SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1557

PREPOZNAVANJE OBJEKATA KLASIFIKACIJOM HISTOGRAMA ORIJENTACIJE GRADIJENTA STROJEM S POTPORNIM VEKTORIMA

Alan Sambol

Zagreb, lipanj 2010.

Sadržaj

Uvod

Razvojem računarstva i povećanjem moći obrade podataka, ljudi su počeli rješavati vrlo teške probleme koji su do tada bili gotovo nerješivi, i to tehnikom grube sile (engl. *brute force*). Međutim, postoji cijeli niz problema koje uporabom ove tehnike ne možemo riješiti, kao što je npr. raspoznavanje objekata. Raspoznavanje uzoraka je znanstvena disciplina u području računarske znanosti i umjetne inteligencije čiji je cilj klasifikacija objekata u kategorije ili razrede.

Postoji velik broj različitih algoritama za klasifikaciju objekata, a jedan od njih je i Stroj s potpornim vektorima. On pripada skupu algoritama za nadgledano strojno učenje (učenje s učiteljem), što znači da je na temelju označenih uzoraka, kojima unaprijed znamo pripadnost određenom razredu, u stanju odrediti da li novi, nepoznati uzorak pripada tom istom razredu.

U ovom radu rješava se problem klasifikacije prometnih znakova u slijedovima slika pribavljenih iz perspektive vozača. Iz slika se izlučuju značajke temeljene na histogramima orijentacije gradijenta, što se zatim koristi kao ulaz za algoritam Stroja s potpornim vektorima.

Rad je strukturiran na način da ćemo se u prvom poglavlju upoznati s algoritmom izlučivanja značajki temeljenih na histogramima orijentacije gradijenta te s algoritmom Stroja s potpornim vektorima, zatim će u drugom poglavlju biti opisana programska implementacija i način korištenja razvijenih alata. Naposlijetku, u trećem poglavlju, bit će opisani eksperimentalni rezultati dobiveni različiti postupcima učenja i izlučivanja značajki.

1. Korišteni algoritmi, metode i izvorne slike

U ovom poglavlju bit će opisano nekoliko različitih načina odabira značajki slika prometnih znakova te će biti opisan postupak učenja i klasifikacije objekata Strojem s potpornim vektorima.

1.1. Reprezentacija prometnih znakova

Slike prometnih znakova potrebno je predstaviti na prikladan način kako bi bilo moguće uspješno provesti klasifikaciju. Odabir značajki koje dobro razdvajaju razrede uvelike utječe na uspješnost klasifikacije.

1.1.1. Ulazne slike

Kao ulazni skup slika korišten je skup T2010 [1]. Slike tog skupa dobivene su iz video zapisa snimljenog tokom vožnje vozilom. Za izlučivanje slika znakova s video zapisa korišten je programski alat *Marker*, koji omogućava ručno označavanje znaka, a nudi i mogućnost detekcije znakova algoritmom *Viola-Jones*.

Različiti vremenski uvijeti, brzina kretanja vozila te udaljenost znaka od objektiva kamere uzroci su znatnih varijacija kvalitete slika. Još jedan neželjen efekt kamere su smetnje nastale preplitanjem slike (engl. *interlacing*) - način izrade slike tako da se prvo snimi svaka druga linija, a zatim preostale te se od takve kombinacije parne i neparne poluslike radi puna slika. Pri brzim pokretima kamere smetnje uzrokovane preplitanjem slike vrlo su izražene. Primjer slike znaka sa smetnjama uzrokovanim preplitanjem prikazan je na slici 1.1.

Slika 1.1 Primjer slike znaka sa smetnjama uzrokovanim preplitanjem

Najjednostavnija metoda za rješenje problema preplitanja (engl. *deinterlacing*) jest metoda odbacivanja polovice linija te interpolacija linija koje nedostaju. Slike istog znaka nastale metodom odbacivanja parnih, odnosno neparnih linija, prikazane su na slici 1.2.

Slika 1.2 Slike nastale odbacivanjem linija

Stroj s potpornim vektorima kao ulaz očekuje vektore značajki jednakih veličina za sve uzorke, zbog čega je potrebno normalizirati dimenzije ulaznih slika.

1.1.2. Vektori značajki temeljeni na boji

Osnovni zapis prikaza boja u slikama jest sustav RGB (engl. *red, green, blue*). Svaki slikovni element (engl. *pixel*) predstavljen je s tri komponente: crvenom, zelenom i plavom, od kojih svaka poprima vrijednost između 0 i 255. Vektor značajki temeljen na RGB vrijednostima formira se na način da se svaki od redom poredanih slikovnih elemenata predstavi s tri vrijednosti, po jednom za svaku komponentu. Za normalizirane slike dimenzija 24x24 slikovna elementa vektor značajki ima $24 \cdot 24 \cdot 3 = 1728$ elemenata.

Sustav boja YUV se također sastoji od tri komponente: luminantne (Y), koja određuje svjetlinu te krominantnih komponenti (U i V), koje određuju boju. Konverzija iz sustava RGB u YUV definirana je jednadžbom (1.1).

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{1.1}$$

Komponenta koju koristimo za formiranje vektora značajki jest luminantna komponenta, a izračunavamo je prema jednadžbi (1.2).

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \quad (1.2)$$

Primjer luminantne komponente prikazan je na slici 1.3.

Slika 1.3 Luminantna komponenta slike

Vektor značajki sada čini po jedna vrijednost za svaki slikovni element. Za normalizirane slike dimenzija 24x24 slikovna elementa vektor značajki ima $24 \cdot 24 = 576$ slikovnih elemenata.

Odbacivanjem značajki koje nisu diskriminacijske (engl. *interset features*) postižemo bolje rezultate klasifikatora. Obično na raspolaganju imamo relativno malen skup uzoraka za učenje, a teorija raspoznavanja uzoraka nam govori da je za uspješnu klasifikaciju poželjan broj uzoraka u skupu za učenje jednak $(3\ do\ 5) \cdot broj\ značajki \cdot broj\ razreda.$ [2]

1.1.3. Histogram orijentacije gradijenta

Osnovna ideja histograma orijentacije gradijenta je podjela slike u više dijelova te izrada histograma koji nam govore o učestalosti smjera gradijenata slikovnih elemenata unutar svakog dijela.

Pri određivanju većine deskriptora slika, prvi korak čini normalizacija boje i svjetline slike. Međutim, taj korak pri izračunu histograma orijentacije gradijenta možemo preskočiti jer je učinak na performanse sustava zanemariv. [3] Prvi korak jest izračun vrijednosti gradijenata slika. Konvolucijom izvorne slike I s filterima $D_x = [-1 \quad 0 \quad 1]$ i $D_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$ dobivamo dvije nove slike, $I_x = I * D_x$ i $I_y = I * D_y$. Primjer konvolucije slike s filtrom dimenzija 3x2 dan je na slici 1.4.

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}	I_{28}	I_{29}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}	I_{38}	I_{39}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}	I_{48}	I_{49}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}	I_{58}	I_{59}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}	I_{68}	I_{69}

K_{11}	K_{12}	K_{13}
K_{21}	K_{22}	K_{23}

Slika 1.4 Primjer slike i filtra a za konvoluciju [4]

Konvolucija se izvodi pomicanjem filtra preko svih dijelova slike koje filtar može u potpunosti prekriti, počevši od gornjeg lijevog ugla. Svaka pozicija filtra određuje jedan izlazni slikovni element, čija vrijednost se računa množenjem vrijednosti filtra i odgovarajućeg slikovnog elementa za svaku ćeliju filtra te zbrajanjem svih tih vrijednosti. Primjer izračuna izlazne vrijednosti slikovnog elementa O_{57} prikazan je jednadžbom (1.3).

$$O_{57} = I_{57} \cdot K_{11} + I_{58} \cdot K_{12} + I_{59} \cdot K_{13} + I_{67} \cdot K_{21} + I_{68} \cdot K_{22} + I_{69} \cdot K_{23} \tag{1.3}$$

Općeniti matematički izraz za izračun konvolucije dan je jednadžbom (1.4), gdje su dimenzije slike $M \times N$, dimenzije filtra $m \times n$, varijabla i ide od 1 do $M - m + 1$, a varijabla j od $N - n + 1$.

$$O(i,j) = \sum_{k=1}^{m} \sum_{l=1}^{n} I(i + k - 1, j + l - 1) \cdot K(k,l) \tag{1.4}$$

Nakon provedenog filtriranja slike, računamo iznos i orijentaciju gradijenta pojedinih slikovnog elementa. Iznos gradijenta računamo jednadžbom (1.5).

$$|G| = \sqrt{I_x^2 + I_y^2} \tag{1.5}$$

Orijentaciju gradijenta računamo jednadžbom (1.6).

$$\theta = \arctan \frac{I_y}{I_x} \tag{1.6}$$

Sljedeći korak je podjela slike u više kvadratnih ćelija te izračun histograma gradijenta za svaku od tih ćelija. Svaki slikovni element unutar ćelije utječe na formiranje histograma, koji nam govori koliko slikovnih elemenata unutar ćelije ima smjer gradijenta u odgovarajućem intervalu (engl. *bin*). Postoji nekoliko opcija za izračun doprinosa slikovnih elemenata histogramu. Najjednostavniji način je u svaki element histograma pohraniti broj slikovnih elemenata koji su glasovali za taj smjer. Drugi, efikasniji način jest pohraniti ukupnu sumu iznosa (ili kvadratnih korijena iznosa) gradijenata svih slikovnih elemenata koji su glasovali za taj smjer. Taj način korišten je u originalnoj implementaciji Dalala i Triggsa [3]. Još jedan mogući način jest uzeti u obzir odstupanje smjera od središta pojedinog intervala. Ukoliko je smjer gradijenta pomaknut za neki δ od središta intervala, doprinos slikovnog elementa tom intervalu iznosi $1-\delta$, dok doprinos bližem od susjedna dva intervala iznosi δ, pod pretpostavkom da širine svih intervala iznose 1.

Zahvaljujući lokalnim varijacijama u osvjetljenju i kontrastu boja, iznosi gradijenata također variraju. Stoga je za poboljšane performansi sustava ključna lokalna normalizacija kontrasta slike, koju postižemo grupiranjem ćelija u veće cjeline (blokove), normaliziranjem kontrasta svake ćelije bloka te konkatenacijom vektora svih ćelija bloka. Dodatno poboljšanje performanse postižemo preklapanjem blokova, tako da svaka ćelija doprinosi konačnom opisniku više puta. Konačni opisnik slike je tada vektor normaliziranih histograma blokova.

Postoji nekoliko načina normalizacije blokova. Neka je *v* nenormalizirani vektor opisnika, $\|v_k\|$ njegova k-norma za *k=1,2* i neka je *e* proizvoljna mala konstanta čija vrijednost neće utjecati na rezultat. Tada normalizacijski faktori mogu biti jedan od sljedećih.

$$\text{L2-norma:} \quad \frac{v}{\sqrt{\|v\|_2^2+e^2}} \tag{1.7}$$

$$\text{L1-norma:} \quad \frac{v}{\|v\|_1+e} \tag{1.8}$$

$$\text{L1-sqrt norma:} \quad \sqrt{\frac{v}{\|v\|_1+e}} \tag{1.9}$$

1.2. Stroj s potpornim vektorima

Stroj s potpornim vektorima (engl. *Support Vector Machine*) je binarni klasifikator koji konstrukcijom hiperravnine u visoko-dimenzionalnom prostoru stvara model koji predviđa kojem od dva razreda pripada novi uzorak.

1.2.1. Linearno odvojivi razredi

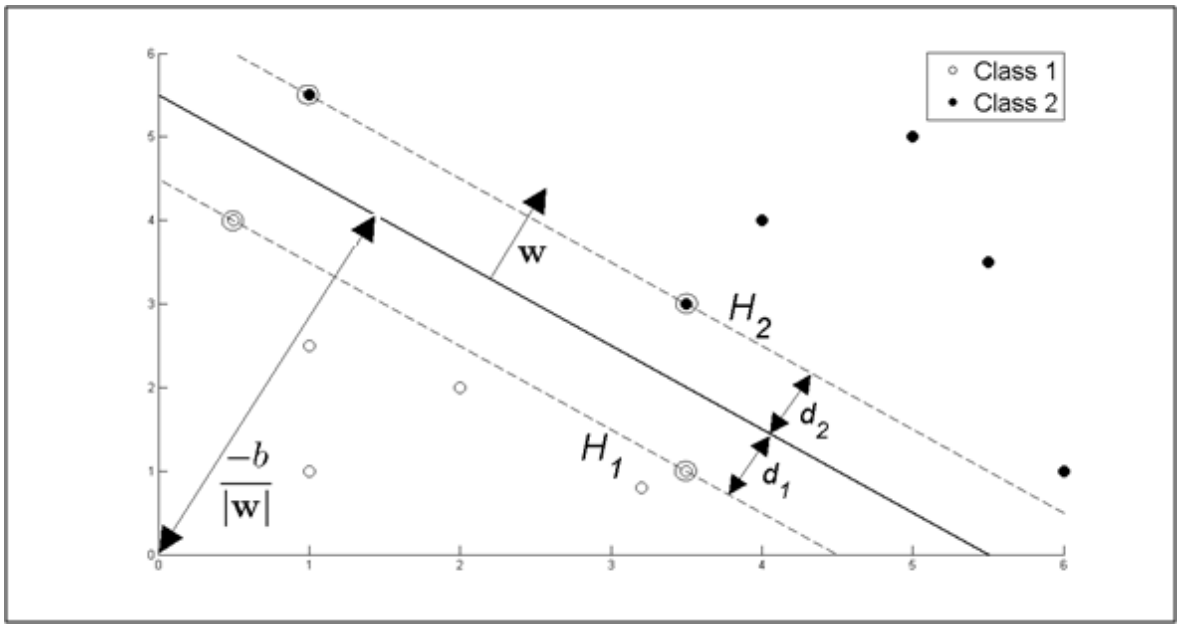
U skupu za učenje imamo *L* vektora (točaka u *D*-dimenzionalnom prostoru), gdje svaki uzorak x_i ima D atributa, odnosno komponenti vektora te pripada jednom od dva razreda $y_i = -1\ ili\ 1$. Oblik jednog ulaznog podatka prikazan je izrazom (1.10).

$$\{x_i, y_i\} \text{ gdje je } i = 1 \dots L,\ y_i \in \{-1,1\}, x \in R^D \tag{1.10}$$

Pretpostavlja se da su podaci linearno odvojivi, što znači da možemo nacrtati pravac u koordinatnom sustavu s osima x_1 i x_2 za slučaj $D = 2$, odnosno hiperravninu za slučaj $D > 2$. Hiperravninu opisujemo izrazom $w \cdot x + b = 0$, gdje je:

- *w* normala hiperravnine
- $\frac{b}{\|w\|}$ okomita udaljenost hiperravnine od ishodišta koordinatnog sustava

Potporni vektori su uzorci najbliži razdvajajućoj hiperravnini i zato se najteže klasificiraju, a cilj Stroja s potpornim vektorima jest da odabere hiperravninu maksimalno udaljenu od najbližih uzoraka oba razreda. Grafički prikaz jednostavnog dvodimenzionalnog slučaja prikazan je na slici 1.5.



Slika 1.5 Grafički prikaz dva linearno odvojiva razreda

Implementacija Stroja s potpornim vektorima sada se svodi na odabir parametara w i b, takvih da ulazne podatke možemo opisati sljedećim izrazima.

$$x_i \cdot w + b \geq +1 \;\; za \;\; y_i = \; +1 \tag{1.12}$$

$$x_i \cdot w + b \leq -1 \;\; za \;\; y_i = \; -1 \tag{1.13}$$

Kombinacijom ta dva izraza dobivamo

$$y_i(x_i \cdot w + b) - 1 \geq 0, \;\; \forall i \tag{1.14}$$

Ravnine H_1 i H_2 na kojima leže potporni vektori (označeni kružićima na slici) možemo opisati sljedećim izrazima.

$$x_i \cdot w + b = +1 \;\; \text{za} \;\; H_1 \tag{1.15}$$

$$x_i \cdot w + b = -1 \;\; za \;\; H_2 \tag{1.16}$$

Definiramo vrijednosti d_1 i d_2 kao udaljenosti od H_1 i H_2 do hiperravnine. Ekvidistantnost hiperravnine od H_1 i H_2 podrazumijeva $d_1 = d_2 = \frac{1}{\|w\|}$. Tu vrijednost nazivamo marginom Stroja s potpornim vektorima. Ako želimo odabrati hiperravninu maksimalno udaljenu od potpornih vektora, moramo maksimizirati marginu, što je ekvivalentno pronalaženju:

$$min\|w\| \quad \text{takav da} \quad y_i(x_i \cdot w + b) - 1 \geq 0, \;\; \forall i \tag{1.17}$$

1.2.2. Linearno neodvojivi razredi

Ako želimo proširiti funkcionalnost Stroja s potpornim vektorima na linearno neodvojive razrede, potrebno je ublažiti uvjete (1.12) i (1.13) uvođenjem nenegativne vrijednosti ξ_i:

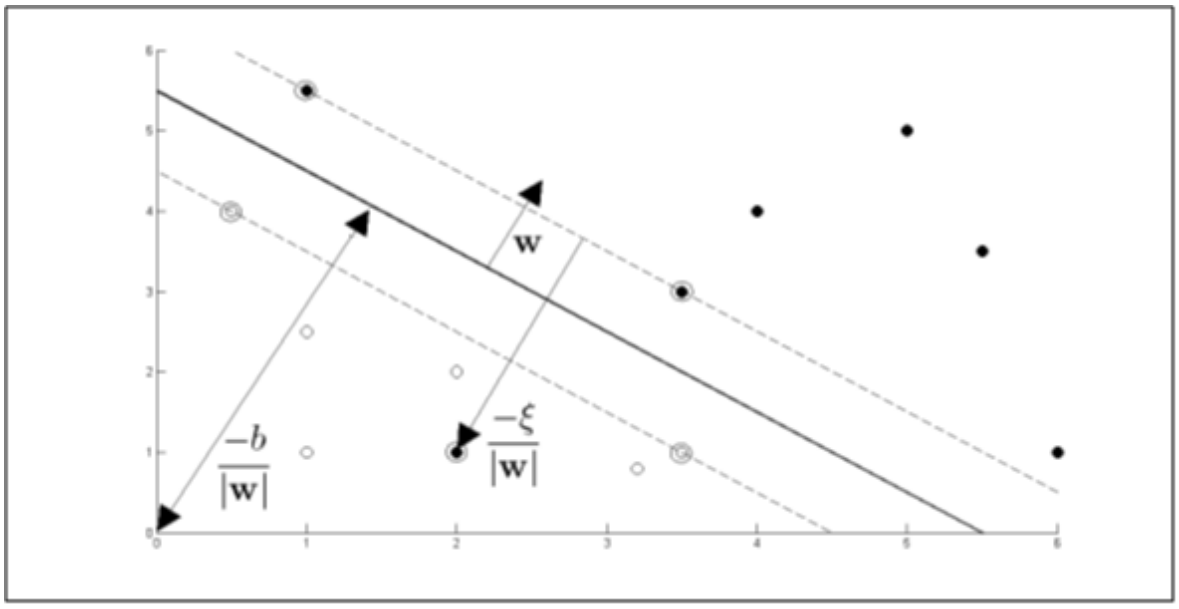
$$x_i \cdot w + b \geq +1 - \xi_i \quad za \quad y_i = +1 \tag{1.18}$$

$$x_i \cdot w + b \leq -1 + \xi_i \quad za \quad y_i = -1 \tag{1.19}$$

Kombinacijom ta dva izraza dobivamo

$$y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0, \quad \xi_i \geq 0 \; \forall i \tag{1.20}$$

Ova metoda naziva se metoda meke margine (engl. soft *margin method* [5]), a izvorno je nastala s idejom dozvoljavanja pogrešnog označavanja razreda prije samog postupka učenja. Na slici 1.6 prikazana je hiperravnina (pravac) kroz dva linearno neodvojiva razreda. Vidljiv je i uzorak s pogrešne strane hiperravnine zbog kojeg prostor nije linearno odvojiv. Mjera udaljenosti tog uzorka od pripadajućeg potpornog vektora jest ξ.



Slika 1.6 Prikaz dva linearno neodvojiva razreda

Sada se odabir razdvajajuće hiperravnine svodi na pronalaženje:

$$min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L}\xi_i \quad \text{takav da} \quad y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0, \; \forall i. \tag{1.21}$$

Vrijednost *C* u izrazu 1.11 predstavlja faktor pogreške, kojim dozvoljavamo određene pogreške pri treniranju, bez čega pronalazak hiperravnine ne bi bio moguć.

1.2.3. Jezgreni trik

Do sada opisani (linearni) klasifikator zapravo se naziva Klasifikator optimalne granice (engl. Maximum Margin Classifier). Stroj s potpornim vektorima je poopćeni Klasifikator optimalne granice za nelinearnu klasifikaciju, što se postiže postupkom poznatim pod nazivom Jezgreni trik (engl. *Kernel Trick* [5]).

Osnovna ideja jest u izrazu (1.21) zamijeniti ulazni vektor značajki x_i s funkcijom $\phi(x_i)$, koja ulazni vektor preslikava iz *n*-dimenzionalnog u *m*-dimenzionalni prostor, uz $m \gg n$. U novom, *m*-dimenzionalnom, prostoru su uzorci linearno odvojivi. Problem predstavlja računanje unutarnjeg produkta vektora $\phi(x_i)$ i *w* jer je nova dimenzionalnost puno veća, ponekad i beskonačna. Međutim, moguće je napisati Jezgrenu funkciju (engl. *Kernel function*) $K(x_i, x_j) = \phi(x_i)^T \cdot \phi(x_j)$, koju je moguće izračunati puno jednostavnije nego eksplicitno računati unutarnji produkt $\phi(x_i)^T \cdot \phi(x_j)$.

Postoji nekoliko standardnih oblika Jezgrenih funkcija:

- Linearna

$$K(x_i, x_j) = {x_i}^T \cdot x_j$$

- Polinomna

$$K(x_i, x_j) = ({x_i}^T \cdot x_j + a)^b$$

- Gaussova (RBF, engl. *Radial Basis Function*)

$$K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}, \ \gamma > 0$$

Gaussova jezgra je jezgra s najširom primjenom. Ona odgovara preslikavanju u beskonačno-dimenzionalni prostor. Sve što je linearno razdvojivo u početnom prostoru značajki, razdvojivo je i u prostoru određenom ovom jezgrom. Parametar γ određuje širinu „zvona“ Gaussove krivulje.

- Racionalna kvadratna [6]

$$K(x_i, x_j) = 1 - \frac{\|x_i - x_j\|^2}{\|x_i - x_j\|^2 + \tau}$$

Ova jezgra je dobra zamjena za Gaussovu jezgru ukoliko je potrebno izbjeći potenciranje.

- Sigmoidalna

$$K(x_i, x_j) = tanh(\kappa x_i{}^T \cdot x_j + a),$$

Ključ dobre klasifikacije Strojem s potpornim vektorima leži u odabiru parametara jezgrene funkcije i ranije spomenutog parametra *C* – faktora pogreške.

1.2.4. Klasifikacija između više razreda

Stroj s potpornim vektorima je binarni klasifikator, što znači da može razvrstati neki nepoznati uzorak u jedan od dva razreda. Problem klasifikacije uzoraka u više od dva razreda ne možemo riješiti samo jednim klasifikatorom. Postoje dva načina kako taj problem možemo riješiti:

1. Konstruirati *n* binarnih klasifikatora od kojih svaki svrstava uzorke ili u jedan od razreda ili u preostalih *n*-1 razreda. Ovakav pristup se naziva „jedan-protiv-svih“ (engl. *one-versuss-all*). Klasifikacija novih uzoraka izvodi se strategijom „pobjednik-odnosi-sve“ (engl. *winner-takes-all*), što znači da svaki klasifikator, osim izlaza, daje i mjeru sigurnosti u svoj izbor. Od svih klasifikatora čiji izbor nije „all“ uzima se izbor onoga koji je najsigurniji u svoj odabir. Ukoliko svi klasifikatori odaberu „all“, vjerojatno se radi o nepostojećem razredu ili uzorku kojeg nije moguće klasificirati, a kao odabir se najčešće uzima odabir suprotan onom klasifikatoru koji je odabrao „all“ s najmanjom sigurnošću.
2. Konstruirati $\frac{n(n-1)}{2}$ binarnih klasifikatora od kojih svaki svrstava uzorke u jedan od dva razreda. Ovakav pristup naziva se „jedan-protiv-jednog“ (engl. *one-versus-one*). Klasifikacija novih uzoraka izvodi se postupkom glasanja. Svaka binarna klasifikacija smatra se jednim glasovanjem za jedan od dva razreda, čime se broj glasova za razred koji je odabran pri toj binarnoj klasifikaciji uvećava za jedan. Nakon što se izvede svih $\frac{n(n-1)}{2}$ postupaka glasovanja, uzorak se pridjeljuje razredu s najviše postignutih glasova. Ukoliko dva razreda imaju jednak broj glasova, najčešće se odabire razred s manjim indeksom.

2. Programska izvedba

U sljedećim potpoglavljima bit će objašnjena programska implementacija pomoćnih programa za pripremu ulaznih slika, programa za ekstrakciju značajki temeljenih na boji, programa koji implementira histograme orijentiranih gradijenata, način njihovog korištenja te način korištenja Stroja s potpornim vektorima implementiranog u biblioteci *libSVM*.

2.1. Priprema podataka i izlučivanje značajki

Pretpostavlja se da su ulazne slike izrezane slike znakova raspoređene po direktorijima koji nose šifru znaka, npr. *A01*. Imena i dimenzije početnih slika znakova nisu bitni za ispravan rad programa. Prvo je potrebno normalizirati dimenzije slika te ih razmjestiti u odgovarajuće direktorije. Za to je implementiran razred `ImageResizer`, koji iz ulaznog direktorija u kojem su poddirektorji nazvani po znakovima odabire znakove, smješta ih u novi direktorij u numerirane poddirektorije te im usput skalira dimenzije na željeni iznos. U polju `signs` pohranjena je lista naziva znakova koji će se koristiti pri učenju ili klasifikaciji. Za svaki razred znaka koji je u tom polju poziva se funkcija `void resizeImages(SignId signId, File inputFolder, File outputFolder, int index, String trainOrTest)`, gdje `index` označava redni broj znaka koji će se koristiti pri klasifikaciji, dok `trainOrTest` određuje radi li se o pripremi skupa podataka za učenje ili testiranje. Sada se poddirektoriji sa normaliziranim znakovima nalaze u direktoriju `<root>/<outputFolder>`, te nose naziv `<trainOrTest><index>`.

Sljedeći korak je izlučivanje značajki iz normaliziranih slika. Za to je implementiran razred `FeatureExtractor`, koji iz strukture direktorija pripremljene u prošlom koraku uzima znak po znak, računa njegove značajke, te stvara datoteku `<trainOrTest>.dat` unutar direktorija `<outputFolder>`, koja sadrži značajke u formatu u kojem ih očekuje implementacija Stroja s potpornim vektorima u biblioteci libSVM. U svakom retku te datoteke nalazi se jedan vektor, odnosno uzorak, u sljedećem formatu:

<ni> 1:<x1> 2:<x2> ... D:<xD>, gdje <ni> predstavlja oznaku razreda (redni broj razreda dobiven u prvom koraku, $n_i \in [1, n]$), a parovi oblika 1:<x1> predstavljaju par vrijednosti rednog broja značajke i njezine vrijednosti (ukupno *D* značajki).

2.1.1. Izlučivanje značajki temeljenih na boji

Unutar razreda `FeatureExtractor` se, ukoliko želimo značajke temeljene na RGB vrijednostima slika, za svaku sliku znaka poziva funkcija

`ArrayList<Double> calculateFeaturesRGB(BufferedImage img).`

Ona čita ulaznu sliku, za svaki slikovni element vadi R, G i B vrijednost te ih redom dodaje u polje koje vraća.

Ukoliko želimo značajke temeljene na luminantnoj komponenti, pozivamo funkciju

`ArrayList<Double> calculateFeaturesLuma(BufferedImage img),`

koja je vrlo slična prethodnoj, samo što na temelju R, G i B vrijednosti računa luminantnu komponentu za svaki slikovni element te ih redom smješta u izlazno polje.

2.1.2. Histogrami orijentacije gradijenta

Želimo li kao značajke koristiti histograme orijantacije gradijenta, pozivamo funkciju `ArrayList<Double> calculateFeaturesHOG(BufferedImage img)`, koja instancira novi primjerak razreda `HOG` te obavlja potrebne izračune. Razred `HOG` sadrži sljedeće članske varijable:

- `int cellWidth` – širina ćelija
- `int cellHeight` – visina ćelija
- `int blockWidth` – širina bloka, mora biti višekratnih od `cellWidth`
- `int blockHeight` – visina bloka, mora biti višekratnik od `cellHeight`
- `int numOfBinsPerCell` – broj diskretizacijskih razina (kanala) za smjer gradijenta

Dimenzije slike moraju biti višekratnik od `cellWidth` i `cellHeight`.

Nakon instanciranja primjerka razreda `HOG`, nad njim se poziva metoda `computeGradients`. Programski jezik *Java* sadrži implementaciju konvolucije nad slikama u razredu `ConvolveOp` paketa `java.awt.image`. Taj razred nudi metodu `void filter(BufferedImage src, BufferedImage dst)` koja provodi konvoluciju nad slikom `src`, te rezultat zapisuje u sliku `dst`. Problem u tome je što naš filtar sadrži negativne vrijednosti te bi nakon konvolucije vrijednosti nekih slikovnih elemenata trebale biti negativne. Međutim, razred `BufferedImage` ne dopušta negativne vrijednosti boje slikovnih elemenata, već ih automatski postavlja u nulu. Stoga je potrebno ručno implementirati operaciju konvolucije. Srećom, imamo vrlo jednostavan filtar [-1 0 1] pa je moguće izbjeći kompletnu implementaciju konvolucije jednostavnim manipulacijama nad slikovnim elementima, što je prikazano u kôdu 2.1.

```
for(int i=0; i<in.getWidth(); i++) {
      for(int j=0; j<in.getHeight(); j++) {
            if(i<(in.getWidth()-2) && j<(inp.getHeight()- 2)) {
                  outputX[j][i] = input[j][i] - input[j][i+2];
                  outputY[j][i] = input[j][i] - input[j+2][i];
            } else {
                  outputX[j][i] = 0;
                  outputY[j][i] = 0;
            }
      }
}
```

Kôd 2.1 – Konvolucija slike filtrom [-1 0 1]

Za svaki slikovni element, unutar iste *for* petlje odmah se izračunava i iznos i smjer gradijenta, instancira se novi primjerak razreda `Gradient` te se on kao vrijednost smješta u mapu `HashMap<Pixel, Gradient> gradients`, dok se kao ključ uzima trenutni slikovni element. Nakon izračuna svih gradijenata, polja izlaznih slika pretvaraju se u slike tipa `BufferedImage`, prije čega je potrebno vrijednosti svih slikovnih elemenata koje su manje od nule postaviti na nulu jer slike u tom formatu ne dopuštaju negativne vrijednosti. To se koristi isključivo za opcionalni prikaz konvoluiranih slika, dok se za daljnje izračune histograma i dalje koriste prethodne (pozitivne i negativne) vrijednosti.

Sljedeća funkcija koja se poziva nad primjerkom razreda `HOG`, `void computeCellHistograms()`, prelazi preko svih ćelija te računa histograme gradijenta svih slikovnih elemenata unutar svake ćelije, od kojih svaki iznosom svog gradijenta doprinosi odgovarajućem smjeru u histogramu. Nakon što izračuna histograme svih ćelija, sprema ih kao vrijednost u mapu `HashMap<Pixel, Histogram> cellHistograms`, dok se kao ključ uzima gornji lijevi slikovni element ćelije.

Nakon izračuna histograma ćelija, slijedi njihova normalizacija i formiranje histograma blokova pozivom funkcije `void computeBlockHistograms()`. Ona prolazi kroz blokove slike, konkatenira histograme svih ćelija unutar bloka te ih zatim normalizira korištenjem L2-norme. Kôd 2.2 prikazuje izračun histograma blokova te njihovu normalizaciju:

```
double l2norm = 0;
double e = 0.01;
for(int y=i; y<i+blockHeight; y+=cellHeight) {
      for(int x=j; x<j+blockWidth; x+=cellWidth) {
            Pixel pixel = new Pixel(x, y);
            Histogram h = this.cellHistograms.get(pixel);

            for(int a=0; a<numOfBinsPerCell; a++) {
                  blockHistogram.channels[a] +=
                  h.channels[a];
            }
      }
}
for(double channel : blockHistogram.channels) {
      l2norm += channel*channel;
}
l2norm += e*e;
l2norm = Math.sqrt(l2norm);
for(int a=0; a<numOfBinsPerCell; a++) {
      double newvalue = blockHistogram.channels[a] / l2norm;
      blockHistogram.channels[a] = newvalue;
}
this.blockHistograms.put(upperLeft, blockHistogram);
```

Kôd 2.2 – Izračun i normalizacija histograma blokova

Kao vrijednosti značajki koje funkcija `calculateFeaturesHOG` stavlja u izlazno polje možemo odabrati normalizirane histograme ćelija ili normalizirane histograme blokova. Originalna implementacija histograma orijentiranih gradijenata kao značajke koristi normalizirane histograme blokova.

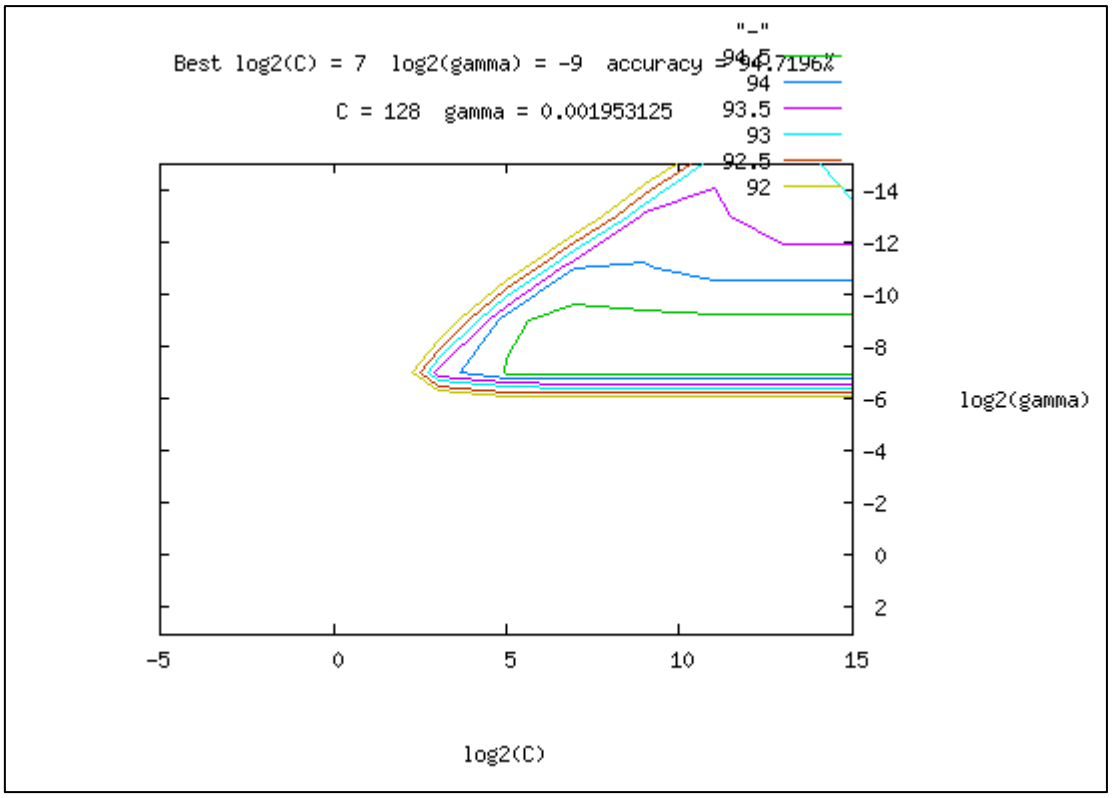
2.2. Klasifikacija pomoću biblioteke libSVM

Biblioteka *libSVM* [7] (engl. *A Library for Support Vector Machines*) sadrži podršku za klasifikaciju uzoraka Strojem s potpornim vektorima, uz niz dodatnih alata koji olakšavaju pripremu ulaznih podataka i odabir ispravnih parametara. Implementirana je u programskim jezicima *C++, Java, Python* i *Matlab*. Korištena je implementacija u jeziku *C++*.

Prije samog treniranja, potrebno je provesti postupak skaliranja ulaznih podataka na raspon [-1, 1] pozivom programa *svm-scale* na sljedeći način:

```
svm-scale <ulazna_datoteka>.dat > <ulazna_datoteka>.scale.dat
```

Nakon toga potrebno je odabrati optimalne parametre C i γ za RBF (Gaussovu) jezgru. Biblioteka *libSVM* nudi alat za odabir optimalnih parametara postupkom unakrsne validacije (engl. *cross-validation*) u skripti *grid.py*. Unakrsna validacija obavlja se tako da se skup ulaznih podataka za učenje podijeli u n podskupova. Tada se svaki od n podskupova testira korištenjem SVM-a naučenog na preostalih (n-1) podskupova. Parametri C i γ se eksponencijalno povećavaju te se svaki put izvodi unakrsna validacija. Nakon što se pronađu najbolji parametri, prelazi se na detaljniju unakrsnu validaciju oko dobivenih parametara kako bi se dodatno povećala točnost. Skripta *grid.py* prima prethodno skalirani skup ulaznih vrijednosti, te crta graf uspješnosti unakrsne validacije s različitim parametrima. Osi grafa su $log_2\gamma$ i $log_2 C$. Primjer grafa prikazan je na slici 2.1.



Slika 2.1 Primjer grafa unakrsne validacije

Na slici je vidljivo da se najbolja unakrsna validacija postiže uz parametre $C = 128$ i $\gamma = 0.001953125$, uz točnost of 94.7196% (zelena linija prikazuje točnost od 94.5%).

Nakon odabira optimalnih parametara *C* i γ možemo početi treniranje SVM-a pokretanjem alata *svm-train* kojem predajemo parametre te ulaznu datoteku sa skaliranim značajkama. Primjer pokretanja za prethodno nađene parametre:

```
svm-train -c 128 -g 0.001953125 train.scale.dat
```

Izlaz ovog programa je datoteka *train.dat.model* koja predstavlja model kojim vršimo daljnju klasifikaciju novih uzoraka.

Klasifikacija novih uzoraka vrši se programom *svm-predict*, na sljedeći način:

```
svm-predict <test_datoteka> <model > <izlazna_datoteka>
```

U izlaznoj datoteci nalaze se indeksi razreda u koje su se klasificirali uzorci iz ispitne datoteke, svaki u svom retku. Osim izlazne datoteke, program na standardni izlaz ispisuje uspješnost testiranja (*broj uspješno klasificiranih uzoraka / ukupan broj uzoraka*).

3. Eksperimentalni rezultati

U tablici 1 prikazani su rezultati dobiveni treniranjem nad skupom od 1979, odnosno 3958 (deinterlaced) slika znakova. Odabrani su samo oni razredi kod kojih u skupu za učenje imamo više od 30 znakova te je time broj razreda jednak 14. To su sljedeći znakovi: A01, A03, A05, A08, A09, A10, A11, A12, A14, A20, A33, A34, A44 i A46. Skup znakova za testiranje sastoji se od 724 znaka. Skupovi za učenje su slike s preplitanjem ili bez preplitanja (engl. *interlaced* / *deinterlaced*), s različitim značajkama: RGB vrijednosti, luminantne vrijednosti (*luma*), histogrami orijentacije gradijenta ćelija (*hog_cell*) i blokova (*hog_block*). Slike znakova su skalirane na dimenzije 24x24 za značajke temeljene na boji te 64x64 za značajke temeljene na histogramima. Dimenzije ćelija kod histograma su 8x8, dok su dimenzije blokova 16x16 (svaki blok sadrži četiri ćelije). Broj diskretizacijskih razina je 9, što znači da kutevi od $[0°,\ 20° >$ pripadaju prvoj diskretizacijskoj razini, kutevi od $[20°,\ 40° >$ drugoj razini itd.

Tablica 1 Eksperimentalni rezultati dobiveni klasifikacijom 14 razreda

Skup za testiranje: / Skup za učenje:	Označene slike	Detekcije
interlaced - RGB	58.0052%	50.5249%
interlaced - hog_block	66.1602%	66.1602%
deinterlaced - RGB	51.3123%	48.0315%
deinterlaced - luma	56.6298%	
deinterlaced - hog_cell	58.5635%	
deinterlaced - hog_block	49.1713%	54.558%

U tablici 2 prikazani su rezultati binarnih klasifikacija nad 14 razreda znakova. Kao skup za učenje uzete su slike s preplitanjem, dok se testiranje vrši nad označenim slikama. Vektor značajki čine histogrami orijentacije gradijenata blokova.

Tablica 2 Eksperimentalni rezultati binarnih klasifikacija

[%]	A01	A03	A05	A08	A09	A10	A11	A12	A14	A20	A33	A34	A44
A03	37.6												
A05	61.2	82.5											
A08	100	99.2	100										
A09	100	99.3	100	100									
A10	89.5	36.5	83.6	100	100								
A11	97.4	56.5	95.8	100	100	100							
A12	100	98.0	100	100	100	100	100						
A14	100	68.0	78.1	100	100	100	94.7	100					
A20	100	88.0	100	100	100	100	100	100	100				
A33	93.8	85.7	97.9	99.5	99.5	96.6	97.0	100	88.1	100			
A34	100	59.7	100	100	100	100	98.6	100	100	100	83.4		
A44	100	99.2	100	98.3	100	100	100	100	100	100	100	100	
A46	100	100	100	100	100	100	100	100	100	100	99.4	100	100

Vidljivo je da su rezultati klasifikacija uglavnom vrlo dobri, s nekoliko iznimaka, npr. klasifikacija između znakova A01 i A03. Promjenom dimenzija blokova sa 16 na 8 slikovnih elemenata te dimenzija ćelija s 8 na 4 slikovna elementa, uspješnost klasifikacije podignuta je s 37.6% na 50.5%. Problem je što je time broj značajki porastao s 324 na 1764 te se vrijeme učenja znatno povećalo. Daljnjim smanjenjem dimenzija blokova i ćelija ne postižu se bolji rezultati.

Zaključak

Prepoznavanje objekata je postupak određivanja razreda (grupe) kojem taj objekt pripada. Stroj s potpornim vektorima jedan je od najraširenijih i najučinkovitijih algoritama za nadgledano strojno učenje i klasifikaciju objekata. Na temelju skupa za učenje gradi se klasifikator kojim se zatim određuje pripadnost nepoznatog objekta nekom razredu. Stroj s potpornim vektorima primarno je binarni klasifikator, no u radu je opisano nekoliko načina proširenja funkcionalnosti na proizvoljan broj razreda.

U ovom radu pokazano je da na uspješnost klasifikacije uvelike utječe broj uzoraka u skupu za učenje, kao i broj razreda u koje uzorke možemo klasificirati.

Osim toga, na uspješnost klasifikacije uvelike utječe i odabir značajki kojima predstavljamo objekte. Korištenje značajki temeljenih na histogramima orijentacije gradijenta polučilo je bolje rezultate pri klasifikaciji od značajki temeljenih na boji. Implementacija histograma dopušta različite parametre, kao što su: dimenzije ćelija, dimenzije blokova te faktor preklapanja blokova, što također znatno utječe na rezultate klasifikacije.

Primjena Stroja s potpornim vektorima na klasifikaciju značajki temeljenih na histogramima orijentacije gradijenta polučila je solidne rezultate te pruža osnovu za daljnje istraživanje.

Literatura

[1] ŠEGVIĆ, S.; BRKIĆ, K.; KALAFATIĆ, Z.; STANISAVLJEVIĆ, V.; ŠEVROVIĆ, M.; BUDIMIR, D.; DADIĆ, I. *A computer vision assisted geoinformation inventory for traffic infrastructure*. ITSC, 2010.

[2] RIBARIĆ, S. *Raspoznavanje uzoraka*. http://www.fer.hr/_download/repository/UURU_23.pdf

[3] DALAL, N.; TRIGGS, B. Histograms of Oriented Gradients for Human Detection. Montbonnot, 2005.

[4] FISHER, R.; PERKINS, S.; WALKER, A.; WOLFART, E. *Convolution*, 2003. http://homepages.inf.ed.ac.uk/rbf/HIPR2/convolve.htm

[5] FLETCHER, T. *Support Vector Machines Explained*, 2009. http://www.tristanfletcher.co.uk/SVM%20Explained.pdf

[6] KUSALIĆ, I. *Raspoznavanje prometnih znakova metodom potpornih vektora*, 2009. http://www.zemris.fer.hr/~kalfa/ZR/KusalicZR_2009.pdf

[7] CHANG, C.; LIN, C. *LIBSVM: a Library for Support Vector Machines*, 2001. http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf

[8] TISDALL-YAMADA, T. *Pattern recognition systems – Lab 5/6 – Histograms of Oriented Gradients*. http://users.utcluj.ro/~raluca/srf_2008/lab/04/lab_04e.pdf

Naslov, sažetak i ključne riječi

Naslov

Prepoznavanje objekata klasifikacijom histograma orijentacije gradijenta Strojem s potpornim vektorima

Sažetak

Osnovni cilj rada je prepoznavanje prometnih znakova u slijedovima slika pribavljenih iz perspektive vozača. Objašnjeni su negativni efekti kamere kao što su preplitanje slike te je predložen jednostavan način rješavanja tog problema. Opisano je više mogućnosti odabira značajki slike, od značajki temeljenih na boji do značajki temeljenih na histogramima orijentacije gradijenta. Detaljno je opisan algoritam izlučivanja značajki temeljenih na histogramima te njegova implementacija. Opisan je način rada klasifikatora temeljenog na potpornim vektorima te način uporabe biblioteke *libSVM* za učenje i testiranje. Na kraju su prezentirani eksperimentalni rezultati dobiveni nad različitim skupovima za učenje i testiranje.

Ključne riječi

Prometni znakovi, izlučivanje značajki, raspoznavanje objekata, klasifikacija, Stroj s potpornim vektorima, histogram orijentacije gradijenta, preplitanje slike.

Title, summary and keywords

Title

Object recognition by applying a support vector machine classifier to histograms of oriented gradients

Summary

The main goal of this thesis is the recognition of traffic signs in sequences of images obtained from the perspective of the driver. Negative effects of camera such as image interlacing are explained, and a simple way to solve that problem is suggested. Multiple feature extraction choices are described, from features based on colour to features based on histograms of oriented gradients. A detailed description of a feature extraction algorithm based on histograms and its implementation are given. A support vector based classifier and its usage through *libSVM* library is described. Finally, experimental results obtained from multiple training and testing sets are presented.

Keywords

Traffic signs, feature extraction, object recognition, classification, Support vector machine, histogram of oriented gradients, image interlacing.

Skraćenice

SVM	*Support Vector Machine*	Stroj s potpornim vektorima
RGB	*Red, green, blue*	crveno, zeleno, plavo
HOG	*Histogram of oriented gradients*	histogram orijentacije gradijenta
libSVM	*A Library for Support Vector Machines*	biblioteka za Stroj s potpornim vektorima
RBF	*Radial basis function*	radijalna temeljna funkcija