristiti bitno utječe na performanse stabla. Kompliciraniji klasifikatori moći će bolje odijeliti podatke, ali će biti računski zahtjevniji za izvođenje od jednostavnijih, linearnih klasifikatora.

Stablo koje na raspolaganju ima samo linearne klasifikatore neće moći najbolje odijeliti podatke koji su predstavljeni distribucijama između kojih se nalazi

nelinearna granica (što je obično slučaj). Ali ipak, nedostatak jednostavnijih linearnih klasifikatora može se umanjiti korištenjem distribucijske šume. U nastavku će biti korišten tip slabog klasifikatora koji odluku donosi usporedbom samo jedne dimenzije vektora značajki (usporedba s nekom konstantom), tzv. ogranak odluke (engl. *decision stump*). Takva usporedba predstavlja linearni klasifikator poravnan s osima.

Kada se odabir dimenzije i konstante za evaluaciju vektora značajki obavlja slučajnim odabirom, govorimo o slučajnim distribucijskim stablima ili u slučaju skupnog modela, slučajnim distribucijskim šumama.

3.2.3. Distribucijske šume

Funkcija gustoće vjerojatnosti distribucijske šume kao skupnog modela može se izraziti kao

$$p_{\mathcal{F}}(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} p_t(\mathbf{x}) \tag{3.10}$$

gdje T predstavlja broj distribucijskih stabala u šumi. Koristeći funkciju gustoće vjerojatnosti možemo izraziti log-izglednost modela kao

$$\mathcal{L}(D \mid \theta_{\mathcal{F}}) = \sum_{i=1}^{N} \ln \frac{1}{T}\sum_{t=1}^{T} p_t(\mathbf{x_i}) \tag{3.11}$$

gdje je $\mathcal{F}$ oznaka distribucijske šume.

Oblik funkcije gustoće vjerojatnosti (i log-izglednosti) distribucijske šume gotovo je identičan onome od Gaussove mješavine. Glavna razlika je u tome što se kod šume, za vektor značajki $\mathbf{x}$, ne evaluiraju sve distribucije modela nego samo one u pripadnim listovima.

3.2.4. Parametri učenja modela

Za učenje nedeterminističkih distribucijskih stabala (i šuma) potrebno je definirati nekoliko parametara. Učenje stabla regulirano je s četiri parametara, [5]:

- Dubina stabla ili broj listova
- Minimalna informacijska dobit
- Minimalni kardinalitet skupa podataka u listu
- Broj dopuštenih podjela

Šuma je regulirana samo brojem stabala.

Dubina stabla ili broj listova

Dubina stabla predstavlja najvažniji parametar učenja stabla. Veća dubina stabla implicira veći broj listova stabala (veći kapacitet modela) što znači da će ulazni prostor biti finije particioniran. Ako je odabrana prevelika dubina, model se može prilagoditi šumu u podacima. S druge strane, ako se odabere premalena dubina, model neće imati dovoljno kapaciteta kako bi opisao distribucije kojima podliježe ulazni prostor.

Minimalna informacijska dobit

Minimalna informacijska dobit definira najmanji dopušteni iznos informacijske dobiti podjele podataka unutarnjeg čvora koji dopušta njegovo račvanje (stvaranje djece). Time se želi spriječiti račvanje čvora koje zapravo ne bi rezultiralo boljom procjenom distribucija. Kada odabrana podjela čvora ne zadovoljava ovaj uvjet, čvor se pretvara u list.

Minimalni kardinalitet skupa podataka u listu

Pošto svaki list predstavlja po jednu distribuciju, ovim parametrom želi se postići da je svaka distribucija zastupljena relevantnom količinom podataka.

Navedena tri parametra služe kao regularizatori prenaučenosti modela.

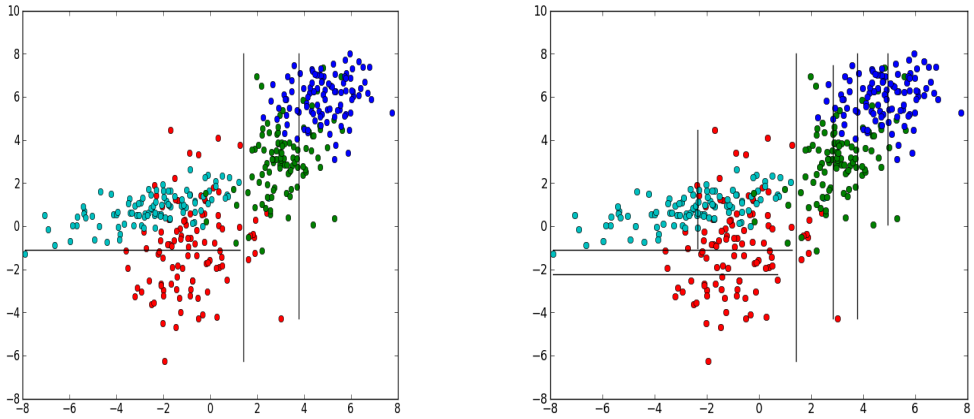
Broj dopuštenih podjela

Broj dopuštenih podjela određuje koliko slučajno generiranih podjela podataka razmatra svaki unutarnji čvor. Za razliku od ostalih parametara učenja stabla, ovaj parametar ne regularizira prenaučenost modela nego utječe na međusobnu zavisnost stabala u šumi. Veći dopušteni broj podjela može rezultirati kvalitetnijim stablima, ali pod cijenu gubitka međusobne nezavisnosti; stabla mogu biti međusobno sličnija.

Broj stabala u šumi

Broj stabala u šumi, uz dubinu stabla, najvažniji je parametar učenja modela. U ovom slučaju više je bolje; šuma s većim brojem stabala u pravilu će bolje aproksimirati distribuciju podataka nego šuma s manjim brojem stabala, ali će zato biti memorijski i računski zahtjevnija za uporabu. Dodavanjem novog stabla u

šumu možemo promatrati kao dodavanje novog pogleda na distribuciju podataka za učenje.



(a) optimalna dubina **(b)** prevelika dubina

Slika 3.1: Utjecaj dubine (broja listova) stabala

3.3. Fisherovi vektori nad slučajnim distribucijskim šumama

U odjeljku 2.3.2 prikazan je oblik Fisherovih vektora kada se kao generativni model koristi Gaussova mješavina. U nastavku će biti prikazan njihov oblik, ali u slučaju kada je generativni model predstavljen slučajnom distribucijskom šumom.

Najprije je potrebno modificirati funkciju l, (3.8), tako da u obzir uzima postojanje više stabala u šumi; $l_t(\mathbf{x})$ gdje je t oznaka stabla koje se razmatra.
Prvi korak kod izračuna Fisherovog vektora je particioniranje ulaznog skupa podataka D po listovima stabala šume. Skup podataka koji je predstavljen listom stabla može se izraziti kao

$$D_l^t = \{\mathbf{x} \in D \mid l_t(\mathbf{x}) = l\} \tag{3.12}$$

Fisherov vektor lista l stabla t računa se na sljedeći način:

$$\phi_\mu(D_l^t) = w_{l_t(\mathbf{x})}^{-0.5} \sum_{\mathbf{x} \in D_l^t} \gamma_{l_t(\mathbf{x})}(\mathbf{x}) \left(\frac{\mathbf{x} - \mu_{l_t(\mathbf{x})}}{\sigma_{l_t(\mathbf{x})}} \right) \tag{3.13}$$

$$\phi_\sigma(D_l^t) = (2w_{l_t(\mathbf{x})})^{-0.5} \sum_{\mathbf{x} \in D_l^t} \gamma_{l_t(\mathbf{x})}(\mathbf{x}) \left(\frac{(\mathbf{x} - \mu_{l_t(\mathbf{x})})^2}{\sigma_{l_t(\mathbf{x})}^2} - 1 \right) \tag{3.14}$$

Vidljivo je da su jednadžbe za izračun Fisherovih vektora nad distribucijskim šumama , (3.13) i (3.14), gotovo identični jednadžbama za izračun Fisherovih vektora nad Gaussovom mješavinom, (2.33) i (2.34). Razlika je u dvije stvari: način izračuna odgovornosti γ i doprinosa ulaznih podataka pojedinoj komponenti (listu) Fisherovog vektora.

Kod Gaussove mješavine, podatak $\mathbf{x}$ utječe na izračun svake pojedine komponente Fisherovog vektora (svaka distribucija mješavine u nekoj mjeri odgovorna je za njegovo generiranje). Kod distribucijskih šuma, broj komponenti Fisherovog vektora na koje utječe podatak $\mathbf{x}$ određen je brojem stabala jer u svakom pojedinom stablu $\mathbf{x}$ pripada točno jednom listu.

Odgovornost pojedinog lista za generiranje vektora značajki $\mathbf{x}$ može se izraziti kao

$$\gamma_{l_t(\mathbf{x})}(\mathbf{x}) = \frac{p_{l_t(\mathbf{x})}(\mathbf{x}|\theta_{l_t(\mathbf{x})})}{\sum_{\tau \in T} p_{l_\tau(\mathbf{x})}\left(\mathbf{x}|\theta_{l_\tau(\mathbf{x})}\right)} \tag{3.15}$$

Iz (3.15) vidljivo je da na izračun odgovornosti utječu samo oni listovi kojima je podatak pridjeljen u pojedinom stablu (kod GMM-a utječu sve komponente mješavine). Npr. u slučaju šume koja se sastoji od T stabala, za vektor značajki $\mathbf{x}$ potrebno je izračunati posterior za samo T distribucija, a ne za W kao što bi bio slučaj kod GMM-a.

Dimenzija fisherovog vektora izgrađenim preko distribucijske šume jednaka je $2 * d * broj\_listova\_šume$, gdje d predstavlja dimenziju podataka u listu (pretpostavka je da je ona jednaka u svakom listu).

4. Korišteni skupovi slika

Za testiranje programske implementacije korišten je skup slika *Pascal 2007 VOC*. Sastoji se od otprilike 5000 slika za učenje i 5000 slika za testiranje (unaprijed definirana podjela, [6]). Slike se mogu svrstati u 20 različitih razreda uz mogućnost da jedna slika pripada jednom ili više od njih. Razredi se u grubo mogu podijeliti u četiri skupine: prijevozna sredstva, životinje, interijer i osobe.

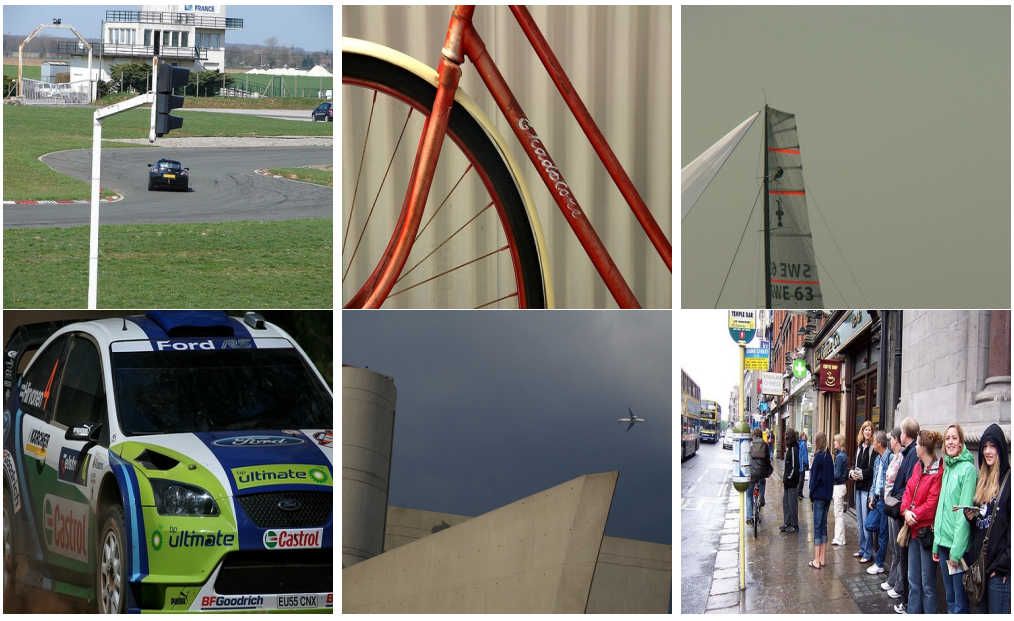
Prijevozna sredstva zrakoplov, bicikl, brod, automobil, motocikl, vlak, autobus

Životinje pas, mačka, krava, ovca, konj, ptica

Interijer boca, stolica, naslonjač, stol, ekran, lončarica

Osobe

Neki primjeri slika iz Pascala mogu se vidjeti na slici 4.1.



Slika 4.1: Pascal 2007 VOC

5. Implementacijski detalji

5.1. Korišteni alati

Programsko rješenje ostvareno je pomoću programskog jezika *Python*. *Python* je veoma ekspresivan jezik koji omogućuje brz razvoj programskih rješenja što ga čini pogodnim za prototipiranje. Glavna mana mu je što je dosta spor, ali tome se može doskočiti korištenjem biblioteka napisanim u C-u.
Za izračun SIFT-ova korištena je biblioteka *VLFeat*, [7].
Za brz rad s matricama korištena je popularna biblioteka *NumPy*.
Za brzu izvedbu *algoritma maksimizacije očekivanja* i *k-srednjih vrijednosti* korištena je biblioteka *Yael*.
Pristup algoritmima strojnog učenja ostvaren je pomoću biblioteke *sckit–learn*, [8].
Za rad sa slikama korištena je biblioteka *PIL–Python Imaging Library*.
Za iscrtavanje grafova korištena je biblioteka *matplotlib*, [9].
Sve navedene vanjske biblioteke napisane su u C-u (optimizirane i paralelizirane za što brže izvođenje), ali pružaju sučelje za rad s Pythonom. Time se postiže visoka ekspresivnost (i visoka razina apstrakcije) Pythona uz brzinu izvođenja sumjerljivom čistom C-u.

5.2. Implementacija

Implementacija stabala i šuma ostvarena je korištenjem objektno orijentirane paradigme. Detalji implementacije pojedinih komponenti dani su u nastavku.

5.2.1. Distribucijska stabla

Distribucijska stabla implementirana su kao razred *DensityTree*. On sam za sebe zapravo ne predstavlja cijelo stablo nego samo jedan čvor koji u slučaju da ne

predstavlja list sadrži pokazivače na svoju djecu (također razreda *DensityTree*). Najvažnija metoda za izgradnju stabla je *growTree*. Njen pojednostavljeni pseudokod dan je u nastavku:

Algorithm 1 *GrowTree* metoda

```
Input: depth, dataset, params
if params.maxDepth == depth then
  createLeaf(dataset)
end if
splits = generateSplits(dataset)
infoGains = [ ]
for each split in splits do
  if min(split.leftSet.size, split.rightSet.size) < params.minSplitSetCardinality
  then
    infoGains.append(0)
  else
    infoGains.append(calculateInformationGain(dataset, split))
  end if
end for
bestSplit, bestInfoGain = chooseBestSplit(splits, infoGains)
if bestInfoGain < params.minInfoGain then
  createLeaf(dataset)
else
  leftTree  = newDensityTree(params)
  rightTree = newDensityTree(params)
  leftTree.growTree(bestSplit.leftSet, depth + 1)
  rightTree.growTree(bestSplit.rightSet, depth + 1)
end if
```

Vidljivo je da metoda implementira pristup opisan u odjeljku 3.1.1. Čvor stabla će se račvati ako nije postignuta najveća dopuštena dubina te ako izabrana podjela zadovoljava uvjete minimalne informacijske dobiti i minimalnog kardinaliteta skupa. Ako samo jedan od tih uvjeta nije zadovoljen, čvor se pretvara u list.

Metoda koja definira tip distribucijskog stabla je *generateSplits()*. U slučaju da se podjele podataka generiraju nasumično, govori se o slučajnim distribucijskim stablima. Dimenzija po kojoj se podaci dijele odabire se uniformno iz skupa

svih mogućih dimenzija. Također, odabir vrijednosti po kojoj se određena dimenzija dijeli odabire se uniformno iz intervala dopuštenih vrijednosti (vrijednost između minimalne i maksimalne vrijednosti odabrane dimenzije za skup podataka u tom čvoru). Utjecaj broja dopuštenih podjela opisan je u odjeljku 3.1.2.

Podjele se također mogu i generirati koristeći Gaussovu mješavinu i algoritam maksimizacije očekivanja. Nakon učenja dvo-komponentnog GMM-a koriste se dobivene srednje vrijednosti i varijance kako bi se generirale podjele. Za svaku dimenziju odabire se točka između srednjih vrijednosti dviju komponenata tako da odabir točke ovisi o varijanci samih komponenata (npr. $dimVal_i = mean_1^{(i)} + (mean_2^{(i)} - mean_1^{(i)})\frac{var_1^{(i)}}{var_1^{(i)}+var_2^{(i)}}$). Iako učenje GMM-a pomoću EM-a inherentno sadrži nedeterminističnost, zbog jednostavnosti mješavine (samo dvije komponente) višestruko pokretanje algoritma može rezultirati veoma sličnim mješavinama. Više zasebnih stabala treniranih pomoću takvog algoritma rezultiralo bi veoma sličnim (ako ne i jednakim) stablima. Pošto ovdje ne postoji mogućnost definiranja broja dopuštenih podjela (on je jednak dimenziji ulaznog prostora) potrebno je na neki drugi način ubaciti nedeterminističnost u proces učenja. To je ostvareno unutar metode *chooseBestSplit* koja se dodatno parametrizira parametrom K. On definira veličinu skupa najboljih podjela iz kojeg se nasumično odabire jedna od njih (npr. za $K = 4$, pronalaze se četiri najbolje podjele iz skupa generiranih podjela te se nasumično odabire jedna od njih). Na taj način neće uvijek biti odabrana najbolja moguća podjela u čvoru nekog stabla, ali će se zato postići veća nezavisnost (i raznolikost) među stablima učenim takvim algoritmom. Ovakvo stablo naziva se *EM distribucijsko stablo*.
Važno je napomenuti da se u metodi za izračun diferencijalne entropije podjele (koju koristi metoda *calculateInformationGain*) koristi znatna relaksacija problema izračuna determinante kovarijacijske matrice. Naime, pretpostavlja se da je kovarijacijska matrica dijagonalna čime se izračun determinante svodi na izračun produkta dijagonalnih vrijednosti matrice.

Metoda *createLeaf* pretvara čvor u list te računa parametre distribucije koju on predstavlja. Prije izračuna parametara distribucije, skup podataka koji pripada listu može se pomoću analize glavnih komponenata (PCA) preslikati u prostor manje dimenzije. Motivacija za to može se pronaći promatranjem samog procesa učenja stabla. Cilj je minimizirati diferencijalnu entropiju podjele podataka što za posljedicu ima da svi podaci koji završe u nekom listu „žive “u nekom potprostoru ulaznog prostora – prostoru manje dimenzionalnosti od početnog ulaznog prostora. Dodatno, potprostor manje dimenzije ima za posljedicu

manju veličinu opisnika lista (manji memorijski otisak Fisherovog vektora kao i brži izračun). Veličina potprostora u listu definira se parametrom R.

Algorithm 2 *createLeaf* metoda

Input: $dataset,\ R$
if R != 0 **then**
  C := $\Sigma(dataset)$
  V := $principalEigenVectors(C)[0:R]$
  D := (dataset - $\mu(dataset)$)$^\top$V
else
  D := dataset
end if
$\mu_l := \mu(D)$
$\sigma_l^2 := diag(\Sigma(D))$

Najvažnija metoda za rad sa stablom je *assignDatasetToLeafs*. Njena svrha je ulazne podatke raspodijeliti po listovima. Uz tu metodu koristi se i *calculateDataLikelihood* koja računa izglednost podataka u listovima. Ona ja važna za izračun odgovornosti pojedinih listova za generiranje nekog podatka.
Navedene dvije metode koriste se za izračun Fisherovih vektora.

5.2.2. Distribucijske šume

Distribucijska šuma implementirana je kao razred *RandomDensityForest*.
U konstruktoru razreda moguće je specificirati željeni broj stabala i tipove stabala (nedeterminističko distribucijsko stablo ili EM stablo).
Treniranje pojedinih stabala međusobno je nezavisno pa se treniranje šume može trivijalno paralelizirati.

5.2.3. Fisherovi vektori

Modul *FisherVector* sadrži metodu za izračun Fisherovih vektora *calculateFV*. Ona je parametrizirana ulaznim podacima (npr. opisnici slike) o distribucijskom šumom.

Prvi korak izračuna Fisherovog vektora je podijeliti ulazne podatke po listovima stabala distribucijske šume. Ovdje se koristi metoda stabla *assignDatasetToLeafs*. Nakon podjele, računa se izglednost podataka u svim listovima.

Ona je potrebna za izračun odgovornosti pojedinog lista za generiranje njemu dodijeljenih podataka, (3.15).

Ovdje dolazi do izražaja razlika korištenja distribucijske šume i Gaussove mješavine kao generativnog modela. Neka se kao primjer uzme generativni model sastavljen od 256 komponenata; Gaussova mješavina sačinjen od 256 distribucija ili distribucijska šuma s četiri stabala gdje svako stablo sadrži 64 lista (jedna od mogućih konfiguracija). Izračun odgovornosti pojedine komponente Gaussove mješavine za generiranje podatka $\mathbf{x}$ zahtjeva računanje izglednosti za svih 256 komponenata. Kod distribucijske šume dovoljno je izračunati izglednost za listove kojima $\mathbf{x}$ pripada što u ovom slučaju odgovara četiri lista. Količina računanja smanjena je 64 puta ($\approx 1.5\%$ računskog zahtjeva Gaussove mješavine).

Problem koji se ovdje javlja je raštrkanost podataka po memoriji (nedostatak koherentnosti podataka među listovima različitih stabala) što može značajno usporiti računanje odgovornosti pojedinih listova. Npr. podatak $\mathbf{x}$ koji pripada listu $l_n^{t_1}$ također pripada i listovima $l_a^{t_2}$, $l_b^{t_3}$ i $l_c^{t_4}$, a podatak $\mathbf{y}$ koji također pripada listu $l_n^{t_1}$ pripada i listovima $l_d^{t_2}$, $l_e^{t_3}$ i $l_f^{t_4}$. Kod (naivnog) izračuna odgovornosti doći će do stalnog skakanja po memoriji što će značajno usporiti izračun te efektivno odbaciti prednost računanja manjeg broja izglednosti. Zato je ključno izglednosti držati na jednoj memorijskoj lokaciji (npr. jednoj tablici) kako bi se to izbjeglo.

Nakon izračuna odgovornosti, preostaje izračunati gradijent log-izglednosti modela i normalizacijske faktore Fisherove informacijske matrice, (3.13) i (3.14), za svaki list šume te opisnike svih listova spojiti u jedan opisnik, čime se dobiva konačan Fisherov vektor.

6. Eksperimentalni rezultati

Kao opisnik slike korišten je *dense SIFT* (2.1.1). Značajke su izlučene pomoću okana veličine 16x16, 24x24, 32x32, 40x40, 48x48, 56x56 i 64x64 piksela uz odgovarajuće korake pomaka okana 2, 3, 4, 5, 6, 7 i 8 piksela. Svi opisnici svedeni su na jediničnu normu.

Nad svakim Fisherovim vektorom provodi se više normalizacija. Prvo se obavlja normalizacija predznačenim korjenovanjem nakon koje se Fisherov vektor stabla svodi na jediničnu normu (l_2 normalizacija). Dodatno se na kraju cijeli Fisherov vektor svodi na jediničnu normu pomoću l_2 normalizacije.

Kao klasifikator korišten je linearni stroj s potpornim vektorima (engl. *support vector machine – SVM*). Za svaku od klasa naučen je klasifikator na principu *jedan–protiv–svih*. Parametar C odabran je k-strukom unakrsnom provjerom (engl. *k-fold cross-validation*) (veći parametar C dozvoljava manju razdvajajuću marginu dok mala vrijednost C zahtjeva da razdvajajuća margina bude veća).

Kao metrika za procjenu kvalitete klasifikatora korištena je prosječna preciznost. Prosječna preciznost predstavlja površinu ispod *preciznost – odziv* krivulje; ona nam govori koliko je klasifikator *siguran* u dobivenu klasifikaciju podataka. U slučaju više različitih klasifikatora koristi se srednja vrijednost prosječne preciznosti svih klasifikatora - mAP (engl. *mean average precision*).

Sve distribucijske šume sačinjene su od jednakog broja slučajnih distribucijskih stabala i EM distribucijskih stabala (npr. šuma koja sadrži 4 stabala sastoji se od 2 slučajna i 2 EM stabla). Za treniranje pojedinog stabla korišteno je oko 8×10^5 SIFT opisnika. Za potrebe mjerenja vremena potrebnog za izračun Fisherovog vektora korišteno je oko 8×10^4 SIFT opisnika. Minimalni kardinalitet skupa u listu postavljen je na 7000. Kod treniranja stabala nije korišteno ograničenje dubine već je korišteno ograničenje broja listova u pojedinom stablu.

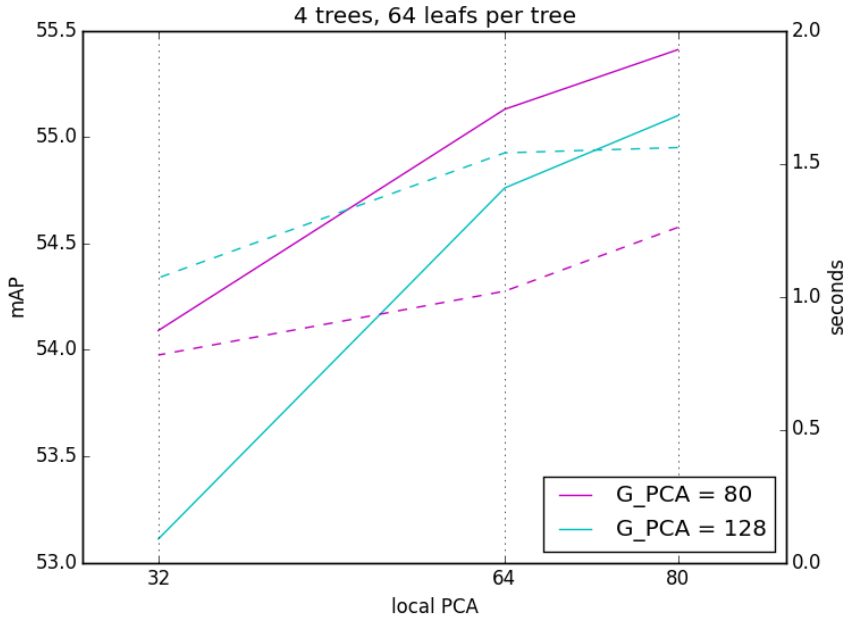
6.1. Utjecaj globalnog i lokalnog PCA

Globalni PCA odnosi se na redukciju dimenzije (ili rotaciju) prostora podataka prije ulaska u distribucijska stabla, dok se lokalni PCA odnosi na redukciju dimenzije u pojedinim listovima. Rezultati testiranja prikazani su na slici 6.1. Puna crta odnosi se na mAP dok se iscrtkana crta odnosi na vrijeme potrebno za izračun jednog Fisherovog vektora. Vidljivo je da se bolji rezultati postižu redukcijom dimenzije ulaznog prostora, a ne samo rotacijom. Moguće objašnjenje tog efekta je u samom načinu izgradnje stabala – odabir dimenzije i podijela skupa podataka temeljem odabira neke vrijednosti odabrane dimenzije. Za očekivati je da će kvalitetnije podjele rezultirati odabirom dimenzija kojima PCA pridaje veće značenje (dimenzije koje nose više informacija). Ovaj efekt najviše dolazi do izražaja u dubljim čvorovima stabala gdje ulazni prostor podataka koji ulaze u čvor, zbog podjela u prethodnim čvorovima, može degenerirati u potprostor.
Redukcija dimenzije početnog ulaznog prostora globalnim PCA također doprinosi i brzini treniranja distribucijskih šuma i izračuna Fisherovih vektora. Rezultati testiranja vidljivi su na slici 6.1.

Dodatno, proveden je eksperiment u kojem je korišten PCA u svakom čvoru stabla (unutarnji čvorovi i listovi). U svakom unutarnjem čvoru smanjena je dimenzionalnost ulaznog prostora za fiksni iznos (u provedenom eksperimentu korištena je redukcija 5 dimenzija po dubini čvora). U listovima je provedena redukcija na prostor dimenzije 64. Vrijeme potrebno za izračun Fisherovog vektora povećalo se na 6.5 sekundi, a mAP je pao na 54.48% naspram 54.76% dobivenih jednakom konfiguracijom šume, ali bez redukcije u unutarnjim čvorima. Zbog povećane računske kompleksnosti ovaj pristup nije dodatno proučavan.

6.2. Utjecaj broja stabala

Utjecaj broja stabala na mAP prikazan je na slici 6.2. Primjetan je znatan porast prosječne preciznosti porastom broja stabala od jedan do četiri. Dodavanjem dvaju dodatnih stabala (sveukupno 6 stabala) ne postiže se značajan porast prosječne preciznosti. Zaključak koji se može izvući iz ovog eksperimenta je da su četiri stabla daju dostatan kapacitet generativnom modelu (distribucijskoj šumi) kako bi (uz intrinzična ograničenja) naučio distribuciju podataka.



Slika 6.1: Utjecaj globalnog i lokalnog PCA na mAP i vrijeme izračuna

6.3. Utjecaj broja listova u stablu

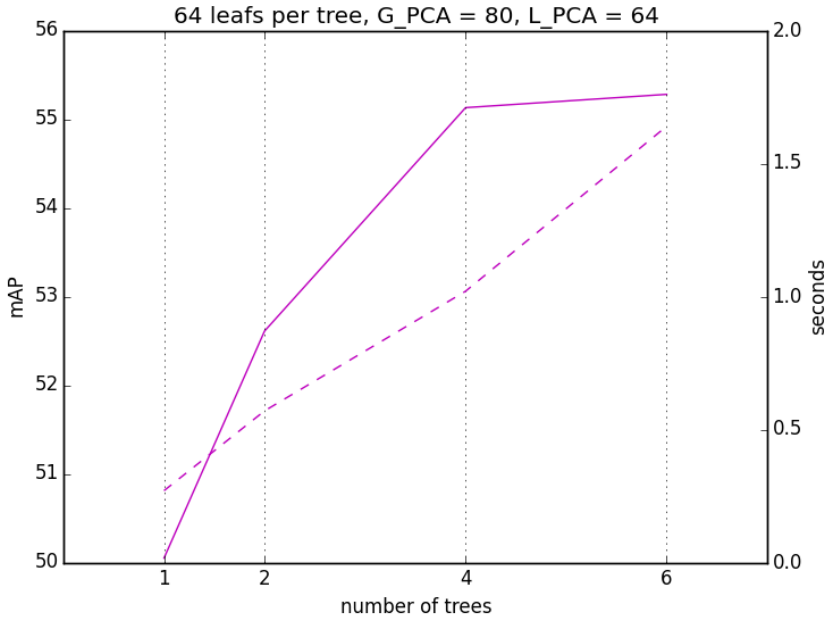
Utjecaj broja listova distribucijskog stabla na mAP prikazan je na slici 6.3. Vidljiva je velika razlika od 3% u korištenju stabla s 32 ili 64 lista. Dodatno povečanje broja listova ne doprinosi značajno povečanju prosječne preciznosti (ali se znatno povećava memorijski otisak pojedinog Fisherovog vektora).

6.4. Utjecaj ostalih parametara

Utjecaj *minimalne informacijske dobiti podjele čvora* nije posebno proučavan. U svim eksperimentima njegova fiksna postavljena vrijednost jednaka je 0.7.

Utjecaj *minimalnog kardinaliteta skupa podataka u listu* (u nastavku *minimalni kardinalitet skupa*) je važan parametar učenja stabala te ga treba prilagoditi s obzirom na broj primjera dostupnih za učenje i željeni broj listova u stablu. Npr. minimalni kardinalitet skupa jednak k implicira da ako želimo imati stablo s l listova, za njegovo treniranje moramo predstaviti barem $k \times l$ primjera za učenje.

Utjecaj *broja dopuštenih podjela* (za slučajna distribucijska stabla) utječe na međusobnu nezavisnost stabala (odjeljak 3.2.4), ali i na kvalitetu odabrane podjele čvora. Vrijednost broja podjela treba biti odabrana s obzirom na dimenziju podataka s kojima će stablo raditi. Trebalo bi omogućiti svakoj dimenziji da bude



Slika 6.2: Utjecaj broja stabala distribucijske šume na mAP i vrijeme izračuna

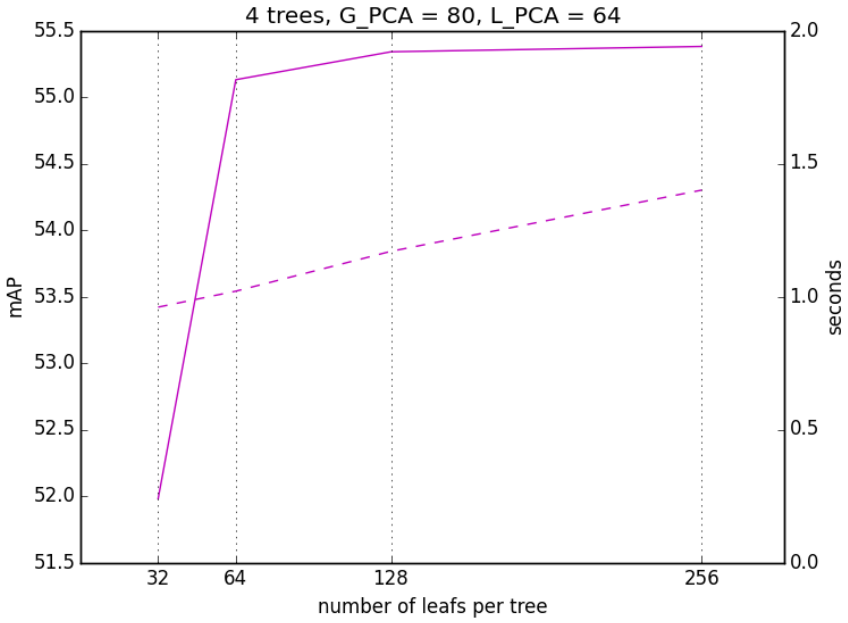
Tablica 6.1: Utjecaj konfiguracije šume na mAP

Konfiguracija šume	**mAP**
4 slučajna dist. stabla	54.21
4 EM dist. stabla	53.76
2 slučajna i 2 EM dist. stabla	55.13

odabrana barem jedanput ili idealno više puta. U svim eksperimentima obično je korištena vrijednost $20 \times d$ koja implicira da (stohastički gledano) se svaka dimenzija razmatra 20 puta.

Utjecaj parametra K (posebno za EM distribucijska stabla) utječe na veličinu skupa najboljih generiranih podjela iz kojeg se odabire konačna podjela čvora. Ovaj parametar služi striktno kako bi se u postupak učenja EM distribucijskog stabla ubacila nedeterminističnost (izbjegavanje sličnih stabala). U eksperimentima vrijednost K postavljena je na 4.

Utjecaj konfiguracije distribucijske šume prikazan je tablicom 6.1. Najbolji rezultat postignut je mješovitom konfiguracijom šume dok se konfiguracija koja koristi samo EM distribucijska stabla pokazala najlošijom.



Slika 6.3: Utjecaj broja listova distribucijskog stabla na mAP i vrijeme izračuna

6.5. Usporeba s Fisherovim vektorima nad Gaussovom mješavinom

U tablici 6.2 prikazana je usporedba korištenja Gaussove mješavine i distribucijske šume kao generativnog modela za izračuna Fisherovih vektora. Kod Gaussove mješavine, ulazni prostor reduciran je na 64 dimenzije s početnih 128 (standardna dimenzija SIFT-a) analiza glavnih komponenata dok je kod distribucijskih šuma ulazni prostor reduciran na 80 dimenzija, a naknadno u listovima na 64 dimenzije. Fisherovi vektori izgrađeni nad oba generativna modela bili su približno jednake dimenzionalnosti (Fisherovi vektori izgrađeni nad Gaussovom mješavinom dodatno sadrže informaciju o težinama pojednih komponenata mješavine, ali se njihov doprinos može zanemariti, [4]).

Iako GMM Fisherovi vektori postižu bolju prosječnu preciznost od RDF Fisherovih vektora za otprilike 3.5%, njihov izračun traje oko 8 puta duže. Važno je napomenuti da u navedeno vrijeme nije uključeno izlučivanje značajki slike, njihova projekcija u potprostor (globalni PCA) te dodatne normalizacije (normalizacija potenciranjem, l_2 normalizacija) nego striktno samo izračun Fisherovog vektora – točno onaj dio u kojem se dva pristupa razlikuju.

Zanimljivo je primjetiti da dvije trećine vremena potrebnog za izračun Fisherovog vektora preko distribucijske šume otpada na postupak podjele podataka po

Tablica 6.2: Usporedba GMM i RDF pristupa

Generativni model	**Broj komponenti**	**mAP**	**vrijeme (sec)**
Gaussova mješavina	256	58.6	8.68
Distribucijska šuma	4 x 64	55.13	1.02

listovima, a samo jedna trećina na računanje Fisherovog vektora.

7. Zaključak

U okviru ovog rada pokazano je da su Fisherovi vektori izgrađeni nad distribucijskim šumama sumjerljivi Fisherovim vektorima izgrađenim nad Gaussovom mješavinom (za problem klasifikacije slika). Iako je korištenjem Gaussove mješavine moguće postići nešto bolje rezultate klasifikacije nego distribucijskim šumama (oko 3% mAP razlike), izračun Fisherovog vektora preko distribucijske šume znatno je brži (oko 8 puta).

Kombinacija slučajnih distribucijskih stabala i EM distribucijskih stabala u distribucijskoj šumi pokazala se boljim nego korištenje samo jedne vrste stabala.

U okviru budućeg rada postoji mogućnost poboljšanja više dijelova cjelokupnog cjevovoda. Kako bi se ubrzao postupak raspoređivanja podataka po listovima stabala (usko grlo izračuna Fisherovog vektora), implementaciju stabala i šuma trebalo bi ostvariti u programskom jeziku C (ili C++).
Također, otvorena je i mogućnost drugačijeg odabira dimenzija (i njima pridruženih vrijednosti) za podjelu podataka tijekom procesa učenja stabala (osim korištenog slučajnog odabira i odabira algoritmom maksimizacije očekivanja).
Za poboljšanje rezultate klasifikacije zanimljivo bi bilo isprobati klasifikacijske modele s posebnom vrstom l_1 regularizacije – umjesto da se pojedinačni parametri klasifikatora pritežu na nulu, ovdje bi se cijeli listovi pritezali na nulu (engl. *block–sparse*). Tako bi se eliminirao utjecaj nekih listova što bi osim potencijalno bolje preciznosti klasifikacije rezultiralo i bržim izračunom Fisherovog vektora (prilagodbom distribucijskih šuma).

Literatura

[1] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 1470–1477, IEEE, 2003.

[2] D. G. Lowe, "Object recognition from local scale-invariant features," *International Conference on Computer Vision, 1999*, pp. 1150–1157, 1999.

[3] J. Krapac, J. Verbeek, and F. Jurie, "Spatial fisher vectors for image categorization," 2011.

[4] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *International journal of computer vision*, vol. 105, no. 3, pp. 222–245, 2013.

[5] A. Criminisi, J. Shotton, and E. Konukoglu, "Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning," *Foundations and Trends® in Computer Graphics and Vision*, no. 7, pp. 81–227, 2011.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results." http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html, 2007.

[7] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," 2008.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[9] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[10] J. M. Phillips and S. Venkatasubramanian, "A gentle introduction to the kernel distance," *arXiv preprint arXiv:1103.1625*, 2011.

[11] T. Jaakkola, D. Haussler, *et al.*, "Exploiting generative models in discriminative classifiers," *Advances in neural information processing systems*, pp. 487–493, 1999.

[12] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1–8, IEEE, 2007.

[13] C. Baecchi, F. Turchini, L. Seidenari, A. D. Bagdanov, and A. D. Bimbo, "Fisher vectors over random density forests for object recognition," in *22nd International Conference on Pattern Recognition, ICPR 2014, Stockholm, Sweden, August 24-28, 2014*, pp. 4328–4333, 2014.

Klasificiranje slika Fisherovim vektorima izgrađenim nad slučajnim distribucijskim šumama značajki slike

Sažetak

U okviru ovog rada proučena je i implementirana metoda korištenja Fisherovih vektora preko distribucijskih šuma za klasifikaciju slika. Motivacija korištenja distribucijskih šuma kao generativnog modela umjesto uvriježene Gaussove mješavine je ubrzanje samog postupka izračuna Fisherovih vektora.
Postignuto je ubrzanje izračuna od oko 8 puta naspram korištenja Gaussove mješavine, uz gubitak prosječne srednje preciznosti (mAP) klasifikacije od oko 3%. Također je i proučen utjecaj globalne i lokalne analize glavnih komponenata, broja stabala u šumi kao i broj listova u stablu na točnost klasifikacije. Dodatno, predstavljen je novi način treniranja distribucijskih stabala; *EM distribucijska stabla.*

Ključne riječi: Fisherovi vektori, distribucijska stabla, distribucijske šume, generativni modeli, klasifikacija slika

Image classification with Fisher vectors over random density forests

Abstract

In this work, usage of Fisher vectors over random density forests for image classification is studied and implemented. Usage of random density forests as generative models instead of Gaussian mixture model was motivated by the desire to reduce computation time of a single Fisher vector.
Computation time was reduced approximately 8 times versus Gaussian mixture usage, while loss of around 3% mean average precision (mAP) is reported. Effect of global and local PCA, number of trees per forest and number of leaves per tree on classification precision is also studied. Additionally, a novel approach to density tree training is introduced; *EM density trees*.

Keywords: Fisher vector, density trees, density forests, generative models, image classification