SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Igor Bonači
Ivan Kovaček
Ivan Kusalić

DETEKCIJA I RASPOZNAVANJE PROMETNIH ZNAKOVA U VIDEO SNIMCI

Zagreb, 2010

Ovaj rad izrađen je u Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave (ZEMRIS), pod vodstvom prof. dr. sc. Zorana Kalafatića i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2009./2010.

SADRŽAJ

1 Uvod

Posljednjih nekoliko desetljeća, računala postaju sve prisutnija u ljudskoj svakodnevici. Postoji mnoštvo problema koje računala rješavaju puno efikasnije od ljudi i stoga su mnogi poslovi nezamislivi bez računalne potpore. Unatoč izrazitoj moći obrade podataka, računalima se i dalje ne mogu efikasno rješavati neki problemi koje ljudi rješavaju svakodnevno i bez velikog napora. Umjetna inteligencija, kao grana računalne znanosti, izučava takve probleme i traži rješenja istih. Računalni vid je grana umjetne inteligencije koja razmatra probleme obrade slike i videa, te izdvajanja korisnih informacija iz njih. Danas ne postoji općeniti postupak koji rješava problem detekcije objekta u slici. Svakom konkretnom problemu detekcije objekta, poput detekcije lica ili prometnog znaka, pristupa se pojedinačno, uzimajući u obzir posebna svojstva objekta.

U ovom se radu rješava problem detekcije i raspoznavanja prometnih znakova u video snimci. Osnovni cilj je za danu sliku locirati poziciju na kojoj se nalazi prometni znak, te potom odrediti o kojem se znaku radi. Ovaj je problem veoma zanimljiv te bi njegovo rješenje imalo široku praktičnu primjenu: kao pomoć u održavanju prometnica, kao pomoć vozačima tijekom vožnje, kao sigurnosni sustav u vozilima…

U sklopu rada izgrađen je sustav temeljen na algoritmu Viole i Jonesa, stroju s potpornim vektorima te umjetnim neuronskim mrežama. Detekcija znaka u slici vrši se algoritmom Viole i Jonesa, dok neuronska mreža i stroj s potpornim vektorima prvenstveno služe za raspoznavanje znaka, to jest određivanje njegove klase. Na temelju usporedbe rezultata stroj s potpornim vektorima je odabran kao primarni klasifikator, a neuronska mreža kao sekundarni. U konačnoj izvedbi dodatnom neuronskom mrežom se obavlja detekcija nad rezultatima algoritma Viole i Jonesa, u svrhu smanjenja razine lažnih detekcija. Kroz razvoj sustava provedena su brojna istraživanja u svrhu identificiranja problema i rješavanja istih. Neka od tih istraživanja iznesena su u radu, primjerice odabir ulaznih značajki, modeliranje skupa za učenje, korištenje neuronske mreže kao dodatne kaskade algoritmu Viole i Jonesa... Na temelju istraživanja sustavno su poboljšavani rezultati, te su naposljetku postignute odlične performanse.

Nakon iscrpnog proučavanja problema, predlažemo rješenje koje daje veoma dobre rezultate. Predloženi postupak je primijenjen na prometne znakove upozorenja, no primjenjiv je i na druge vrste prometnih znakova. Korišteni postupci, njihova teorijska osnova i dobiveni rezultati slijede u nastavku.

2 Ciljevi

Osnovni cilj u ovome radu je automatizirana identifikacija prometnih znakova kroz video snimku. Rad je ograničen na prometne znakove upozorenja, koji su trokutastog oblika. Za potrebe detekcije i raspoznavanja znakova drugih oblika, primjerice okruglih znakova, također je moguće, uz manje promjene, primijeniti opisani postupak.

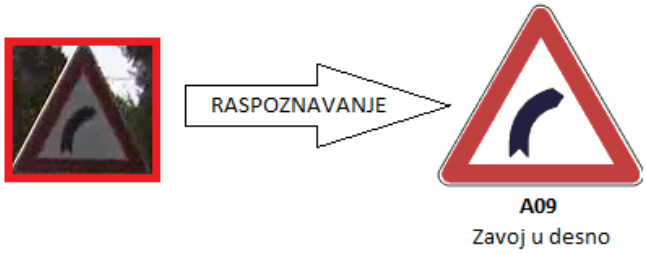
Obrada video snimke svodi se na obradu slika koje čine istu, te se problem detekcije i raspoznavanja prometnih znakova sastoji od dva osnovna potproblema:

- Detekcija i lociranje prometnog znaka u slici
- Raspoznavanje detektiranog znaka

Problem detekcije znaka u slici svodi se na pretraživanje svih mogućih dijelova slike, te utvrđivanja za svaki pojedini dio predstavlja li on prometni znak ili ne.

Slika 2.1. Detekcija prometnog znaka

Slika 2.1 prikazuje primjer željenog rezultata detekcije. Izlaz procesa detekcije čini opis položaja detektiranog znaka unutar slike, te predstavlja ulaz za proces raspoznavanja. Detektirani položaj prometnog znaka je na slici prikazan crvenim kvadratom koji uokviruje detektirani znak.



Slika 2.2 Raspoznavanje prometnih znakova

Na slici (Slika 2.2) je prikazan rezultat postupka raspoznavanja. Raspoznavanjem se za dani isječak slike utvrđuje koji prometni znak taj isječak predstavlja.

Konačan rezultat detekcije i raspoznavanja kazuje da li se u zadanoj slici nalazi znak (ili više njih), kojeg je tipa taj znak te njegovu poziciju u slici.

Na osnovu podataka o detekciji i raspoznavanju znakova u slikama, naposljetku je potrebno identificirati znakove na razini video snimke. Rezultat konačne analize govori u kojim se vremenskim periodima kroz video snimku nalaze znakovi, te kojeg su oni tipa.

U ovome radu se za potrebe detekcije koriste algoritam Viole i Jonesa, te umjetne neuronske mreže, dok se problem raspoznavanja rješava pomoću umjetnih neuronskih mreža i stroja s potpornim vektorima.

U idućim poglavljima detaljno su analizirani principi rada triju navedenih algoritama, dok je u kasnijim poglavljima opisan koncept izvedenog sustava te su prezentirani dobiveni rezultati.

3 Značajke slike

Da bi se iz slike izdvojila korisna informacija, potrebno je odrediti način izlučivanja značajki iz slike. Također, prije izlučivanja značajki potrebno je obraditi sliku da bi se ispravile eventualne smetnje, ili postigli željeni efekti. Ovo poglavlje obrađuje dva navedena problema: obrada slike, te izlučivanje značajki.

3.1 Obrada slike

3.1.1 Ispravljanje smetnji uzrokovanih preplitanjem slike

[1]Slike nad kojima se vrši klasifikacija su dijelovi video snimke. Kamera pri snimanju koristi preplitanje *(eng. interlacing)*: prvo snimi svaki drugi redak slike, a zatim preostale retke. Kao rezultat toga, javljaju se smetnje na slici pri brzim pokretima kamere. Slika 3.1 prikazuje primjere slika sa smetnjom pri utjecaju preplitanja.

Slika 3.1 Preplitanje slike

U skupu slika postoji okvirno 10% slika sa smetnjom prouzročenom preplitanjem, te bi adekvatna obrada takvih slika poboljšala rezultate klasifikacije. Jednostavna metoda s dobrim rezultatima je odbacivanje polovice linija, te interpolacija linija koje nedostaju.

Slika 3.2 Slike ispravljene odbacivanjem linija

[1] Dijelovi poglavlja su preuzeti iz I. Kovaček: Sustav za detekciju i raspoznavanje prometnih znakova

Slika 3.2 prikazuje slike s prijašnje slike (Slika 3.1) ispravljene metodom odbacivanja linija. Slike su vidno bolje nakon obrade, ali ipak nisu kvalitetne.

Problem u odbacivanju linija je gubitak informacije o slici, te kada bi se provodila obrada slika u tome obliku, informacija bi se izgubila i na kvalitetnim slikama.

Očuvanje informacije je motivacija za sljedeći pristup ispravljanju smetnji uzrokovanih preplitanjem. Promatrana smetnja se očituje u pomaknutim horizontalnim linijama, te bi aproksimacija inverza smetnje bio pomak tih linija natrag. Problem je otkrivanje najboljeg pomaka. U frekvencijskom spektru ispravne slike u pravilu se ističu niske frekvencije, dok su visoke relativno male. Promotrimo li sliku sa smetnjom uzrokovanom preplitanjem, zamijetit ćemo da su istaknutije vertikalne frekvencije slike. Takva priroda smetnje omogućuje odabir udjela visokih vertikalnih frekvencija spektra slika za mjeru kvalitete.

Dvodimenzionalna diskretna kosinusna transformacija (3.1) je oblik diskretne Fourierove transformacije:

$$X_{k1,k2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x_{n_1,n_2} \cos\left[\frac{\pi}{N_1}\left(n_1 + \frac{1}{2}\right)k_1\right] \cos\left[\frac{\pi}{N_2}\left(n_2 + \frac{1}{2}\right)k_2\right] \qquad (3.1)$$

Pojašnjenje:

$X_{k1,k2}$ - vrijednost spektra za prostorne frekvencije k1 i k2

N_1 , N_2 - dimenzije slike (prostorna domena)

x_{n_1,n_2} - vrijednost slike na poziciji n_1 , n_2

Osnovna svojstva 2D DCT transformacije povoljna su za izdvajanje vertikalnih frekvencija slike. Veličina spektra jednaka je veličini slike, te je spektar razapet dimenzijama horizontalne i vertikalne frekvencije. Tako se u 0-tom stupcu i n-tom retku nalazi faktor visoke vertikalne prostorne frekvencije slike.

Odabir mjere kvalitete slike Q svodi se na odabir funkcije visokog vertikalnog spektra slike. Odabrana mjera je negativna suma 0.25% spektra, za frekvencije najbliže najvećoj vertikalnoj frekvenciji:

$$Q = -\sum_{k_1=0}^{H}\sum_{k_2=V}^{N_2-1} X_{k_1,k_2} \quad , \begin{matrix} H = \lceil 0.05 \cdot N_1 \rceil \\ V = \lceil 0.95 \cdot N_2 \rceil \end{matrix} \tag{3.2}$$

Nakon što je odabrana mjera kvalitete slike, možemo pristupiti procesu ispravljanja slike metodom pomaka horizontalnih linija ili odbacivanjem parnih (ili neparnih) linija.

Osnovna ideja je pomakom parnih (ili neparnih) horizontalnih linija za različite iznose, odrediti pomak za koji je slika imala najveću mjeru kvalitete Q. Diskretna kosinusna transformacija se računa nad luminantnom komponentom slike. Promotrimo ispravljanje slike na kojoj nema pojave preplitanja.

Slika 3.3 Ispravljanje preplitanja pomakom

Slika 3.3. prikazuje primjer ispravljanja preplitanja metodom pomaka. Veličina d je pri tome horizontalni pomak linija, dok je Q mjera kvalitete predložena izrazom (3.2). Očito je da je početna slika, s pomakom 0, najkvalitetnija od ponuđenih, a za tu sliku je i mjera kvalitete Q najveća. To odražava bitno svojstvo ovog pristupa ispravljanju smetnje: očuvanje informacije i kvalitete slika bez smetnje uzrokovane preplitanjem. Pokazalo se da se ovom metodom ispravne slike ne mijenjaju.

Na slici (Slika 3.4) nalazi se usporedba dvaju metoda popravljanja. Metoda ispravljanja pomakom popravlja sliku, ali se ipak pokazuje lošijom od metode ispravljanja odbacivanjem linija. Uzrok tome je priroda smetnje. Pri preplitanju, podaci na zakašnjelim linijama nisu samo horizontalno pomaknuti od željenih podataka, nego su pomaknuti ovisno o smjeru gibanja kamere.

Slika 3.4 Usporedba metoda ispravljanja preplitanja

Ispravljanje pomakom može poslužiti za detekciju preplitanja: ako je najbolji pomak različit od 0, slika sadrži smetnje uzrokovane preplitanjem.

Kombinacija dvaju metoda daje najbolje rezultate. Ispravljanje pomakom služi kao detektor preplitanja. Ako je detektirana smetnja, slika se ispravlja metodom odbacivanja linija.

3.2 Sustavi boja

Boje su važne značajke slike, te je potrebno dobro odabrati reprezentaciju boja pri rješavanju problema klasifikacije.

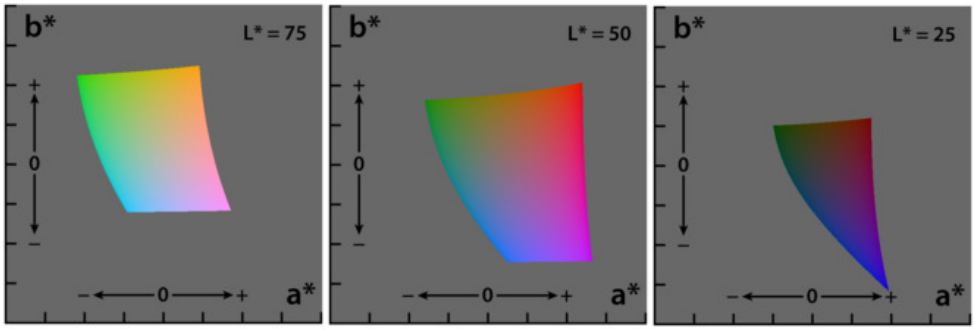
Osnovni zapis slike je u RGB *(eng. red, green, blue)* formatu. Slika je razložena na tri komponente: crvenu, plavu i zelenu, za svaki slikovni element. U praksi se pokazalo da su podaci u ovakvome zapisu često loše značajke za predstavljanje uzoraka.

Često se koristi samo luminantna komponenta koja odgovara komponenti Y iz YUV sustava boja. U i V su krominantne komponente. Formula (3.3) je izraz za izračun luminantne komponente iz komponenti RGB sustava. Slika 3.5 prikazuje odnos slike i njene luminantne komponente.

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \qquad (3.3)$$

Slika 3.5 Luminantna komponenta slike

Lab sustav boja dizajniran je da bi aproksimirao ljudsko viđenje boja, to jest komponente su mu perceptualno uniformne u odnosu na ljudski vid. L komponenta predstavlja luminantnu komponentu, dok a i b komponente razapinju nijanse boje. Slika 3.6 prikazuje dinamiku Lab sustava. Kao što se može očitati sa grafova, u Lab sustavu su definirane i boje koje nisu vidljive ljudskom oku.



Slika 3.6. Lab sustav boja [16]

3.3 Gradijent slike

Rubovi slike sadrže bitnu informaciju o obliku. Gradijent luminantne komponente relativno dobro izdvaja rubove slike, pošto je gradijent smjer najbržeg rasta funkcije.

Gradijent funkcije dvije varijable:

$$\nabla f(x,y) = \frac{\partial f(x,y)}{\partial x}\vec{i} + \frac{\partial f(x,y)}{\partial y}\vec{j} \tag{3.4}$$

Pošto je slika diskretna, koristimo aproksimacije derivacija po x i y:

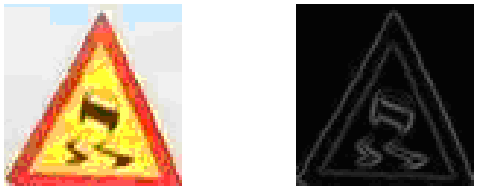
$$dx = \frac{\partial f(x,y)}{\partial x} = \frac{f(x+1,y) - f(x-1,y)}{2} \tag{3.5}$$

$$dy = \frac{\partial f(x,y)}{\partial y} = \frac{f(x,y+1) - f(x,y-1)}{2} \tag{3.6}$$

Pomoću izraza (3.5) i (3.6) gradijent se aproksimira na temelju vrijednosti horizontalnih i vertikalnih susjeda slikovnog elementa. Prikaz gradijenta pomoću kuta i iznosa često je prikladniji:

$$r = \sqrt{dx^2 + dy^2} \ , \ \phi = arctg(\frac{dy}{dx}) \tag{3.7}$$

Slika 3.7 prikazuje iznos gradijenta luminantne komponente slike. Može se zamijetiti sposobnost izdvajanja rubova.



Slika 3.7 Gradijent luminantne komponente slike

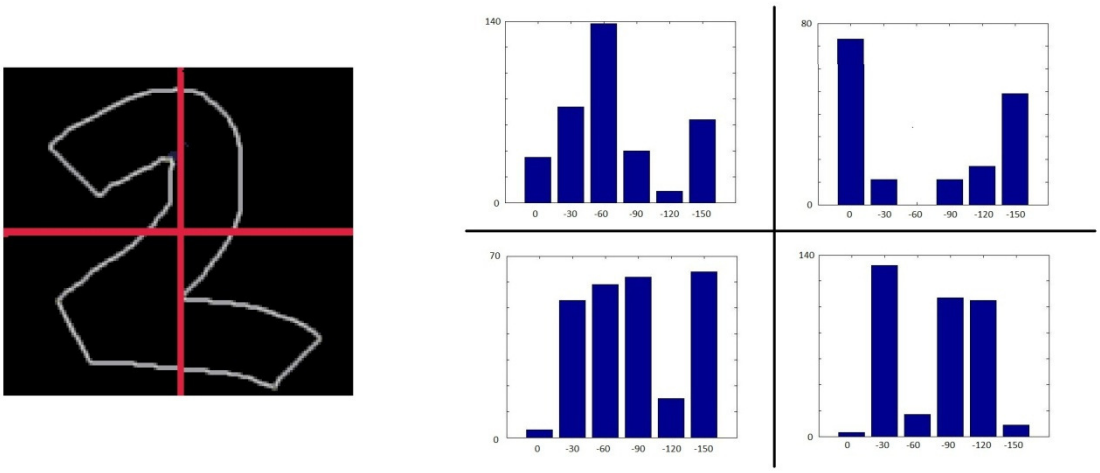
Precizniji gradijent, koji uzima u obzir i dijagonalne vrijednosti, može se postići filtriranjem (konvolucijom) Schaarovim operatorima, koji su cjelobrojne aproksimacije:

$$G_y = \begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix} \qquad G_x = \begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix} \tag{3.8}$$

3.4 Histogram orijentacije gradijenta

Upotreba histograma je jednostavan način izlučivanja značajki nad dijelovima slike. Adekvatnim odabirom veličine i položaja segmenata slika nad kojima se rade histogrami značajki, postiže se određena otpornost na pomake slike.

Posebno zanimljiv je histogram orijentacije gradijenata. Nakon odabira segmenata slike nad kojima se rade histogrami, za svaki slikovni element u segmentu promatra se kut gradijenta. Histogram kutova gradijenata daje opis orijentacije rubova za dotični segment slike, te time daje informaciju o obliku koji se nalazi u segmentu.



Slika 3.8 Histogram orijentacije gradijenata

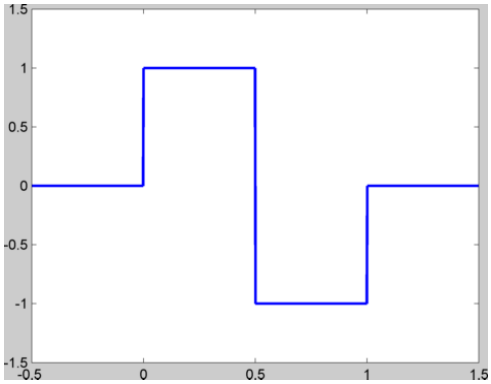
Slika 3.8 prikazuje gradijent slike podijeljen na segmente, te histograme smjera gradijenata. Vrijednosti kuteva linearno su raspoređene od 0 do -150 stupnjeva. Histogrami vjerno prikazuju mogućnost izdvajanja informacije o smjeru rubova slike.

U navedenom primjeru histogram se temelji na smjeru, to jest, na rasponu od 180 stupnjeva. Detaljniju informaciju o rubovima može se dobiti proširenjem histograma na orijentaciju, to jest na 360 stupnjeva, te proširenjem na dvodimenzionalni histogram dodavajući dimenziju iznosa gradijenta.

Uz histogram orijentacije gradijenta često se koriste i histogrami bazirani na značajkama iz različitih sustava boja. Takve značajke su otporne na rotaciju i translaciju objekta.

3.5 Haarove značajke

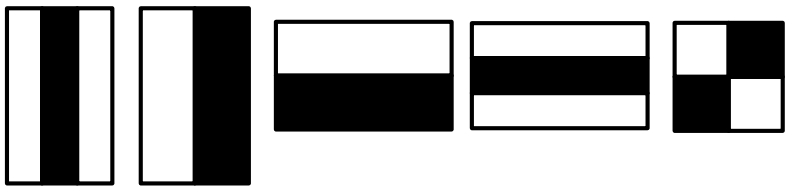
U matematici su poznati Haarovi valići (*eng. Haar wavelet*) koji su prvi poznati prebrojiv slijed ortonormalnih funkcija koje su baza u Lebesqueovom prostoru kvadratno integrabilnih funkcija[19]. Slika 3.9 prikazuje osnovni Haarov wavelet.



Slika 3.9 Prikaz Haarovog waveleta

Svaka kontinuirana funkcija može se aproksimirati linearnom kombinacijom $\phi(t), \phi(2t) \dots, \phi(2^k t), \dots$ i njihovih pomaka. Također je poznata i diskretna Haarova wavelet transformacija koja se danas često koristi u kompresiji slike, primjerice u JPEG 2000 standardu. Diskretna Haarova transformacija koristi ortonormalni skup Haarovih značajki. Prednosti korištenja Haarove transformacije u odnosu na DCT[2] transformaciju je brzina kompresije te bolja kompresija naglih promjena odnosno visokih frekvencija u slici koje nastaju kao rezultat rubova objekta.

Po uzoru na Haarove valiće osmišljene su Haarove značajke. Definiraju se kao razlika sume slikovnih elemenata unutar određenih pravokutnih područja. Slika 3.10 prikazuje osnovne oblike Haarovih značajki. Crne regije slike označavaju da se slikovni elementi u njima zbrajaju, a bijele regije da se slikovni elementi oduzimaju od ukupne vrijednosti.



Slika 3.10 Prikaz Haarovih značajki

Primjena valića u kompresiji koristi ortonormalnu bazu u Haarovom prostoru značajki. Nasuprot tome u metodama računalnog vida koristi se potpun skup svih Haarovih značajki na određenom području. Taj skup ne čini bazu jer nije linearno nezavisan, te se primjerice za regiju veličine $24x24$ slikovna elementa sastoji od čak $45\,396$ elemenata, dok je dimenzija tog vektorskog prostora jednaka 576.

Ukoliko je izvorna slika siva tada se generiraju značajke samo na tom jednom kanalu slike. Ukoliko postoji veći broj kanala slike (primjerice u RGB, HSV ili LAB sustavu boja) tada je moguće izračunavati Haarove značajke na svakom od kanala nezavisno [23].

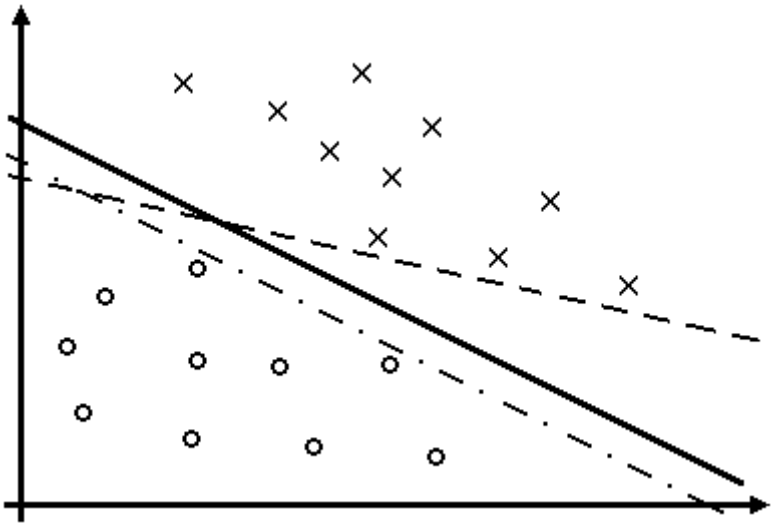
[2] DCT – diskretna kosinusna transformacija (discrete cosinus transform)

4 Stroj s potpornim vektorima

[3]Stroj s potpornim vektorima (eng. *Support Vector Machine*, [1]) će biti objašnjen postepeno. Prvo će biti iznesene dvije općenite intuicije o klasifikaciji, nakon toga slijedi objašnjenje klasifikatora optimalne granice (eng. *Optimal Margin Classifier*, [2]), te će konačno klasifikator optimalne granice biti modificiran tako da nastane stroj s potpornim vektorima.

4.1 Intuicija o najboljoj granici odluke

Promatra se binarna klasifikacija s grupama „1“ i „-1“. Svaki objekt se za potrebe klasifikacije predstavlja kao vektor značajki (eng. *Feature vector*). Vektor značajki $x \in \mathbb{R}^n$ sadrži n realnih brojeva koji predstavljaju pojedine osobine objekta (npr. cijena objekta, veličina, površina koju zauzima, težina i slično). Budući da je $x \in \mathbb{R}^n$, vektor značajki se može promatrati kao točka u n-dimenzionalnom prostoru. Pretpostavlja se da su grupe „1“ i „-1“ linearno odvojive u n-dimenzionalnom prostoru. To znači da postoji hiperravnina koja dijeli n-dimenzionalni prostor na dva dijela tako da se svi elementi (točke) grupe „1“ nalaze s jedne strane hiperravnine, a svi elementi (točke) grupe „-1“ nalaze s druge strane hiperravnine. U slučaju $n = 2$ dobivamo 2D prostor, a hiperravnina postaje pravac.



Slika 4.1. Linearno odvojive grupe u 2D prostoru

Na slici 4.1 su prikazane dvije linearno odvojive grupe (grupa „1“ je označena kružićem, a grupa „-1“ iksićem). Hiperravnina koja odvaja grupe je pravac. Na slici 4.1 su prikazana tri pravca koja uspješno odvajaju grupe. Od tri prikazana odvajajuća pravca najbolji je pravac prikazan punom linijom. Taj je pravac najudaljeniji od obje grupe, te time smanjuje mogućnost pogrešne klasifikacije u rubnom području

[3] Dijelovi poglavlja su preuzeti iz I. Kusalić: Raspoznavanje prometnih znakova metodom potpornih vektora

grupa, uz sam pravac. To je druga intuicija: najbolji odvajajući pravac dvije linearno odvojive grupe je onaj koji je najudaljeniji od obje grupe.

4.2 Klasifikator optimalne granice

Stroj s potpornim vektorima (eng. *Support Vector Machine*, SVM) se temelji na starijem klasifikatoru optimalne granice (eng. *Optimal Margin Classifier*, poznat i pod nazivom *Maximum Margin Classifier*), te će prije samog SVM-a biti objašnjen ovaj algoritam.

4.2.1 Korištena notacija

Da bi bilo moguće matematički postaviti temelje algoritma potrebno je uvesti određenu notaciju.

Neka je $y \in \{-1, +1\}$ oznaka pojedine grupe i neka je definirana funkcija

$$g(z) = \begin{cases} 1, \text{ ako z} \geq 0 \\ -1, \text{ ako je z<0} \end{cases}$$

Nadalje, neka je $h(x)$ hipoteza koja vraća oznaku grupe (+1 ili -1). Hipoteza $h(x)$ je parametrizirana s parametrima w i b. Pri tome je $x \in \mathbb{R}^n$ vektor značajki, a $w \in \mathbb{R}^n$ i $b \in \mathbb{R}$ su parametri hipoteze.

Vrijedi: $$h_{w,b}(x) = g(w^T x + b)$$

Skup za učenje je skup primjera za učenje koji su predstavljeni kao uređeni parovi $(x^{(i)}, y^{(i)})$ gdje je $x^{(i)} \in \mathbb{R}^n$ vektor značajki i-tog objekta, a $y^{(i)}$ je oznaka grupe kojoj pripada i-ti objekt.

4.2.2 Funkcijska i geometrijska margina

Par parametara (w, b) definira klasifikator, tako što definira određenu razdvajajuću hiperravninu koja razdvaja grupe objekata.

Funkcijska margina

Definicija : Funkcijska margina hiperravnine (w, b) *s obzirom na neki određeni primjer za učenje* $(x^{(i)}, y^{(i)})$ *je* $\hat{\gamma}^{(i)} = y^{(i)}(w^T x^{(i)} + b)$.

Ovako definirana funkcijska margina zapravo matematički iznosi intuiciju o pouzdanosti klasifikacije. Naime, ako je $y^{(i)} = 1$, da bi funkcijska margina bila što veća, mora vrijediti $w^T x^{(i)} \gg 0$, odnosno ako je $y^{(i)} = -1$, mora vrijediti $w^T x^{(i)} \ll 0$. Što je veća funkcijska margina, to je veća i

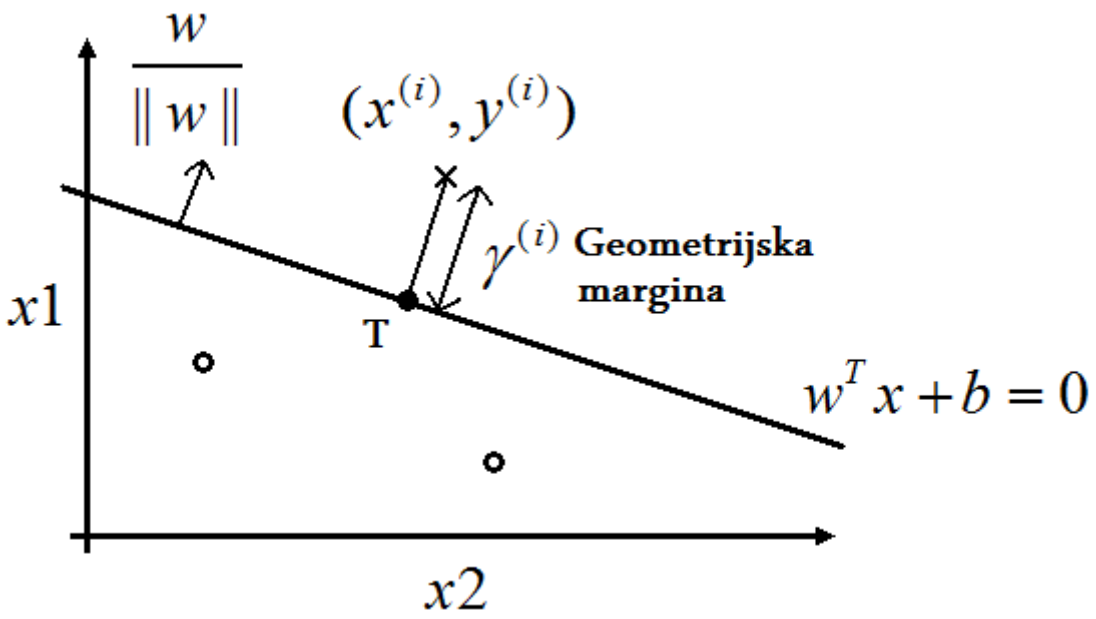
pouzdanost klasifikacije. Vrijedi da je i-ti primjer za učenje $(x^{(i)}, y^{(i)})$ ispravno klasificiran vrijedi li $\hat{\gamma}^{(i)} > 0$, što slijedi direktno iz same definicije.

Definicija: Funkcijska margina hiperravnine (w,b) *s obzirom na cijeli skup za učenje je* $\hat{\gamma} = \min_i \hat{\gamma}^{(i)}$.

Odnosno, funkcijska margina s obzirom na cijeli skup za učenje je najlošiji slučaj funkcijske margine s obzirom na neki određeni primjer za učenje iz danog skupa za učenje.
Rečeno je kako je poželjno imati što veću funkcijsku marginu. Problem s tim zahtjevom je taj da je funkcijsku marginu moguće povećati za željeni faktor tako da se za taj faktor skaliraju parametri w i b (npr. $(w \to 2w \wedge b \to 2b) \Rightarrow \hat{\gamma} \to 2\hat{\gamma}$). Ovaj problem je moguće riješiti dodavanjem normalizacijskog uvjeta, npr. ako se zahtjeva da vrijedi $\| w \| = 1$.

Geometrijska margina

Definicija: Geometrijska margina $\gamma^{(i)}$ *hiperravnine* (w,b) *s obzirom na neki određeni primjer za učenje* $(x^{(i)}, y^{(i)})$ *je geometrijska udaljenost točke koja predstavlja taj primjer za učenje od hiperravnine* (w,b)
.



Slika 4.2. Geometrijska margina u 2D

Na slici 4.2 je prikazana geometrijska margina $\gamma^{(i)}$ u 2D prostoru određenom s x1 i x2 osima. Hiperravnina je zadana s $w^T x + b = 0$, pa je normala na tu hiperravninu dana s $\vec{n} = \frac{w}{\| w \|}$. Točka T je

dana s $x^{(i)} - \gamma^{(i)} \frac{w}{\| w \|}$. Kako točka T leži na hiperravnini, mora zadovoljavati jednadžbu hiperravnine, pa vrijedi:

$$w^T \left(x^{(i)} - \gamma^{(i)} \frac{w}{\| w \|} \right) + b = 0$$

$$w^T x^{(i)} + b = \gamma^{(i)} \frac{w^T w}{\| w \|} = \gamma^{(i)} \| w \|$$

$$\gamma^{(i)} = \left(\frac{w}{\| w \|} \right)^T x^{(i)} + \frac{b}{\| w \|}$$

čime je izražena udaljenost primjera za učenje $(x^{(i)}, y^{(i)})$ od hiperravnine zadane s (w, b).

Uzme li se u obzir mogućnost da primjer za učenje nije ispravno klasificiran, geometrijska je margina dana s: $\gamma^{(i)} = y^{(i)} \left[\left(\frac{w}{\| w \|} \right)^T x^{(i)} + \frac{b}{\| w \|} \right]$.

Dobiveni izraz za geometrijsku marginu identičan je izrazu za funkcijsku marginu, samo je još i normaliziran s $\| w \|$ (tj. $\gamma^{(i)} = \frac{\hat{\gamma}^{(i)}}{\| w \|}$). Cilj je imati što veću geometrijsku marginu. Odnosno, ako je objekt ispravno klasificiran, bolje je da je što dalje od odvajajuće hiperravnine.

Definicija: Geometrijska margina hiperravnine (w, b) *s obzirom na cijeli skup za učenje je* $\gamma = \min_i \gamma^{(i)}$

4.2.3 Formulacija optimizacijskog problema

Klasifikator optimalne granice je algoritam koji bira parametre w i b tako da maksimizira geometrijsku marginu.

Algoritam postavlja sljedeći optimizacijski problem:

$\max_{\gamma, w, b} \gamma$, pod uvjetom da vrijedi: $y^{(i)}(w^T x^{(i)} + b) \geq \gamma \wedge \| w \| = 1$

Budući da se geometrijska margina ne mijenja u ovisnosti o $\| w \|$, moguće je postaviti $\| w \|$ na proizvoljnu vrijednost. U ovoj formulaciji optimizacijskog problema zahtjeva se $\| w \| = 1$ što izjednačava geometrijsku marginu s funkcijskom marginom.

Problem trenutne formulacije optimizacijskog problema je što $\| w \| = 1$ nije konveksno ograničenje (zahtjeva da w leži na jediničnoj hiperkugli).

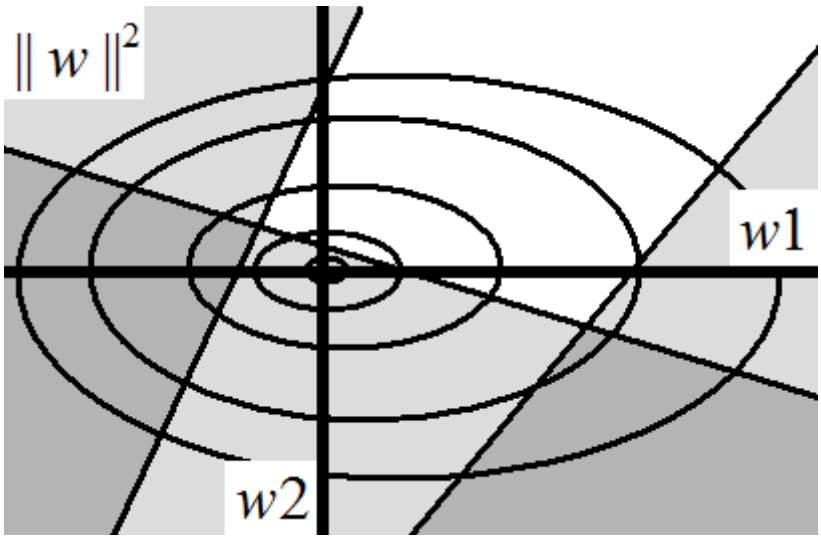
Poželjno je naći formulaciju optimizacijskog problema koja ima samo konveksna ograničenja, jer tada će postojati samo globalni optimum, pa će ispravno raditi bilo koji algoritam lokalne pretrage, poput gradijentnog spusta.

Optimizacijski problem moguće je reformulirati na sljedeći način:

$\min_{w,b} \| w \|^2$, pod uvjetom da vrijedi: $y^{(i)}(w^T x^{(i)} + b) \geq 1$

U konačnoj formulaciji je funkcija cilja kvadratna funkcija (dakle konveksna), a ograničenja su linearna ograničenja na parametre koja izbacuju poluprostore kao nedozvoljene.

Slika 4.3 prikazuje funkciju cilja i ograničenja u konačnoj formulaciji optimizacijskog problema.



Slika 4.3. Funkcija cilja i ograničenja u konačnoj formulaciji optimizacijskog problema

Dobivenu formulaciju problema moguće je riješiti modifikacijom gradijentnog spusta, no još je bolje ubaciti ovu formulaciju u *Quadratic programming (QP) Solver* i time je riješen problem klasifikatora optimalne granice.

4.2.4 Primal i Dual optimizacijski problem

Ovdje je moguće stati, jer je problem zapravo riješen. Ipak, ovaj optimizacijski problem ima određena svojstva zbog kojih je moguće izvesti efikasnije rješenje koje će kasnije biti modificirano tako da nastane stroj s potpornim vektorima.

Lagrangeovi multiplikatori

Pomoću metode Lagrangeovih multiplikatora moguće je riješiti problem minimizacije $\min_w f(w)$, pod uvjetom da je zadovoljen skup ograničenja: $h_i(w) = 0$, $i = 1,...,l$.

Konstruira se Lagrangian: $L(w,\beta) = f(w) + \sum_{i=1}^{l} \beta_i h_i(w)$, pri čemu su parametri β_i Lagrangeovi multiplikatori.

Problem se rješava tako da se $\frac{\partial L}{\partial w}$ postavi na 0, i da se $\frac{\partial L}{\partial \beta}$ postavi na 0. Da bi w^* bilo rješenje, mora vrijediti da: $\exists \beta^*$, takav da $\frac{\partial L(w^*,\beta^*)}{\partial w} = 0 \wedge \frac{\partial L(w^*,\beta^*)}{\partial \beta^*} = 0$

Primal problem

Optimizacijski problem klasifikatora optimalne granice se može riješiti primjenom generaliziranih Lagrangeovih multiplikatora.

Neka je potrebno riješiti optimizacijski problem zadan s:

$\min_w f(w)$, tako da bude zadovoljeno: $g_i(w) \le 0$, $i = 1,...,k$ i $h_i(w) = 0$, $i = 1,...,l$

Konstruira se generalizirani Lagrangian: $L(w,\alpha,\beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w)$

Neka je $\theta_p(w) = \max_{\substack{\alpha,\beta \\ \alpha_i \ge 0}} L(w,\alpha_i,\beta_i)$ i neka je $p^* = \min_w \theta_p(w) = \min_w \max_{\substack{\alpha,\beta \\ \alpha_i \ge 0}} L(w,\alpha_i,\beta_i)$,

Oznaka „p“ stoji za „primal problem“. Vrijedi: $\theta_p(w) = \begin{cases} +\infty, \text{ ako } g_i(w) > 0 \\ +\infty, \text{ ako } \mathrm{h}_i(w) \ne 0 \\ f(w), \text{ inače} \end{cases}$

Vrijedi li $g_i(w) > 0$, narušeno je ograničenje optimizacije. Postavljanjem $\alpha_i = +\infty$ dobije se $\theta_p(w) = +\infty$. Isto vrijedi i za $\mathrm{h}_i(w) \ne 0$, jer je opet narušeno ograničenje optimizacije.

Ako nisu narušena ograničenja optimizacije, sumacije u $L(w,\alpha,\beta)$ su jednake 0, jer se maksimum postiže kad su svi $\alpha_i = 0$ i $\beta_i = 0$, stoga je $\theta_p(w) = f(w)$.

Zato je: $\theta_p(w) = \begin{cases} +\infty, \text{ ako su ograničenja narušena} \\ f(w), \text{ inače} \end{cases}$

To znači da je $p^* = \min_w \theta_p(w)$ zapravo originalni problem.

Dual problem

Neka je $\theta_d(\alpha,\beta) = \min_w L(w,\alpha,\beta)$, neka je $d^* = \max_{\substack{\alpha,\beta \\ \alpha_i \ge 0}} \theta_d(\alpha,\beta) = \max_{\substack{\alpha,\beta \\ \alpha_i \ge 0}} \min_w L(w,\alpha,\beta)$,

gdje oznaka „d“ stoji za „dual problem“. Razlika između primal i dual problema,to jest između p^* i d^*, je jedino u poretku maksimizacije i minimizacije.

Uvijek vrijedi $d^* \leq p^*$ (tj. $\max \min(...) \leq \min \max(...)$).

Pod određenim uvjetima vrijedi $d^* = p^*$. Ovi su uvjeti poznati pod nazivom Karush–Kuhn–Tuckerovi (KKT) uvjeti [7]. Tada je moguće riješiti dual problem umjesto primal problema.

Primjena na klasifikator optimalne granice

Promjena u notaciji za primjenu na klasifikator optimalne granice:

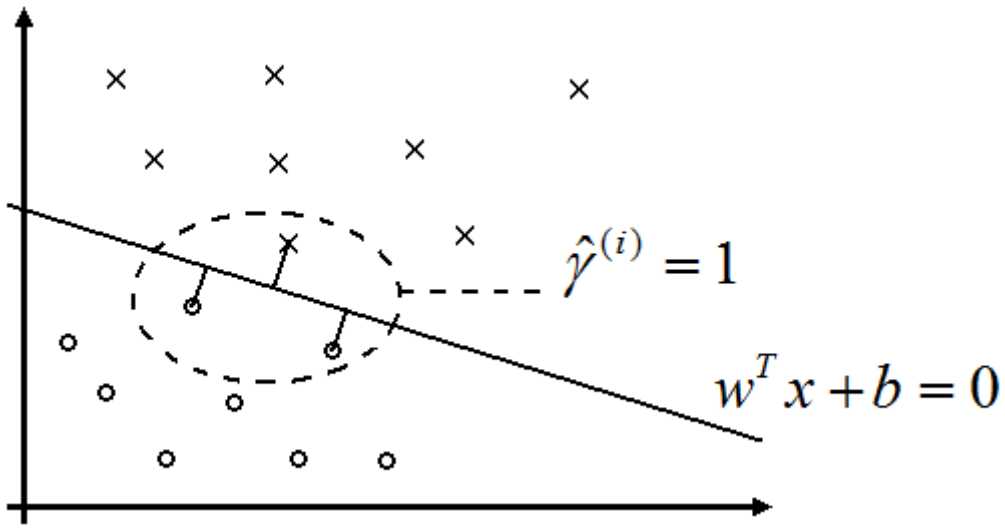
generalizirani Lagrangeovi multiplikatori $\alpha_i, \beta_i \rightarrow \alpha_i$ (postoje samo g_i nejednakosti u optimizacijskom problemu, nema h_i jednakosti), parametri $w \rightarrow w, b$.

Optimizacijski problem:

$\min_{w,b} \frac{1}{2} \| w \|^2$, $\frac{1}{2}$ dodana radi ljepšeg računa, pod uvjetom da vrijedi $y^{(i)}(w^T x^{(i)} + b) \geq 1$

Uvjet se može napisati kao: $g_i(w,b) = -y^{(i)}(w^T x^{(i)} + b) + 1 \leq 0$.

Iz KKT uvjeta: $\alpha_i^* > 0 \Rightarrow g_i(w^*) = 0$. Zato $\alpha_i^* > 0 \Rightarrow \hat{\gamma}^{(i)} = 1$, tj. $(x^{(i)}, y^{(i)})$ ima funkcijsku marginu jednaku jedan, što je prikazano na slici 4.4.



Slika 4.4. Potporni vektori

Obično vrijedi $\hat{\gamma}^{(i)} = 1 \Leftrightarrow \alpha_i \neq 0$.

Primjeri za učenje koji su dalje od odvajajuće hiperravnine imaju $\hat{\gamma}^{(i)} > 1$.

Kao što slika 1.4 sugerira, obično primjera za učenje s $\hat{\gamma}^{(i)} = 1$ ima relativno malo (s obzirom na cijeli skup za učenje). Primjeri za učenje s $\hat{\gamma}^{(i)} = 1$ nazivaju se potpornim vektorima (eng. *Support Vectors*). Odavde potječe ime stroj s potpornim vektorima (eng. *Support Vector Machine*, SVM). Kako je relativno

malo potpornih vektora, većinom vrijedi $\alpha_i = 0$, jer je $\alpha_i = 0$ za primjere za učenje koji nisu potporni vektor.

Generalizirani Lagrangian je: $L(w,b,\alpha) = \frac{1}{2} \| w \|^2 - \sum_{i=1}^{m} \alpha_i (y^{(i)} (w^T x^{(i)} + b) - 1)$.

Za dual problem vrijedi: $\theta_d(\alpha) = \min_{w,b} L(w,b,\alpha)$.

Rješavanjem pomoću generaliziranih Lagrangiana dobije se:

$$W(\alpha) = L(w,b,\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j < x^{(i)}, x^{(j)} >$$

Dual problem je: $\max_{\alpha} W(\alpha)$, uz ograničenja: $\alpha_i \geq 0$ i $\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$

Po KKT vrijedi: $\theta_d(\alpha) = W(\alpha)$.

Postupak rješavanja problema klasifikatora optimalne granice

Prvo se riješi dual problem za α,

zatim se odredi w iz izraza $w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$,

te je još potrebno odrediti b iz $b = -\frac{\max_{i:y^{(1)}=-1} w^T x^{(i)} + \min_{i:y^{(1)}=1} w^T x^{(i)}}{2}$.

Interpretacija formule za b: nađe se najgori primjer za učenje iz grupe „1“ i najgori primjer za učenje iz grupe „-1“, te se stavi hiperravnina na pola njihove međusobne udaljenosti.

Opisani postupak ima dobro svojstvo da samo potporni vektori imaju $\alpha_i \neq 0$, zbog čega je potrebno pamtiti samo potporne vektore jednom kada je klasifikator izgrađen.

Radi potpunosti treba spomenuti da su uvjeti potrebni da vrijedi $d^* = p^*$ (i ispravnost KKT uvjeta) ispunjeni, pa je sav provedeni postupak ispravan.

4.3 Jezgreni trik

Klasifikator optimalne granice je prilično dobar linearni klasifikator. Ipak, baš zato što je linearan, ima ograničenu primjenu jer je primjenjiv samo na klasifikaciju linearno odvojivih grupa objekata (u odnosu na izabrani skup značajki). Stroj s potpornim vektorima je modificirani klasifikator optimalne granice koji nije ograničen samo na linearnu klasifikaciju. Mogućnost primjene na linearno neodvojive grupe objekata postiže se metodom poznatom pod nazivom jezgreni trik (eng. *Kernel Trick*, [3]).

4.3.1 Motivacija

Postoji cijeli niz problema koje je moguće izraziti (točnije, koji su vezani uz podatke) samo preko unutarnjeg produkta (eng. *Inner product)*, umnoška dva vektora koji kao rezultat daje realan broj. Klasifikator optimalne granice takav je problem.

Hipoteza je dana s $h_{w,b}(x) = g(w^T x + b)$, gdje je $w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$.

Vrijedi $w^T x + b = \sum_{i=1}^{m} \alpha_i y^{(i)} < x^{(i)}, x > + b$, pri čemu je $< x^{(i)}, x >= (x^{(i)})^T x$ oznaka za unutarnji produkt vektora $x^{(i)}$ i x.

Također vrijedi $W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j < x^{(i)}, x^{(j)} >$, gdje je opet $< x^{(i)}, x^{(j)} >= (x^{(i)})^T x^{(j)}$ oznaka za unutarnji produkt vektora $x^{(i)}$ i $x^{(j)}$.

Zato je moguće postupak izgradnje klasifikatora (određivanje w i b), kao i postupak klasifikacije novog objekta povezati s podacima koristeći samo unutarnji produkt. Slijedi da nikada nije potrebno vektore značajki $x^{(i)}$ odnosno x izraziti direktno.

4.3.2 Ideja

Neka je dan vektor značajki $x \in \mathbb{R}^n$, i neka je dana funkcija $\phi : \mathbb{R}^n \mapsto \mathbb{R}^m$, gdje je $m > n$.

Funkcija $\phi(x)$ preslikava točku (vektor značajki) x iz n-dimenzionalnog vektorskog prostora u m-dimenzionalni vektorski prostor, gdje su n- i m-dimenzionalni prostori tzv. Hilbertovi prostori (eng. *Hilbert space*). Pri tome m-dimenzionalni prostor obično ima puno više dimenzija, a čak je dozvoljeno da je $\phi(x) \in \mathbb{R}^{\infty}$, tj. da se preslikava u beskonačno-dimenzionalan prostor.

Budući da se cijeli algoritam može izraziti pomoću unutarnjeg produkta, zamjenom $< x^{(i)}, x^{(j)} >$ s $< \phi(x^{(i)}), \phi(x^{(j)}) >$ cijeli algoritam sada radi s m-dimenzionalnim vektorima značajki $\phi(x)$.

Javlja se problem efikasnog računanja unutarnjeg produkta vektora koji imaju veoma velik (ili čak beskonačan) broj značajki.

Srećom, u mnogim bitnim specijalnim slučajevima moguće je napisati tzv. jezgrenu funkciju (eng. *Kernel Function*) $K(x, z)$, tako da vrijedi $K(x, z) =< \phi(x), \phi(z) >$ i da je pri tome moguće izračunati $K(x, z)$ puno efikasnije nego da se eksplicitno računa $< \phi(x), \phi(z) >$.

Primjer

Neka je $x, z \in \mathbb{R}^n$ i neka je $K(x, z) = (x^T z)^2$, tada za $K(x, z)$ vrijedi:

$$K(x,z) = (x^T z)^2 = (\sum_{i=1}^{n} x_i z_i)(\sum_{j=1}^{n} x_j z_j) = \sum_{i=1}^{n}\sum_{j=1}^{n} (x_i x_j)(z_i z_j)$$

Nadalje, neka je $\phi(x) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ \vdots \\ x_n x_{n-1} \\ x_n x_n \end{bmatrix}$. Vrijedi: $K(x,z) = (\phi(x))^T \phi(z)$

Računa li se $K(x,z)$ kao $K(x,z) = (\phi(x))^T \phi(z)$, složenost je $O(n^2)$, a ako se $K(x,z)$ računa kao $K(x,z) = (x^T z)^2$, tada je složenost $O(n)$.

4.3.3 Izbor jezgre

Da bi se izgradio kvalitetan klasifikator, izbor jezgre je prilično bitan. Neka su x i z vektori značajki, te $x \mapsto \phi(x)$ i $z \mapsto \phi(z)$ slike dobivene jezgrom. Ako su x i z jako slični, tada su vektori $\phi(x)$ i $\phi(z)$ usmjereni u približno istom smjeru, pa je njihov unutarnji produkt velik, a ako su x i z veoma različiti, tada je njihov unutarnji produkt malen. Ova intuicija je korisna, a nije stroga. Pri susretu s novim klasifikacijskim problemom jedan od načina kako početi je pokušati naći funkciju $K(x,z)$ koja za objekte koji se promatraju kao slični vraća veliku vrijednost, a za objekte koji se promatraju kao različiti vraća malu vrijednost.

Ispravnost jezgre

Pitanje ispravnosti jezgre je ekvivalentno pitanju postoji li funkcija $\phi(x)$ koja preslikava vektor iz jednog prostora u drugi.
Odnosno, $K(x,z)$ je ispravna jezgra ako i samo ako $\exists \phi$ takav da vrijedi: $K(x,z) = <\phi(x), \phi(z)>$

Mercerov teorem (nužan i dovoljan uvjet da bi $K(x,z)$ bila ispravna jezgra):

Neka je dan proizvoljan skup od $m < \infty$ točaka $\{x_1, ..., x_m\}$. Jezgra $K(x,z)$ je ispravna (Mercerova) jezgra ako i samo ako za pripadajuću matricu $K \in \mathbb{R}^{n \times n}$, definiranu kao $K_{ij} = K(x^{(i)}, x^{(j)})$ vrijedi da je simetrična i da je $K \geq 0$.

Gaussova jezgra

Ako se razvije specijalizirana jezgra za određeni problem, mogu se znatno poboljšati dobivene performanse. Do sada su kreirane i jezgre specijalizirane za rad sa slikama [9],[10]. Dobra alternativa

razvoju vlastite jezgre je korištenje Gaussove jezgre koja je ovdje ukratko opisana (osim Gaussove jezgre, postoje još neke standardne jezgre opće namjene, [4].

Gaussova jezgra (eng. *Gaussian kernel*) je vjerojatno jezgra s najraširenijom primjenom. Definirana je formulom $K(x,z) = \exp(-\tau \| x - z \|^2)$, gdje je $\tau \in \mathbb{R}$. Naziva se Gaussovom jezgrom zbog sličnosti formule jezgre s formulom Gaussove krivulje. Ova jezgra odgovara preslikavanju u $\mathbb{R}^\infty$ prostor, tj. dobije se beskonačno-dimenzionalan prostor.

Sve što je originalno linearno razdvojivo u prostoru početnih značajki, razdvojivo je i u višem prostoru određenom ovom jezgrom. Za velik broj problema je Gaussova jezgra dobar izbor početne jezgre.

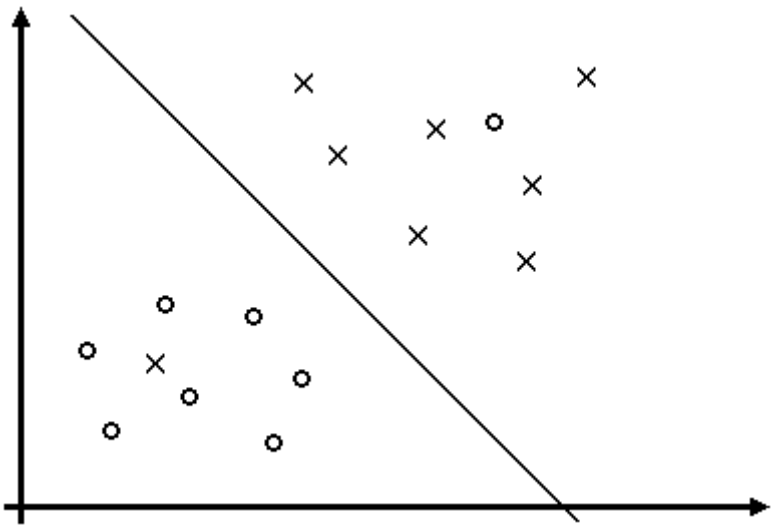
4.4 Stroj s potpornim vektorima L1 meke granice

U nastavku je opisana modifikacija stroja s potpornim vektorima kojom se postižu bolje performanse i veća upravljivost izgradnjom klasifikatora.

4.4.1 Motivacija

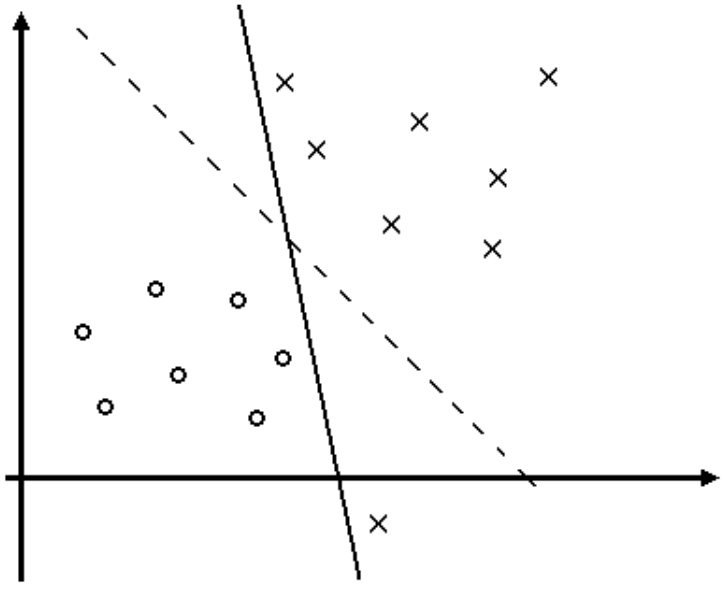
Stroj s potpornim vektorima je klasifikator optimalne granice na koji je primijenjen jezgreni trik. Zato je u klasifikatoru optimalne granice svaka pojava unutarnjeg produkta zamijenjena s jezgrenom funkcijom ($< x, z > \rightarrow K(x,z)$). Upotrebom jezgrenog trika, stroj s potpornim vektorima može dati nelinearnu granicu između dvije grupe, pa je primjenjiv i na linearno neodvojive grupe objekata (u odnosu na izabrani skup značajki). Zapravo, stroj s potpornim vektorima i dalje rezultira linearnom granicom, no ta je granica linearna u prostoru u koji su preslikani vektori značajki funkcijom $\phi(x)$. Funkcija $\phi(x)$ često nije eksplicitno poznata, a ponekad ni sam prostor, u koji se preslikava upotrebom jezgre, nije poznat.

Ovakav stroj s potpornim vektorima i dalje zahtijeva da su grupe objekata potpuno odvojive (u suprotnom algoritam neće uspješno završiti). Vjerojatnost da su grupe odvojive u nekom prostoru raste s brojem dimenzija. Unatoč tome, upotreba određene jezgre ne garantira da će dani skup za učenje biti (linearno) odvojiv u prostoru određenom s upotrijebljenom jezgrom. Iz tog je razloga korisno modificirati stroj s potpornim vektorima tako da radi i ako je određeni dio skupa za učenje linearno neodvojiv, čak i u prostoru određenom upotrebom jezgre. Na slici 4.5 su prikazane dvije linearno neodvojive grupe objekata. Iako grupe nisu odvojive, ipak postoji granica koja odvaja grupe relativno uspješno.



Slika 4.5. Linearno neodvojive grupe objekata

Osim iznesenog problema, javlja se i problem prevelikog utjecaja pojedinog primjera za učenje na rezultantnu granicu grupa. Obično je bolje donekle smanjiti utjecaj pojedinog primjera za učenje. Time se izbjegava i slučaj kada je u skupu za učenje neki objekt greškom smješten u krivu grupu. Slika 4.6 ilustrira ovaj problem.



Slika 4.6. Problem prevelikog utjecaja pojedinog primjera za učenje

Jedna modifikacija stroja s potpornim vektorima koja rješava prethodno izložene probleme naziva se stroj s potpornim vektorima L1 meke granice (eng. *L1 norm softmargin Support Vector Machine*, [5]).

4.4.2 Modifikacija

Modificira se formulacija problema stroja s potpornim vektorima:

$\min_{w,b,\xi_i} \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{m} \xi_i$ pod uvjetom da vrijedi $y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i$ i $\xi_i \geq 0, i = 1,...,m$, gdje su ξ_i „kaznene" varijable.

Ako je $y^{(i)}(w^T x^{(i)} + b) > 0$, tada je objekt ispravno klasificiran, u suprotnom je pogrešno klasificiran. Postavi li se neki $\xi_i > 1$, tada je $\hat{y} < 0$, pa se dozvoljava da primjer iz skupa za učenje bude krivo klasificiran. Ipak, takav izbor se ne preporuča jer u (minimizacijskom) optimizacijskom cilju stoji: $+C \sum_{i=1}^{m} \xi_i$, pri čemu zadani parametar C određuje koliko se lošim smatra izbor $\xi_i > 1$.

Izvođenjem dual problema se u konačnici dobije:

$$\max_{\alpha} W(\alpha) = \max_{\alpha} \left[\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j < x^{(i)}, x^{(j)} > \right],$$

uz ograničenja: $\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$ i $0 \leq \alpha_i \leq C, i = 1,...,m$.

Odnosno, jedina modifikacija je $0 \leq \alpha_i \leq C$ umjesto $\alpha_i \geq 0$.

Iz KKT uvjeta se dobije kriterij konvergencije, kada su α_i konvergirali k globalnom optimumu. Ovaj se kriterij može koristi u algoritmu koji rješava optimizacijski problem po α_i:

$\alpha_i = 0 \Rightarrow y^{(i)}(w^T x^{(i)} + b) \geq 1$
$\alpha_i = C \Rightarrow y^{(i)}(w^T x^{(i)} + b) \leq 1$
$0 \leq \alpha_i \leq C \Rightarrow y^{(i)}(w^T x^{(i)} + b) = 1$

4.5 Slijedna minimalna optimizacija

Slijedna minimalna optimizacija (eng. *Sequential Minimal Optimization*, SMO) je jedan od algoritama [6] koji rješava optimizacijski problem stroja s potpornim vektorima.

Slijedna minimalna optimizacija je inspirirana algoritmom poznatim pod nazivom koordinatni uspon (*eng. Coordinate Ascent*). Ovaj algoritam rješava optimizacijski problem: $\max_{\alpha}[W(\alpha_1, \alpha_2, \ldots, \alpha_m)]$, pri čemu nisu zadana nikakva ograničenja. Ideja algoritma je u svakom koraku držati sve α_i konstantnim osim jednog. Tako $W(.)$ postaje funkcija samo jedne varijable, te se maksimizacija u danom koraku može rješti analitički. Svakim se korakom algoritam približava rješenju duž jedne osi, što ukupno dovodi do konvergencije algoritma.

Koordinatnim usponom nije moguće riješiti optimizacijski problem stroja s potpornim vektorima, jer postoje ograničenja na α_i. Nije moguće sve α_i, osim jednog, držati konstantnim u jednoj iteraciji algoritma, zbog ograničenja $\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$. Ovo ograničenje zahtjeva da je bilo koji α_i u potpunosti određen s preostalim α_j (npr. vrijedi $\alpha_1 = -\frac{\sum_{i=2}^{m} \alpha_i y^{(i)}}{y^{(1)}}$). Iako nije moguće mijenjati samo jedan α_i, moguće je mijenjati bilo koja dva α_i, a da se pri tome ne naruše ograničenja na α_i. Kao posljedica ove ideje nastao je algoritam slijedne minimalne optimizacije (eng. *Sequential Minimal Optimization*, [7]).
U nazivu algoritma je minimalna optimizacija jer se mijenja najmanji mogući broj α_i, a da su pri tome ograničenja zadovoljena.
Nacrt algoritma:

Ponavljaj do konvergencije {

Odaberi α_i, α_j pomoću heuristike

Drži sve $\alpha_k, k \neq i, j$ konstantnim

Optimiziraj $W(\alpha)$ u odnosu na α_i, α_j

(uvažavajući pri tome sva ograničenja)

}

Efikasnost SMO algoritma uvelike ovisi o dobrom odabiru heuristike koja izabire α_i, α_j koji će se sljedeći optimizirati [11].

Uzastopnom primjenom optimizacijskih koraka algoritam se približava traženom rješenju. Kako se u pojedinom koraku mijenja par α_i, α_j, ponekad se dogodi da se optimizacijom α_i (približavanjem α_i njegovoj konačnoj vrijednosti), α_j udalji od njegovog konačnog rješenja. Unatoč ovom udaljavanju α_j od njegovog rješenja, ukupni učinak svakog pojedinog koraka je pozitivan, pa se algoritam svakim korakom približava traženom rješenju [8].

4.6 Izvedba stroja s potpornim vektorima

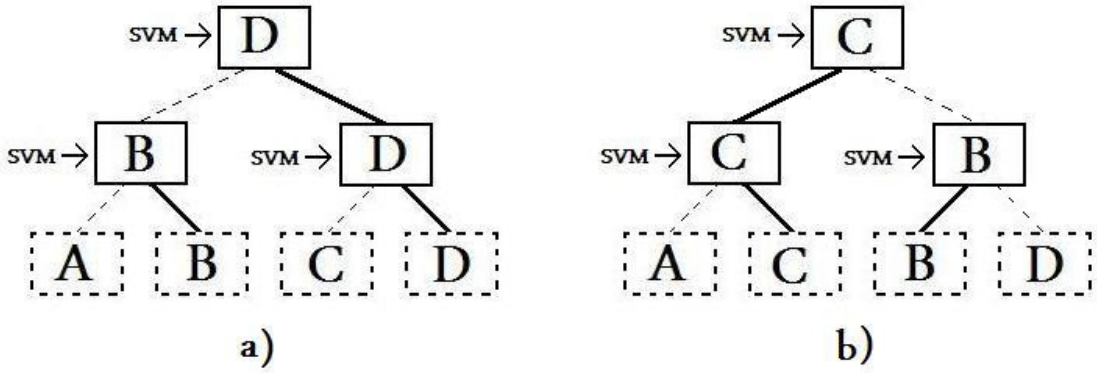
Stroj s potpornim vektorima je zapravo binarni klasifikator, a naš cilj je detektirati trokutaste prometne znakove te ih potom klasificirati u jednu od nekoliko desetaka klasa. Problemu klasifikacije s više klasa se uobičajeno pristupa na dva načina [13]: klasifikacija jedan na jedan (eng. *One-Against-One*) i klasifikacija jedan protiv svih (eng. *One-Against-All*).

Kod klasifikacije jedan na jedan se za sve moguće parove klasa napravi po jedan binarni klasifikator. Uzorak se najčešće klasificira u klasu koja dobije najviše glasova nakon upotrebe svih klasifikatora.

Za potrebe klasifikacije jedan protiv svih se napravi onoliko binarnih klasifikatora koliko je klasa klasifikacije. Jedan se binarni klasifikator izgradi tako da se jedna klasa izdvoji, a sve se ostale tretiraju kao druga klasa. Svaki klasifikator prihvaća po jednu klasu a odbacuje sve ostale.

U ovom su projektu isprobane obje prethodno izložene metode, no u finalnoj se verziji ne koriste u osnovnom obliku, već modifikacija klasifikacije jedan na jedan. Modifikacija je u načinu kombiniranja klasifikatora. Klasifikacija se vrši tako da se nad svim klasama „izgradi binarno stablo“. Na početku se izabere minimalan broj klasifikatora koji pokrivaju sve klase. Jedan klasifikator pokriva dvije klase, pa se ukupno izabere dvostruko manje klasifikatora nego je klasa. Provede se klasifikacija sa svim ovim klasifikatorima i time se odbaci pola klasa (jer svaki binarni klasifikator izabere jednu od dvije klase kao istinitu). Za sve klase koje su ostale, odabere se ponovno minimalan broj klasifikatora koji im odgovaraju. Postupak se ponavlja sve dok ne ostane samo jedna klasa. Time je dobiven rezultat klasifikacije.

Kako sam proces klasifikacije nije vremenski zahtjevan u odnosu na ostatak programa, ovaj se postupak provede n puta, gdje je n unaprijed zadani neparan broj (najčešće 3, 5 ili 7). Za svaku provedbu procesa se odabiru različiti klasifikatori koji pokrivaju sve početne klase. Time se dobivaju potencijalno različiti rezultati, jer se kroz cijeli postupak koriste različiti klasifikatori (Slika 4.7). Konačno se kao rezultat klasifikacije izabire ona klasa koja je odabrana barem n/2 puta (dobila većinu glasova). Ako takve nema, ne donosi se odluka o klasifikaciji pomoću stroja s potpornim vektorima, već se odluka prepušta umjetnoj neuronskoj mreži.

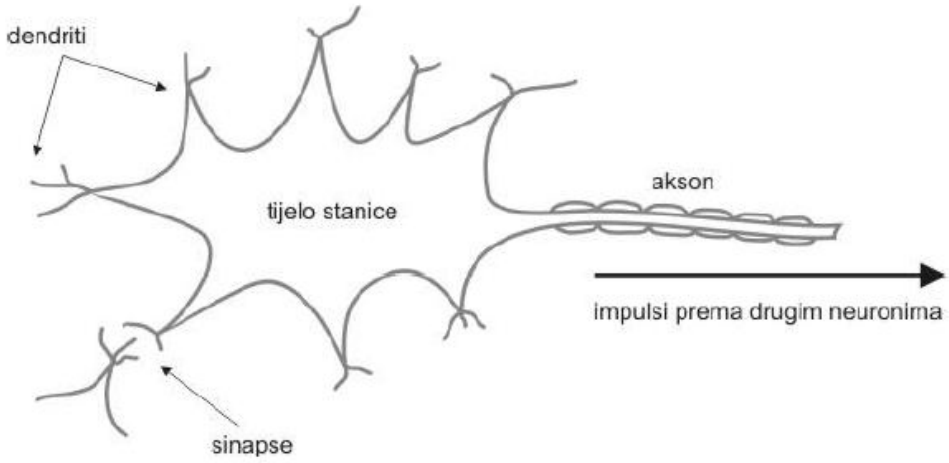


Slika 4.7. Ilustracija mogućnosti različite klasifikacije za različita stabla. Klasifikator je predstavljen punim pravokutnicima u kojima je njegova odluka, a početne klase su predstavljene iscrtkanim pravokutnicima

5 Umjetne neuronske mreže

[4]Probleme poput detekcije i klasifikacije slika, rijetko je moguće rješavati pomoću tradicionalnih algoritama. Pokazalo se kako je računalo vrlo loše u rješavanju mnogih problema u čijoj je srži sposobnost generalizacije. Za razliku od računala, čovjek takve probleme rješava brzo i efikasno. Upravo je ljudski mozak inspiracija za razvoj umjetnih neuronskih mreža.

Ljudski mozak sastoji se od međusobno povezanih neurona. Osnovni princip rada mozga je prolazak električnih impulsa kroz veze među neuronima.



Slika 5.1 *Pojednostavljeni model neurona [14]*

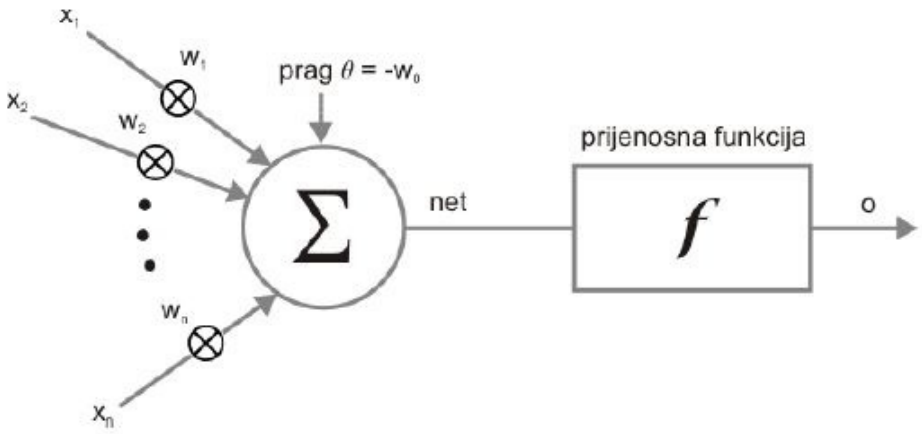
Slika 5.1 prikazuje osnovne dijelove pojednostavljenog modela neurona. Električni impulsi iz susjednih neurona preko sinapsa i dendrita dolaze u tijelo stanice, te čine pobudu neurona. Odziv na pobudu se šalje preko aksona prema drugim neuronima.

Umjetne neuronske mreže [14] su matematički modeli zasnovani na pojednostavljenom modelu ljudskog mozga.

[4] Dijelovi poglavlja su preuzeti iz: I. Kovaček: Sustav za detekciju i raspoznavanje prometnih znakova

5.1 Umjetni neuron

Već 1943. godine McCulloch i Pitts predložili su matematički model umjetnog neurona [14], prikazan na slici (Slika 5.2).



Slika 5.2 *Umjetni neuron [14]*

Dijelovi umjetnog neurona su:

- $x_{1..n}$ - ulazni signali
- $w_{0..n}$ - težinski faktori
- zbrajalo
- f - prijenosna funkcija

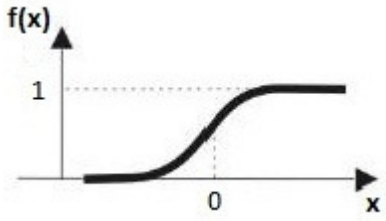
Analogija između umjetnog neurona i ljudskog neurona je sljedeća: x_i su ulazni signali iz drugih neurona, w_i su težine koje predstavljaju jakost sinapse, zbrajalo je tijelo stanice, a prijenosna funkcija predstavlja akson.

Vrijednost izlaza neurona određena je relacijom 5.1:

$$o = f(\sum_{i=0}^{n} x_i w_i) \qquad (5.1)$$

gdje je $x_0 = 1$.

Odabir prijenosne funkcije je proizvoljan, te ovisi o problemu. Pogodna prijenosna funkcija za klasifikaciju je sigmoidalna funkcija (5.2).



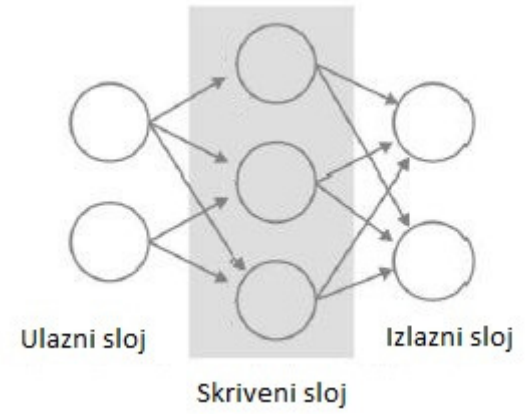
$$f(x) = \frac{1}{1+e^{-x}} \tag{5.2}$$

Slika 3. Sigmoidalna funkcija [14]

Bitno svojstvo sigmoidalne funkcije je nelinearnost, koja omogućava razdvajanje linearno nerazdvojivih klasa pomoću neuronske mreže. U okviru ovog rada, neuronske mreže se koriste samo za klasifikaciju, te se u nastavku podrazumijeva da je prijenosna funkcija neurona funkcija (5.2).

5.2 Arhitektura unaprijedne neuronske mreže

Neuronska mreža nastaje povezivanjem neurona. Unaprijedna neuronska mreža *(eng. feedforward network)* je osnovna arhitektura mreže (Slika 5.3).



Slika 5.3 Arhitektura unaprijedne neuronske mreže [14]

Ulazni sloj predstavlja ulazne podatke, dok su izlazi neurona izlaznog sloja rezultati. Skriveni sloj se sastoji od jednog ili više podslojeva neurona. Veze postoje samo između susjednih slojeva i podslojeva: izlazi prethodnog sloja su ulazi sljedećeg, promatrajući od strane ulaznog sloja.

5.3 Učenje

Svaki od izlaza neuronske mreže je funkcija ulaznih podataka. Općenito je cilj konstruirati traženu funkciju. To se postiže postavljanjem težinskih faktora na adekvatne vrijednosti. Ručno određivanje težinskih faktora je teško već i za vrlo nizak broj neurona, a u konkretnim problemima je nemoguće. Stoga se težinski faktori određuju učenjem.

Jedan primjerak za učenje sastoji se od ulaznih podataka i traženog izlaza. Skup primjera za učenje je diskretni uzorak tražene funkcije, te je cilj učenja što bolje rekonstruirati funkciju iz njenih diskretnih uzoraka. Osnovna ideja je minimizirati funkciju pogreške u ovisnosti o težinskim faktorima. Odabrana funkcija pogreške $E(\cdot)$ je polovina srednje kvadratne pogreške:

$$E(\vec{w}) = \frac{1}{2N} \cdot \sum_{i=1}^{N} (t_i - o_i)^2 \qquad (5.3)$$

gdje su:

$\vec{w}$ - težinski faktori mreže

N - broj primjera za učenje

t_i - ciljana vrijednost i-tog primjerka

o_i - vrijednost izlaza mreže za i-ti primjerak.

Osnovna ideja je odrediti metodu traženja minimuma pogreške za jedan neuron, te pomoću propagacije pogreške unazad proširiti traženje minimuma na cijelu mrežu.

5.3.1 Gradijentni spust

Promotrimo funkciju pogreške izlaza zbrajala, zanemarujući trenutno aktivacijsku funkciju neurona. Odabrana metoda za traženje minimuma funkcije pogreške jednog neurona je gradijentni spust [14]. Gradijent je smjer najbržeg rasta funkcije, te se težine mijenjaju u suprotnom smjeru:

$$\vec{w}(k+1) = \vec{w}(k) - \eta \cdot \nabla_w E(\vec{w}) \qquad (5.4)$$

gdje je η stopa učenja, $\vec{w}$ faktori težine neurona,a k korak učenja.

Gradijent funkcije pogreške iznosi:

$$\nabla_w E(\vec{w}) = \frac{dE(\vec{w})}{d\vec{w}} = \frac{d}{d\vec{w}}\left(\frac{1}{2N}\sum_{i=1}^{N}\varepsilon_i^{\,2}\right) = \frac{1}{N}\sum_{i=1}^{N}\varepsilon_i \frac{d}{d\vec{w}}\varepsilon_i \qquad (5.5)$$

gdje je N broj primjeraka za učenje, a ε_i iznos pogreške na i-tom primjeru.

Primijetimo da izračun gradijenta pomoću izraza (5.5) omogućuje samo grupno učenje. U praksi se pokazalo boljim pojedinačno učenje. Da bi se omogućilo pojedinačno učenje, funkcija pogreške se aproksimira koristeći samo jedan primjerak za učenje:

$$E(\vec{w}) \approx \frac{1}{2}\varepsilon^2 \qquad (5.6)$$

Tada je aproksimacija gradijenta funkcije pogreške:

$$\nabla_w E(\vec{w}) \approx \varepsilon \frac{d}{d\vec{w}}\varepsilon \qquad (5.7)$$

Ograničimo se trenutno na izlaz zbrajala neurona, za kojeg vrijedi:

$$o_{net} = \vec{w} \cdot \vec{x} \qquad (5.8)$$

Tada vrijedi:

$$\varepsilon = t_{net} - o_{net} = t_{net} - \vec{w} \cdot \vec{x} \qquad (5.9)$$

Iz formula (5.9) i (5.7) slijedi:

$$\nabla_w E(\vec{w}) \approx -\varepsilon \cdot \vec{x} \qquad (5.10)$$

Promotrimo sada derivaciju sigmoidalne aktivacijske funkcije (5.2) :

$$\frac{d}{dx} f(x) = \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = f(x)(1-f(x)) \qquad (5.11)$$

Veza između greške na izlazu zbrajala, i greške na izlazu prijenosne funkcije je:

$$\varepsilon = (t-o)\cdot f'(o_{net}) = o(1-o)(t-o) \qquad (5.12)$$

Iz (12), i (10) slijedi:

$$\nabla_w E(\vec{w}) \approx -o \cdot (1-o) \cdot (t-o) \cdot \vec{x} \qquad (5.13)$$

Iz (5.13) i (5.4) slijedi konačan izraz za traženje minimuma funkcije pogreške jednog neurona pojedinačnim učenjem gradijentnim spustom:

$$\vec{w}(k+1) = \vec{w}(k) + \eta \cdot o \cdot (1-o) \cdot (t-o) \cdot \vec{x} \qquad (5.14)$$

5.3.2 Propagacija greške unatrag

Pomoću izraza (5.14) moguće je gradijentnim spustom ugađati težine izlaznih neurona, ali da bi se pravilo primijenilo na neurone skrivenog sloja, potrebno je definirati pogrešku na izlazu neurona (t – o). Upravo to radi propagacija greške unatrag [14] *(eng. backpropagation).*

Veza greške δ na izlazu skrivenog neurona i greške ε (5.10) na izlazu zbrajala izlaznog neurona je:

$$\delta = w \cdot \varepsilon \tag{5.15}$$

gdje je w težinski faktor između dva neurona,

Greška na izlazu skrivenog neurona δ_h s obzirom na sve izlazne neurone, je:

$$\delta_h = t_h - o_h = \sum_{i=1}^{K} w_{hi}\varepsilon_i \tag{5.16}$$

gdje je K broj neurona izlaznog sloja, w_{hi} težinski faktor između h-tog neurona skrivenog sloja i i-tog neurona izlaznog sloja, a ε_i greška na izlazu iz zbrajala i-tog neurona izlaznog sloja.

Iz (5.16) i (5.14) slijedi pravilo ugađanja skrivenog neurona gradijentnim spustom:

$$\vec{w}_h(k+1) = \vec{w}_h(k) + \eta \cdot o_h \cdot (1-o_h) \cdot \delta_h \cdot \vec{x}_h \tag{5.17}$$

Pomoću gradijentnog spusta i propagacije greške unazad, moguće je učiti kompletnu mrežu. Slijedi algoritam za učenje:

```
Inicijaliziraj težinske faktore slučajne vrijednosti
Dok nije ispunjen uvjet zaustavljanja radi:
    Za svaki primjer za učenje radi:
                Izračunaj izlaz mreže o
   Ugodi faktore izlaznog sloja po (14)
   Ugodi faktore skrivenog sloja po (17)
    Kraj
Kraj
```

5.3.3 Ubrzanje konvergencije

Modifikacijom algoritma učenja često se može postići brža konvergencija. Neke od modifikacija su adaptacija stope učenja te moment učenja[15].

Nakon jedne epohe učenja, to jest nakon jednog učenja svakog od primjeraka, stopa učenja se adaptira u odnosu na promjenu funkcije greške: ako se greška smanjila, stopa se povećava, ako se greška povećala, promjene težina se zanemaruju i stopa se smanjuje.

Negativna strana metode gradijentnog spusta je rizik pronalaženja lokalnih minimuma. Dodavanjem momenta učenju [15] postiže se mogućnost bijega iz malih lokalnih minimuma, kao i nastavljanje učenja na ravnim plohama funkcije pogreške.

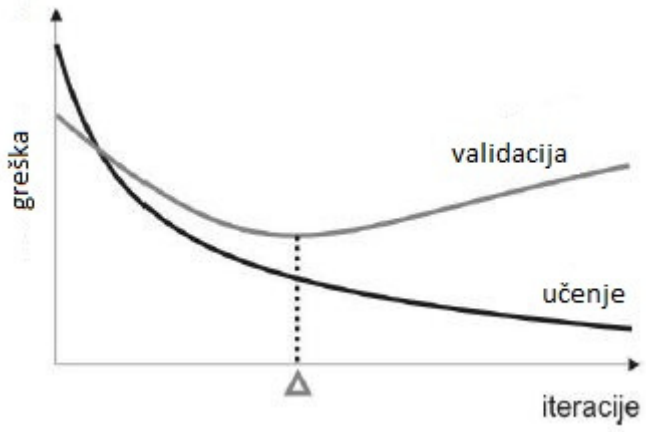
$$\vec{w}(k+1) = \Delta\vec{w} + m \cdot (\vec{w}(k) - \vec{w}(k-1)) \qquad (5.18)$$

Izraz (5.18) predstavlja učenje sa momentom. Faktor m je faktor momenta i predstavlja udio prošle promjene težina u trenutnoj promjeni. Član $\Delta\vec{w}$ je promjena težina određena metodom gradijentnog spusta.

5.3.4 Validacija

Neuronska mreža pretjeranim učenjem ulaznog skupa može naučiti šumove skupa za učenje, i izgubiti svojstvo generalizacije. Tako naučena neuronska mreža će dati loše rezultate u primjeni.

Spriječavanje pretjeranog učenja postiže se uvođenjem skupa za validaciju [14]. Mreža uči na primjerima skupa za učenje, a kao rezultantna mreža odabire se ona sa najboljim performansama na skupu za validaciju. Uz oprezan odabir skupova za učenje i validaciju, ovim pristupom se postiže očuvanje svojstva generalizacije mreže.



Slika 5.4 Korištenje validacije [14]

Važno je napomenuti da rezultati nad skupom za validaciju nisu rezultati testiranja. Iako mreža nije učila na primjerima za validaciju, samim odabirom mreže po performansama validacije, mreža je pod utjecajem tog skupa. Za testiranje je potrebno odvojiti još jedan skup primjeraka.

Osim što se po validaciji odabire najbolja mreža, po njoj se također i odabiru ulazne značajke, što se manifestira u još većem utjecaju validacije na mrežu.

5.4 Izvedba umjetnih neuronskih mreža

Sve korištene neuronske mreže su unaprijedne (Slika 5.3). Sastoje se od ulaznog sloja, skrivenog sloja sa jednim podslojem, te izlaznog sloja. Prijenosna funkcija svih neurona skrivenog i izlaznog sloja je sigmoidalna funkcija.

Ulazni sloj predstavlja ulazne značajke, te je broj neurona ulaznog sloja jednak broju ulaznih značajki. Broj neurona u skrivenom sloju mora biti dovoljno velik da bi mreža mogla naučiti traženu funkciju, a ipak je potrebno da bude dovoljno nizak da bi brzina konvergencije bila prihvatljiva. U pravilu je u izvedenim neuronskim mrežama veličina skrivenog sloja približno jednaka trećini veličine ulaznog sloja neurona.

Broj izlaznih neurona u mrežama za detekciju znaka je 1. Zbog odabira sigmoidalne prijenosne funkcije, izlaz iz mreže je realan broj između 0 i 1. Ako je izlaz veći od određene granice k (primjerice k = 0), promatranu sliku proglašujemo znakom. Izlazne vrijednosti primjera za učenje postavljamo na 1, ako je primjerak znak, a na 0 inače.

Broj izlaznih neurona u mrežama za klasifikaciju znaka je jednak broju klasa. Odabrana klasa znaka je redni broj onog izlaznog neurona čija je izlazna vrijednost najveća. Izlazne vrijednosti primjera za učenje se postavljaju u skladu s time: izlazna vrijednost neurona koji predstavlja klasu primjerka postavlja se na 1, dok se izlazne vrijednosti ostalih neurona postavljaju na 0.

Zbog prirode sigmoidalne funkcije, očekivanja vrijednosti ulaznih neurona normiraju se oko 0, na temelju skupa za učenje. Nakon toga vrijednosti se skaliraju u ovisnosti o zadanom rasponu.

Početne vrijednosti težina neuronske mreže se postavljaju na slučajne brojeve između -1 i 1. Ovakvo postavljanje početnih vrijednosti težina i normiranje ulaza osiguravaju rad prijenosne funkcije (2) u prijelaznom području.

6 Algoritam Viole i Jonesa

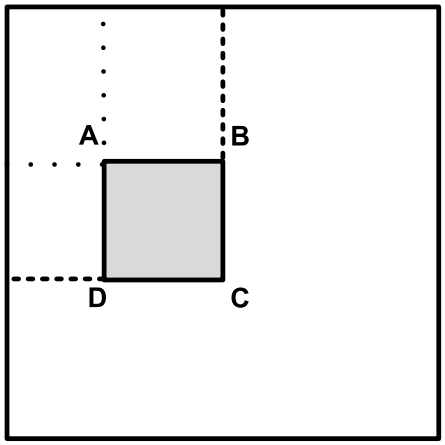
Algoritam Viole i Jonesa je generički postupak za detekciju objekata na slici koji se odlikuje veoma brzom detekcijom, a unatoč tome osigurava visok postotak detekcije. Danas je jedan od najšire korištenih te najefikasnijih algoritama za detekciju objekata na slici. Zbog svoje brzine često se primjenjuje u analizi video sekvenci u stvarnom vremenu.

6.1 Integralna slika

Integralna slika je pomoćni alat za brzi izračun Haarovih značajki (Slika 3.10). Svaka točka integralne slike sadrži sumu svih slikovnih elemenata koji se nalaze iznad i lijevo od pozicije točke. Time je omogućen izračun sume slikovnih elemenata u određenom pravokutnom području. Primjerice osjenčan prostor sa slike (Slika 6.1) moguće je dobiti formulom

$$S = I(C) - I(B) - I(D) + I(A)$$

gdje $I(X)$ predstavlja vrijednost integralne slike u točki X.



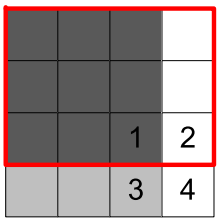
Slika 6.1 Prikaz korištenja integralne slike za izračun sume slikovnih elemenata

Izračun Haarovih značajki sastoji se zbroja i razlike različitih pravokutnih područja čiju je sumu moguće dobiti upravo pomoću integralne slike. Za svaki okvir video sekvence se jednom unaprijed izračuna integralna slika. Zatim je integralnu sliku moguće koristiti prilikom izračuna bilo koje Haarove značajke neovisno o njenom položaju i veličini.

Izračun integralne slike moguće je provesti jednim prolazom kroz sve slikovne elemente, primjerice ukoliko želimo izračunati integralnu sliku u točki 4, a poznajemo integralnu sliku u točkama 1, 2, 3 moguće je primijeniti sljedeću relaciju (Slika 3.10)

$$I(4) = I(2) + I(3) - I(1) + i(4)$$

gdje je s I označena integralna slika, a s i izvorna slika.



Slika 6.2 Prikaz izračuna integralne slike

Postavlja se pitanje logičke opravdanosti ovakvog odabira značajki. Naime može se primijetiti da Haarove značajke vjerno opisuju pojavu ruba u slici. Primjerice prva značajka sa slike 3.10 opisuje pojavu vertikalnog ruba.

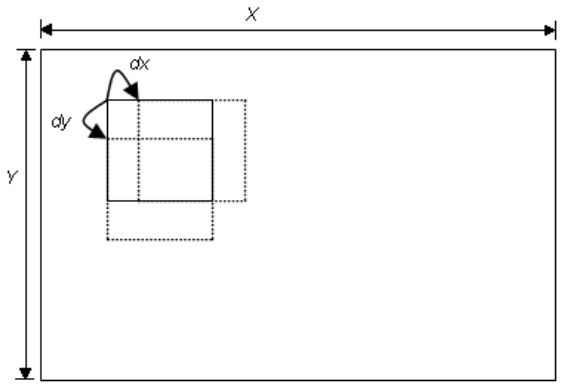
Glavna prednost korištenja Haarovih značajki je brzina njihovog izračuna te se stoga koriste prilikom detekcije objekata u slici u algoritmu Viole i Jonesa koji je opisan u nastavku.

6.2 Detekcija objekata na slici

Najjednostavniji način detekcije objekata u slici je detekcija pomičnim prozorom. Slika 6.3 prikazuje pomicanje posmičnog prozora po slikovnoj ravnini. Prozor se pomiče u oba smjera s pomacima u iznosu dx i dy slikovnih elemenata, te je za svako dobiveno okno potrebno procijeniti da li je ono objekt od interesa.

Širina i visina slike su označeni s X i Y. Jednostavnim izračunom moguće je izračunati ukupan broj različitih prozora koje je potrebno ispitati (Slika 6.3):

$$N = \frac{X}{dx} * \frac{Y}{dy}$$



Slika 6.3 Prikaz detekcije metodom posmičnog okna

S obzirom da u većini primjena nije poznata veličina objekta kojeg je potrebno pronaći u slici potrebno je provesti opisani postupak za različite veličine početnog okna. Neka je faktorom s označen

faktor uvećanja (*eng. scale factor)* koji određuje koliko je svako sljedeće okno veće od prethodnog, a s m i M označeno minimalna odnosno maksimalna veličina objekta u slici kojeg je potrebno pronaći.

Uzevši u obzir ove pretpostavke moguće je izračunati ukupan broj prozora koje je potrebno klasificirati u dvije grupe: objekt od interesa ili pozadina.

$$N = \frac{X}{dx} * \frac{Y}{dy} * \log_s \frac{M}{m}$$

U ovom trenutku moguće je odrediti N za veličinu slike od $640x480$ slikovnih elemenata, najmanju moguću veličinu objekta $m = 24$ slikovna elementa te najveću veličinu u iznosu $M = 200$ slikovnih elemenata, te parametre $s = 1.1,\ dx = 1,\ dy = 1$.

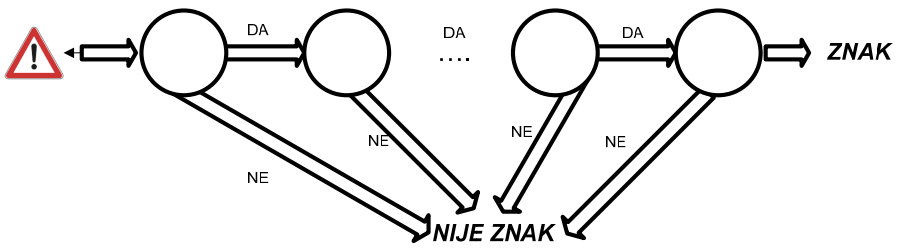
$$N \approx 6.8 * 10^6$$

Tipična video sekvenca sastoji se od 25 slike po sekundi, te broj dijelova slike koje je potrebno klasificirati u jednoj sekundi višestruko raste.
Ovaj primjer stoga ukazuje na visoku računalnu zahtjevnost ovakvog pristupa. Niti jedan od prije opisanih klasifikatora (*SVM* i *neuronska mreža*) ne zadovoljavaju ovaj uvjet. Također je važno da je udio lažno negativnih primjeraka vrlo nizak s obzirom na potreban broj klasifikacija za svaki okvir video sekvence.

Oba navedena problema moguće je riješiti kaskadom klasifikatora koja je opisana u sljedećem odjeljku.

6.2.1 Kaskada klasifikatora

U općenitom slučaju za klasifikaciju je moguće istovremeno koristiti veći broj klasifikatora te njihove rezultate ujediniti u jedinstveni odgovor zajedničkog klasifikatora. Tipični načini izgradnje zajednice klasifikatora se nazivaju: *boosting, bagging i cascading.*
U ovom poglavlju biti će posebno opisana metoda kaskadiranja (*eng. cascading*) upravo na način na koji se ona koristi u algoritmu Viole i Jonesa.



Slika 6.4 Prikaz kaskade klasifikatora

Slika 6.4 prikazuje kaskadu klasifikatora. Svaki od klasifikatora na svoj ulaz odgovara s da ili ne. Ukoliko je odgovor ne tada je ulaz odmah označen kao ne-znak, a ukoliko je odgovor *da* ulaz se prosljeđuje na sljedeću razinu kaskade koja ponovno odlučuje. Ulaz je proglašen znakom ako zadovoljava sve razine kaskade. Upravo ovaj način izgradnje kaskade omogućuje brzo pretraživanje slike, jer dijelovi slike koji nisu znakovi gotovo uvijek će biti odbačeni već na prvih nekoliko razina, a tek manji broj potencijalnih znakova će doći do krajnjih razina kaskade.

Iz gornjeg razmatranja, uz pretpostavku nezavisnosti klasifikatora, moguće je izvesti formule za postotak detekcije (*eng. detection rate*) i lažno pozitivnih primjera (*eng. false positive*):

$$F = \prod_{i=1}^{K} f_i$$

$$D = \prod_{i=1}^{K} d_i$$

gdje je f_i broj lažno pozitivnih primjera za *i-tu* razinu kaskade, a d_i postotak detekcije za *i-tu* razinu.

Broj lažno pozitivnih primjera pada s povećanjem razine kaskade, ali se ujedno i smanjuje postotak detekcije. Primjerice, ukoliko se želi postići broj lažno pozitivnih primjera 10^{-6}, a klasifikatori koji grade kaskadu imaju visok postotak detekcije i postotak lažno negativnih primjera, primjerice $d = 0.99, f = 0.33$ trebamo spojiti u kaskadu ukupno 13 klasifikatora. Time se dobije:

$$D = d^{13} = 0.877;\ F = f^{13} = 5.5 * 10^{-7}$$

Opravdano je dovesti u sumnju pretpostavku o nezavisnosti klasifikatora koja je uzeta prilikom provedenih izračuna. Pretpostavka ne vrijedi u općem slučaju izgradnje kaskade, ali je ovdje navodimo jer vrijedi za algoritam Viole i Jonesa, zbog posebnog načina izgradnje kaskade, koji je opisan u kasnijem tekstu.

6.3 Boosting

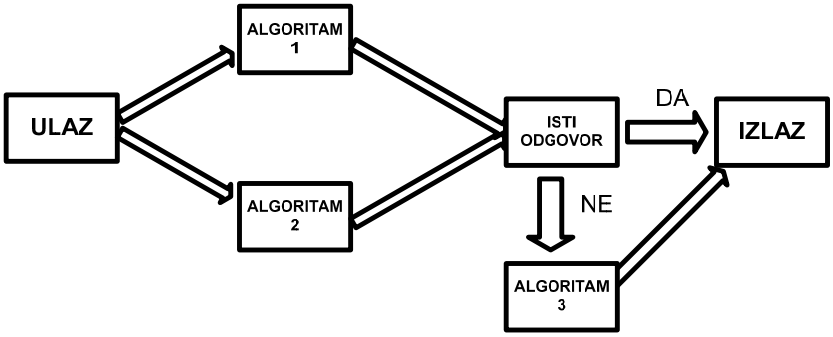
Boosting je metoda strojnog učenja čiji je cilj od većeg broja slabih[5] klasifikatora formirati jedan jaki klasifikator[6]. Primjerice razmotrimo sliku (Slika 6.5), skup ulaznih značajki se prosljeđuje na ulaz prvih dvaju algoritama. Ukoliko oba algoritma daju isti odgovor tada izlaz tih algoritama možemo proglasiti konačnim izlazom. Ukoliko se dva algoritma ne slažu oko rješenja, konačnu odluku donosi treći algoritam. Ovakvom kombinacijom algoritama očekujemo da će konačni rezultat biti bolji nego

[5] Termin slabi klasifikator se koristi za klasifikator s visokom pogreškom klasifikacije

[6] Termin jaki klasifikator označava pouzdan klasifikator s malom pogreškom

pojedinačni rezultati klasifikatora. Ovaj algoritam razvio je Schapire 1990 godine. Da bi se ovakvom kombinacijom klasifikatora postigli bolji rezultati, potrebno je osigurati njihovu statističku nezavisnost.

Schapire je predložio metodu podjele skupa za učenje X u tri disjunktna skupa X_1, X_2, X_3; $X_1 \cup X_2 \cup X_3 = X$. Prvi algoritam uči na skupu X_1, drugi na skupu pogrešaka prvog algoritma na skupu X_1 i jednakom broju točnih klasifikacija prvog algoritma na skupu X_2. Konačno, treći algoritam uči na dijelu skupa X_3 na kojem se algoritmi 1 i 2 ne slažu. Schapire je formalno dokazao da se ovakvim kombiniranjem klasifikatora smanjuje pogreška klasifikacije, te da se pogreška može proizvoljno smanjiti rekurzivnim kombiniranjem ovako izgrađenih klasifikatora.



Slika 6.5 Prikaz izvornog boosting algoritma

Iako se opisana metoda pokazala uspješnom u praksi, njena negativna strana je što zahtijeva velik skup primjeraka za učenje. Freund i Schapire su 1996. predložili izmjenu izvornog *boosting* algoritma za proizvoljno velik broj klasifikatora koji za učenje cijele kombinacije koristi isti skup. Time je potreba za velikim skupom za učenje izbjegnuta. Do danas je predstavljen veoma velik broj različitih algoritama za boosting: *AdaBoost, GentleBoost, RankBoost, BrownBoost, CoBosting, LPBoost*[20]. U ovom radu ograničit ćemo se na izvorni *AdaBoost.M1* postupak.

6.3.1 AdaBoost algoritam

Neka je s $(\boldsymbol{x}^t, r^t)$ označen jedan primjerak iz skupa za učenje. $\boldsymbol{x}^t$ označava vektor značajki koji pripadaju tom uzorku, a r^t njegovu ispravnu klasifikaciju. Osnova ideja algoritma je modulirati vjerojatnosti izvlačenja pojedinog uzorka iz skupa uzoraka za učenje. Neka je ta vjerojatnost označena s p_j^t gdje j označava redni broj slabog klasifikatora.

Početne vjerojatnosti su sve jednake i iznose $p_1^t = \frac{1}{N}$, gdje je N veličina skupa za učenje. Zatim počevši od $j = 1$ dodaje se novi slabi klasifikator u skup. Sa ε_j neka je označena pogreška j-tog klasifikatora. Pogreška se definira na skupu uzoraka moduliranih sa vjerojatnostima p_j^t. Pogreška mora biti strogo manja od $\frac{1}{2}$, a ukoliko nije zaustavlja se učenje klasifikatora. Definirajmo $\beta_j = \frac{\varepsilon_j}{1-\varepsilon_j} < 1$. Vjerojatnost svakog točno klasificiranog primjera smanjujemo faktorom β_j odnosno vrijedi $p_{j+1}^t = \beta_j p_j^t$,

inače vjerojatnost ostaje jednaka kao i u trenutnom koraku, odnosno vrijedi: $p_{j+1}^t = p_j^t$. S obzirom da je p_j^t funkcija gustoće vjerojatnosti potrebno ju je normalizirati tako da vrijedi da je ukupna suma jednaka 1, odnosno $p_{j+1}^t = \frac{p_{j+1}^t}{\sum_{i=1}^n p_{j+1}^i}$. Ovim postupkom se vjerojatnost točno klasificiranih primjera smanjuje, a vjerojatnost krivo klasificiranih primjera povećava. Time se sljedeći klasifikatori više fokusiraju na pogreške prethodnih razina te time popravljaju ukupan rezultat klasifikacije. Tablica 6.1 prikazuje pseudokod *AdaBoost.M1* algoritma [17].

Tablica 6.1 Algoritamski zapis AdaBoost algoritma

Učenje klasifikatora:

Za svaki $\{(\boldsymbol{x}^t, r^t)\} \in X$, postavi $p_1^t = \frac{1}{N}$

Za sve slabe klasifikatore $j = 1 \ldots L$

Slučajno odaberi podskup X_j iz X s vjerojatnostima p_j^t

Treniraj klasifikator d_j na skupu X_j.

Za svaki (x^t, r^t) klasificiraj uzorak $y_j^t = d_j(\boldsymbol{x}^t)$

Izračunaj pogrešku $\varepsilon_j = \sum_t p_j^t * \delta(y_j^t \neq r^t)$

Ako je $\varepsilon_j \geq \frac{1}{2}$ izađi

$\beta_j = \frac{\varepsilon_j}{1-\varepsilon_j}$

Za svaki $(\boldsymbol{x}^t, r^t)$ ukoliko je ispravno klasificiran: $p_{j+1}^t = \beta_j p_j^t$, inače $p_{j+1}^t = p_j^t$

Normaliziraj vjerojatnosti $p_{j+1}^t = \frac{p_{j+1}^t}{\sum_{i=1}^n p_{j+1}^i}$

Težina w_j klasifikatora je $w_j = \log(\frac{1}{\beta_j})$

Testiranje:

Za dani vektor značajki $\boldsymbol{x}$ izračunaj $d_j(\boldsymbol{x})$ za $j = 1 \ldots L$

Izračunaj ukupan izlaz $j = 1 \ldots K$

$$y_i = \sum_{j=1}^{L} w_j * d_{ji}(\boldsymbol{x})$$

Klasifikacija se izvodi jednostavnom linearnom kombinacijom izlaza svakog pojedinog klasifikatora u kaskadi. Težina pojedinog klasifikatora označena je s $w_j = \log(\frac{1}{\beta_j})$ te raste s padom

greške pojedinog klasifikatora. Schapire (1998.) je pokazao da AdaBoost optimira marginu razdvajanja na sličan način kao i SVM [18]. Ukoliko se margina razdvajanja poveća, uzorci za učenje se bolje razdvajaju te je time greška manje vjerojatna.

Logično je zapitati se u kojim slučajevima AdaBoost može popraviti performanse klasifikatora. Ukoliko se primjenjuje nad kombinacijom jakih linearnih klasifikatora, primjerice SVM-a, svaki sljedeći klasifikator će učiti samo nad malim brojem uzoraka koji sadrže visoku razinu šuma, stoga dodavanje novih klasifikatora u kaskadu ne bi utjecalo na povećanje performansi. Stoga je AdaBoost idealno koristiti nad slabim i slabo koreliranim klasifikatorima visoke varijance. Neka je σ^2 varijanca pojedinog klasifikatora, σ_u^2 ukupna varijanca njihove linearne kombinacije, a težina pojedinog klasifikatora neka je zbog jednostavnosti $w_i = \frac{1}{L}$, slijedi:

$$\sigma_u^2 = D\left(\sum(1/L * d_i)\right) = \frac{L}{L^2} D(d_i) = \frac{1}{L} * \sigma^2$$

te je ukupna varijanca zajednice klasifikatora manja nego varijanca jednog klasifikatora.

6.4 Algoritam Viole i Jonesa

Algoritam Viole i Jonesa je najpoznatiji i najkorišteniji detektor objekata na slici. Njegove ključne odlike su brzina koja omogućava detekciju na videu u stvarnom vremenu te njegova pouzdanost i otpornost na šum.

6.4.1 Pregled algoritma

Algoritam koristi integralnu sliku za izračun Haarovih značajki korištenih u klasifikaciji. Izgrađuje se kaskada klasifikatora od kojih svaki uči *AdaBoost* algoritmom na greškama prethodnih razina. *AdaBoost* odabire manji broj najboljih značajki koje diskriminiraju između traženog objekta i pozadine te time omogućava vrlo visoku brzinu izvršavanja. Parametri svake razine kaskade su odabrani na način da imaju izuzetno visok postotak detekcije, primjerice *0.999*, ali istovremeno broj lažno negativnih razmjerno visok, primjerice *0.5*. Kaskadiranjem klasifikatora na način opisan u prethodnim poglavljima postepeno se smanjuje postotak detekcije, ali značajno brže pada postotak lažno negativnih primjera.

6.4.2 Ulazne značajke

Tipično se ulazne slike skaliraju na veličinu 24x24 slikovna elementa, te se zatim izračunava potpun skup Haarovih značajki nad tako dobivenim slikama. Ukoliko se slika sastoji od više kanala značajke se izračunavaju na svakom kanalu zasebno. Ovakav potpun skup Haarovih značajki može biti vrlo velik, primjerice gotovo 10^6 značajki.

6.4.3 Jedna razina klasifikatora

Prvi korak prilikom izgradnje jedne razine klasifikatora je evaluacija svih značajki na svim slikama iz skupa za učenje. Glavni zadatak ove faze izgradnje konačnog klasifikatora je odabir ključnih značajki iz potpunog skupa Haarovih značajki. U ovoj fazi svakoj stvorenoj značajki pridajemo prag θ(*threshold*) i polaritet p (*polarity*). Ideja ja da se od svake značajke može stvoriti jedan slabi klasifikator na sljedeći način:

$$h(x, f, p, \theta) = \begin{cases} 1, pf(x) < p\theta \\ 0, inače \end{cases}$$

Odabir najboljeg praga i polariteta za pojedini klasifikator biti će jedan od zadataka AdaBoost algoritma. Prag i polaritet se odabiru na način da minimiziraju grešku klasifikacije svakog pojedinog slabog klasifikatora.

AdaBoost algoritam, opisan u prethodnom poglavlju možemo shvatiti kao pohlepan algoritam koji odabire najbolje slabe klasifikatore te formira jedan konačan jaki klasifikator. S obzirom na izmjene koje je potrebno napraviti na izvornom AdaBoost algoritmu, u tablici (Tablica 6.2) naveden je pseudokod inačice koja se koristi u algoritmu Viole i Jonesa.

Tablica 6.2 Algoritamski zapis AdaBoost algoritma korištenog u algoritmu Viole i Jonesa

Učenje klasifikatora:

Za zadane $\{(\boldsymbol{x}^t, r^t)\} \in X$, gdje $r^t = 1$ za pozitivne uzorke i $r^t = 0$ za negativne

- *M je broj negativnih primjera, L broj pozitivnih*

Inicijaliziraj težine: $w_{1,i} = \frac{1}{2M}$ za negativne uzorke i $w_{1,i} = \frac{1}{2L}$ za pozitivne

Za *t = 1, … , T*

1. Normaliziraj težine: $w_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$
2. Odaberi najbolji slabi klasifikator s obzirom na pogrešku klasifikacije
   - $\varepsilon_t = \min_{p,f,\theta} \sum_i w_i |h(x_i, f, p, \theta) - y_i|$
   - $\beta_j = \frac{\varepsilon_j}{1-\varepsilon_j}$
3. Definirajmo $h_t(x) = h(x_i, f_t, p_t, \theta_t)$ tako da f_t, p_t, θ_t minimiziraju ε_t
4. Osvježi težine:
   - $w_{t+1,i} = w_{t,i} * \beta_t^{1-e_i}$, gdje je $e_i = 1$ ako je uzorak krivo klasificiran, odnosno $e_i = 0$ ako je uzorak ispravno klasificiran

Testiranje:

Rezultantni jaki klasifikator je:

$$C(x) = \begin{cases} 1, \sum_{t=1}^{T} \alpha_t * h_t(x) > \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0, inače \end{cases}, \text{gdje je } \alpha_t = \log \frac{1}{\beta_t}$$

Ovaj korak je računalno vrlo zahtjevan jer je tipična veličina ulaznog skupa slika nekoliko tisuća uzoraka, što pomnoženo s brojem značajki rezultira velikim zahtjevima za memorijom i procesorskom snagom.

Posebnu pažnju potrebno je posvetiti drugom koraku opisanog algoritma, s obzirom da je on računalno najzahtjevniji. Opis efikasne implementacije opisanog koraka može se pronaći u izvornom članku P. Viole i M. Jonesa [21].

6.4.4 Izgradnja kaskade

Razine kaskade se grade treniranjem slabih klasifikatora AdaBoost algoritmom. Zatim se obavlja postupak promjene finalnog praga AdaBoost algoritma u svrhu promjena odnosa između postotka detekcije i postotka lažno pozitivnih primjera. Naime, postotak detekcije se tipično postavlja na preko 99%, dok broj lažno pozitivnih primjera tipično iznosi oko 30%. Cilj je izgraditi kaskadu koja će imati što jednostavnije početne razine tako da bude računalno efikasna.

Najčešće se prilikom učenja algoritma odabire postotak detekcije i postotak lažno pozitivnih primjera jedne razine kaskade, te ukupan postotak lažno pozitivnih. Algoritam zatim sam određuje potreban broj značajki koje će se koristiti u svakoj razini na način da zadovolji postavljene parametre.

Važno je napomenuti da svaka sljedeća razina kaskade uči na lažno pozitivnim primjerima prethodnih razina. Prilikom izgradnje kaskade negativni primjeri se izrezuju iz slika u kojima nema objekata od interesa. Izrezivanje negativnih primjera za krajnje razine kaskade je također računalno zahtjevan zadatak, jer je potrebno obraditi izuzetno velik broj slika da se prikupi dovoljan broj lažno negativnih primjera. Pozitivni primjeri za učenje su isti za svaku razinu kaskade. Također je važno napomenuti da skup pozitivnih i negativnih primjera treba razdvojiti na dva disjunktna skupa; skup za učenje te skup za validaciju. Skup za učenje koristi se prilikom učenja AdaBoost algoritma, dok se skup za validaciju koristi kod određivanja postotka detekcije i postotka lažno negativnih primjera.

Tablica 6.3 Algoritamski zapis Viola i Jones postupka

Izgradnja kaskade:

Odabire se:

- f - postotak lažno pozitivnih primjera na jednoj razini kaskade(maksimalni dopušteni)
- d – postotak detekcije jedne razine kaskade (minimalni dopušteni)
- F_{target} – postotak lažno pozitivnih primjera cijele kaskade

P je skup pozitivnih primjera, N je skup negativnih primjera

$F_0 = 1.0; D_0 = 1.0$

$i = 0$

Dok je $F_i > F_{target}$

$i = i + 1$

$n_i = 0; F_i = F_i + 1$

Dok je $F_i > f * F_{i-1}$

$n_i = n_i + 1$

Treniraj klasifikator sa n_i značajki na slikama P, N pomoću AdaBoost algoritma

Smanji prag dok klasifikator postotak detekcije ne postane barem $d * D_{i-1}$ te odredi F_i

$N = 0$

Ako je $F_i > F_{target}$ onda evaluiraj kaskadu na skupu negativnih slika i sve krive detekcije postavi u skup N

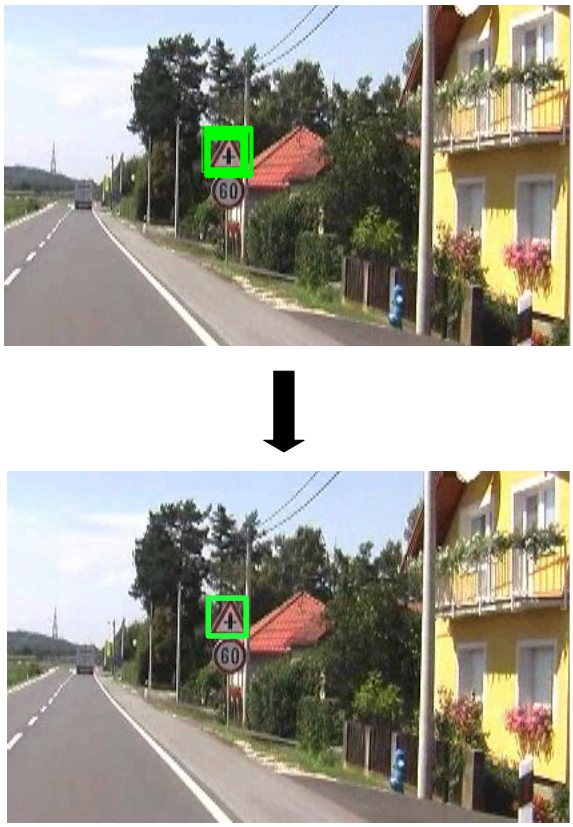
6.4.5 Detekcija

Prvi korak prilikom postupka detekcije objekta je izgradnja integralne slike. U prethodnim poglavljima opisan je način izgradnje kaskade klasifikatora. Naučeni klasifikator se zbog svojstava integralne slike može na jednostavan način pomicati preko slike, na sve postojeće koordinate slikovne ravnine.

Osim translacije sam klasifikator je moguće i skalirati. Naime kod većine drugih klasifikatora bilo bi potrebno skalirati ulaznu sliku na ispravnu veličinu i zatim provesti klasifikaciju, ali takav pristup bi prilikom detekcije objekata bio suviše računalno zahtjevan. Stoga izgrađena kaskada, osim što je neovisna na translaciju, ima i mogućnost skaliranja. Naime, s obzirom da se svaka značajka izračunava kao linearna kombinacija nekoliko elemenata integralne slike moguće je jednostavnim skaliranjem koordinata koje treba zbrojiti skalirati i sam klasifikator, pri tome posebna pažnja mora biti posvećena

pravilnom skaliranju praga svakog slabog klasifikatora. Na ovaj način jednostavnim pretraživanjem slike moguće je pronaći sve objekta od interesa.

Vrlo često se događa da je jedan te isti objekt detektiran nekoliko puta jer se okna za pretraživanje međusobno preklapaju, stoga je potrebno na izlaz Viola i Jones algoritma primijeniti dodatnu algoritamsku logiku za eliminaciju preklapanja. Čak se često, ukoliko je cilj smanjiti postotak lažno negativnih primjera, objekt smatra točnom detekcijom samo ako je više puta detektiran uz određeni pomak ili skaliranje.



Slika 6.6 Prikaz rada algoritma za određivanje rezultantnog okna

U ovom radu korišten je *UnionFind*[7] algoritam. Osnovna je ideja upariti detekcije koje sigurno međusobno odgovaraju, metodom preklapanja površina. Ukoliko je postotak preklapanja detekcija visok tada oni sigurno odgovaraju istom objektu. Zatim se, primjenom *UnionFind* strukture, detekcije uparuju u međusobno vezane skupove, a kao predstavnika uzima se prosjek svih elemenata skupa. Amortizirana složenost opisanog algoritma je $O(n^2 * \alpha(n))$, gdje je s $\alpha(n)$ označena inverzna Ackermanova funkcija.

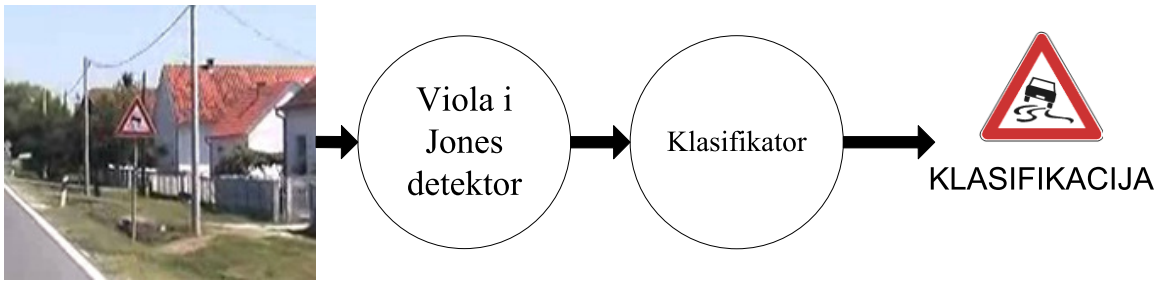
[7] Union find se u literaturi još često naziva: *Disjoint-set data structure.* Algoritam je u svojoj osnovnoj inačici detaljno opisan u T. H. Cormen, Leiserson, Rivest, Stein: Introduction to Algorithms[24].

Vrlo često se $\alpha(n)$ smatra konstantom s obzirom da je $\alpha(n) < 5$ za sve praktične vrijednosti n. S obzirom da je jedna od najvažnijih odlika Viola i Jones algoritma njegova brzina, stoga je posvećena posebna pozornost da i ova komponenta sustava radi na najbrži mogući način. Slika 6.6 prikazuje primjer rada opisanog algoritma.

Najjednostavnija moguća primjena algoritma Viole i Jonesa na detekciju objekata u video sekvencama je evaluacija cjelokupne kaskade na cijelom slikovnom okviru. Na prvi pogled ovakav postupak je pretjerano zahtjevan, s obzirom da su promjene između dvaju uzastopnih okvira malene, te da stoga nije potrebno uvijek pretraživati cijelu sliku. Međutim upravo ovakav pristup donosi mnoge prednosti jer je moguće značajno smanjiti broj lažno pozitivnih primjera te povećati postotak detekcije upravo primjenom algoritamskih postupaka nad izlazima detekcije za svaki pojedini slikovni okvir. Taj postupak je opisan u daljnjem tekstu.

7 Realizacija

U prethodnim poglavljima izložena je teoretska podloga metoda za detekciju i raspoznavanje objekata u slici. Realizirani sustav većim se dijelom oslanja na iste. Kroz ovo poglavlje iznesen je koncept sustava, podaci o procesima te usporedba klasifikatora. Predstavljeni su uočeni problemi te pristup rješavanju istih. Naposljetku je predstavljena konačna inačica sustava[8].



Slika 7.1 Osnovni koncept sustava

7.1 Učenje

Svi korišteni algoritmi pripadaju kategoriji algoritama nadziranog učenja. Stoga im je potreban skup za učenje, koji se sastoji od uzoraka i pripadnog točnog rješenja. U nastavku je detaljnije opisano učenje pojedinih algoritama.

7.1.1 Učenje algoritma Viole i Jonesa

Viola i Jones algoritam učen je na skupu od 1802 znaka. Nezavisno su generirane Haarove značajke na sve tri komponentne *LAB* sustava boja. Rezultantna kaskada se sastoji od 16 razina, a prva razina se sastoji od tri značajke, što je bitno za brzinu izvođenja. Negativni primjerci za svaku razinu kaskadu se dobiju tako da se pretraže slike za učenje na kojima prethodne razine kaskade daju lažno pozitivnu detekciju.

Parametri korišteni prilikom učenja iznose:

- postotak lažno negativnih po razini: 0.35
- ukupan postotak lažno negativnih: 10^{-7}
- postotak detekcije na jednoj razini: 0.998

[8] Kroz poglavlje je predstavljen koncept sustava. Opis same implementacije se nalazi u dodatku A.

7.1.2 Učenje neuronske mreže i stroja s potpornim vektorima

Učenje neuronske mreže i stroja s potpornim vektorima odvija se na način opisan u prethodnim poglavljima. Kroz ovo poglavlje obrađuje se odabir ulaznih značajki i skupovi za učenje. U prvome dijelu izneseno je istraživanje o najboljem odabiru ulaznih značajki za detekciju i klasifikaciju. Iako je istraživanje provedeno korištenjem neuronskih mreža, odabrane značajke se koriste i kao ulaz stroja s potpornim vektorima. Isto tako, opisani skup za učenje se koristi i za neuronsku mrežu, i za stroj s potpornim vektorima.

Odabir ulaznih značajki

[9]Pri korištenju klasifikatora poput neuronske mreže, ili stroja s potpornim vektorima, potrebno je odrediti skup ulaznih značajki nad kojima će klasifikator raditi. U ovom poglavlju opisana je usporedba performansi sustava u odnosu na odabir značajki. Usporedba je provedena korištenjem neuronskih mreža. U skladu s rezultatima je odabran najbolji način izlučivanja značajki.

Detekcija znaka

Detekcija znaka u slici neuronskom mrežom ili strojem s potpornim vektorima nije praktična zbog brzine izvođenja. Ipak, takav detektor se može koristiti za dodatnu provjeru detekcija algoritma Viole i Jonesa, kada je potrebno provjeriti mali broj isječaka slike.

Za rješavanje problema detekcije znaka, poželjno je odabrati ulazne značajke po kojima se znak može lako razlikovati od okruženja. Prometni znakovi se ističu svojom bojom i oblikom, te su ulazne značajke odabrane u skladu s time.

Histogrami boje i zasićenja

Prvi način odabira ulaznih značajki temelji se na svojstvima znakova vezanim za boju. Slika se prije izlučivanja značajki skalira na veličinu 24x24.

Informaciju o boji slike dobro opisuje histogram boja obrađen u poglavlju 3.4. Pošto je u znakovima udio bijele boje također značajan, koristi se i histogramom S komponente (zasićenja) HSV

[9] Dijelovi poglavlja su preuzeti iz I. Kovaček: Sustav za detekciju i raspoznavanje prometnih znakova

sustava koji daje informaciju o udjelu bijele boje u slici. Oblik znaka se također uzima u obzir, te se histogrami izrađuju samo na dijelovima slike.

Slika 7.2 Odabir slikovnih elemenata

Slika 7.2 prikazuje područja nad kojima se izrađuju histogrami. Histogram boje se izrađuje nad slikovnim elementima unutar vanjskog ruba, dok se histogram zasićenja izrađuje na manjem području, na slikovnim elementima unutar unutrašnjeg ruba.

Veličina histograma boje i histograma zasićenja je 20. Time je veličina ulaznog sloja neuronske mreže 40, dok je veličina skrivenog sloja 32.

Histogrami orijentacije gradijenata

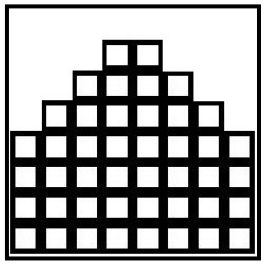
Drugi način izlučivanja značajki oslanja se na trokutasti oblik znakova. Histogrami orijentacije gradijenata (poglavlje 3.4) luminantne komponente dobro izdvajaju informaciju o rubovima slike, koji su istaknuti u slikama prometnih znakova.

Prije izlučivanja značajki, slika se skalira na veličinu 48x48, zatim se vrijednosti luminantne komponente normaliziraju na raspon 0 – 255. Nakon normalizacije, luminantna komponenta se filtrira Gaussovim filtrom:

$$f = \frac{1}{15} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad (7.1)$$

Filtriranjem slike Gaussovim filtrom postiže se uglađivanje rubova slike, te time smjer gradijenta postaje ravnomjerno raspoređen na rubovima, što je bitno pri izradi histograma orijentacije gradijenata.

Histogrami se izrađuju nad blokovima veličine 6x6, te su raspoređeni uzevši u obzir poziciju znaka u slici. Slika 7.3 prikazuje raspored blokova.



Slika 7.3 Raspored blokova

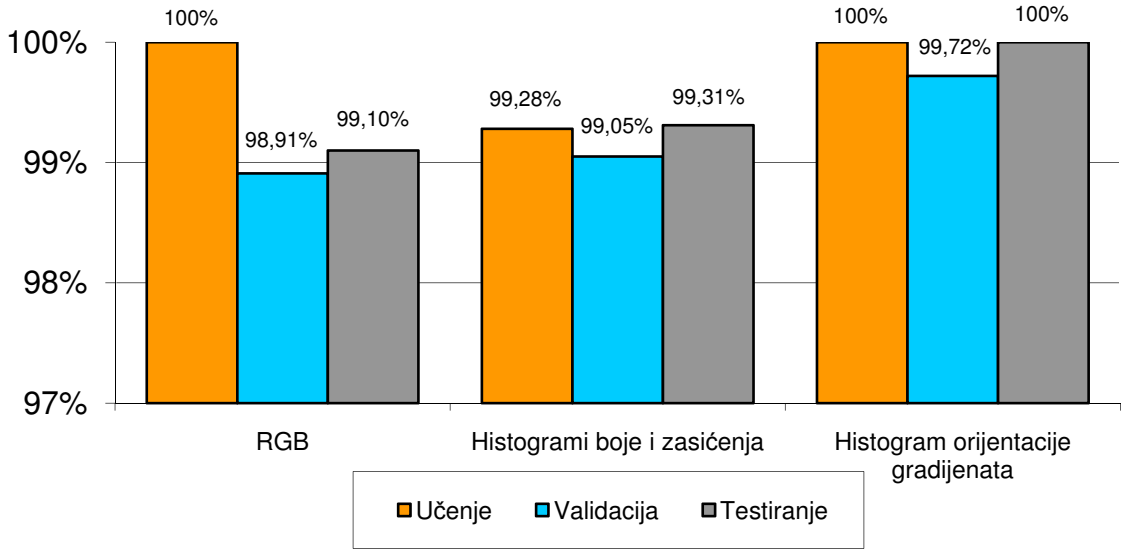
Histogrami se izrađuju za šest kuteva te uzimaju u obzir samo smjer, to jest, 180 stupnjeva. Veličina ulaznog sloja neuronske mreže je 264, dok je veličina skrivenog sloja 79.

Rezultati

Skupovi za učenje, validaciju i testiranje sastoje se od dvije vrste slika, znakova, i okoline. Sve slike su izabrane iz jednog skupa slika metodom slučajnog odabira. Početni skup se sastoji od slika izdvojenih iz video snimaka. Tablica 7.1 prikazuje podatke o veličini skupova za učenje, validaciju i testiranje.

Tablica 7.1. Veličine skupova za učenje,validaciju, i testiranje

	Učenje	Validacija	Testiranje
Znakovi	1426	353	348
Ostalo	1223	384	339



Slika 7.4. Usporedba rezultata za različit odabir značajki

Slika 7.4 prikazuje rezultate na skupovima za učenje, validaciju i testiranje, u ovisnosti o odabiru ulaznih značajki. Odabir ulaznih značajki utječe na performansu, ali manje nego što je očekivano. Razlog tomu leži u velikom skupu za učenje, te neuronska mreža može naučiti i generalizirati, iako je odabir značajki loš.

Najbolji rezultati se postižu na histogramima orijentacije gradijenata, te se stoga odabiru kao ulazne značajke za detekciju.

Klasifikacija znakova

Za razliku od detekcije znaka, pri klasifikaciji znakova poželjno je odabrati ulazne značajke koje dobro opisuju razliku između različitih klasa znakova. Informacija o razlici između klasa se nalazi na području središta znaka.

Luminantna komponenta

Prvi način odabira značajki je luminantna komponenta središta znaka. Slike se skaliraju na veličinu 24x24, zatim se izdvajaju središnji elementi slike, na način prikazan na slici 7.5.

Slika 7.5 Izdvajanje središnjeg dijela slike

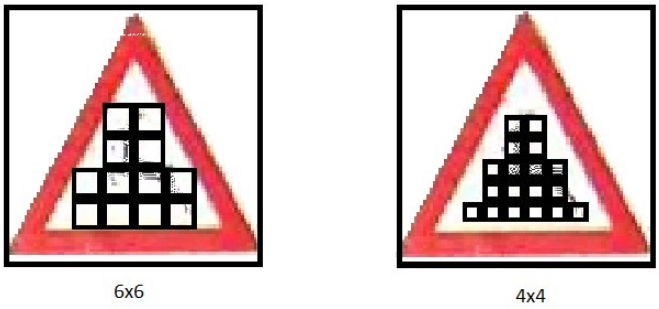
Dobivena luminantna komponenta dijela slike se zatim normalizira na raspon od 0 do 255.

Ukupan broj promatranih slikovnih elemenata je 112, te je toliki i broj neurona ulaznog sloja neuronske mreže. Broj neurona skrivenog sloja je 34.

Histogrami orijentacije gradijenata

Razlika između različitih klasa znakova je u oblicima koji se nalaze u središtu, te se histogram orijentacije gradijenata koji nosi informaciju o rubovima na središtu slike nameće kao logičan odabir ulaznih značajki.

Slike se skaliraju na veličinu 48x48, te se luminantna komponenta normalizira na raspon od 0 do 255. Nakon toga slijedi filtriranje Gaussovim filtrom (7.1). Zatim se izrađuju histogrami orijentacije gradijenata nad blokovima smještenim u središtu slike. Postoje dva skupa blokova, jedni veličine 4x4, te drugi veličine 6x6. Slika 7.6 prikazuje raspored blokova.



Slika 7.6 Raspored blokova

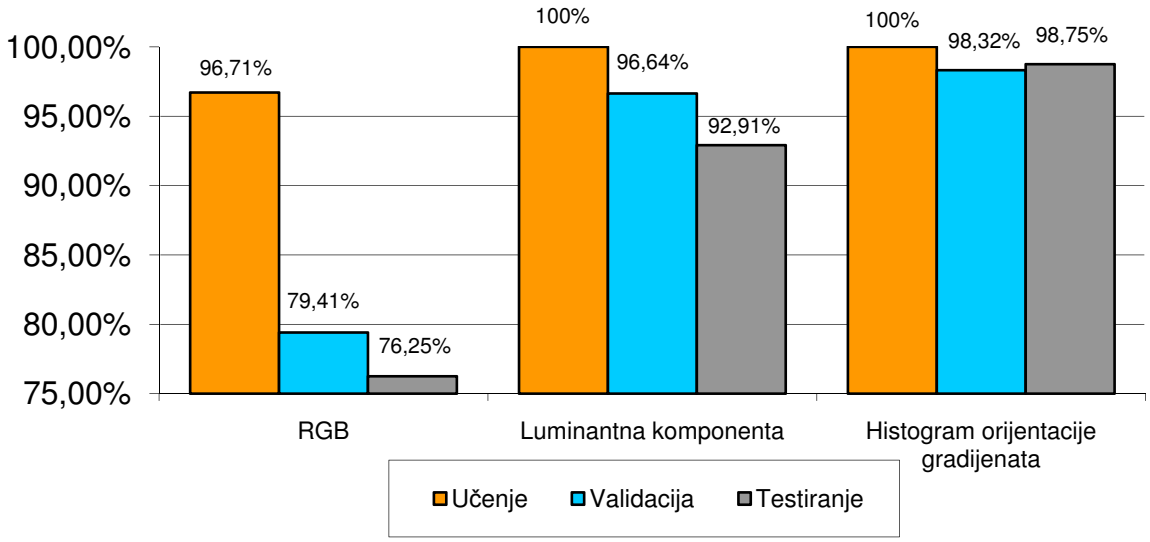
Histogrami nad blokovima veličine 6x6 se rade za četiri kuta i 180 stupnjeva, dok se histogrami nad blokovima veličine 4x4 rade za sedam kuteva i 360 stupnjeva. Broj neurona ulaznog sloja neuronske mreže je 174, te je broj neurona u skrivenom sloju jednak 52.

Rezultati

Skupovi za učenje, validaciju i testiranje odabrani su na isti način kao i skupovi za detekciju znaka. Odabrane su klase znakova čijih je primjeraka na raspolaganju više od 140. Takvih je klasa ukupno 8. Korištene klase prikazane su na slici 7.7. Skup za učenje sastoji se od 80 primjeraka po klasi, skup za validaciju od 30 primjeraka po klasi, te skup za testiranje od 30 primjeraka po klasi.

Slika 7.7 Korištene klase znakova.

Učenje je provedeno za tri načina odabira ulaznih značajki: RGB format, luminantna komponenta središta slike, te histogrami orijentacije gradijenata.

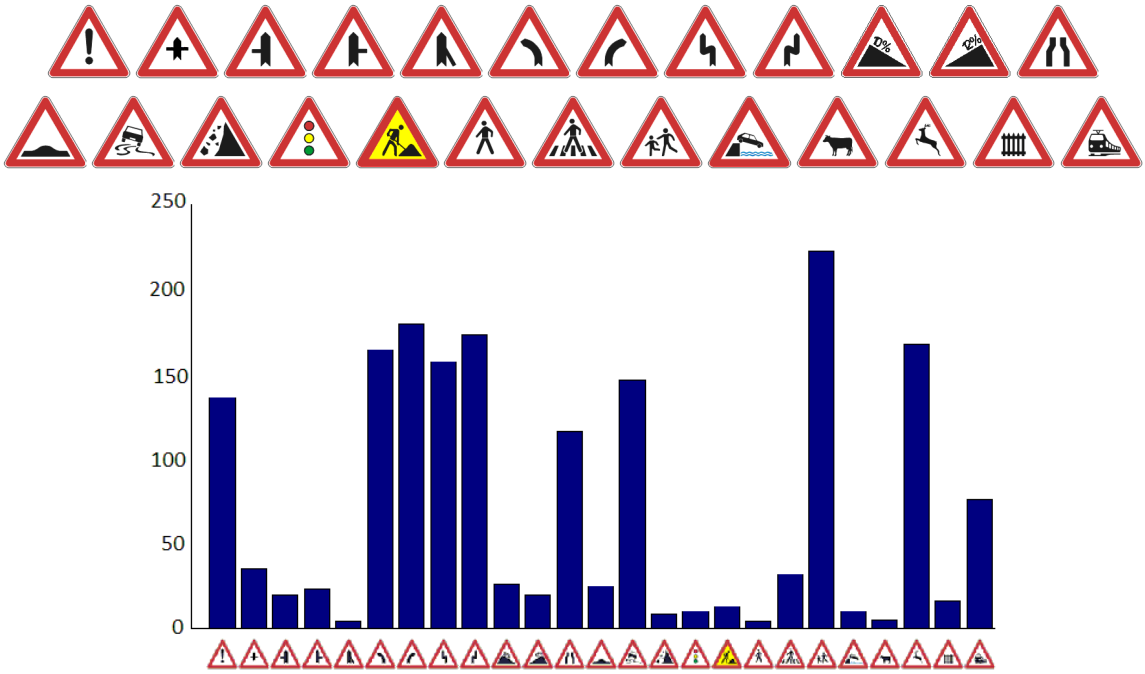


Slika 7.8 Rezultati klasifikacije u ovisnosti o odabiru značajki

Najbolji rezultat je postignut korištenjem histograma orijentacije gradijenata, te se on odabire kao skup ulaznih značajki za klasifikaciju.

Skup za učenje

Skup za učenje klasifikatora sastoji se od 1802 ručno označena isječka slike na kojima se nalaze znakovi. U skupu je zastupljeno 25 od ukupno 50 razreda znakova upozorenja. Pri tome je broj uzoraka po razredu različit, te se kreće od 4 do 236. Slika 7.9 prikazuje zastupljene razrede, i razdiobu broja primjeraka po razredima.



Slika 7.9 Zastupljeni razredi znakova i broj primjeraka po razredima

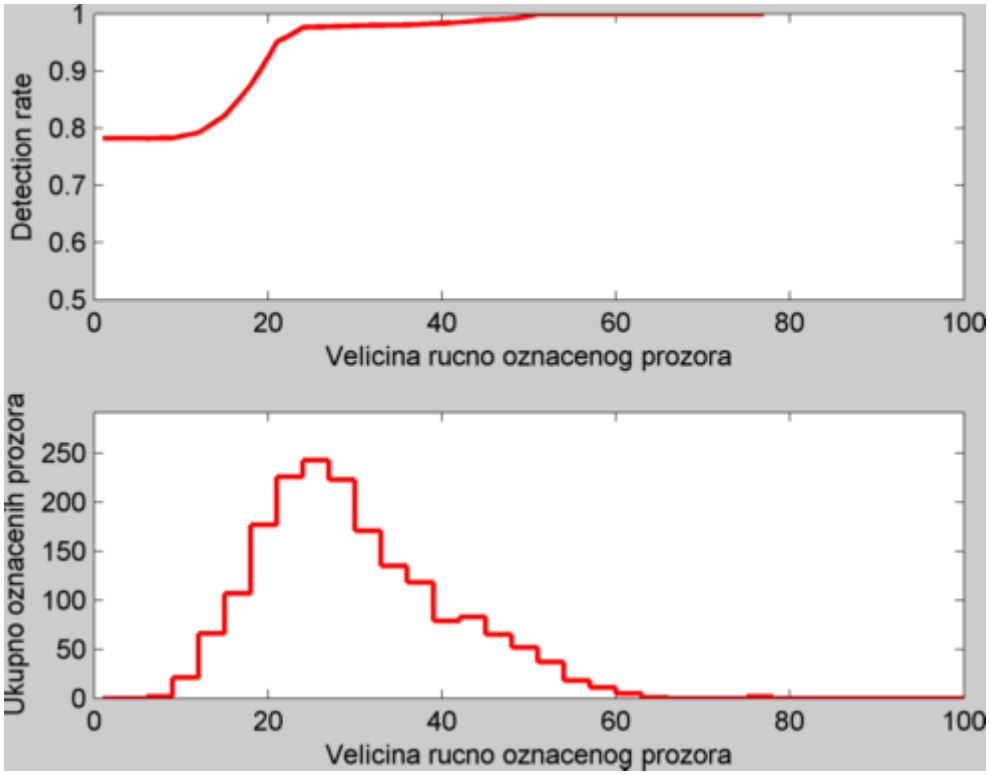
Pri učenju, kao i pri evaluaciji, izlučuju se značajke na način opisan u prethodnom poglavlju. Slike iz skupa su pod utjecajem smetnje uzrokovane preplitanjem, te se za njih provodi uklanjanje smetnje, na način opisan u poglavlju 3.1.1.

7.2 Rezultati i odabir klasifikatora

Proces detekcije i raspoznavanja ostvaren je na dva načina s ciljem usporedbe različitih klasifikatora. U oba se koristi algoritam Viole i Jonesa za detekciju objekata na slici, dok za klasifikaciju jedan koristi stroj s potpornim vektorima, a drugi neuronsku mrežu.

Uzorci za provedeno testiranje su 1842 slike iz video snimke. Na svakoj od slika se nalazi barem jedan prometni znak. Važno je napomenuti da su skupovi za učenje klasifikatora generirani na temelju ovih slika te rezultati testiranja nisu mjerodavni. Cilj ovakvog testiranja je identifikacija problema, čijim rješavanjem će se sustav unaprijediti.

Rezultati detekcije prikazani su na slici 7.10. Na gornjem grafu je prikazana uspješnost detekcije u ovisnosti o minimalnoj promatranoj veličini okvira[10]. Na donjem grafu je prikazana raspodjela veličine slike u skupu.



Slika 7.10 Rezultati detekcije

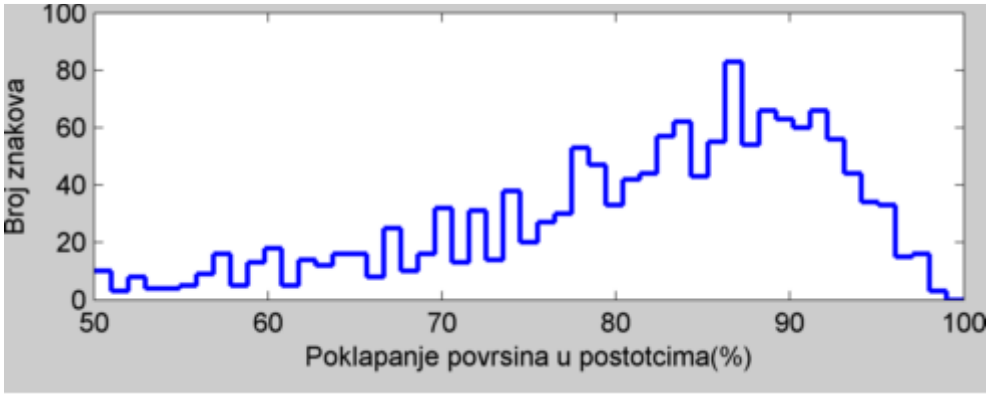
[10] Pojašnjenje: uspješnost detekcije ukoliko u testiranje ulaze samo oni znakovi veći od mjere na apscisi.

Uspješnost detekcije je 78.23%. Taj rezultat je prilično loš. Ipak, sa grafa se očitava da se uspješnost detekcije popravlja sa povećanjem veličine minimalnog promatranog okvira. Tako je za znakove veće od 24x24 uspješnost detekcije 97.67%. Ta činjenica omogućava solidne performanse na razini video snimke.

Niska razina detekcije uzrokovana je i činjenicom da su u slikama za testiranje izbačeni parni vertikalni i horizontalni redci, zbog smetnje uzrokovane preplitanjem. Time je velik broj znakova postao manji od 24x24, što se vidi sa donjeg grafa. Pošto je 24x24 najmanja veličina za koju algoritam Viole i Jonesa detektira znak, ovakav nizak rezultat ne prikazuje stvarnu situaciju, te je time opravdano promatrati rezultat za okvire veće od 24x24.

Na ukupno 1840 slika algoritam je imao 326 lažnih detekcija. Razina krivih detekcija je 17.69%. Ona predstavlja odnos krivih detekcija i broja znakova. Točnije, ovaj rezultat nam govori da se u periodu pojave 6 znakova pojavljuje 1 lažna detekcija. Očito je navedeni rezultat nedovoljno dobar za primjenu, te predstavlja veliki problem.

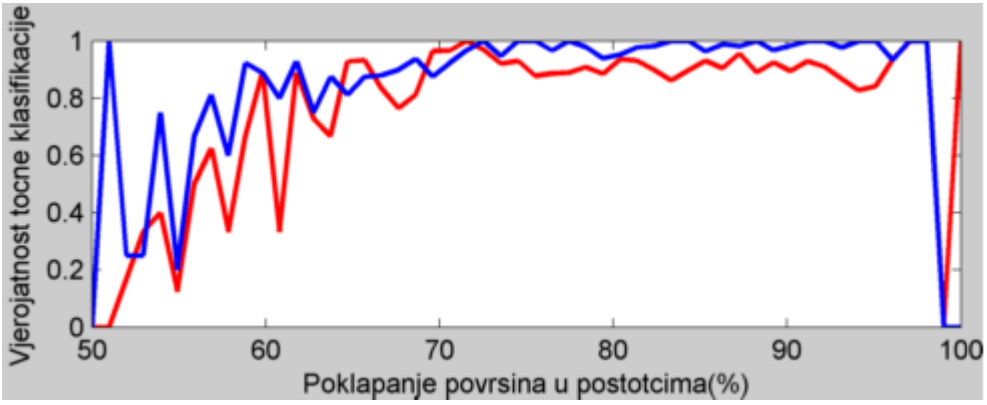
U svrhu određivanja kvalitete pozicioniranja za ispravne detekcije, uvedena je mjera preklapanja površina između označenog i detektiranog isječka. Definirana je kao udio površine preklapanja u većoj od površina. Slika 7.11 prikazuje razdiobu preklapanja površina za ispravne detekcije.



Slika 7.11 Razdioba kvalitete pozicioniranja

Sa slike se može vidjeti da je položaj većine detekcija odmaknut od označenog položaja znaka. Ipak, prikazani graf potrebno je promatrati sa oprezom. Naime, u obzir se mora uzeti greška ručnog označavanja znakova, te činjenica da je pozicioniranje prilično dobro već za preklapanje od 90%, jer greška raste kvadratno sa kombinacijom vertikalnog i horizontalnog odmaka. Ipak postoji udio detekcija sa relativno malom površinom preklapanja, to jest, nepreciznim pozicioniranjem. Za očekivati je da će takva pozicioniranja imati negativan utjecaj na uspješnost klasifikacije.

Usporedba klasifikacije neuronskom mrežom (crvena) i strojem s potpornim vektorima (plava) prikazana je na slici 7.12.



Slika 7.12 Usporedba uspješnosti klasifikatora

Graf prikazuje uspješnost klasifikacije u ovisnosti o udjelu preklapanja detekcije sa znakom. Očito je da je uspješnost klasifikacije bolja za preciznije locirani znak. Na krajnjem lijevom i desnom rubu testiranje se odvija nad vrlo malim brojem uzoraka stoga se pojavljuju samo ekstremne vrijednosti odnosno detekcija od 0 ili 1. Slika 7.11 prikazuje broj detektiranih uzoraka s točno tim preklapanjem.

Ukupna uspješnost klasifikacije neuronskom mrežom iznosi 88.16%, dok za stroj s potpornim vektorima iznosi 91%. Stroj s potpornim vektorima postiže bolji rezultat, te je on odabran kao primarni klasifikator.

Rezultat od 91% uspješnosti klasifikacije je osrednji, ali nije dovoljno dobar za primjenu.

Jedan od važnih aspekata performansi samog sustava je brzina. Sustav je sposoban obraditi 8 slika u sekundi, dok je za obradu video snimke u realnom vremenu potrebna brzina od barem 20 slika u sekundi.

7.3 Poboljšanja

Kao što je već spomenuto u prethodnom poglavlju, rezultati nisu zadovoljavajući niti u smislu detekcije, niti klasifikacije. Sljedeći korak u razvoju sustava je identifikacija i rješavanje problema koji rezultiraju lošim performansama.

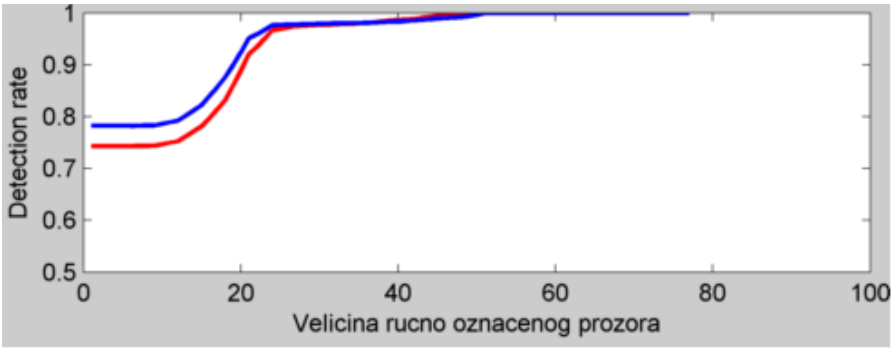
7.3.1 Problem lažnih detekcija

Osnovni problem detekcije je visoka razina lažnih detekcija. To je velika mana sustava, pošto bi za izložene performanse kroz video snimku postojalo više lažnih detekcija nego samih znakova.

Kao rješenje problema visoke razine lažnih detekcija nameće se korištenje dodatnog detektora koji bi se ponašao kao dodatna razina kaskade klasifikatora algoritma Viole i Jonesa. Za dodatnu razinu kaskade upotrebljena je umjetna neuronska mreža. Ulazne značajke su opisane u poglavlju 7.1.2. Skup za učenje ovakve neuronske mreže se sastoji od dva razreda: znakovi i ostali isječci. U skupu za učenje postoji 1639 znakova. Primjerci okoliša su slučajno odabrani isječci slika, te ih ima 1000. Osim toga, da bi se neuronska mreža ponašala kao razina kaskade, u skup za učenje je dodano i 1000 lažnih detekcija algoritma Viole i Jonesa, iz prethodno prezentiranog testiranja. Dobivena neuronska mreža naučena je ispravljati pogreške detekcije.

Potrebno je još odrediti koristiti li neuronsku mrežu prije ili poslije ujedinjavanja prozora. Pokazalo se da postavljanje neuronske mreže poslije ujedinjavanja prozora značajno smanjuje razinu lažnih detekcija, ali i uvelike smanjuje uspješnost detekcije, što je neželjeni učinak.

Postavljanje neuronske mreže prije procesa ujedinjenja prozora pokazuje se prilično dobrim rješenjem. Razina lažnih detekcija smanjena je s 17.69% na 4.83%, što je značajno poboljšanje. Uspješnost detekcije je snižena za 4%, što je ipak u razumnim razmjerima. Slika 7.13 prikazuje usporedbu uspješnosti detekcije s (crvena) i bez (plava) korištenja neuronske mreže za detekciju, u ovisnosti o minimalnoj promatranoj veličini prozora.



Slika 7.13 Utjecaj neuronske mreže na uspješnost detekcije.

Očito je smanjenje razine detekcije manje za veće znakove, te već za minimalnu veličinu od 30x30 postaje zanemariva, što se može očitati sa grafa. Pošto se uspješnost detekcije nije promijenila za veće znakove, očekivano je da ovakvo sniženje performanse neće utjecati na performansu kroz video

snimku. Stoga je razumno prihvatiti navedenu redukciju detekcije u svrhu značajnog uklanjanja lažnih detekcija.

7.3.2 Problem nepreciznog lociranja

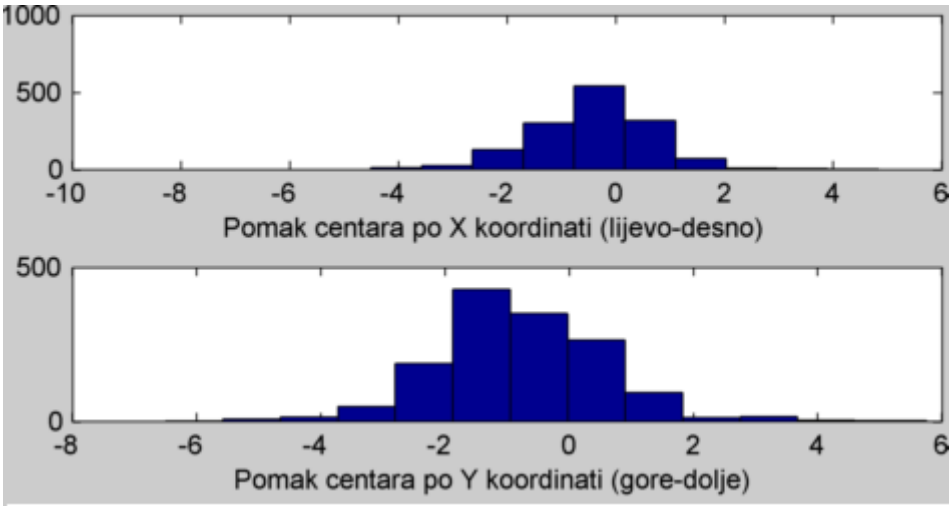
Uspješnost klasifikacije pokazala se relativno niskom. Uzrok tomu je pojava nepreciznog pozicioniranja znaka algoritmom Viole i Jonesa. Slika 7.14 prikazuje primjerke neprecizno pozicioniranih detekcija. Očiti su pomaci od idealne lokacije znaka.

Slika 7.14 Neprecizno pozicioniranje algoritmom Viole i Jonesa

Jedan pristup rješavanju ovog problema je pokušaj naknadnog preciznog pozicioniranja znaka. Istražen je rad neuronske mreže i Houghove transformacije za detekciju trokuta na ovom problemu, no pokušaji nisu polučili značajne rezultate.

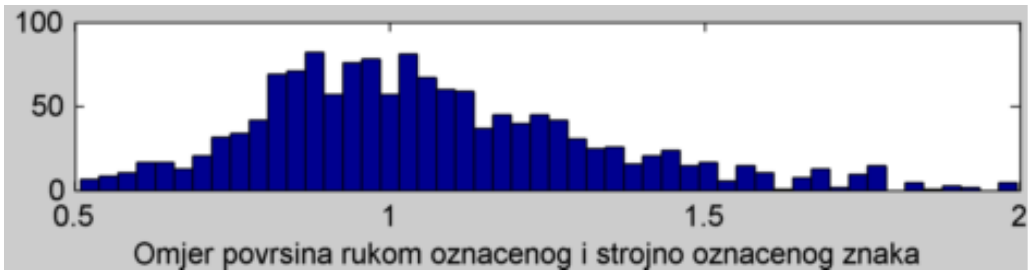
Drugi pristup rješavanju ovoga problema je prilagodba skupa za učenje klasifikatora. Osnovna ideja je dodavanje lošije pozicioniranih primjeraka znakova u skup, poput isječaka prikazanih na slici 7.14.

Dodavanje lošije pozicioniranih primjeraka u skup za učenje svodi se na generiranje istih pomoću operacija pomaka, uvećanja i smanjenja nad početnim skupom za učenje. Da bi ovakav pristup bio adekvatan, potrebno je modelirati razdiobe pomaka i uvećanja tako da odgovaraju razdiobama pomaka i uvećanja detekcija algoritma Viole i Jonesa. Rezultati sa slika 7.15 i 7.16 dobiveni su usporedbom izlaza algoritma Viole i Jonesa sa ručno označenim pozicijama znakova.



Slika 7.15 Razdiobe pomaka detekcija algoritma Viole i Jonesa

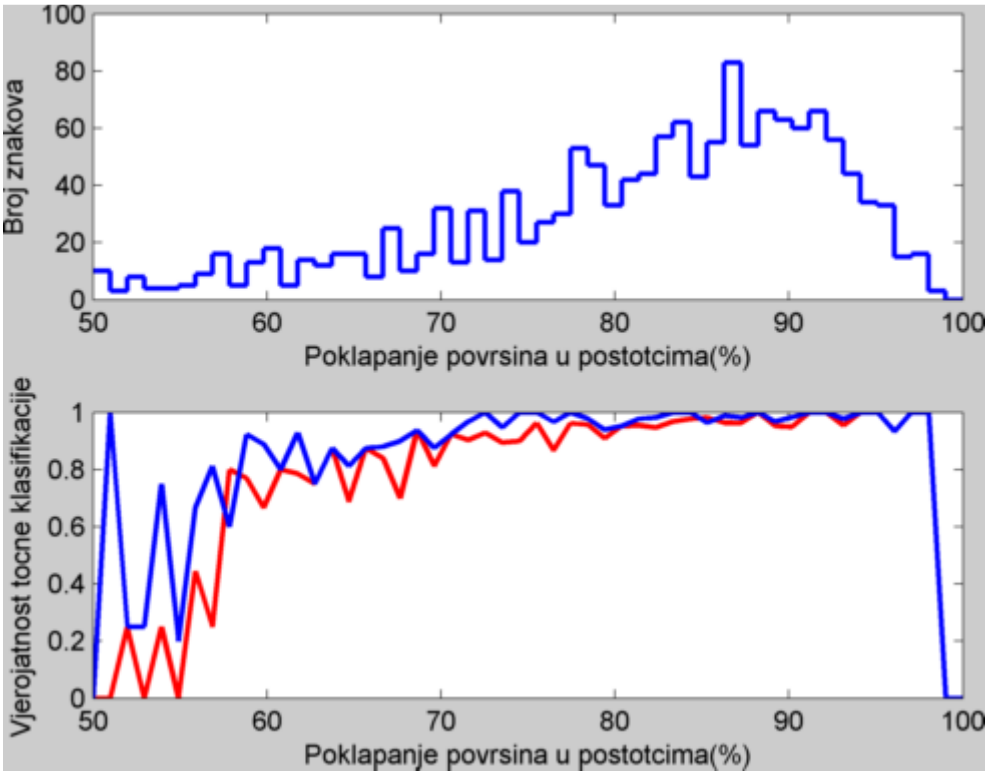
Razdiobu uvećanja može se aproksimirati razdiobom omjera preklapanja površina detektiranog i označenog isječka, što prikazuje slika 7.16.



Slika 7.16 Razdioba preklapanja površina

Sve tri razdiobe, prikazane na gornjim grafovima, mogu se aproksimirati kombinacijom normalnih razdioba. Pri tome se razdioba pomaka po x-osi modelira normalnom razdiobom s parametrima $(\mu = -0.48, \sigma = 1.15)$, a razdioba pomaka po y-osi se modelira s normalnom razdiobom s parametrima $(\mu = -0.74, \sigma = 1.31)$. Razdioba faktora uvećanja je malo kompleksnija, jer nije simetrična, ali i dalje nalikuje na normalnu razdiobu, no devijacije lijeve i desne stane nisu iste. Zato je svaka strana modelirana s zasebnom normalnom razdiobom (tj. njenom odgovarajućom stranom). Pri tome je desna strana modelirana normalnom razdiobom s parametrima $(\mu = 1, \sigma = 0.3)$, a lijeva strana normalnom razdiobom s parametrima $(\mu = 1, \sigma = 0.2)$.

Na slici 7.17 prikazani su rezultati klasifikacije stroja s potpornim vektorima učenog na skupu generiranom na opisan način (plava), te stroja s potpornim vektorima treniranog na originalnom, nemodeliranom skupu za učenje. Promjena skupa za učenje donijela je značajno poboljšanje : sa 91.33% na 95.42%.



Slika 7.17 Rezultati klasifikacije SVM-a na različitim skupovima

Graf prikazuje kako klasifikator naučen na modeliranom skupu značajno bolje klasificira lošije pozicionirane primjerke, što je i bio cilj modeliranja skupa za učenje. Pri tome je klasifikator zadržao dobre performanse za preciznije pozicionirane primjerke.

7.3.3 Ostala poboljšanja

Zbog načina određivanja klase pomoću stroja s potpornim vektorima, moguće je da ishod bude neodlučen. Stoga se u tome slučaju kao klasifikator koristi neuronska mreža.

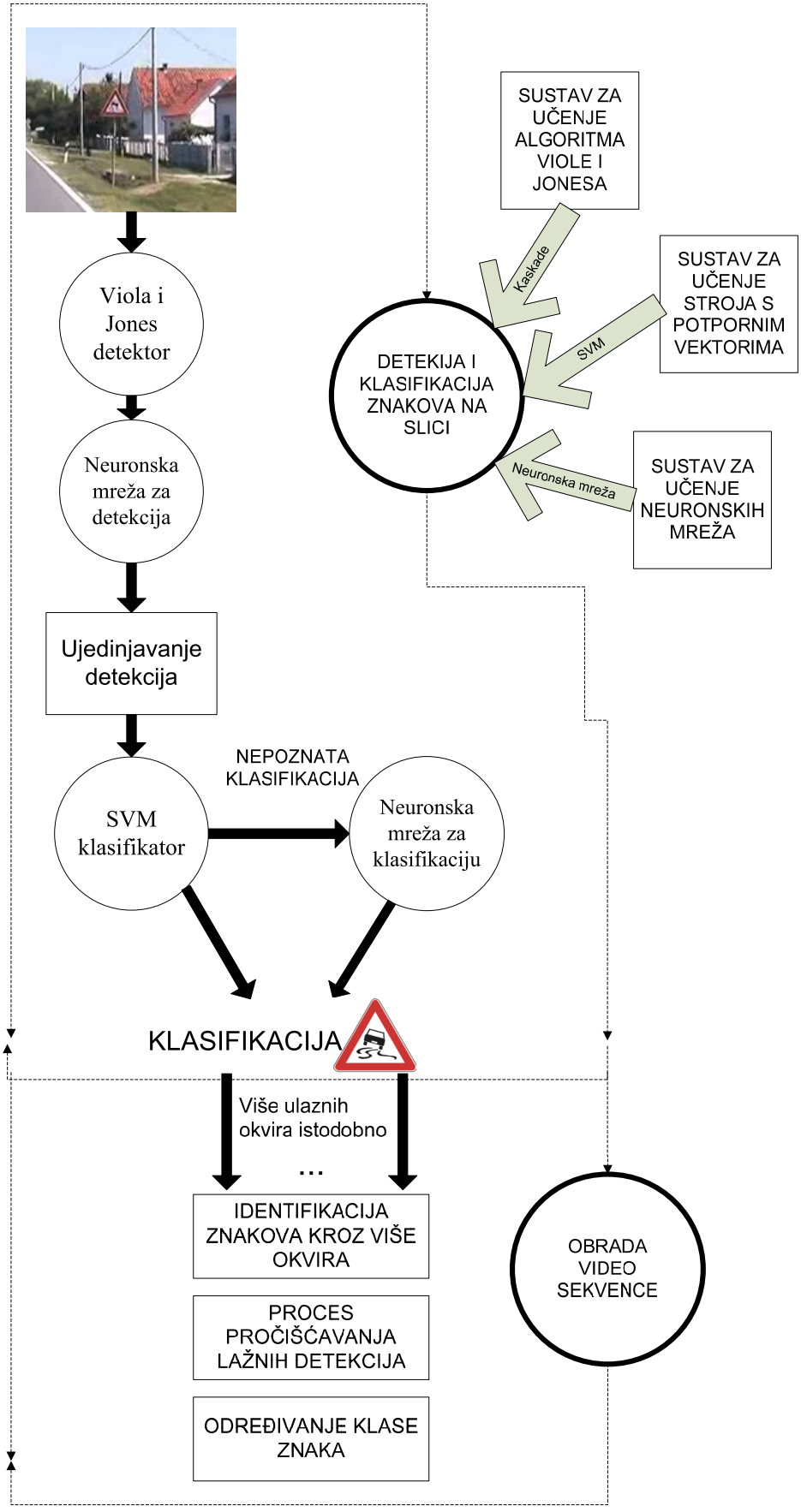
Poboljšana je implementacija detekcije algoritma Viole i Jonesa nad okvirima video sekvence primjenom paralelizma. Naime, algoritam je moguće jednostavno paralelizirati istovremenom detekcijom nad susjednim okvirima. Testiranjem na dvojezgrenom procesoru je pokazano dvostruko ubrzanje detekcije, sa osam na 14 okvira okvira po sekundi. S obzirom da je implementaciju jednostavno prilagoditi većem broju dretvi na procesorima s više jezgri ili grozdu računala (*eng. cluster)* računala postignuta ubrzanja mogu biti višestruka. Dodatno ubrzanje je postignuto izmjenom operacija nad realnim brojevnim tipovima s operacijama nad cijelim brojevima, čime je postignuto dodatno ubrzanje od 20%, a rezultantna pogreška je gotovo zanemariva.

7.4 Konačna izvedba

Slika 7.18 prikazuje konačnu izvedbu sustava. Prva faza je detektiranje potencijalnih regija znakova algoritmom Viole i Jonesa. Potom se detekcije pročišćavaju neuronskom mrežom te se zatim obavlja proces ujedinjavanja susjednih detekcija. Nad detekcijama iz slike pokreće se SVM klasifikator koji određuje tip znaka sa slike. Ukoliko SVM ne može odrediti razred znaka tada se pokreće neuronska mreža za klasifikaciju. Rezultati detekcije i klasifikacije kroz nekoliko susjednih okvira video sekvence se prosljeđuju u sustav za obradu video sekvence.

Na video sekvencama moguće je, zbog brzine implementiranog algoritma, koristiti nezavisno pretraživanje svakog okvira video sekvence. Ovakvo pretraživanje uvodi veliku zalihost u metodu detekcije i klasifikacije s obzirom da se znak pojavljuje na više uzastopnih okvira te s druge strane omogućava implementaciju algoritma koji prati uzastopne detektirane okvire te na temelju njih određuje koji su nastali kao posljedica greške klasifikatora ili šuma u podacima.

Nakon detekcije, regije koje sustav označi kao potencijalne znakove na slikama ulaze u proces identifikacije znaka na razini video snimke. Primarni cilj ovog procesa je točno određivanje položaja znaka u video sekvenci. Nakon toga slijedi proces pročišćavanja lažnih detekcija i klasifikacije. U tom procesu razmatra se određeni broj susjednih okvira video sekvence (tipično 30), te ukoliko je u njima pronađena detekcija koja odgovara istom području tada se regija proglašava ispravnom detekcijom, a inače se odbacuje. Ovaj proces uvelike smanjuje udio lažno pozitivnih detekcija te čini klasifikaciju pouzdanijom s obzirom da se za sve susjedne detektirane regije klasifikacija obavlja nezavisno te se zatim odabire ona klasa koja se najveći broj puta pojavila kao rezultat pojedinačne klasifikacije.



Slika 7.18 Shematski prikaz izvedenog sustava

8 Rezultati

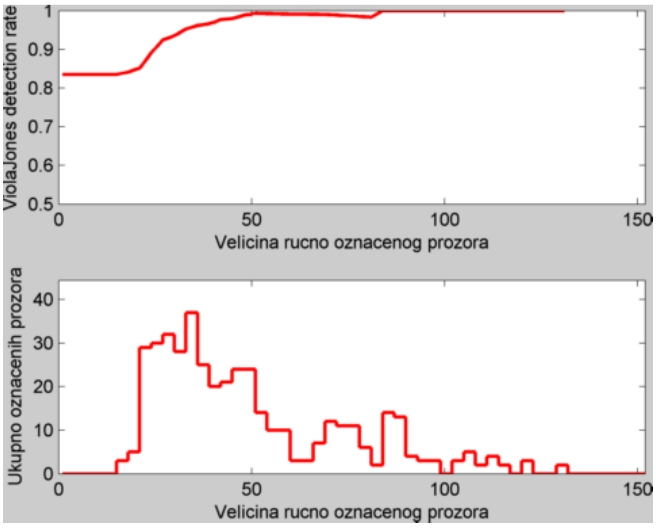
Rezultati izloženi u prethodnim poglavljima poslužili su u svrhu identifikacije problema, te poboljšanja sustava. Time je konačno rješenje ovisno o snimkama, odnosno slikama nad kojima su takvi rezultati izrađeni.

Da bi rezultati bili mjerodavni, provedeno je završno testiranje nad video snimkom o kojoj je sustav u potpunosti neovisan. Točnije, niti jedan uzorak za učenje i validaciju nije iz testne video snimke. Jednako tako, niti jedno istraživanje, niti poboljšanje sustava nije temeljeno na istoj. Također, pri snimanju snimke za testiranje korištena je video kamera različitih svojstava od one koja je korištena pri snimanju prethodnih snimki.

Kroz daljnja poglavlja su prezentirani rezultati detekcije i klasifikacije na razini slika te na razini video snimke, postignuti na opisanoj snimci za testiranje.

8.1 Detekcija

Testiranje je provedeno nad slikama iz video snimke, za koje je označena pozicija i tip znaka. Slika 8.1 prikazuje rezultate u ovisnosti o minimalnoj promatranoj veličini znaka (gornji graf). Na donjem grafu je prikazana razdioba veličine znakova u skupu za testiranje.



Slika 8.1 Rezultati detekcije

Uspješnost detekcije je **83.53%,** što je relativno loš rezultat. Uzrok tomu je što algoritam Viole i Jonesa provodi detekciju samo za okvire veće od 24x24 slikovna elementa[11]. Sa gornjeg grafa se očitava rezultat detekcije za okvire veće od 24x24, te iznosi **89.18%,** što je još uvijek loš rezultat za primjenu.

Očevidan je rast uspješnosti detekcije s porastom minimalne promatrane veličine, te se stoga može zaključiti da algoritam Viole i Jonesa bolje radi nad velikim znakovima. Uspješnost detekcije već za okvire veće od 50x50 iznosi **99.14%,** što je prihvatljiv rezultat. Uzevši u obzir da se znak u video snimci pojavljuje u većem broju slika, te da se njegova veličina povećava, za očekivati je da će u pravilu znak postići veličinu 50x50, te da će performanse kroz video snimku biti prihvatljive.

Parametri korišteni prilikom detekcije su:

- Faktor uvećanja okna za detekciju (*eng. scale factor*): 1.20
- Korak pomicanja okna (*eng. step size*): 5% veličine okvira
- Minimalna veličina grupe (koristi se kod koraka ujedinjavanja okvira): 2

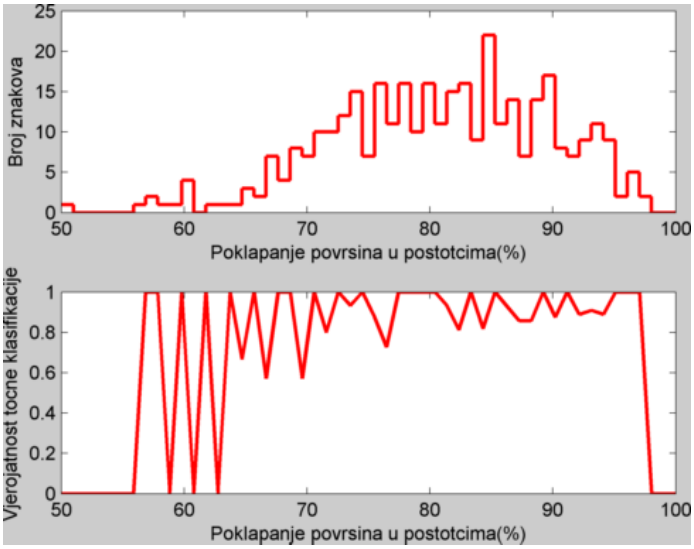
Izmjenom navedenih parametara moguće je mijenjati odnos između broja lažno negativnih detekcija i postotka detekcije. Parametri također utječu na brzinu izvršavanja algoritma Viole i Jonesa.

8.2 Klasifikacija

Testiranje je provedeno nad istim slikama kao i za detekciju, i to samo za ispravne detekcije.

Slika 8.2 prikazuje rezultate klasifikacije u ovisnosti o postotku preklapanja površina detekcije i označenog znaka (donji graf). Gornji graf prikazuje raspodjelu udjela preklapanja površina između označenog znaka i detekcije.

[11] Veličina od 24x24 slikovna elementa je najčešće korištena minimalna veličina prilikom detekcije Viola i Jones algoritmom. Važno je da odabrana minimalna veličina okvira sadrži dovoljno informacije za razlikovanje objekta od pozadine



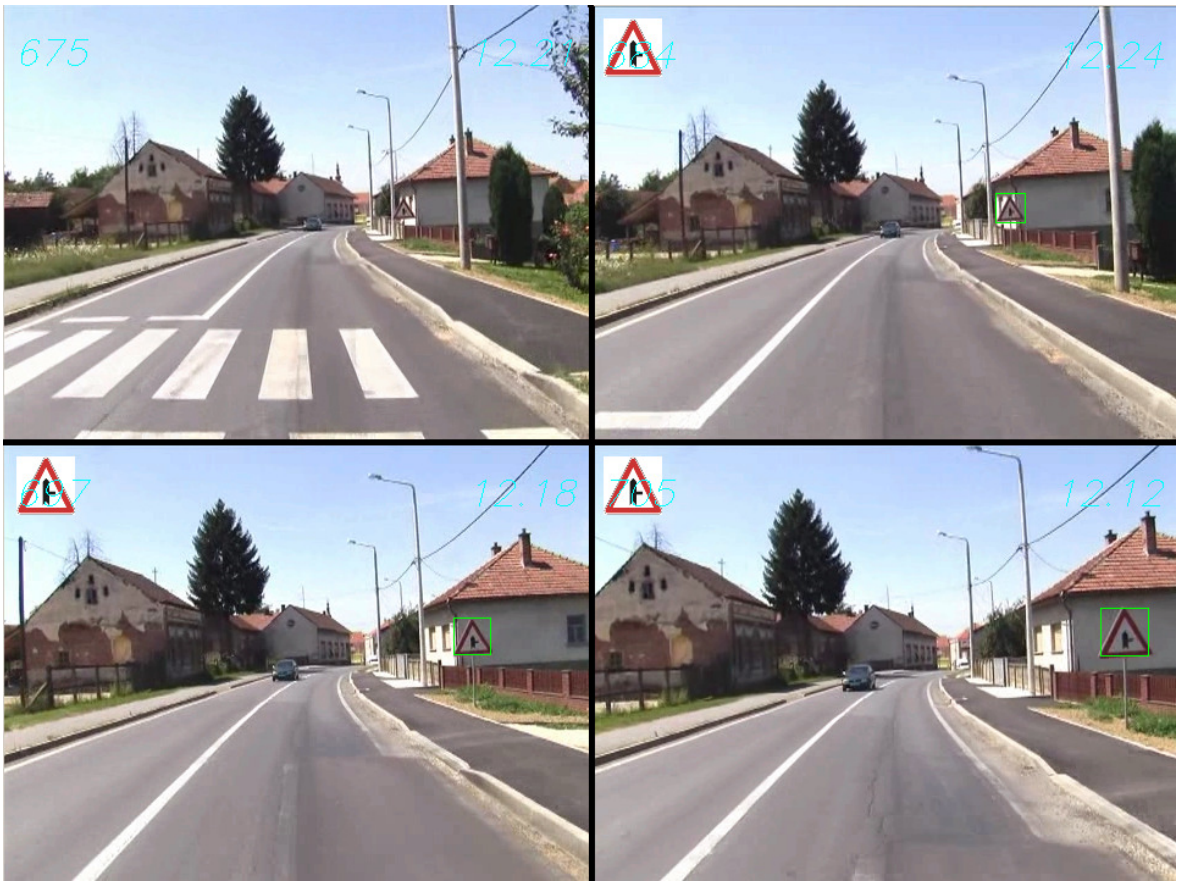
Slika 8.2 Rezultati klasifikacije

Uspješnost klasifikacije iznosi 90.42%. Kao i u slučaju detekcije, to je na prvi pogled loš rezultat. Ipak, promatrajući graf ovisnosti performanse o postotku preklapanja, očigledno je da je klasifikacija bolja za preciznije locirane znakove.

Za očekivati je da su znakovi male veličine, kao i znakovi velike veličine, neprecizno locirani, za razliku od onih prosječne veličine. Pošto se kroz video sekvencu znak najviše puta pojavljuje u veličini oko prosječne, za očekivati je da će ga se više puta precizno locirati, te da će u pravilu klasifikacija glasanjem kroz snimku polučiti prihvatljive rezultate.

8.3 Performanse sustava

Konačna performansa sustava očituje se u kvaliteti detekcije i raspoznavanja znakova kroz video snimku. Slika 8.3 prikazuje rad sustava na video snimci.



Slika 8.3 Rad sustava na video snimci

Na primjeru je pokazano kako je sustav identificirao znak, iako ga je detektirao tek kada je postigao određenu veličinu. Ovakvo ponašanje sustava ukazuje na pretpostavku da bi rezultati kroz video snimku trebali biti osjetno bolji od rezultata na slikama.

Testiranje je provedeno ručno, nad video snimkom sljedećih karakteristika:

- Rezolucija: 480x360 px
- Trajanje: 1h 28m 16s
- Ukupno slika: 132420
- Slika po sekundi: 25
- Ukupno znakova: 265

Rezultati detekcije prikazani su u tablici 8.1:

Tablica 8.1

REZULTATI DETEKCIJE	
Ukupno znakova:	265
Detektiranih:	260
Nedetektiranih:	5
Lažnih detekcija:	2
Uspješnost detekcije:	**98.07%**
Razina lažnih detekcija:	**0.76%**

Kao što se vidi iz priloženog, rezultati detekcije su prilično dobri, te prihvatljivi. Važno je napomenuti da su nedetektirani znakovi uglavnom male veličine i lošije kvalitete. Ovaj problem bi se lako mogao riješiti korištenjem video snimke veće razlučivosti.

Rezultati klasifikacije se odnose samo na detektirane znakove, kakvih je ukupno 260. Tablica 8.2 prikazuje rezultate klasifikacije.

Tablica 8.2

REZULTATI KLASIFIKACIJE	
Ukupno znakova:	260
Ispravno klasificiranih:	241
Pogrešno klasificiranih:	19
Uspješnost klasifikacije:	**92.69%**

Na prvi pogled, uspješnost od 92% je relativno loš rezultat. No uzrok tome je u nepotpunom skupu za učenje. Naime, u skupu za učenje na raspolaganju su primjerci iz 25 razreda znakova, dok ukupno postoji 50 razreda znakova upozorenja. U testnoj video snimci pojavljuje se ukupno 17 znakova za koje klasifikator nije naučen, te ih nema sposobnost prepoznati.

Iz navedenih razloga, prikladno je mjeriti rezultate klasifikacije bez uloge znakova koji nisu zastupljeni u skupu za učenje. Takvi rezultati prikazani su u tablici 8.3.

Tablica 8.3 Efektivni rezultati klasifikacije

REZULTATI KLASIFIKACIJE 2	
Ukupno znakova:	243
Ispravno klasificiranih:	241
Pogrešno klasificiranih:	2
Uspješnost klasifikacije:	**99.17%**

Ovakav pristup mjerenju pokazuje da su rezultati klasifikacije vrlo dobri. U oba slučaja krive klasifikacije radi se o sličnim znakovima.

Konačno, ukupna performansa sustava je **97.25%**, odnosno, 90.90%, ukoliko se uzima u obzir kriva klasifikacija znakova čijih predstavnika nema u skupu za učenje.

9 Moguća poboljšanja

Projekt ostavlja velik prostor za buduća poboljšanja i daljnji rad. Moguće je postići bolje rezultate od dobivenih upotrebom većeg skupa za učenje. Nažalost takav skup nije bio dostupan u trenutku izrade ovog projekta. Novi, veći skup za učenje bi trebao sadržavati sve klase trokutastih znakova (a ne samo 50% klasa što ima trenutni skup za učenje). Sve klase bi trebale imati dovoljno velik broj znakova da budu statistički značajne (u trenutnom skupu za učenje pojedine klase imaju manje od 10 znakova što nikako nije dovoljno).

Projekt bi se također mogao proširiti i na druge kategorije znakova osim znakova upozorenja (tj. trokutastih znakova). Znakovi zabrane (okrugli znakovi) su također jedna velika kategorija znakova koja bi se mogla obraditi iznesenim postupcima. Svi postupci primijenjeni na trokutaste znakove bi se mogli jednostavno i bez većih promjena primijeniti i na okrugle znakove.

Moglo bi se također detektirati i stanje pojedinog prometnog znaka, poput: istrošenosti boje, deformiranost nosača znaka, deformiranosti samog znaka, zakrenutosti znaka od željene ravnine...

Nažalost ograničeni danim skupom za učenje nismo bili u mogućnosti ostvariti predložena poboljšanja. Nadamo se da će u budućnosti biti na raspolaganju odgovarajući skup za učenje, te da će navedene promjene jednom zaživjeti u nekom obliku nastavka ovog projekta.

10 Zaključak

U sklopu rada razvijen je sustav za detekciju i raspoznavanje prometnih znakova upozorenja (trokutastih znakova). Razvijeni sustav pokazuje izrazito dobre performanse, no postoji još prostora za daljnja poboljšanja. Jedna od prednosti sustava koja doprinosi njegovoj primjenjivosti je i rad u realnom vremenu. Sustav je moguće direktno ugraditi u bilo koju aplikaciju koja zahtijeva raspoznavanje trokutastih prometnih znakova. Predloženi postupak je konceptualno primjenjiv na široku klasu problema iz domene računalnog vida, pri čemu su potrebne manje izmjene da se sustavom može opisati i riješiti željeni problem.

Korišteni algoritam Viole i Jonesa je najpouzdaniji predstavnik algoritama za brzu detekciju objekata u slikama. Stroj s potpornim vektorima također predstavlja sam vrh klasifikacijskih algoritama. S druge strane, algoritam umjetne neuronske mreže je stariji algoritam koji je izrazito prilagodljiv različitim domenama, te se može koristiti u kontekstima različite naravi (u projektu su različite neuronske mreže korištene kao detektor i kao klasifikator).

Osim glavnih komponenti sustava (algoritma Viole i Jonesa, umjetnih neuronskih mreža i stroja s potpornim vektorima) pokazalo se izrazito bitnim maksimizirati raspoložive informacije detaljnim testiranjem i pogodnim statističkim modeliranjem. Primjena gotovih (eng. “of the shelf”) algoritama, bez prilagodbe specifičnostima domene raspoznavanja prometnih znakova rezultira u značajno lošijim performansama sustava. Ova činjenica opravdava opširna testiranja provedena tijekom ostvarenja sustava.

Za potrebe rada su svi dijelovi sustava programski ostvareni u cijelosti. Kako je dio sustava koji vrši detekciju i klasifikaciju ostvaren u programskom jeziku C, sustav se može izvršavati i na računalima različitim od standardnih osobnih računala. Moguće je i dizajnirati posebno sklopovlje da bi se sustav integrirao u vozila čime bi direktno mogao pomagati vozačima za vrijeme vožnje.

11 Zahvale

Zahvaljujemo se mentoru prof.dr.sc. Zoranu Kalafatiću na materijalima, usmjeravanju, interesu te uloženom vremenu. Također, zahvaljujemo se doc.dr.sc Siniši Šegviću na sugestijama u oblikovanju finalne dokumentacije.

12 Dodatak A – Opis implementacije

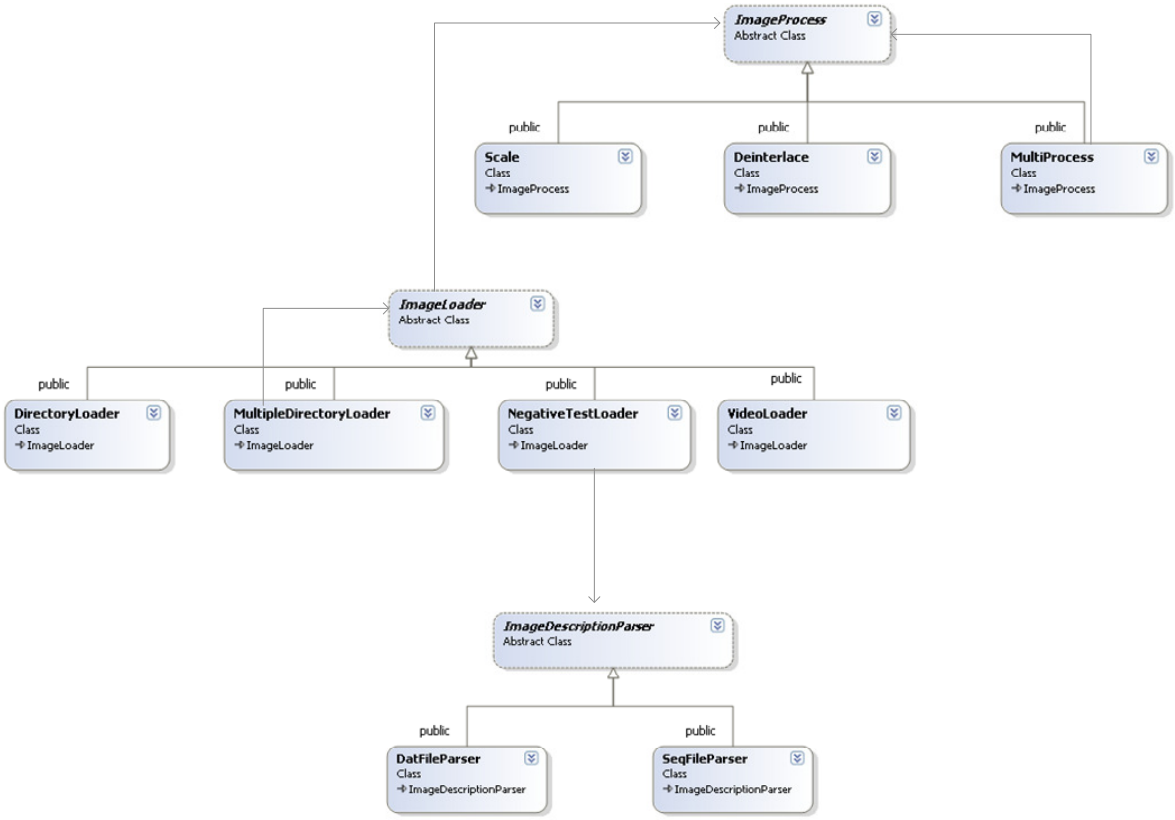
A.1 Algoritam Viole i Jonesa

Cjelokupni algoritam Viole i Jonesa implementiran je u programskom jeziku C++. Programski jezik je odabran zbog visoke računalne zahtjevnosti algoritma. Primjerice učenje kaskade Viola i Jones algoritmom može trajati i do tjedan dana. U implementaciji korištene su dvije besplatne gotove biblioteke *OpenCV* [26] za prikaz, obradu i učitavanje slike, te biblioteka *Boost* [25] za čitanje XML konfiguracijskih datoteka, višedretveni rad te rad s datotečnim podsustavom.

S obzirom da je hijerarhija razreda suviše složena da se prikaže jednim UML dijagramom, u sljedećim poglavljima biti će prikazani i ukratko objašnjeni dijagrami prema podsustavima.

A.2 Učitavanje i obrada slike:

Za učitavanje i osnovne metode obrade slike koristi se biblioteka *OpenCV 1.1*. Za potrebe algoritma Viola i Jonesa razvijena je hijerarhija razreda *ImageLoader* čiji izvedeni razredi omogućavaju učitavanje pozitivnih i negativnih primjeraka za učenje. Pozitivni primjerci, odnosno slike prometnih znakova, se učitavaju iz jednog direktorija, dok se negativni primjerci izrezuju iz velikih slika. Prilikom učenja kaskade u algoritmu Viole i Jonesa potrebno je analizirati izrazito veliki broj slika u potrazi za greškama kaskade koji se zatim koriste za učenje narednih razina. Taj posao obavlja razred *NegativeTestLoader.* Hijerarhija razreda *ImageDescriptionParser* iz datoteke učitava podatke o slikama i poziciji znakova u njima te time omogućava izrezivanje pozitivnih i negativnih primjera za učenje. Programom *Marker*[22] ručno su označeni znakovi na video sekvencama te su stvorene datoteke koje opisuju položaj znakova u pojedinim okvirima video sekvence.



Slika A.1 Osnovna hijerarhija razreda korištena za učenje algoritma Viola i Jonesa

Hijerarhija razreda *ImageProcess* služi za implementaciju različitih metoda obrade slike. Podržane operacije su skaliranje, te eliminacija preplitanja (*deinterlace*) opisana u ovom radu.

Razred *Image.h*[12] enkapsulira sve podatke vezane uz sliku, integralnu sliku te omogućava njen prikaz i obradu. Implementirani su i još poneki pomoćni razredi: *ColorSpace* za enkapsulaciju podataka o trenutnom korištenom sustavu boju te različitim mogućnostima konverzije među njima te razred *Settings* koji služi za učitavanje postavki iz datoteke *Settings.xml*. U toj datoteci moguće je postaviti veći broj parametara kako za učenje algoritma Viole i Jonesa tako i za postavke evaluacije kaskade.

A.1.1 Učenje kaskade:

Temeljni razredi važni za učenje kaskade su *AdaBoost* i *ViolaJones.* Naučena kaskada se sprema u razredu *Cascade.* Razred *ViolaJones* konfigurira se pomoću razreda *ImageLoader* za učitavanje pozitivnih i negativnih primjera za učenje. Razredi *AdaBoost i ViolaJones* implementiraju odgovarajuće algoritme opisane u radu.

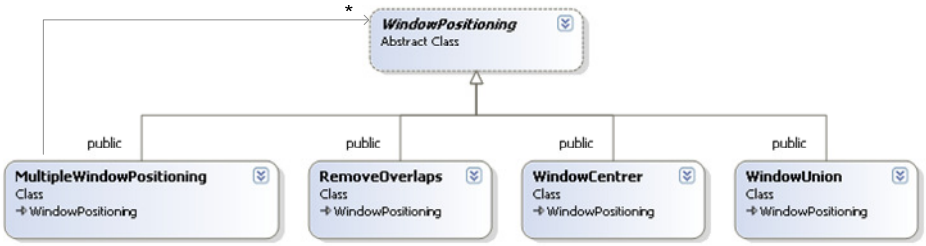
[12] Razred *Image.h* zbog jednostavnosti nije prikazan na gornjem dijagramu, a povezan je s gotovo svim razredima u hijerarhiji

A.1.2 Detekcija znakova

Viola i Jones detektor implementiran je u razredu *Image*, odnosno funkciji *Image::evaluateCascade()*. Funkcija je posebno optimirana na način da ne koristi izračune nad brojevima sa pomičnim zarezom[13] već se za sve operacije koriste cijeli brojevi. Ovaj postupak optimizacije donio je poboljšanja u smislu brzine od 20% dok je nastala pogreška zbog zaokruživanja gotovo zanemariva.

Prilikom evaluacije kaskade nad video sekvencama, zbog povećanja brzine izvođenja, koristi se višedretveni rad. Korištenjem biblioteke *Boost.Multithreading[25]* implementirana je paralelna inačica detekcije pomoću algoritma Viole i Jonesa. Testiranjem na *Intel Dual-Core* procesoru pokazano da je paralelna inačica gotovo dvostruka brža od osnovne. Povećanjem broja procesorskih jezgri ili procesora može se postići i značajno veće ubrzanje.

Hijerarhija razreda *WindowPositioning* omogućava dodatnu obradu rezultantnih regija iz algoritma Viole i Jonesa. Razred *WindowUnion* implementira *UnionFind* algoritam opisan u radu, *WindowCentrer* omogućava dodatno preciznije centriranje znaka pomoću neuronske mreže. Razred *MultipleWindowPositioning* omogućava izvođenje više različitih metoda obrade pomoću oblikovnog obrasca *Composite*[14].



Slika A.2 Hijerarhijski prikaz podsustava za ujedinjenje i centriranje detekcija

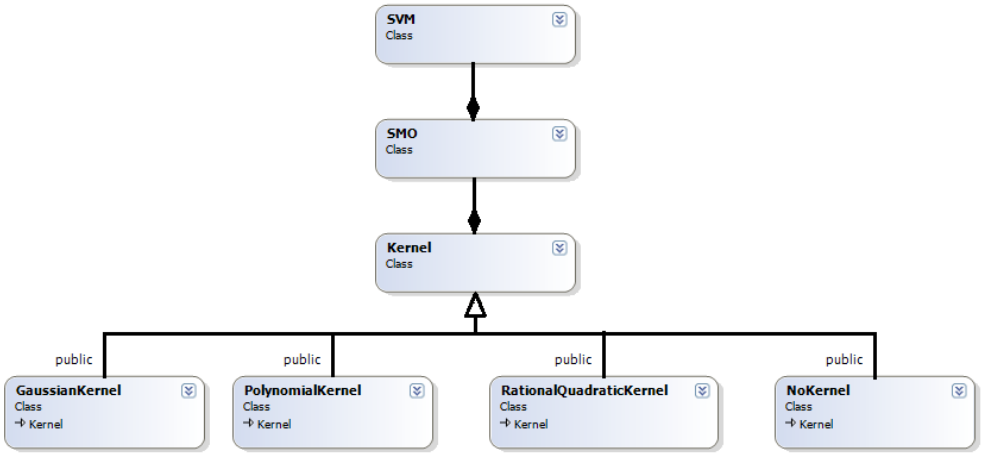
Prilikom detekcije također je omogućena uporaba različitih klasifikatora (iz hijerarhije *Classifier)* za dodatno pročišćavanje detekcija koje pronađe algoritam Viole i Jonesa.

[13] Odnosi se na *float* odnosno *double* tip podataka u programskom jeziku C++

[14] Oblikovni obrazac *Composite* omogućava da se grupa objekata tretira kao jedna cjelina

A.2 Programsko ostvarenje stroja s potpornim vektorima

U sklopu ovog rada su korištene dvije implementacije stroja s potpornim vektorima: naša vlastita implementacija i besplatna biblioteka LIBSVM [12].



Slika A.3 Hijerarhijski prikaz podsustava za učenje SVM klasifikatora

Obje implementacije se temelje na prethodno iznesenom algoritmu slijedne minimalne optimizacije i stroju s potpornim vektorima s mekom granicom. Kako je SVM optimalni klasifikator, obje implementacije daju iste rezultate te su potpuno ravnopravne i uvijek se mogu međusobno zamijeniti. Osnovni formati zapisa dvije implementacije se razlikuju, pa su ostvarene skripte koje pretvaraju jedan format u drugi.

LIBSVM biblioteka je implementirana u nekoliko jezika, a korištene su prvenstveno implementacije u C++u i Javi, pri čemu je Java verzija korištena za proučavanje biblioteke i ispitivanje djelovanja jezgre, a C++ verzija, zbog brzine, za sam proces učenja. Slika A.3 prikazuje UML dijagram Java implementacije učenja klasifikatora.

Glavna prednost biblioteke LIBSVM je u brzini, te je za red veličine brža od naše vlastite implementacije, što je bilo i za očekivati jer koristi puno sofisticiranije heuristike i statističke metode ubrzanja procesa učenja.

S druge strane, biblioteku LIBSVM je puno teže modificirati i eksperimentirati s djelovanjem pojedinih elemenata stroja s potpornim vektorima. Tu je naša vlastita implementacija puno robusnija i dozvoljava veće zahvate u funkcionalnost. Ova razlika se posebno pokazala pri eksperimentiranju s djelovanjem pojedinih jezgara.

Naša implementacija stroja s potpornim vektorima je ostvarena u Javi. Algoritam slijedne minimalne optimizacije je programski ostvaren u svom osnovnom obliku, uz relativno malen broj modifikacija. Upotrebljena je standardna, preporučena heuristika za odabir para α_i, α_j jer se pokazala dovoljno efikasnom za dani problem. Istraživanja su pokazala [11] da postoje i bolje heuristike za ovaj odabir, no njihova je implementacija znatno složenija, a kako je dani problem relativno uspješno riješen i jednostavnijom heuristikom, složenije heuristike nisu ni razmatrane u sklopu ovog rada.

Kao što je i uobičajeno [7], implementirano je spremanje trenutnih odstupanja od tražene vrijednosti u pričuvnu memoriju (eng. *Error Cache*). Spremanje odstupanja u pričuvnu memoriju dovodi do značajnog ubrzanja u postupku rješavanja optimizacijskog problema stroja s potpornim vektorima, odnosno skraćuje vrijeme učenja.

Ostvareno je i pamćenje vrijednosti jezgrene funkcije za parove primjera za učenje $x^{(i)}, x^{(j)}$ u pričuvnoj memoriji (eng. *Kernel Cache*). Ovim se postupkom postiže značajno ubrzanje jer se prilikom učenja algoritma za sve parove primjera za učenje vrijednosti jezgre izračunavaju samo jednom, a ne u svakom koraku.

Nad ulaznim vektorima značajki provedena je normalizacija na interval $<0,1>$ da bi stroj s potpornim vektorima radio ispravno s bilo kojom od realiziranih jezgri. Osim mogućnosti ne korištenja jezgre, kada je stroj s potpornim vektorima praktički identičan klasifikatoru optimalne granice, implementirane su i jezgre:

- polinomna jezgra,
- Gaussova jezgra,
- racionalna kvadratna jezgra.

A.2.1 Učenje stroja s potpornim vektorima

Pri upotrebi naše vlastite implementacije, za velike skupove za učenje može doći do prevelike potrošnje memorije ako se koristi pamćenje vrijednosti jezgre u pričuvnoj memoriji. Ovaj je problem riješen na dva načina: ili se nije koristilo pamćenje vrijednosti jezgrene funkcije, ili se skup za učenje razbio na nekoliko dijelova te se učenje odvijalo u više koraka. Drugo rješenje je bolje jer se njime ne gubi ubrzanje koje pamćenje u pričuvnoj memoriji donosi. Skup za učenje se razbije na dijelove koji su dovoljno mali da nemaju problema s memorijom. Dijelovi se posebno uče, a na kraju se potporni vektori iz svih dijelova stave u novi skup za učenje. Taj novi skup za učenje se nadopuni do maksimalne veličine sa slučajno odabranim uzorcima za učenje. Ovaj skup se upotrebljava za učenje konačne verzije klasifikatora. Izneseni proces je automatiziran i stoga jednostavan za provesti, a učinkovit.

Stroj s potpornim vektorima i bez korištenja jezgara daje zapanjujuće dobre rezultate, prosječna točnost jednog SVM-a iznosi 99.81%. Upotrebom jezgri na ovako malim skupovima za učenje dolazi do prenaučenosti klasifikatora, pa jezgrene funkcije nisu korištene u konačnoj verziji programa.

A.3 Programsko ostvarenje umjetnih neuronskih mreža

Iako trenutno postoji mnoštvo dostupnih implementacija neuronskih mreža, u ovom radu sve su neuronske mreže samostalno implementirane.

U fazi istraživanja korišten je Matlab, programski alat sa razvijenim okruženjem za obradu slike. U Matlabu je testiran rad neuronskih mreža te razne metode obrade slike i izlučivanja značajki. Posljedica toga je i konačna izvedba dijelova aplikacije u Matlabu. Pošto je aplikacija izvedena u programskom jeziku C++, svi dijelovi o kojima ovisi rad konačne verzije aplikacije, iznova su implementirani u programskom jeziku C++.

Osnovne cjeline implementirane u Matlabu su:

- Ispravljanje smetnje nastale preplitanjem, nad uzorcima za učenje
- Sustav za rad neuronskih mreža
- Istraživanje odabira najboljih ulaznih značajki

Implementirano u programskom jeziku C++ :

- Izlučivanje značajki – za najbolje odabire
- Evaluacija neuronske mreže

Sustav za rad neuronskih mreža sastoji se od učenja, te evaluacije i testiranja. Za rad aplikacije, u programskom jeziku C++ je implementirana samo evaluacija i izlučivanje značajki , dok se učenje odvija nezavisno u Matlabu. Definicija neuronske mreže, koja je rezultat učenja u Matlabu, zapisuje se u datoteku, te aplikacija na temelju te datoteke koristi neuronsku mrežu za evaluaciju. U suštini, nema razlike između neuronske mreže za detekciju i neuronske mreže za klasifikaciju, te se obavljaju isti postupci za oba tipa.

13 Literatura

[1] Burges, Christopher J. C.: A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, vol. 2, pp. 121-164, 1998.

[2] Boser et al: A Training Algorithm for Optimal Margin Classifiers, Proceedings Fifth ACM Workshop on Computational Learning Theory, pp. 144–152, 1992.

[3] Shawe-Taylor, John; Cristianini, Nello: Kernel Methods for Pattern Analysis, Cambridge University Press, New York, USA, 2004.

[4] Genton, Marc G.: Classes of kernels for machine learning: a statistics perspective, Journal of Machine Learning Research, vol. 2, pp. 299-312, 2002.

[5] Wu, Q.; Zhou, D.: SVM Soft Margin Classifiers: Linear Programming versus Quadratic Programming. Neural Computation, vol. 17, pp. 1160-1187, 2005.

[6] Hush, Don; Scovel, Clint: Polynomial-time decomposition algorithms for support vector machines, Machine Learning, vol. 51, pp. 51–71, 2003.

[7] Platt, John C.: Fast Training of Support Vector Machines using Sequential Minimal Optimization, u knjizi Schölkopf et al: Advances in Kernel Methods - Support Vector Learning, Cambridge, MA, 1998. MIT Pres

[8] Lin, Chih-Jen: Asymptotic Convergence of an SMO Algorithm Without Any Assumptions, IEEE Transactions on Neural Networks, vol. 13, pp. 248–250, 2002.

[9] O. Chapelle, P. Haffner, and V. Vapnik, "SVMs for histogram-based image classification," IEEE Transactions on Neural Networks, vol. 9, 1999.

[10] Zamolotskikh, A; Cunningham, P.: An Assessment of Alternative Strategies for Constructing EMD-Based Kernel Functions for Use in an SVM for Image Classification, CBMI '07 International Workshop, pp. 11-17, 2007.

[11] Keerthi et al: Improvements to Platt's SMO algorithm for SVM classifer design, Neural Computation, vol. 13, pp. 637–649, 2001.

[12] Chang, Chih-Chung; Lin, Chih-Jen: LIBSVM: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[13] Chih-Wei Hsu, Chih-Jen Lin: A Comparison of Methods for Multi-class Support Vector Machines, *IEEE Transactions on Neural Networks* , Vol. 13 , Nr. 2 (2002) , p. 415--425.

[14] Bojana Dalbelo Bašić, Marko Čupić, Jan Šnajder: "Umjetne neuronske mreže" , 08. lipnja 2009. http://www.fer.hr/_download/repository/UI_14_umjetne_neuronske_mreze.pdf

[15] Yann LeCun, Leon Bottou, Genevieve B. Orr, Klaus-Robert Müller: "Efficient BackProp", 1998, http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

[16] "Lab color space", 24.01.2010. , http://en.wikipedia.org/wiki/Lab_color_space

[17] E. Alpaydin: Introduction to Machine Learning, MIT Press, 2004.

[18] T. M. Mitchell: Machine Learning, McGraw Hill, 1997.

[19] "Haar wavelet", 24.04.2010., http://en.wikipedia.org/wiki/Haar_wavelet

[20] "Boosting", 15.04.2010.,http://en.wikipedia.org/wiki/Boosting

[21] P. Viola, M. Jones: Robust Real-time Object Detection, Second international workshop on statistical and computational theories of vision – modeling, learing, computing, and sampling, 2001

[22] K. Brkić, A. Pinzl, S. Šegvić: Traffic sign detection as a component of an automated traffic infrastructure inventory system, Proceedings of the annual Workshop of the Austrian Association for Pattern Recognition (OAGM/AAPR), Stainz, Austria, 2009

[23] Claus Bahlmann , Ying Zhu , Visvanathan Ramesh Martin Pellkofer , Thorsten Koehler: A System for Traffic Sign Detection, Tracking, and Recognition Using Color, Shape, and Motion Information

[24] T. H. Cormen, Leiserson, Rivest, Stein: Introduction to Algorithms, McGraw-Hill, 2002

[25] "Boost documentation", 02. 02. 2010., http://www.boost.org/doc/libs/1_42_0

[26] G. Bradski, A. Kaehler: Learning OpenCV, O'Reilly Media, 2008

Naslov, sažetak i ključne riječi

Naslov

Detekcija i raspoznavanje prometnih znakova u video snimci

Sažetak

Opisana je metoda za detekciju i raspoznavanje trokutastih znakova iz video sekvenci dobivenih kamerom postavljenom na krov pokretnog vozila. Iako je sustav primijenjen na specifičan problem, opisane ideje i algoritmi su primjenjivi na mnogo šire područje iz domene računalnog vida. Jednostavnim proširenjem sustava moguće je ostvariti detekciju i raspoznavanje ostalih tipova prometnih znakova. Korištena metoda se sastoji od tri glavne komponente, prva komponenta je sustava za detekciju baziran na kombinaciji algoritma Viole i Jonesa te neuronske mreže koja smanjuje postotak lažnih detekcija. Drugi dio se sastoji od kombinacije SVM klasifikatora i neuronske mreže za raspoznavanje tipa prometnog znaka i konačno treći dio koji objedinjuje pojedinačne detekcije i klasifikacije u robustan sustav za raspoznavanje prometnih znakova iz video sekvenci. Konačni rezultati sadrže informaciju o položaju i tipu prometnih znakova sadržanih u video sekvenci.

Ključne riječi

raspoznavanje prometnih znakova, umjetne neuronske mreže, stroj s potpornim vektorima, algoritam Viole i Jonesa

Title, abstract and keywords

Title

Traffic sign detection and recognition in a video sequence

Abstract

We have presented an efficient system for detection and classification of triangular traffic signs from video sequences captured by a camera mounted on top of a moving vehicle.
Although the system described here is applied to a specific problem, the ideas presented have a much broader application. On the other hand the presented system is easily adoptable to other types of traffic signs. The described system consists of three main parts: Viola Jones cascaded detector followed by a Neural network trained to decrease false positive rate; a combination of SVM and Neural network for classification and finally a robust method which unifies the results of individual frames. Final results give information about the position and label, calculated from several consecutive detections, for all traffic signs contained in the video sequence.

Keywords

traffic signs recognition, artificial neural networks, support vector machine, Viola Jones detector