SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

PROJEKT IZ STROJNOG UČENJA

KLASIFIKACIJA ŽANRA GLAZBE POMOĆU STROJA S POTPORNIM VEKTORIMA

Đuretec Krešimir
Fabek Ivan
Gulić Matija
Lučanin Dražen
Vidačić Katarina

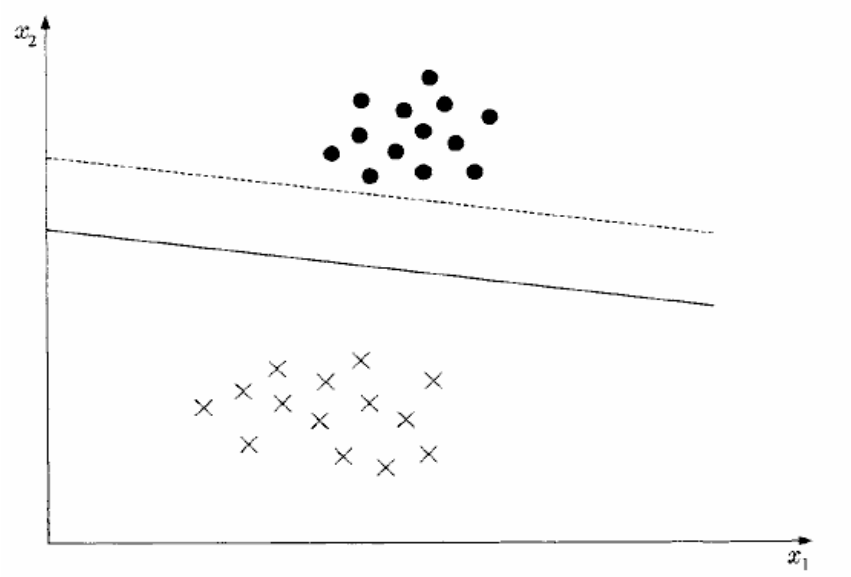
Zagreb, siječanj 2010.

Sadržaj

1. Stroj s potpornim vektorima (SVM)

1.1 Binarna klasifikacija

Mnogi binarni problemi u raspoznavanju uzoraka se rješavaju pronalaskom decizijske funkcije (hiperravnina) koja razdvaja dva skupa uzoraka. Takvih decizijskih funkcija ima beskonačno mnogo (uz pretpostavku da su razredi linearno separabilni).



Slika 1.1 Dvije moguće decizijske funkcije (preuzeto iz [6])

Dvije decizijske funkcije su prikazane na slici 1.1. Kao što se može uočiti obje pravilno razvrstavaju uzorke za učenje, no kada bi se moglo birati koju kasnije koristiti logičan odabir bi bio ona koja je označena punom linijom. Razlog tome je taj da funkcija označena punom linijom omogućuje i jednom i drugom razredu podjednake fluktuacije, tj. isprekidana linija ne omogućuje uzorcima iz razreda koji je označen crnim krugovima nikakve pomake prema dolje. To svojstvo se naziva mogućnost generaliziranja i vrlo je važno svojstvo svakog klasifikatora jer ono određuje koliko će dobro klasifikator klasificirati uzorke izvan skupa za učenje. Cilj stroja s potpornim vektorima (*eng.support vector machine ,SVM*) je upravo pronaći takvu decizijsku funkciju kod koje je udaljenost do oba razreda maksimalna.

1.1.1 Linearno separabilni razredi

Neka je zadano N uzoraka x_i (i = 1,…,N) koji pripadaju razredu ω_1 ili ω_2. Budući da su razredi linearno separabilni decizijska funkcija je oblika

$$d(x) = w^T x + w_0 \tag{1.1}$$

uz uvjet razvrstavanja

$$w^T x_i + w_0 \begin{cases} > 0 & x_i \in \omega_1 \\ < 0 & x_i \in \omega_2 \end{cases} \tag{1.2}$$

Cilj je maksimizirati udaljenost najbližih elemenata iz razreda ω_1 i ω_2 do decizijske funkcije.

Poznato je da je udaljenost neke točke do decizijske funkcije d(x) jednaka

$$\frac{|d(x)|}{\|w\|} \tag{1.3}$$

Funkcija d(x) se može skalirati na taj način da u najbližoj točci razreda ω_1 poprimi vrijednost 1 ,a u najbližoj točci razreda ω_2 poprimi vrijednost -1.
Duljina margine nakon skaliranja iznosi

$$\frac{1}{\|w\|}+\frac{1}{\|w\|}=\frac{2}{\|w\|} \tag{1.4}$$

Sad se i uvjet razvrstavanja može zapisati u obliku:

$$w^T x_i + w_0 \begin{cases} \geq 1 & x_i \in \omega_1 \\ \leq -1 & x_i \in \omega_2 \end{cases} \tag{1.5}$$

Optimizacijski problem koji se mora riješiti je pronaći minimum funkcije (1.6) uz ograničenje (1.7).

$$\mathrm{J(w)} = \frac{1}{2}\|w\|^2 \tag{1.6}$$

$$y_i(w^T x_i + w_0) \geq 1 \tag{1.7}$$

y_i je pomoćna varijabla koja ovisno o uzorku poprima vrijednost 1 ili -1 ($y_i = 1$ ako $x_i \in \omega_1$ odnosno $y_i = -1$ ako $x_i \in \omega_2$).
Ovo predstavlja kvadratni optimizacijski problem ograničen linearnom nejednadžbom.
Problem se može zapisati u neograničenom obliku.

$$F(w, w_0, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i \left[y_i (w^T x_i + w_0) - 1 \right] \tag{1.8}$$

Ovaj oblik se još naziva i Lagrangeova funkcija, a α_i Lagrangeovi multiplikatori.
Optimum zadovoljava KKT(Karush-Kuhn-Tucker) uvjete dane jednadžbama (1.9) - (1.12).

$$\frac{\partial}{\partial w} F(w, w_0, \alpha) = 0 \tag{1.9}$$

$$\frac{\partial}{\partial w_0} F(w, w_0, \alpha) = 0 \tag{1.10}$$

$$\alpha_i \geq 0 \tag{1.11}$$

$$\alpha_i \left[y_i (w^T x_i + w_0) - 1 \right] = 0 \tag{1.12}$$

Uvrštavanjem (1.8) u (1.9) i (1.10) dobivaju se jednadžbe (1.13) i (1.14).

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i \tag{1.13}$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \qquad \textbf{(1.14)}$$

Iz jednadžbe (1.12) se može uočiti da je umnožak Lagrangeovog multiplikatora i iznosa decizisjke funkcije umanjenog za 1 pripadne točke uvijek jednak 0. Točke koje su udaljene za 1 od decizijske funkcije (te je time njihov iznos umanjen za 1 jednak 0) su upravo točke koje se nalaze na rubovima margina. U tim slučajevima pripadni Lagrangeov multiplikator može biti različit od 0. Te točke se nazivaju potporni vektori. Tu valja obratiti pozornost da nije nužno da sve točke koje se nalaze na rubu margina budu ujedno i potpornji vektori. Pripadni Lagrangeov multiplikator točaka koje se ne nalaze na rubu margine tj. za njih vrijedi d(x)>1 iznosi 0. To se može protumačiti na način da te točke nisu važne za određivanje decizijske funkcije i njihovim uklanjanjem se ne utječe na samu decizijsku funkciju.
Uvrštavanjem (1.13) i (1.14) u (1.8) dobiva se sljedeći optimizacijski problem uz uvjet (1.14).

$$\max_{\alpha} f(\alpha) = \left(\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \right) \qquad \textbf{(1.15)}$$

Ovaj oblik se još nazva i dualni oblik.
Normala ravnine je zadana jednadžbom (1.13) dok je slobodna konstanta (w_0) dana izrazom.

$$w_0 = y_i - w^T x_i \qquad \textbf{(1.16)}$$

Sa gledišta numeričke preciznosti bolje je uzeti izraz (1.17)

$$w_0 = \frac{1}{S} \sum_{i \in S} \left(y_i - w^T x_i \right) \qquad \textbf{(1.17)}$$

Sada se decizijska funkcija može zapisati u obliku

$$d(x) = \sum_{i \in S} \alpha_i y_i x_i^T x + w_0 \qquad \textbf{(1.18)}$$

sa uvjetom razvrstavanja

$$d(x_i) \begin{cases} > 0 & x_i \in \omega_1 \\ < 0 & x_i \in \omega_2 \end{cases} \qquad \textbf{(1.19)}$$

1.1.2 Linearno neseparabilni razredi

Najčešće su razredi linearno neseparabili i model predstavljen u prethodnom poglavlju ne može pronaći decizijsku funkciju.
Da bi se prilagodio postojeći model SVM-a linearno neseparabilnim razredima, uvodi se nova varijabla – nenegativna varijabla ξ_i.
Uvjet zadan jednadžbom (1.7) postaje uvjet zadan jednadžbom (1.20)

$$y_i (w^T x_i + w_0) \geq 1 - \xi_i \qquad \textbf{(1.20)}$$

Uvođenjem varijable ξ_i postoji prihvatljiva decizijska funkcija. Ovisno o varijabli ξ_i postoje tri kategorije uzoraka:

1) $\xi_i=0$ uzorci koji pripadaju linearno separabilnom dijelu i obrađeni su u prethodnom poglavlju
2) $\xi_i > 0$ i $\xi_i \leq 1$ uzorci koji više nemaju maksimalnu marginu, ali još uvijek su točno klasificirani
3) $\xi_i > 1$ uzorci koji su pogrešno klasificirani

Sada je potrebno pronaći takvu decizijsku funkciju u kojoj je broj uzoraka koji nemaju maksimalnu marginu minimalan.
Potrebno je pronaći minimum funkcije

$$J(w) = \sum_{i=1}^{M} I(\xi_i) \tag{1.21}$$

gdje je

$$I(\xi_i) = \begin{cases} 1 & \xi_i > 0 \\ 0 & \xi_i = 0 \end{cases} \tag{1.22}$$

Ako se uzme u obzir i maksimizacija margine dobiva se sljedeća funkcija koju je potrebno minimizirati.

$$J(w, w_0, \xi) = \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{N} I(\xi_i) \tag{1.23}$$

uz uvjete

$$y_i (w^T x_i + w_0) \geq 1 - \xi_i \tag{1.24}$$

$$\xi_i \geq 0 \tag{1.25}$$

C je parametar koji određuje važnost širine margine i broja uzoraka koji nemaju maksimalnu marginu.
Lagrangeov oblik danog problema dan je funkcijom

$$F(w, w_0, \xi, \alpha, \mu) = \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{N} I(\xi_i) - \sum_{i=1}^{N} \mu_i \xi_i - \sum_{i=1}^{N} \alpha_i \left[y_i \left(w^T x_i + w_0 \right) - 1 + \xi_i \right] \tag{1.26}$$

KKT uvjeti su

$$\frac{\partial}{\partial w} F(w, w_0, \xi, \alpha, \mu) = 0 \tag{1.27}$$

$$\frac{\partial}{\partial w_0} F(w, w_0, \xi, \alpha, \mu) = 0 \tag{1.28}$$

$$\frac{\partial}{\partial \xi} F(w, w_0, \xi, \alpha, \mu) = 0 \tag{1.29}$$

$$\alpha_i \geq 0, \mu_i \geq 0 \tag{1.30}$$

$$\alpha_i \left[y_i (w^T x_i + w_0) - 1 + \xi_i \right] = 0 \tag{1.31}$$

$$\mu_i \xi_i = 0 \tag{1.32}$$

Iz (1.26) i (1.29) slijedi

$$C - \mu_i - \alpha_i = 0 \tag{1.33}$$

Dualni oblik je

$$\max_{\alpha} f(\alpha) = \left(\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \right) \tag{1.34}$$

uz uvjete

$$0 \le \alpha_i \le C \tag{1.35}$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{1.36}$$

1.2 Preslikavanje u višedimenzionalni prostor

U slučaju kad razredi nisu linearno odvojivi iako će SVM naći optimalnu decizijsku hiperravninu to ne znači da će ta ravnina dobro odvajati uzorke koji pripadaju različitim razredima. Rješenje ovog problema je da se originalan prostor značajki proširi tj. preslika u prostor značajki većih dimenzija.
Korištenjem nelinearnog vektora funkcija

$$g(x) = (g_1(x), ..., g_l(x))^T \tag{1.37}$$

koji preslikava m-dimenzionalni ulazni vektor značajki x u l-dimenzionalni prostor značajki, dobivamo linearnu decizijsku funkciju u prostoru značajki.

$$d(x) = w^T g(x) + w_0 \tag{1.38}$$

Gdje je w l-dimenzionalan vektor, a w_0 utežna vrijednost (pomak).

Prema Hilbert-Schmidt teoriji ako simetrična funkcija K(x,x') zadovoljava uvjet

$$\sum_{i,j=1}^{M} h_i h_j K(x_i, x_j) \ge 0 \tag{1.39}$$

za sve M, x_i i h_i, tada postoji funkcija preslikavanja g(x) koja preslikava ulazni uzorak u prostor značajki i g(x) zadovoljava

$$K(x, x') = g^T(x) g(x'). \tag{1.40}$$

Ako prethodno vrijedi, vrijedi također i

$$\sum_{i,j=1}^{M} h_i h_j K(x_i, x_j) = \left(\sum_{i=1}^{M} h_i g^T(x_i) \right) \left(\sum_{i=1}^{M} h_i g\ (x_i) \right) \ge 0 \tag{1.41}$$

Ovaj uvjet naziva se Mercerov uvjet, a funkcija koja ga zadovoljava naziva se pozitivna semidefinitna jezgra ili Mercerova jezgra.

Prednost korištenja jezgre u stroju s potpornim vektorima je što nije potrebno raditi s visokodimenzionalnim prostorom značajki eksplicitno već je dovoljno poznavanje K(x,x') kod treniranja i klasifikcije.
Korištenjem jezgre ,postavljen je dualni problem maksimizacije funkcije

$$Q(\alpha) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha j_j y_i y_j K(x_i, x_j) \quad \textbf{(1.42)}$$

s uvjetima

$$\sum_{i=1}^{M} y_i \alpha_i = 0, \quad \textbf{(1.43)}$$

$$0 \le \alpha_i \le C, \quad \textbf{(1.44)}$$

za $i = 1,..., M$.

Zbog toga što je K(x,x') pozitivna semidefinitna jezgra, problem optimizacije funkcije se svodi na rješavanje konkavnog problema metodom kvadratnog programiranja. Također, zbog toga što je rješenje $\alpha = 0$ prihvatljivo problem mora imati globalno optimalno rješenje.
Komplementarni KKT uvjeti su dani sa

$$\alpha_i \left(y_i \left(\sum_{i=1}^{M} y_j \alpha_j K(x_i, x_j) + b \right) - 1 + \xi_i \right) = 0 \quad \textbf{(1.45)}$$

za $i = 1,...M$,

$$(C - \alpha_i)\xi_i = 0 \quad \textbf{(1.46)}$$

za $i = 1,...M$,

$$\alpha_i \ge 0, \ \xi_i \ge 0 \quad \textbf{(1.47)}$$

za $i = 1,...M$,
Decizijska funkcije je dana sa

$$D(x) = \sum_{i \in S} \alpha_i y_i K(x_i, x) + b, \quad \textbf{(1.48)}$$

gdje je b dan sa

$$b = y_j - \sum_{i \in S} \alpha_i y_i K(x_i, x_j). \quad \textbf{(1.49)}$$

U ovom slučaju x_j je potporni vektor značajki. Kako bi se osigurala stabilnost proračuna, uzima se prosječna vrijednost:

$$b = \frac{1}{U} \sum_{j \in U} \left(y_j - \sum_{i \in S} \alpha_i y_i K(x_i, x_j) \right) \quad \textbf{(1.50)}$$

gdje je U skup svih indeksa potpornih vektora.

Poznavajući prethodno novi nepoznati uzorci se klasificiraju korištenjem decizijske funkcije na sljedeći način:

$$x \in \begin{cases} razred\ 1 & ako \quad D(x) > 0 \\ razred\ 2 & ako \quad D(x) < 0 \end{cases} \tag{1.51}$$

ako je $D(x) = 0$, uzorak x se ne može klasificirati.

Jezgre

Linearne jezgre

U slučajevima kada je problem klasifikacije linearno odvojiv u prostoru ulaznih značajki nije potrebno preslikavanje u višedimenzionalna prostor značajki. U takvim slučajevima koriste se linearna jezgra:

$$K(x, x') = x^T x'. \tag{1.52}$$

Polinomijalne jezgre

Polinomijalne jezgre stupnja d, gdje je d prirodan broj, oblika su

$$K(x, x') = \left(x^T x' + 1\right)^d. \tag{1.53}$$

Za vrijednost d=1 polinomijalna jezgra je zapravo linearna. Za vrijednosti d=2 i m=2 polinomijalna jezgra je sljedećeg oblika

$$K(x, x') = 1 + 2x_1 x_1' + 2x_2 x_2' + 2x_1 x_1' x_2 x_2' + x_1^2 x_1'^2 + x_2^2 x_2'^2 = g^T(x) g(x) \tag{1.54}$$

gdje je $g(x) = \left(1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2\right)^T$. Stoga, za vrijednosti d=2 i m=2 polinomijalna jezgra zadovoljava Mercerov uvjet. Može se pokazati da svaka polinomijalna jezgra stupnja d zadovoljava Mercerov uvjet.

Jezgre s radijalnim baznim funkcijama

Radijalne bazne funkcije su oblika

$$K(x, x') = \exp\left(-\gamma \|x - x'\|^2\right) \tag{1.55}$$

Gdje je γ pozitivan broje koji kontrolira radijus. Drugačije zapisano prethodna formula izgleda

$$K(x, x') = \exp\left(-\gamma \|x\|^2\right) \exp\left(-\gamma \|x'\|^2\right) \exp\left(2\gamma x^T x'\right) \tag{1.56}$$

Budući da je

$$\exp\left(2\gamma x^T x'\right) = \sum_{i=0}^{\infty} \frac{(2\gamma)^i}{i!} \left(x^T x'\right)^i \tag{1.57}$$

beskonačna suma, slijedi da je i jezgra. Isto tako se dokaže da su i $\exp\left(-\gamma \|x\|^2\right)$ i $\exp\left(-\gamma \|x'\|^2\right)$ jezgre, a budući da vrijedi da je umnožak jezgri i dalje jezgra slijedi da je i $\exp\left(-\gamma \|x\|^2\right) \exp\left(-\gamma \|x'\|^2\right) \exp\left(2\gamma x^T x'\right)$ jezgra.

Rezultirajuća decizijska funkcija je

$$d(x) = \sum_{i \in S} \alpha_i y_i \exp\left(-\gamma \left\| x_i - x \right\|^2 + b\right). \quad \textbf{(1.58)}$$

1.3 Klasifikacija više razreda uzoraka

Iako možda SVM nije najpogodnija metoda za rješavanje problema raspoznavanja velikog broja razreda uzoraka(npr. 200 razreda) razvijene su mnoge metode koje daju zadovoljavajuće rezultate. Najveće probleme naravno, predstavlja samo učenje koje se zasniva na kvadratnom programiranju. Veliki broj razreda i veliki broj uzoraka za učenje može pretvoriti kvadratno programiranje u vrlo neefikasnu metodu koja zahtijeva veliku količinu memorije, a samo učenje traje vrlo dugo. Drugi problem koji se javlja kod klasifikacije više razreda su mogućnost pojavljivanja nedefiniranih regija. Za oba problema postoje efikasni pristupi koji ih rješavaju.
Neke od metoda su :

1. jedan nasuprot svih metoda
2. usporedba po parovima
3. stabla odluke

Ideja jedan nasuprot svih metode je odvajanje decizijskom funkcijom jednog razreda od svih preostalih. Pritom je potrebno izgraditi k decizijskih funkcija, gdje je k broj razreda.
Neka je formulom (1.59) zadana decizijska funkcija $d_i(x)$.

$$d_i(x) = w_i^T x + w_0 \quad \textbf{(1.59)}$$

Tada će uzorak biti razvrstan u razred ω_i ako je zadovoljen uvjet (1.60)

$$d_i(x) > 0 \quad \textbf{(1.60)}$$

Može se odmah uočiti jedan vrlo važan nedostatak ove metode, a to je pojava nedefiniranih regija. Tako klasifikator neće znati klasificirati slučaj kada je $d_1(x)>0$, $d_2(x)>0$, $d_3(x)<0$ ili slučaj kada je $d_1(x)<0$, $d_2(x)<0$, $d_3(x)<0$. Ovo je takozvana diskretna verzija decizijskih funkcija jer se uzima samo podatak je li iznos funkcije u danoj točci veći ili manji od 0.
Da bi se izbjeglo pojavljivanje nedefiniranih regija, uvodi se kontinuirani oblik decizijske funkcije dan izrazom (1.61).

$$\arg\max_{i=1,\ldots,k} d_i(x) \quad \textbf{(1.61)}$$

Sada će uzorak biti razvrstan u razred čija pripadna decizijska funkcija ima najveću vrijednost. Ovo je samo jedan od načina kojim se može riješiti problem nedefiniranih područja.

Usporedba po parovima se temelji na izračunavanju $\frac{k(k-1)}{2}$ decizijskih funkcija, gdje je k broj razreda. Prilikom klasifikacije potrebno je izračunati vrijednosti $\frac{k(k-1)}{2}$ decizijskih funkcija i odlučiti se na temelju rezultata kojem razredu pripada nepoznati uzorak. Samo učenje decizijskih funkcija kao i kasnije testiranje i klasifikacija mogu biti neefikasni upravo zbog vrlo velikog broja samih decizijskih funkcija.

1.3.1 Klasifikacija temeljena na stablima odluke

Stabla odluke su jedna od metoda kojom se može riješiti problem nedefiniranih područja koja se pojavljuju kod metode jedan nasuprot svih. Ideja je da se u svakom čvoru stabla provodi binarna klasifikacija, tj. da se odvoji jedna grupa razreda od druge. Potrebno je dati algoritam koji će po nekom pravilu grupirati razrede u posebne grupe. Algoritam grupiranja je posebice važan za ovaj postupak jer on ima vrlo veliki utjecaj na konačnu točnost samog klasifikatora. Do sada je razvijeno više različitih algoritama grupiranja , a osnovna razlika im je rade li grupiranje u neproširenom prostoru značajki ili proširenom prostoru značajki koji nastaje nakon proširenja pomoću jezgrenih funkcija (*eng. kernel space*).Ovdje će biti predstavljen samo jedan algoritam grupiranja i to algoritam koji radi u neproširenom prostoru značajki.

Cilj grupiranja je dobivanje dvije grupe razreda u pojedinom čvoru stabla. Ako se svi uzorci iz jedne grupe (dobivene grupiranjem) označe s y=+1, a iz druge grupe s y=-1 dobiva se problem binarne klasifikacije koja se može riješiti s SVM-om. Da bi se odredilo koji razred pripada kojoj grupi potrebno je imati mjeru udaljenosti. Također je potrebno za svaki razred imati predstavnika. Taj predstavnik jednostavno može biti centar razreda koji se jednostavno dobiva formulom (1.62).

$$c_i = \frac{1}{|X_i|}\sum_{x \in X_i} x \qquad \textbf{(1.62)}$$

Sada se udaljenost između dva razreda može jednostavno dobiti kao Euklidska udaljenost između centara razreda što je i dano u formuli (1.63)

$$l_{ij} = \| c_i - c_j \| \qquad \textbf{(1.63)}$$

1) $grupa_1 = \{\emptyset\}$;
2) $grupa_2 = \{\emptyset\}$;
3) izračunaj c_i svih razreda
4) $\{r1,r2\} = \max_{i \neq j} l_{ij}$;
5) $grupa_1 = grupa_1 \cup r_1$;
6) $grupa_2 = grupa_2 \cup r2$;
7) **ponavljaj dok** nisu svi razredi u jednoj od grupa
8) $r = \min_{i \neq grupa1} l_{igrupa1}$;
9) $p = \min_{i \neq grupa2} l_{igrupa2}$;
10) **ako je** $l_{r,grupa1} < l_{p,grupa2}$
11) $grupa_1 = grupa_1 \cup r$;
12) **inače**
13) $grupa_2 = grupa_2 \cup p$;
14) **kraj**
15) **vrati** $\{grupa_1, grupa_2\}$

Slika 1.2 Pseudokod algoritma grupiranja

U koracima 1) do 3) radi se inicijalizacija algoritma. U koracima 4) do 6) traže se dva najudaljenija razreda koji se dodjeljuju svaki u po jednu grupu. U koracima 7) do 13) traži se razred koji je najbliži pojedinoj grupi te se taj razred i toj grupi dodjeljuje. Ti koraci se ponavljaju sve dok se svi razredi ne dodijele u jednu od dvije grupe.
Izgradnja stabla je vrlo jednostavna. Nakon što se provede grupiranje dobivaju se dvije grupe. Pronalazak decizijske funkcije za te dvije grupe se temelji na binarnom SVM-u. Nakon toga se nad dobivenim razredima iz grupa (dobivenih grupiranja) rekurzivno ponavlja postupak i tako sve dok u jednom čvoru ne preostane samo jedan razred. Taj čvor je list i on nosi oznaku tog razreda.
Klasifikacija uzorka se provodi u svakom čvoru stabla koji se nalazi na putu do određenog lista. Klasifikacija u pojedinom čvoru se koristi za usmjeravanje u lijevo ili desno podstablo ovisno o rezultatu klasifikacije.
Postupak rješavanja problema klasifikacije više razreda koji se temelji na binarnim stablima odluke ima nekoliko dobrih svojstava.

I. potrebno je pronaći svega k-1 decizijskih funkcija
II. samo se u korijenu razmatraju svi uzorci za treniranje dok je u ostalim čvorovima taj skup sve manji i manji. To znatno pridonosi samo brzini treniranja.
III. tijekom klasifikacije se ispituju samo neke decizijske funkcije

.

1.4 Treniranje SVM-a

U prethodnim poglavljima je pokazano da se rješenje nalazi rješavanjem problema koji je zadan u obliku kvadratnog programiranja. Postoje numerički postupci koji rješavaju probleme kvadratnog programiranja, no oni se pokazuju kao neučinkoviti za SVM. Dva su razloga za to: 1) Q matrica može biti jako velika 2) samo rješavanje problema može trajati jako dugo. Kao što je poznato do nedavno (prije 10 godina) radna memorija je bila veliki luksuz i gotovo niti jedno kućno računalo nije imalo preko 100 MB. Ako se uzme u obzir da problem od 4000 uzoraka za učenje (koji se inače smatra malim problemom) može imati 128 MB veliku matricu očito je bilo potrebno smisliti nove načine rješavanja problema.
Radi jednostavnijeg pisanja u sljedećim poglavljima sve će biti pisano u obliku matrica i vektora.

Dualni problem se sada može zapisati u sjedećem obliku

$$\min_{\alpha} f(\alpha) = \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \qquad \textbf{(1.64)}$$

uz uvjete

$$\begin{aligned} &0 \le \alpha_i \le C, i = 1,...n \\ &y^T\alpha = 0 \end{aligned} \qquad \textbf{(1.65)}$$

Dodatno još vrijedi da je vektor *e* vektor koji sadrži sve jedinice. Matrica Q je n×n matrica. Također vrijedi da je $Q_{ij}=y_i y_j K(x_i,x_j)$. $K(x_i,x_j)$ je jezgrena funkcija.

1.4.1 Dekompozicija

Ideja dekompozicije jest dijeljenje skupa Lagrangeovih multiplikatora(W) u dva skupa B i N. Pritom mora vrijediti $B \bigcup N = W$ i $B \bigcap N = 0$ Na skup B se tada može primijeniti neka od metoda za rješavanje kvadratnog problema , a skup N se drži konstantnim.
Sada se jednadžba (1.64) može prepisati u sljedeći oblik

$$\min_{\alpha} f(\alpha) = \frac{1}{2}\begin{bmatrix}\alpha_B^T, \alpha_N^T\end{bmatrix}\begin{bmatrix}Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN}\end{bmatrix}\begin{bmatrix}\alpha_B \\ \alpha_N\end{bmatrix} - \begin{bmatrix}e_B^T & e_N^T\end{bmatrix}\begin{bmatrix}\alpha_B \\ \alpha_N\end{bmatrix} =$$
$$= \frac{1}{2}\alpha_B^T Q_{BB}\alpha_B + \left(-e_B + Q_{BN}\alpha_N\right)^T \alpha_B + kons\tan ta \qquad \textbf{(1.66)}$$

uz uvjete

$$0 \le \alpha_B \le C, i = 1,...n$$
$$y_B^T \alpha = -y_N^T \alpha \qquad \textbf{(1.67)}$$

Iz drugog uvjeta se uočava jedno vrlo važno svojstvo dekompozicije , a to je $|B| \ge 2$. Jasno je kada bi se samo jedan Lagrangeov multiplikator mijenjao prilikom promjene odmah bi se prekršilo pravilo iz jednadžbe (1.65), no ako se mijenja dva ili više multiplikatora tada se uvijek mogu namjestiti tako da jednadžba (1.65) ostane zadovoljena.
Budući da optimalno rješenje mora zadovoljavati KKT uvjete nakon optimiranja jednog podskupa Lagrangeovih multiplikatora odabire se novi i tako sve dok nisu zadovoljeni KKT uvjeti.
Ovisno o skupu B postoje dvije vrste metoda dekompozicije.

- metoda koja u svakom koraku ima konstantu veličinu skupa B
- metoda koja ima varijabilnu veličinu skupa B

```
1) dok je prekršeno pravilo optimalnosti
2)         odaberi |B| varijabli za skup B;
3)         preostale varijable su fiksirane;
4)         riješi QP - podproblem nad B ;
5) kraj
6) vrati α;
```

Slika 1.3 Pseudo kod algoritma dekompozicije

1.4.2 SMO (Sequential Minimal Optimization)

SMO metoda je metoda dekompozicije koja ima stalnu veličinu skupa B i to $|B| = 2$.
Sada se jednadžba (1.66) može napisati kao

$$\min_{\alpha} f(\alpha) = \frac{1}{2}\begin{bmatrix}\alpha_i, \alpha_j\end{bmatrix}\begin{bmatrix}Q_{ii} & Q_{ij} \\ Q_{ij} & Q_{jj}\end{bmatrix}\begin{bmatrix}\alpha_i \\ \alpha_j\end{bmatrix} - \left(-e_B + Q_{BN}\alpha_N\right)^T\begin{bmatrix}\alpha_i \\ \alpha_j\end{bmatrix} + kons\tan ta \qquad \textbf{(1.68)}$$

uz uvjete

$$0 \le \alpha_i, \alpha_j \le C, i = 1,...n$$
$$y_i\alpha_i + y_j\alpha_j = -y_N^T\alpha \qquad \textbf{(1.69)}$$

Budući da se optimizacija u jednom koraku provodi samo nad dva Lagrangeova multiplikatora rješenje se dobiva analitički. Takav postupak je vrlo brz što je pozitivno za algoritam, jer više nije potrebno gubiti puno vremena na optimizatoru koji koristi neki numerički postupak.
Odabir samih multiplikatora u jednom koraku nije toliko jednostavan. Do sada je razvijeno više postupaka.
Keerthi u [3] predlaže korištenje sljedećeg postupka.
Neka su definirana dva skupa I_{up} i I_{low} na sljedeći način.

$$I_{up}(\alpha) \equiv \{t \mid \alpha_t < C, y_t = 1 \vee \alpha_t > 0, y_t = -1\}$$
$$I_{low}(\alpha) \equiv \{t \mid \alpha_t < C, y_t = -1 \vee \alpha_t > 0, y_t = 1\} \quad \textbf{(1.70)}$$

Tada se par (B={i,j}) za optimizaciju može odabrati na sljedeći način.

$$i \in \arg\max_t \{-y_t \nabla f(\alpha)_t \mid t \in I_{up}(\alpha)\}$$
$$j \in \arg\min_t \{-y_t \nabla f(\alpha)_t \mid t \in I_{low}(\alpha)\} \quad \textbf{(1.71)}$$

gdje je

$$\nabla f(\alpha) \equiv Q\alpha - e \quad \textbf{(1.72)}$$

$\nabla f(\alpha)$ je gradijent kriterijske funkcije.
Ovaj postupak se može izvesti iz KKT uvjeta jer je poznato da optimalno rješenje mora zadovoljavati KKT uvjete. Izvod se može pronaći u [3]. Postupak u svakom koraku pronalazi par koji najviše krši KKT uvjete te nad njima provodi optimizaciju.
Joachims([5]) predlaže korištenje aproksimacije prvog reda kriterijske funkcije za određivanje skupa varijabli (u slučaju SMO samo dvije varijable) čije će računanje dovesti do najvećeg napretka.
Neka je aproksimacija prvog reda kriterijske funkcije zadana sljedećom formulom

$$f(\alpha + d) \approx f(\alpha) + \nabla f(\alpha) d = f(\alpha) + \nabla f(\alpha)_B^T d_B \quad \textbf{(1.73)}$$

Vektor d označava smjer najbržeg spusta. Da bi se dobio vektor d potrebno je riješiti sljedeći optimizacijski problem.

$$\min_{d_B} \nabla f(\alpha)_B^T d_B$$
$$\textit{uvjeti}:$$
$$y_B^T d_B = 0$$
$$d_t \geq 0 \quad ako \quad \alpha_t = 0, t \in B$$
$$d_t \leq 0 \quad ako \quad \alpha_t = C, t \in B$$
$$-1 \leq d_t \leq 1, t \in B \quad \textbf{(1.74)}$$

Može se pokazati da je postupak koji traži par koji najviše krši KKT uvjete povezan s postupkom koji se temelji na aproksimaciji prvog reda kriterijske funkcije.

Budući da je kriterijska funkcija kvadratna logično je koristiti aproksimaciju drugog reda.

$$\Delta f(\alpha) = f(\alpha + d) - f(\alpha) = \nabla f(\alpha)^T d + \frac{1}{2} d^T \nabla^2 f(\alpha) d = \nabla f(\alpha)_B^T d_B + \frac{1}{2} d_B^T \nabla^2 f(\alpha)_{BB} d_B \quad \textbf{(1.75)}$$

Da bi se sada našao smjer d potrebno je riješiti sljedeći optimizacijski problem.

$$\begin{aligned} &\min_{d_B} \frac{1}{2} d_B^T \nabla^2 f(\alpha)_{BB} d_B + \nabla f(\alpha)_B^T d_B \\ &uvjeti: \\ &y_B^T d_B = 0 \\ &d_t \geq 0 \quad ako \quad \alpha_t = 0, t \in B \\ &d_t \leq 0 \quad ako \quad \alpha_t = C, t \in B \end{aligned} \quad \textbf{(1.76)}$$

Rješavanje ovog problema nije jednostavno i potrebno je ispitati sve moguće parove. Ispitivanje svih mogućih kombinacija se može zaobići na sljedeći način. Multiplikator *i* se odabire na način predložen u (1.71). Za multiplikator *j* se uvodi modifikacija.
Da bi se došlo de te modifikacije potrebno je proučiti (1.76).
Definiraju se varijable :

$$\begin{aligned} &\hat{d}_i = y_i d_i \\ &\hat{d}_j = y_j d_j \\ &\hat{d}_i = -\hat{d}_j \; iz \quad y_B^T d_B = 0 \end{aligned} \quad \textbf{(1.77)}$$

Raspisivanjem (1.76) dobiva se sljedeći izraz:

$$\frac{1}{2}\begin{bmatrix} d_i, d_j \end{bmatrix} \begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ij} & Q_{jj} \end{bmatrix} \begin{bmatrix} d_i \\ d_j \end{bmatrix} + \begin{bmatrix} \nabla f(\alpha)_i & \nabla f(\alpha)_j \end{bmatrix} \begin{bmatrix} d_i \\ d_j \end{bmatrix} = \frac{1}{2}\left(K_{ii} + K_{jj} - 2K_{ij}\right)\hat{d}_j^2 + \left(-y_i \nabla f(\alpha)_i + y_j \nabla f(\alpha)_j\right)\hat{d}_j \quad \textbf{(1.78)}$$

Kako je $K_{ii}+K_{jj}-2K_{ij}>0$ definiraju se varijable

$$\begin{aligned} a_{ij} &= K_{ii} + K_{jj} - 2K_{ij} > 0 \\ b_{ij} &= -y_i \nabla f(\alpha)_i + y_j \nabla f(\alpha)_j > 0 \end{aligned} \quad \textbf{(1.79)}$$

(1.78) ima minimum u

$$\hat{d}_j = -\hat{d}_i = -\frac{b_{ij}}{a_{ij}} < 0 \quad \textbf{(1.80)}$$

a iznos (1.76) u točki minimuma jest

$$-\frac{b_{ij}^2}{2a_{ij}}$$

Ako je K pozitivno definitna matrica tada je za svaki $i \neq j$ vrijedi $K_{ii} + K_{jj} - 2K_{ij}>0$.
Sada se može napisati novi kriterij za odabir para {i,j}.

$$i \in \arg\max_t \left\{ -y_t \nabla f(\alpha)_t \mid t \in I_{up}(\alpha) \right\}$$
$$j \in \arg\min_t \left\{ -\frac{b_{it}^2}{a_{it}} \mid t \in I_{low}(\alpha), -y_t \nabla f(\alpha)_t < -y_i \nabla f(\alpha)_i \right\} \quad \textbf{(1.81)}$$

Ako je $K_{ii} + K_{jj} - 2K_{ij} \leq 0$ tada se kriterij (1.81) ne može primijeniti za izbor para {i,j}. Pokazano je u [4] da se taj problem rješava uvođenjem dodatnog izraza u kriterijsku funkciju što na kraju rezultira rezultatom $a_{ij} \equiv \tau$, gdje je τ proizvoljno mali broj.

Sada se konačno može dati postupak izbora para Lagrangeovih multiplikatora u jednoj iteraciji optimizacije.
1. Definiraju se a_{ts} i b_{ts} kao u (1.79).

$$\bar{a}_{ts} \equiv \begin{cases} a_{ts} & ako \quad a_{ts} > 0 \\ \tau & ako \quad a_{ts} \leq 0 \end{cases} \quad \textbf{(1.82)}$$

2. Odabir {i,j}

$$i \in \arg\max_t \left\{ -y_t \nabla f(\alpha)_t \mid t \in I_{up}(\alpha) \right\}$$
$$j \in \arg\min_t \left\{ -\frac{b_{it}^2}{\bar{a}_{it}} \mid t \in I_{low}(\alpha), -y_t \nabla f(\alpha)_t < -y_i \nabla f(\alpha)_i \right\} \quad \textbf{(1.83)}$$

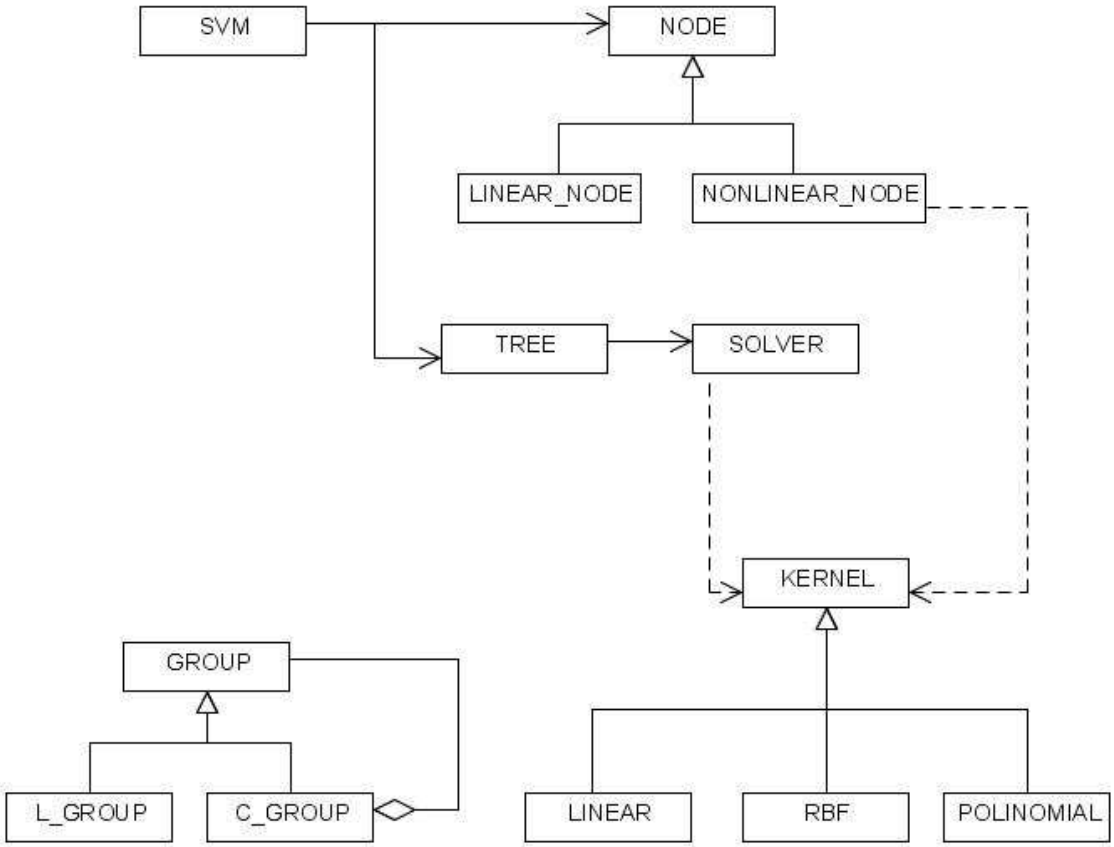
3. Vraćanje B ={i,j}

1.4.3 Dodatna poboljšanja

Dodatna poboljšanja gore navedenog postupka se mogu postići na dva područja. To su korištena memorija i korišteno vrijeme.
Ako za treniranje nije dostupna velika količina radne memorije (to nekada može biti i samo nekoliko MB) tada opcija spremanja cijele jezgrene matrice (Q) u memoriju ne dolazi u obzir. Ponovno računanje jezgrene funkcije za par uzoraka prilikom svakog korištenja moglo bi znatno usporiti sami algoritam. Uz pretpostavku da je ipak dostupna određena količina radne memorije (možda samo 100KB) može se ipak dobiti dovoljno brzi algoritam. Ideja je da se u memoriji čuvaju oni stupci (ili redci ,budući da je matrica simetrična svejedno je) matrice koji su korišteni najčešće u posljednjih nekoliko koraka. Kada se dogodi promašaj tada se briše onaj stupac koji je najdavnije korišten , a dodaje se novi. Računa se cijeli stupac , a ne samo jedan element jer kad je potreban jedan element najčešće je potreban čitavi stupac u kojem se taj element nalazi.
Kako bi se dodatno ubrzao sami algoritam koristi se metoda smanjivanja problema (*eng. shrinking*). Ideja metode se zasniva na činjenici da su za decizijsku funkciju važni samo potporni vektori. Ako se uklone preostali uzorci na sami rezultat se neće utjecati. Također broj potpornih vektora je redovito puno manji od samog broja uzoraka pa je i za očekivati da će ubrzanje biti znatno.
Nažalost ove metode još nisu implementirane u samoj implementaciji klasifikatora. Implementirani klasifikator sprema cijelu Q matricu u memoriju. To mu omogućuje da računanje jezgrene funkcije za jedan par uzoraka obavi samo jedanput što pridonosi brzini. S druge strane postupak ovisno o broju uzoraka za treniranje može zauzeti i nekoliko desetaka MB radne memorije. Ako je problem veliki (nekoliko desetaka tisuća uzoraka za učenje) može se dogoditi da neće biti dovoljno radne memorije.

Tijekom računanja Lagrangeovih multiplikatora klasifikator uzima u obzir sve uzorke te taj način donekle usporava sami postupak treniranja. Opet za velike probleme klasifikator bi se mogao pokazati kao neučinkovit.

1.5 Programsko ostvarenje SVM-a



Slika 1.4 Dijagram razreda

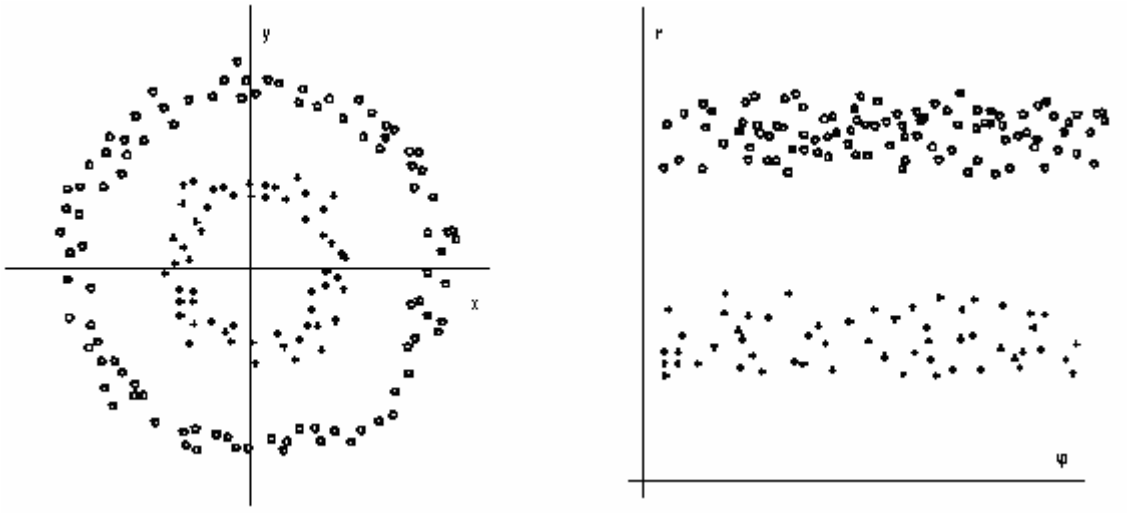
Dijagram razreda je prikazan na slici 1.4. Glavni razred preko kojeg se komunicira s cijelim klasifikatorom je razred *svm*. Taj razred podržava četiri operacije koje su bitne za sami klasifikator a to su : treniranje (train), klasifikacija nepoznatog uzorka (classify) spremanje (save) i učitavanje (load) svm-a iz datoteke. Razred *svm* koristi razred *tree* za izgradnju samog stabla. Iako se na slici ne može uočiti razred *tree* koristi razred *algorithm* koji definira sami postupak grupiranja. Razred *tree* koristi i razred *solver* u kojemu je ostvaren SMO postupak. Ovdje je važno napomenuti da prilikom izgradnje samog stabla traženje Lagrangeovih multiplikatora za svaki čvor pomoću *solver* razreda je neovisno. To omogućuje uvođenje paralelizacije samog postupka čime bi samo treniranje bilo još brže.
Na slici se također uočavaju tri vrlo važne komponente za sami sustav. To su *group*, *kernel* i *node*.
Skup uzoraka koji pripadaju istom razredu se smještaju u razred *l_group*. Taj razred još sadrži neke vrlo važne informacije za sami skup uzoraka kao što je njihov centar. Razred *c_group* omogućuje bilo kakvo grupiranje razreda (koji su predstavljeni razredom *l_group*) i preostalih grupa razreda (koje su predstavljene razredom *c_group*). Također *c_group* se

ponaša kao običan razred pa je moguće saznati i neke važne informacije kao što su centar grupe.
Razred *kernel* omogućuje u klasifikatoru korištenje različitih jezgrenih funkcija na vrlo jednostavan način.
Razred *node* predstavlja jedan čvor stabla. Potrebne su dvije vrste čvora ovisno koja se jezgrena funkcija koristi. Razred *linear_node* koristi se prilikom korištenja linearne jezgrene funkcije dok se razred *nonlinear_node* koristi za preostale jezgrene funkcije. Razlika između ove dvije vrste čvorova je u tome što se kod korištenja linearne jezgrene funkcije može eksplicitno izračunati decizijska funkcija dok kod preostalih jezgrenih funkcija se mora pamtiti potporni vektori i pripadajući Lagrangeovi multiplikatori. Ovdje se treba napomenuti da (iako nije naznačeno na slici) stvaranjem čvorova u stablu upravljaju same jezgrene funkcije tako da je korisnik oslobođen tog posla.
Ovakva arhitektura omogućuje vrlo lagano proširenje jer su glavne komponente nezavisne tj. sva komunikacija se odvija preko apstraktnih razreda.

2. Izlučivanje značajki iz glazbe

2.1 Značajke u raspoznavanju glazbe

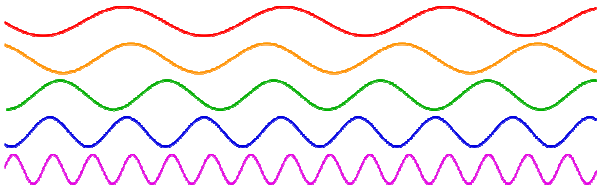
Izbor pravih značajki je ključan proces u klasifikaciji glazbe. Klasifikacijski algoritam će uvijek naći nekakvo rješenje, ali loš prikaz značajki dovest će do rješenja koje ne predstavlja pravu prirodu dotičnog podatka. Kako bi se optimizirao pronalazak pravog izbora značajki, treba se pridržavati sljedećih kriterija:
Objekti koje smatramo sličnima, odnosno da pripadaju istom tipu glazbe, trebaju se nalaziti u blizini ostalih točaka iz tog razreda u prostoru inačica i udaljenost između razreda bi trebala biti što veća. Ovaj kriterij ne mora biti zadovoljen za sve slučajeve, ponekad je dovoljno da točke koje predstavljaju jednu kategoriju glazbe leže u regiji prostora inačica koja se jasno može razdijeliti od regija koje predstavljaju ostale kategorije. Odnosno, granice decizijske funkcije trebaju biti jasno odvojene. Drugi, jednako važni kriterij je da značajke trebaju očuvati sve važne informacije koje određena pjesma posjeduje, a koje je obilježavaju.
Obično je poželjno smanjiti dimenzionalnost ulaznih podataka, kako bi se poboljšale performanse i računalna produktivnost i u većini slučajeva to je moguće ostvariti određenim transformacijama nad podatcima. Te transformacije se odnose na proces izlučivanja značajki, drugim riječima, podatci se trebaju pojednostaviti bez gubitka informacije. To bi u najboljem slučaju, podrazumijevalo reverzibilnu transformaciju, kao što je, primjerice, Fourierova. Stoga i ne čudi što je Fourierova transformacija jedna od najpoznatijih i najučinkovitijih tehnika u domeni analize audio podataka



Slika 2.1 Primjer ilustrira važnost pravog izbora značajki [7]

Uzorak je prikazan točkama u 2D prostoru. Lijeva slika predstavlja skup značajki predstavljenih kartezijevim koordinatama (x,y), a desna skup značajki predstavljen polarnim koordinatama. Na lijevoj slici je lako uočljivo da će većina algoritama teže točno pronaći granice decizijske funkcije. No, ako za prikaz istog uzorka koristimo polarne koordinate (r, φ) gdje je r radijus a φ kut, jednostavnije je odijeliti dvije regije iz prostora inačica. Ovo pokazuje, da i jednostavna transformacija kao što je ova, može bitno poboljšati rezultate klasifikacije.
Značajke koje se koriste u klasifikaciji audio signala, obično su podijeljene u dvije kategorije: fizičku i perceptivnu. Fizičke značajke su stvarni parametri signala, temelje se na obilježjima signala s realnim vrijednostima, primjerice: frekvencija, energija sinala,

snaga signala, duljina zvučnog vala, amplituda,... itd. Perceptivne značajke su temeljene na ljudskom doživljaju zvuka, kako ljudi doživljavaju visinu tona, ritam te primjerice, kako razlikuju pojedine instrumente. Zasigurno vrijedi činjenica da su sve perceptivne značajke povezane sa fizičkim značajkama, budući da je čovjekova percepcija zvuka zasnovana na fizičkim obilježjima signala zvuka, koje su stvarne vrijednosti.



Slika 2.2 Sinusoidalni valovi zvuka različitih frekvencija sustava

Na slici 2.2 donji valovi imaju više frekvencije od gornjih. Horizontalna os predstavlja vrijeme tijekom kojeg se val širi. Svojstva valova su: period, duljina vala, frekvencija, amplituda, jačina, brzina širenja, itd.

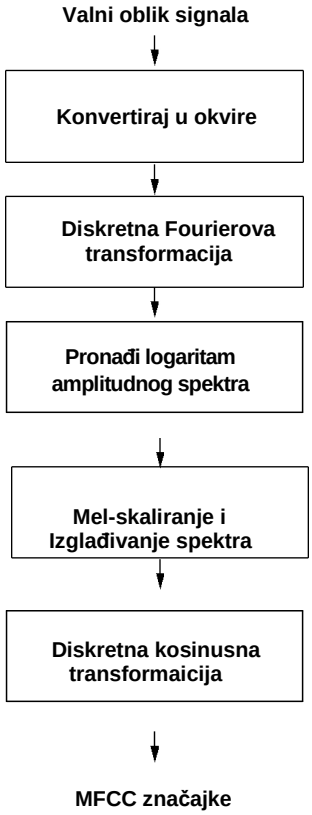
2.2 Jednostavne značajke

2.2.1 Spektar frekvencije

Frekvencija je jedna od glavnih značajki u raspoznavanju žanra muzike i iz nje se mogu dobiti ostale značajke. Distribucija frekvencije signala može se izračunati preko Fourierove transformacije. Fourier je otkrio da se svaki periodički valni oblik može aproksimirati određenim brojem sinusoidalnih komponenata koje su harmonične s osnovnom frekvencijom. Za neperiodične signale to znači da ih treba ponavljati n puta, gdje $n \to \infty$, kako bi postigli „prisilnu“ periodičnost.
Jedna od osnovnih i najuočljivijih komponenti audio signala, zasigurno je visina tona. Visina tona predstavlja osnovnu frekvenciju zvučnog vala neke melodije.

2.2.2 MFFC značajke

Postoji širok izbor mogućnosti dobivanja značajki iz audio signala, no MFCC (eng. *Mel-Frequency Cepstral Coefficients*) je vjerojatno prema [3, 4, 5, 6] najpoznatija i najbolja te je zbog toga korišten u projektu. Korjeni korištenja MFCC-a su u raspoznavanju govora, no sve se više primjenjuje i u klasifikaciji glazbe [7]. MFCC se temelji na poznatim varijacijama frekvencija u širini frekvencijskog pojasa koji je prepoznatljiv ljudskom uhu. Kako bi se „uhvatile“ bitne karakteristike pjesme kao uzorka, signal se predočuje Mel skalom (eng. *Mel* – skr. od *Melody*). Dakle, dobivanje značajki omogućeno je koeficijentima MFCC-a, koji su rezultat kosinusne transformacije logoritma kratkoročnog spektra energije signala izraženih Mel-frekvencijskom skalom [8].
MFC koeficijenti kao sažeti predstavnici amplitudnog spektra signala, modeliraju boju tona (eng. *Timbre*). Primjerice, boju tona predstavlja skup značajki koje ljudi koriste kada razlikuju saksofon od trube u nekoj jazz pjesmi, čak i kada oba instrumenta sviraju iste note na istoj visini i glasnoći. Računanje MFC koeficijenata odvija se sljedećim postupkom:



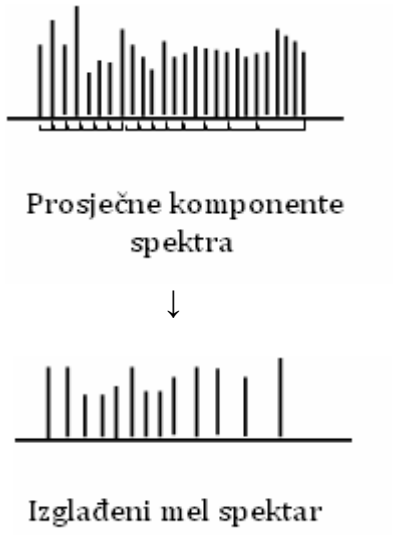
Slika 2.3 Postupak dobivanja MFCC značajki

1. Uokviravanje signala - U prvom koraku potrebno je sinusoidalni signal podijeliti u okvire, obično se radi tako da se primjeni Hammingova funkcija prozora (eng. *Hamming window*). To znači da će vrijednost signala izvan odabranog intervala biti jednak nuli.
2. Diskretna Fourierova transformacija - U drugom koraku se nad svakim od tako dobivenih „prozora" primjeni diskretna Fourierova transformacija (DFT) za vremenski diskretni signal x(n) duljine *N*

$$X(k) = \sum_{i=0}^{N-1} w(n)x(n)\exp(-j2\pi kn/N) \qquad \textbf{(2.1)}$$

za k=0,1,....N-1 gdje k odgovara frekvenciji $f(k) = kf_s/N$, a f_s predstavlja frekvenciju uzorkovanja.
3. Traženje logaritma – Ljudsko uho raspoznaje lagane tonove od jačih tonova, a percepcijska mjera koju tada doživljavamo je glasnoća. Glasnoća odgovara fizičkoj mjeri jačine tona. Diskretnom Fourierovom trensformacijom zadržavamo samo logaritam amplitudnog spektra. Odbacuju se informacije o faznom spektru jer se smatra da je njegova vrijednost nebitna i gotovo neprepoznatljiva za ljusko uho, te se je pokazalo da je amplitudni spektar mnogo bitniji od faznog [1] u terminima raspoznavanja glazbe i govora. Uzima se logaritam amplitudnog spektra jer je uočljiva glasnoća signala aproksimativno logaritamska.
4. Mel skaliranje – Ljudi razvrstavaju zvukove na skali od najnižih tonova do najviših, određujući time visinu tona (eng. *Pitch*). Visina tona kojeg predstavlja sinusoidalni signal je usko povezana s količinom frekvencije kod jednostavnih tonova i osnovnom frekvencijom kompleksnih tonova, kao što je to ranije napomenuto. Mel – skala je

procjena relacije između percipirane visine tona i stvarne frekvencije i izračunava se izjednačavanjem 1000 mela sa 1000 Hz sinusnog tona pri 40 dB. Koristi se u izračunavanju MFCC-a kako bi se transformirale frekvencije iz spektralnog prikaza u perceptivnu visinsku skalu pa prema tome mel – skaliranje ima formu filtarskog sloga.
Ovaj korak obuhvaća Mel skaliranje i izglađivanje spektra, jer se tako ističu samo percepcijski bitne frekvencije.



Slika 2.4 Izglađivanje mel-spektra

5. Diskretna kosinusna transformacija - Zadnji korak koji obuhvaća kosinusnu transformaciju (DCT) koristi se kako bi se reducirala dimenzionalnost izlaza, najčešće na nekoliko najbitnijih koeficijenata.

2.3 Kompleksne značajke

Navedene značajke pod 2.3.1 i 2.3.2 nisu eksplicitno korištene u ovom projektu, no to ne znači da se inače ne koriste u mnogim implementacijama sustava koji se bave raspoznavanjem govora i klasifikacije audio signala te su zato ukratko opisane u nastavku.

2.3.1 Ritam

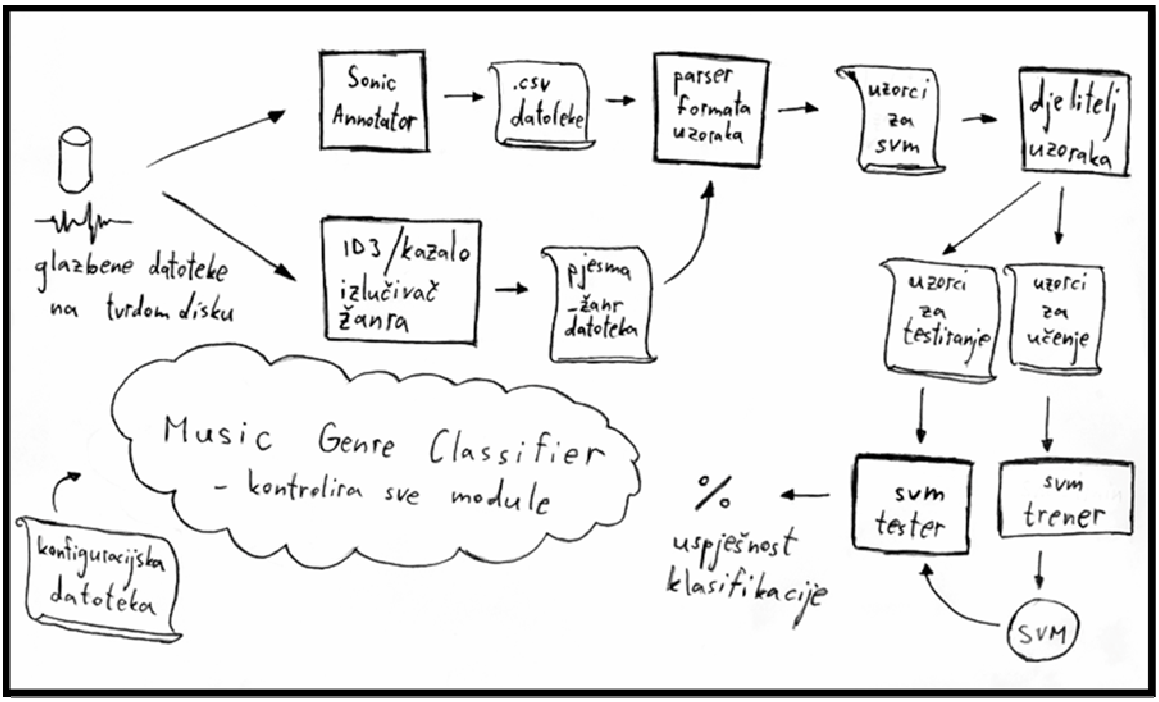
Praćenje i detekcija ritma je veliko i brzorastuće područje istraživanja samo za sebe, Koliko je god prepoznavanje ritma jednostavno za ljude, za većinu sustava je to složen proces. Primjerice, promjeni tempa u pjesmi ljudi se prilagode unutar sekunde, dok to većina automatskih sustava teže usvaja (primjer: improvizacija, neočekivane promjene).
Međutim, precizna detekcija ritma nije neophodna za raspoznavanje glazbenih žanrova, već je bitniji primjećeni tempo i jačina pojedinog ritma. Tempo predstavlja brzinu izvođenja danog glazbenog dijela, odnosno percipirani doživljaj periodičnog ponavljanja zvuka u određenim intervalima (najčešće duljine intervala između 250 ms i 2s) i kao takav predstavlja subjektivnu značajku uzorka. Jačina ritma je mjera jačine signala pri udarcima a određena je periodičnim vrhovima u signalu i relativnim amplitudama ritma u odnosu na prosječnu amplitudu signala dane pjesme.

2.3.2 Boja tona

Boja tona (eng. Timbre) je općenito definirana kao kvaliteta zvuka koja omogućuje raspoznavanje razlike između zvukova istog nivoa i glasnoće, a da je pritom proizvode različiti instrumenti ili glasovi. Kao takva, ovisi o spektru, pritisku pod kojem zvuk izlazi, frekvenciji i trenutačnim karakteristikama instrumenta ili glasa. Zbog toga je vjerojatno najkompleksnija i najsubjektivnija značajka audio zapisa te zbog toga nije modelirana u ovom radu kao značajka. Jer koliko je poznato, ne postoji sustav koji bi je uspio modelirati u smislu da daje zadovoljive rezultate. Spektrogrami se ponekad koriste za dobivanje informacija istih, no to je nedovoljno za područje kategoriziranja glazbe po stilovima. Buduće implementacije sličnih sustava uvelike bi poboljšale mnoge rezultate istraživanja u području klasifikacije audio signala.

3. Arhitektura sustava

U našem projektu aplikaciju smo razvijali tako da smo razbili funkcionalnost na više manjih modula koji obavljaju točno određene zadatke.
Sama izvršna datoteka Music Genre Classifier je tako u stvari samo jedna skripta koja kao ulaz prima konfiguracijsku datoteku "conf-file.txt" i kontrolira sve druge module (poziva ih).
Arhitektura ovog sustava se najbolje može vidjeti iz grafičkog prikaza (slika 3.1).



Slika 3.1 Slika arhitekture našeg sustava

U sljedećim cjelinama se ukratko s visoke razine objašnjavaju detalji pojedinih modula.

3.1 Sonic Annotator

Jedini modul koji nismo sami implementirali, već smo koristili gotov produkt. Detalji korištenja Sonic Annotatora nalaze se u zasebnom poglavlju. Ukratko - kao izlaz daje tekstualnu datoteku za svaku obrađenu pjesmu s izračunatim vrijednostima značajki.

3.2 ID3/kazalo izlučivač žanrova

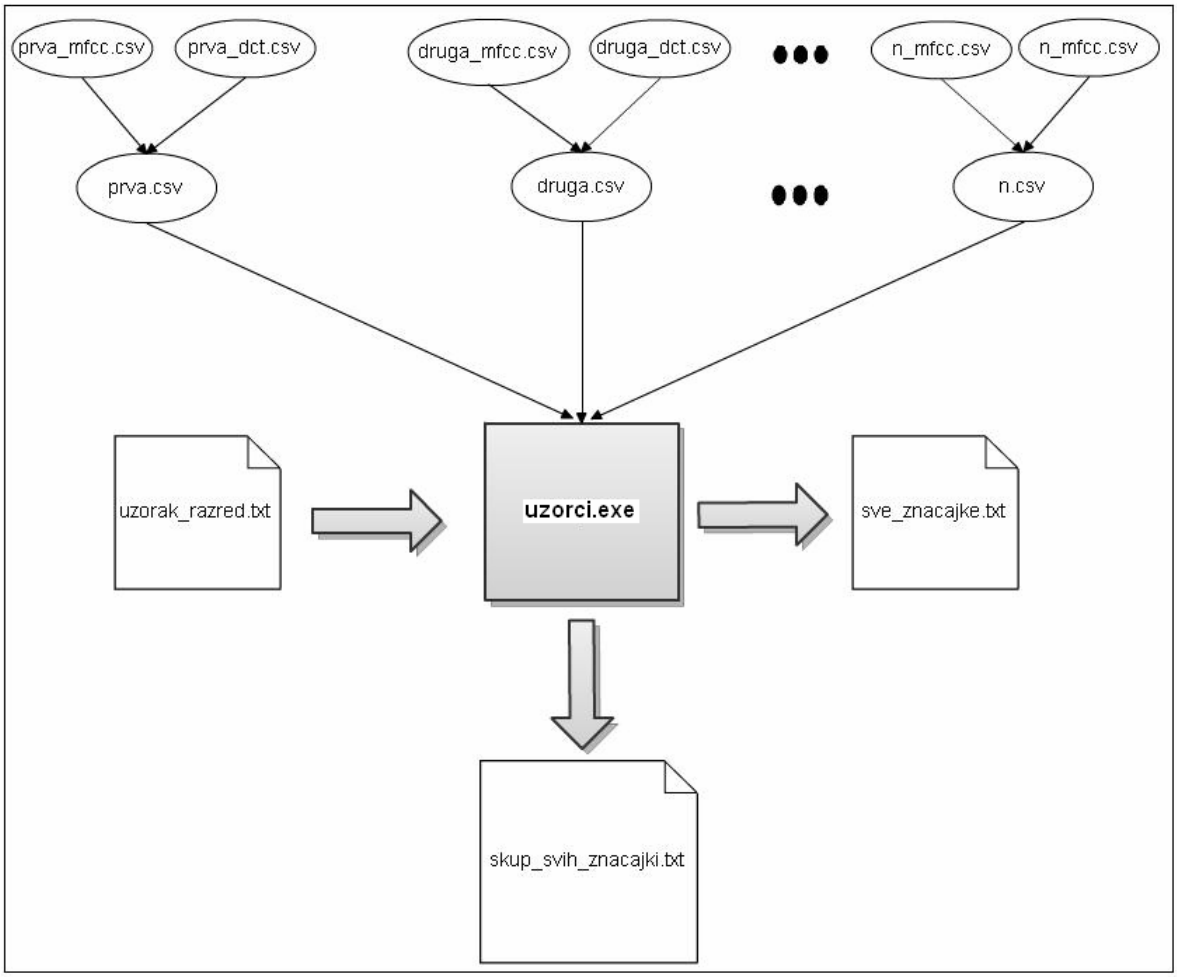
Dio je same skripte prevedene u izvršnu datoteku „Music Genre Classifier.exe“. Rekurzivno čita sve datoteke iz ulaznog glazbenog kazala i određuje "pravi" žanr svake pjesme (koja je jednog od podržanih formata - zadano u „conf-file.txt“) na jedan od 2 načina:

- čitajući žanr (engl. genre) id3 tag glazbene datoteke
- gledajući naziv vršnog kazala (dakle pretpostavlja se da su sve pjesme rock žanra negdje u kazalu naziva „rock“)

Kao izlaz daje datoteku „uzorak_razred.txt“ u kojoj piše žanr svake pjesme. Ovi podaci će se kasnije koristiti kao ocjenjivač pri treniranju i testiranju klasifikatora.

3.3 Parser formata uzoraka

Nakon što su značajke izlučene potrebno je tako dobivene vrijednosti zapisati na način prikladan za daljnju obradu. U ovom slučaju za treniranje i testiranje SVM-a.
Skripta „uzorci.pl“ radi upravo taj posao. Pronalazi sve datoteke u kojima su zapisane vrijednosti značajki (svaka pjesma ima svoju datoteku sa značajkama), te ih sprema u jednu datoteku. Za naš rad izabrali smo segmente pjesama u trajanju 5 sekundi. U prethodnom koraku moguće je odrediti koliko takvih segmenata želimo, upravo broj tih segmenata utječe na broj značajki. Uzmimo za primjer MFCC izlučivanje značajki, ono u segmentu od 5 sekundi sadrži 21 značajku, recimo da će biti izabrana tri proizvoljna segmenta od 5 sekundi tada će postojati 63 značajke.
Važno je spomenuti da navedena skripta ne ovisi o izboru značajki, te je moguće kombinirati više vrsta različitih značajki kako bi se dobio što bolji rezultat. Pretraživanje različitih vrsta značajki je potpuno automatizirano i ukoliko postoji više vrsta značajki sve značajke će biti pripremljene za daljnji rad sa SVM-om.
Kako bi se odredilo kojem razredu pripada pojedini uzorak koristi se tekstualna datoteka koja sadrži imena svih pjesama s pripadajućom oznakom razreda „uzorak_razred.txt“.
Pogledajmo sada kako izgleda dio arhitekture sustava vezan za opisanu jedinicu posla:



Slika 3.2 Detaljniji prikaz arhitekture vezan uz uzorci.exe

Izvršni program "uzorci.exe" radi slijedeće:

1) pretražuje sve datoteke u trenutnom direktoriju i provjerava ako imaju ekstenziju „.csv“
2) zatim sve koji imaju tu ekstenziju i jednako ime pjesme spaja u jednu datoteku
3) nakon toga sve datoteke sa značajkama zapisuje u jednu novu datoteku, dodatno zapisuje podatak o kojem razredu se radi za svaki uzorak (to se provjerava u datoteci "uzorak_razred.txt")
4) tako dobivene značajke sprema u "sve_znacajke.txt"
5) završni korak je odabir željenih značajki (one od 5 sekundi) i pretvara ih u oblik razumljiv SVM-u, te značajke zapisuju se u "skup_svih_znacajki.txt"

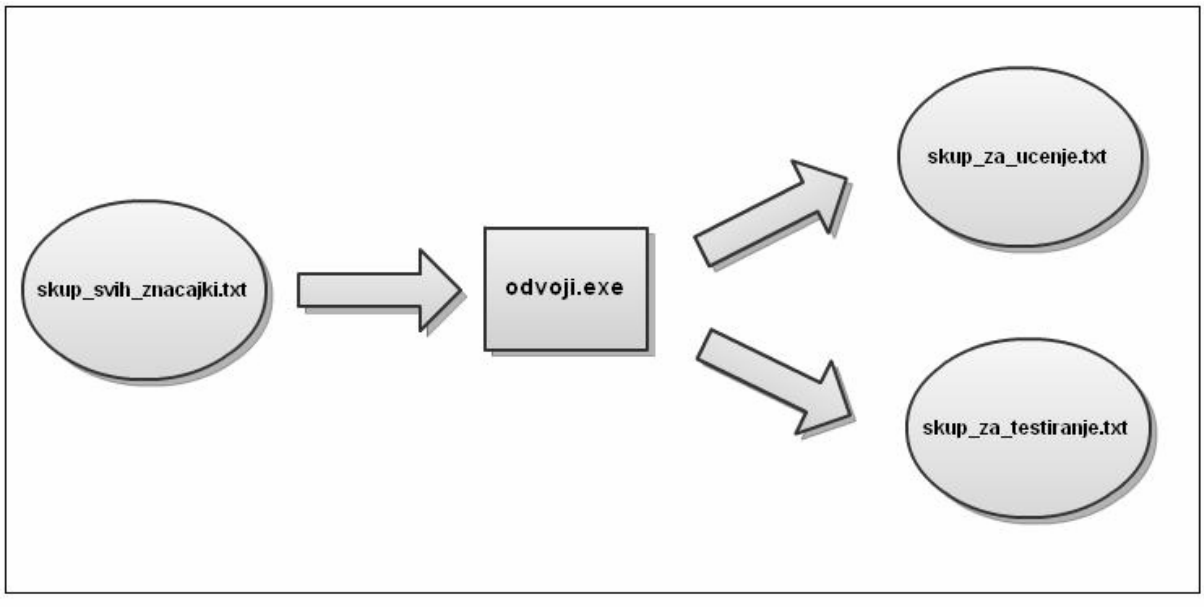
Upute za upotrebu:
Da bi program radio u istom direktoriju moraju biti:

- datoteke s ekstenzijom .csv
- uzorci.exe
- uzorak_razred.txt

Nakon završetka rada u istom direktoriju stvorit će se datoteka s međurezultatima „sve_znacajke.txt“ i datoteka s podacima koji se koriste za SVM „skup_svih_znacajki.txt“

3.4 Djelitelj uzoraka

Nakon što je prethodno opisanim postupkom dobivena datoteka sa pravilno oblikovanim značajkama, još je potrebno razdvojiti uzorke za učenje i uzorke za testiranje. Ovaj postupak implementiran je u skripti odvoji.pl. Skripta kao argument prima postotak uzoraka koji će se koristiti za učenje. Ovisno o zadanom postotku uzoraka po razredu koji će se koristiti za učenje, datoteka „skup_svih_znacajki.txt“ će biti razdvojena na dvije nove datoteke. Jedna od njih je „skup_za_ucenje.txt“ u kojoj će biti zadani postotak uzoraka i druga novonastala datoteka je „skup_za_testiranje.txt“ u kojoj će biti svi preostali uzorci. Pogledajmo kako izgleda dio arhitekture koji se bavi ovim problemom:

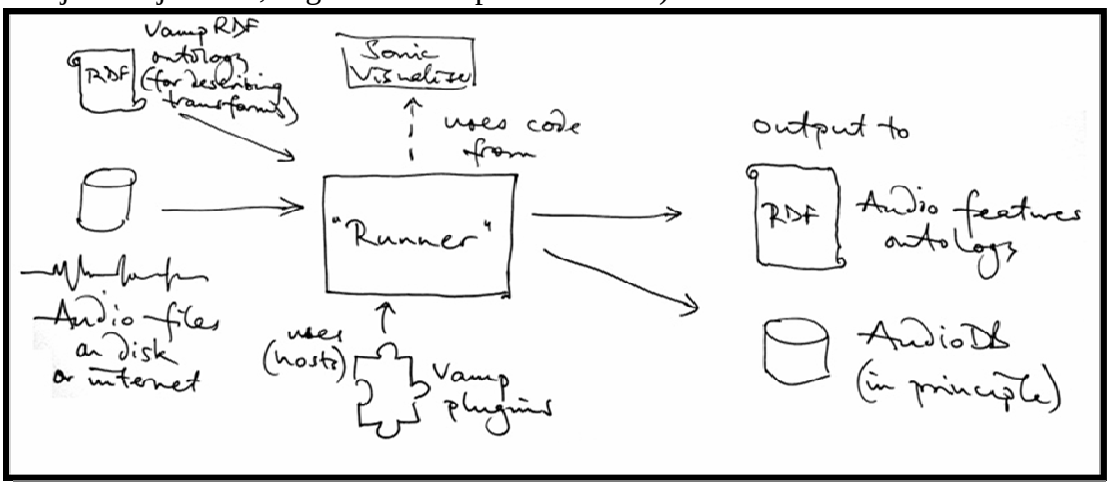


Slika 3.3 Grafički prikaz djelitelja uzoraka

3.5 Korišteni alati

3.5.1 Sonic Annotator

Sonic Annotator (ranije znan kao „Runner“) je program naredbenog retka (engl. command line) za izlučivanje glazbenih značajka iz više glazbenih datoteka. Osnovna ideja je pružiti proces izvlačenja značajki, a pritom sakriti konkretne metode, koristeći Vamp priključke (engl. Vamp plugins) za izlučivanje značajki. Izlazni formati su RDF, CSV (zarezom odvojene vrijednosti, engl. comma separated values).



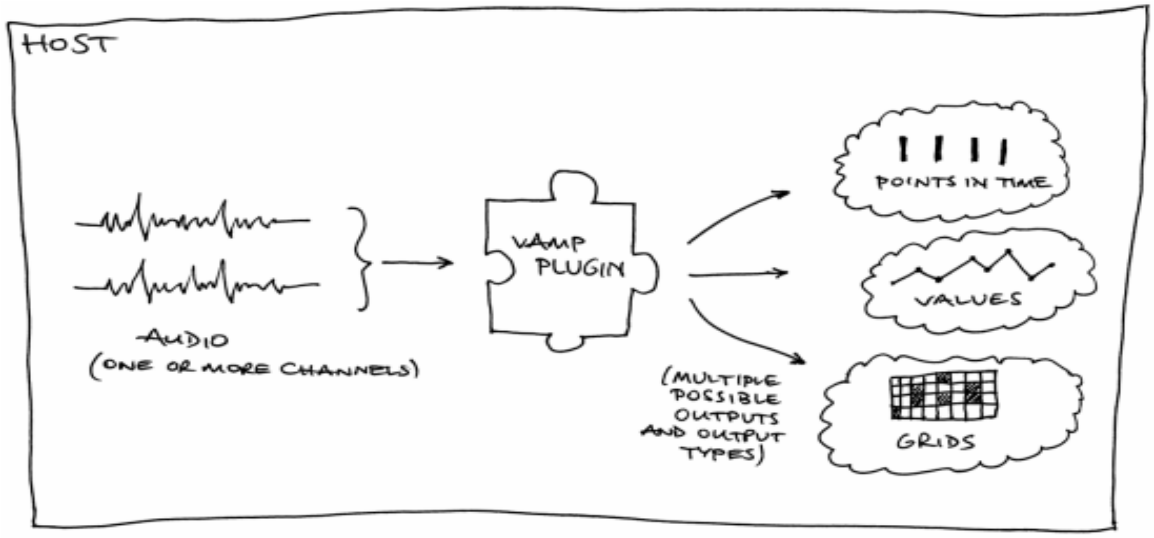
Slika 3.4 Arhitektura Sonic Annotatora

Sonic anotator kao ulaz koristi:
- RDF datoteku (tzv. kostur, engl. skeleton) koje opisuju Vamp priključke na koje se treba spojiti.
- lokaciju glazbenih datoteka koje treba obraditi (mogu biti čak i na Internetu)

Izlaz je skup .csv datoteka (ili drugačijih, ovisi kako se zada) za svaku pjesmu i svaku značajku koje sadrže vrijednosti tih (što te vrijednosti točno znače, zavisi od korištenih značajki).

3.5.2 Vamp plugins

Vamp je sustav priključaka za obradu zvuka koji izlučuje opisne informacije iz zvučnih podataka - drugim riječima vrši zvučnu analizu ili izlučivanje značajki.



Slika 3.5 Arhitektura Vamp priključaka

U našem projektu nisu bili direktno korišteni, već kroz Sonic Annotator, na koji se može gledati kao sučelje prema ovim priključcima.
Priključci koje smo instalirali za potrebe projekta su „Vamp libxtract plugins“.
To su priključci za izlučivanje značajki niske razine koji koriste libxtract biblioteku Jamiea Bullocka kako bi pružili oko 50 spektralnih i drugih značajki.

3.6 Korištene značajke

Pri testiranju smo kao značajke koristili mfcc (engl. Mel-frequency cepstral coefficients), koje se i u literaturi preporučaju za klasifikaciju glazbe. Računali smo ih na 3 intervala trajanja 5 sekundi u različitim dijelovima pjesme (<0,5> sec, <20,25> sec i <50,55> sec).
Dobivanje značajki mfccom je pobliže objašnjeno u teorijskom dijelu dokumentacije.

Sonic Annotator smo koristili naredbom:

```
sonic-annotator.exe -t mfcc -S mean --summary-only --segments 5,20,25,50,55 -w csv --csv-force --csv-basedir . -r d:\staza\do\kazala
```

Datoteka „mfcc“ je kostur za istoimeni Vamp priključak, a dobiva se izvođenjem naredbe:

```
sonic-annotator -s vamp:vamp-libxtract:mfcc:mfcc > mfcc
```

Oznake „mean“, „summary-only“ i „segments“ služe da se smanji inače jako velik broj vrijednosti. Segments određuje granice za vremenske intervale u kojima se onda svake sekunde radi sažetak (engl. summary) tako da se izračuna aritmetička sredina (engl. mean) svih vrijednosti.
U projektu smo kasnije parsirali izlazne .csv datoteke i uzeli samo vrijednosti iz 3 prije spomenuta intervala (zanemarujući ostale intervale, npr. <5,20> sec).
Tim procesom za svaku glazbenu datoteku (pod uvjetom da je duža od 55 sekundi) dobijemo 63 značajke. Za SVM klasifikator se i inače preporuča stotinjak vrijednosti po uzorku.

3.7 Upute za instalaciju i korištenje

Kako aplikacija zavisi o programu Sonic Annotator za izvlačenje značajki iz glazbenih datoteka, a Sonic Annotator nadalje koristi Vamp priključke (engl. Vamp plugins), potrebno je imati Vamp priključke na točno zadanom mjestu. Sam Sonic Annotator ne treba posebno instalirati jer je to samo jedna izvršna datoteka koja se nalazi u „working bundle“ kazalu.

3.7.1 Instalacija Vamp priključaka

1) skinite željene Vamp priključke (u projektu smo koristili libxtract) sa ove stranice http://www.vamp-plugins.org/download.html
2) raspakirajte .rar datoteku i stavite.dll i .cat datoteke u kazalo „C:\Program Files\Vamp Plugins“ - ako koristite 64-bitni Windows OS, umjesto prethodnog koristite kazalo "C:\Program Files (x86)\Vamp Plugins"
3) testirajte dostupnost Vamp priključaka tako što se u naredbenom retku (engl. command prompt) pozicionirate na „working bundle“ kazalo našeg projekta

(gdje je i sonic-annotator.exe) i pokrenete Sonic Annotator naredbom: „sonic-annotator.exe –l (ukoliko se ne ispisuju razni vamp:libxtract priključci (osim „example“ priključaka) onda nešto nije prošlo dobro u koraku 2)

3.7.2 Pokretanje aplikacije

Sve izvršne datoteke (i jedna konfiguracijska) potrebne za pokretanje aplikacije (Sonic Annotator također) dostupne su u kazalu „working bundle“.
Glavna skripta koja kontrolira sve druge dijelove programa je prevedena u izvršnu datoteku „music_genre_classifier.exe“. Potpunu funkcionalnost programa moguće je vidjeti njenim pokretanjem (najbolje u naredbenom retku za bolji pregled ispisa).

3.7.3 Konfiguracijska datoteka

Sve postavke i parametre aplikacije podešava se kroz tekstualnu datoteku „conf-file.txt“. U komentarima datoteke objašnjeno je ukratko što točno koji parametar znači i koje su moguće vrijednosti.

NAPOMENA: poredak redaka u konfiguracijskoj datoteci je bitan i ne smije ga se mijenjati!

Primjer izgleda konfiguracijske datoteke:

```
##
##  KONFIGURACIJSKA DATOTEKA za skriptu koja kontrolira pojedine dijelove aplikacije
##


# direktorij gdje se nalazi glazba za učenje i testiranje
D:\glazba


# tipovi glazbe koji se žele podržati (ostali se ignoriraju) - sonic annotator isto mora podržavati tip
.mp3 .wav


# žanrovi koji se žele podržavati
classical,rock,jazz


# naziv datoteke u koju će se zapisati pripadnost datoteke razredu (NE MIJENJATI)
uzorak_razred.txt


# željena metoda za određivanje žanra: folder ili id3
# ili na eng...
```

```
# desired method for determining the actual genre (for validation purposes): "folder" or "id3"
# with "folder" all the songs are supposed to be in a folder of it's genre's name
# with "id3" the value is extracted from id3 tags
folder

# pozovi sonic annotator (ovisno o broju datoteka može trajati jako dugo): 1 ili 0
# ako se zada 0, potrebno je ručno kopirati neke unaprijed izračunate značajke
1

# naredba koja se koristi za sonic annotator (bez staze do foldera s muzikom, to se konkatenira na kraj)
sonic-annotator.exe -t mfcc -S mean --summary-only --segments 5,20,25,50,55 -w csv --csv-force --csv-basedir . -r

# naredba za skaliranje
svm-scale.exe -l 0 -u 1

# klasifikator: libsvm ili kreshvm
# "libsvm" znači koristi biblioteku libsvm
# "kreshvm" je naš klasifikator
kreshvm

# naredba za libsvm-ov svm-train
svm-train -s 0 -c 100 -g 0.1 -v 5

# naredba za treniranje našeg klasifikatora kreshvm
# kreshvm_train.exe 63 0.5 1
# kreshvm_train parametri
# drugi parametar - konstanta C
5

# drugi parametar - tip svm-a
#  1 - linearni
#  2 - rbf kernel
#  3 - polinomni kernel
```

```
1

# treći parametar - gama za rbf, stupanj za polinomni kernel, nebitno za linearni svm
4.3
# testiranje našeg klasifikatora
kreshvm_test.exe

#naredba za odvajanje skupa podataka na učenje i testiranje
odvoji.exe 80%
```

3.7.4 Obrazac korištenja aplikacije 1 - testiranje klasifikacije

Pretpostavimo da se na računalu pod "D:\glazba" nalaze 3 kazala „jazz“, „classical“ i „rock“ u kojima se dalje nalaze kazala izvođača i albuma (proizvoljne dubine, npr. „D:\glazba\rock\Led Zeppelin\The Song Remains The Same\Rain Song.mp3“ je dozvoljena staza), a negdje u njima razne glazbene datoteke odgovarajućeg žanra (dakle sve rock pjesme su negdje u folderu „rock“).
Mi želimo naučiti SVM s 80% tih glazbenih datoteka (ravnomjerno odabranih pripadnika svakog žanra) i testirati ga s preostalih 20%.
Tada postavljamo sljedeće parametre konfiguracijske datoteke conf-file.txt:
- ulazni glazbeni direktorij - D:\glazba
- žanrovi koji se žele podržavati - jazz,rock,classical
(moraju se poklapati s kazalima žanrova)
- željena metoda za određivanje žanra - folder
(zato što kazalo (engl. folder) određuje ispravni žanr, a ne id3 tag)
- naredba za odvajanje skupa podataka na učenje i testiranje - odvoji.exe 80%
- pozovi sonic annotator - 1
(ako nemamo od prije izračunate značajke za glazbene datoteke)
- klasifikator - kreshvm
(ovo je naš klasifikator pa vjerojatno njega želimo testirati)
Ostale vrijednosti su parametri drugim dijelovima aplikacije te su detaljnije objašnjene u drugim dijelovima dokumentacije, a pretpostavljene vrijednosti bi trebale biti dovoljno dobre u većini slučajeva.
Kako to točno zapisati u "conf-file.txt" vidljivo je u prethodnom odjeljku jer upravo ta datoteka odgovara ovom primjeru.
Nakon što smo to podesili dovoljno je pokrenuti "music_genre_classifier.exe" i ako sve odradi dobro, na kraju će se ispisati postotak uspješno klasificiranih testnih glazbenih datoteka (onih 20% koje nisu bile korištene za učenje SVM-a).

3.7.5 Obrazac korištenja aplikacije 2 - ponovno testiranje

Recimo da želimo promijeniti neki parametar i ponovno testirati klasifikator, s novim parametarom korištenim pri učenju. Daleko najsporiji dio rada aplikacije je izlučivanje značajki uz pomoć Sonic Annotatora.
Da ne bi ponovno morali čekati vrlo dug proces obrade glazbenih datoteka, možemo ubrzati proces tako da ne pozivamo Sonic Annotator.

To se postavlja u konfiguracijskoj datoteci pišući pod **pozovi sonic annotator - 0**

NAPOMENA: ukoliko se želi koristiti ovo ubrzanje, u kazalu "working bundle" zajedno s izvršnim datotekama moraju biti **.csv datoteke** svih onih glazbenih datoteka koje želimo koristiti za treniranje i testiranje i **uzorak_razred.txt datoteka** koja sadržava sve te pjesme s oznakama žanrova. Ako ništa nije brisano od zadnjeg računanja, ovaj uvjet bi trebao biti zadovoljen.
Kako bi koristili već izračunate značajke (i ubrzali testiranje različitih parametara svm-a) pripremljena je zaliha unaprijed izračunatih značajki u kazalu „precomputed examples“ (najviše za GTZAN). Te značajke se mogu iskoristiti tako da se odabere neku od ovih baza (npr. gtzan) i iz kazala "gtzan" koji se nalazi u kazalu "precomputed examples" kopira u "working bundle" kazalo datoteku uzorak_razred.txt i sve značajke koje se žele računati.
Ako bi npr. htjeli računati klasifikaciju po mfcc-u i nonzero-count-u iskopirali bi sadržaje ta 2 kazala u "working bundle".

4. Rezultati

Točnost klasifikacije je računata tako da se skup uzoraka (pjesama) podijelio na skup za treniranje i skup za testiranje u nekom omjeru (koji će biti naveden, uglavnom je 80% korišteno za treniranje). Presjek ta dva skupa je uvijek prazan skup. Ta se metoda naziva još i hold-out metoda jer je jedan dio skupa ostao „tajan“ i korišten samo za testiranje.
Točnost predstavlja omjer točno klasificiranih uzoraka i svih uzoraka iz skupa za testiranje.

4.1 Baze pjesama

Za testiranje smo koristili više baza:

1) ISMIR – Na konferenciji ISMIR 2004. je objavljena baza pjesama za natjecanje u raspoznavanju žanrova. Sastoji se od 729 pjesama u 8 žanrova. Mana je što neki razredi imaju svega nekoliko pjesama (pop npr. ima samo 6 pjesama). Baza je dostupna za skidanje na adresi http://ismir2004.ismir.net/genre_contest/index.html
2) GTZAN – Ovaj skup pjesama je bio korišten za rad u raspoznavanju žanrova „Musical genre classification of audio signals“ G. Tzanetakisa i P. Cooka. Dobro je što od svakog žanra ima po 100 isječaka pjesama (u trajanju od 30 sekundi), no mana je što su stavili dosta (čak i za ljude) sličnih žanrova – recimo blues,country, reggae, pop, disco. Razlike između nekih pjesama tih žanrova su prilično diskutabilne što se može vidjeti i iz rezultata klasifikacije. Može se skinuti na adresi http://marsyas.sness.net/download/data_sets
3) osobna – Ovaj skup smo sami sastavili od 57 pjesama, žanrova rock, classical i jazz (19 pjesama svaki žanr) i predstavlja malo lakši primjer.

BAZA	BROJ ŽANROVA	ZNAČAJKE	KLASIFIKATOR	PARAMETAR S KOJIM JE DOBIVENA NAJBOLJA TOČNOST	TRENIRANJE (%)	NAJBOLJA DOBIVENA TOČNOST (%)
ismir	8	mfcc	kreshvm	C=2, rbf, gama=0.1	80	56,7376
gtzan	10	mfcc	kreshvm	C=0.6, polinomni, 1. Stupanj	80	48
osobna	3	mfcc	kreshvm	C=2, rbf, gama=0.1 i također C=0.1, linearni	80	91,6667

Tablica 4.1 Rezultati s variranjem baza

Iz ispisa klasifikatora su se potvrdile slutnje navedene kao mane u opisima baza. Samo je za našu, „relativno čistu“ bazu klasifikator imao svega jednu grešku.

4.2 Značajke

Izbor značajki je jedna od presudnih elemenata uspješne klasifikacije. Ovdje je dan pregled njihovih različitih kombinacija (kada je navedeno više značajki, znači da se za svaki uzorak izračunaju sve značajke i nanižu). Broj vrijednosti je ukupan broj vrijednosti svih korištenih značajki koje se dobiju za svaki uzorak.
Korištena je GTZAN baza, 80% treniranje, a parametri svm-a su: C=0.6, polinomni kernel 1. stupnja.

ZNAČAJKE	BROJ VRIJEDNOSTI	TOČNOST (%)
mfcc	63	48
loudness	3	18,5
mfcc, loudness	66	41,5
nonzero count	3	14,5
spectral centroid	3	17
mean	3	16
mean, nonzero count	6	13
mean, nonzero count, loudness	9	18,5
mean, nonzero count, loudness, mfcc, spectral_centroid	75	42,5
mean, nonzero count, loudness, mfcc, spectral centroid, spectral slope, spectral standard deviation	81	39

Tablica 4.2 Rezultati s variranjem značajki

Vidi se da više značajki može dati bolje rezultate. Iako niti jedna značajka sama za sebe nije dobra kao mfcc, mfcc uz 4 druge značajke daje bolje rezultate. Moguća je, međutim, i suprotna tendencija što se vidi iz slučaja sa 7 značajki koje daju manju točnost od slučaja s 5 značajki.

4.3 Usporedba s libsvm-om

Uz klasifikator koji smo sami napravili, koristili smo i gotov klasifikator „libsvm“ za usporedbu.
Za oba klasifikatora su korišteni najbolji pronađeni parametri i baza GTZAN . Za naš kreshvm je korišteno C=0.6, polinomni kernel 1. stupnja. Za kreshvm je korišten rbf kernel uz gama=0.1 i C=100 (naredba svm-train -s 0 -c 100 -g 0.1 -v 5). Točnost je određena peterostrukom unakrsnom usporedbom (engl. cross-validation), što smo smatrali najbliže našem dijeljenju skupa na 80% za učenje, 20% za testiranje.

KLASIFIKAOR	ZNAČAJKE	TOČNOST
kreshvm	sve	39
libsvm	sve	60.7
kreshvm	mfcc	48
libsvm	mfcc	58.2

Tablica 4.3 Rezultati usporedbe kreshvm-a i libsvm-a

Vidi se da je libsvm nešto bolji od našeg kreshvm-a, ali ne mnogo. Treba, također, uzeti u obzir da su na libsvmu radili stručnjaci u tom području više godina (9 godina).

4.4 Veličina skupa za treniranje

Usporedili smo rezultate u ovisnosti o tome koliko je velik bio skup za učenje, tj. koliki je dio korišten za učenje, a koliki za testiranje klasifikatora.
Korištena je GTZAN baza, 80% treniranje, a parametri svm-a su: C=0.6, polinomni kernel 1. stupnja.

DIO CIJELE BAZE KOJI SE KORISTI ZA UČENJE (OSTATAK ZA TESTIRANJE)	TOČNOST(%)
50	28,2
60	28,5
70	33,6667
80	39
90	55

Tablica 4.4 Rezultati u ovisnosti o veličini skupa za učenje

Vidi se da povećanjem skupa za učenje gotovo jednoliko raste točnost klasifikacije.

4.5 Parametri kreshvm-a

Usporedili smo rezultate u ovisnosti o parametrima kernela kresvm-a.
Korištena je GTZAN baza, 80% treniranje, korištene značajke su MFCC, te svih 10 razreda navedene baze. Pogledajmo rezultate za linearni kernel:

VRSTA KERNELA	C	REZULTAT (%)
linearni	0,1	31
linearni	0,5	47,5
linearni	0,6	48
linearni	0,7	47
linearni	0,8	44,5
linearni	0,9	43
linearni	1	42,5
linearni	1,1	43,5
linearni	1,2	43
linearni	1,3	44,5
linearni	1,5	42
linearni	2	40,5
linearni	3	39,5
linearni	4	37,5
linearni	5	39,5

Tablica 4.5 Rezultati za linearni kernel

Možemo zaključiti da kod linearnog kernela izbor parametra C ovisi o dobrim rezultatima. Korištenje dobrog parametra dobivamo točnost od 48%.

VRSTA KERNELA	C	GAMMA	REZULTAT (%)
rbf	0,5	0,7	36
rbf	0,5	4,3	10
rbf	0,6	0,1	31,5
rbf	0,6	0,3	42
rbf	0,6	0,4	41,5
rbf	0,6	0,5	38,5
rbf	0,6	0,7	34,5
rbf	0,6	0,8	34
rbf	0,6	0,9	32,5
rbf	1	0,1	40,5
rbf	1	0,2	44,5
rbf	1	0,3	43,5
rbf	1	0,4	41,5
rbf	1	0,5	41,5
rbf	1	0,6	39,5
rbf	1	0,7	37
rbf	1	0,8	35,5
rbf	2	0,01	21,5
rbf	2	0,05	44
rbf	2	0,07	45,5
rbf	2	0,08	46,5
rbf	2	0,1	46,5
rbf	2	0,2	44,5
rbf	2	0,3	43,5
rbf	2	0,4	41,5
rbf	2	0,5	39
rbf	2	0,6	36
rbf	3	0,35	40,5
rbf	5	4,3	10,5
rbf	50	0,1	34

Tablica 4.6 Rezultati za RBF kernel

Iz tablice se može zaključiti da odabir dobrih parametara uvelike ovisi o rezultatu. Loš izbor parametara C = 5 i gamma = 4.3 daje točnost od samo 10,5%, dok bi izbor dobrih parametar bio C = 2 i gamma = 0,1 davao 46,5%.

VRSTA KERNELA	C	STUPANJ	REZULTAT (%)
poli	0,6	1	48
poli	0,6	2	29
poli	0,6	3	24,5
poli	0,6	4	23
poli	1	2	29,5
poli	2	2	26

Tablica 4.7 Rezultati za polinomni kernel

U slučaju ove baze dolazi do smanjenja točnosti povećanjem stupnja polinoma.
Nakon što promotrili prethodno navedene tablice može se zaključiti koliko je važno postaviti prave parametre.

4.6 Broj razreda

Pogledajmo sada kako rezultat ovisi o broju razreda. Za ovo testiranje korištena je baza GTZAN koja ima 10 razreda. Razredi u ovoj bazi su: blues(1), classical(2), country(3), disco(4), hip-hop(5), jazz(6), metal(7), pop(8), reggae(9), rock(10). Brojevi navedeni u zagradi će biti u tablici kako bi zadržali preglednost.
Korištene su značajke MFCC, klasifikator je kreshvm, postotak za treniranje 80%.

VRSTA KERNELA	C	GAMMA/STUPANJ	BROJ RAZREDA	REZULTAT (%)	TESTIRANI RAZREDI
linearni	0,6		10	48	svi
linearni	0,6		8	60	bez razreda 4,6
linearni	0,6		7	63,57	bez razreda 1,4,6
linearni	0,6		7	53,57	bez razreda 3,5,10
linearni	0,6		5	75	bez razreda 1,4,5,6,9
linearni	0,6		3	98,33	bez razreda 1,4,5,6,7,9,10
poli	0,6	2	10	29	svi
poli	0,6	2	4	63,75	bez razreda 3,4,5,7,9,10
rbf	2	0,1	10	46,5	svi
rbf	2	0,1	7	62,15	bez razreda 1,4,6
rbf	2	0,1	5	75	bez razreda 1,4,5,6,7
rbf	2	0,1	3	95	bez razreda 1,3,4,5,6,7,9

Tablica 4.8 Rezultati uz variranje broja razreda

Iz tablice se može pročitati da rezultati uvelike ovise o broju razreda. Za svih 10 razreda dobiva se točnost 48%, dok za 7 razreda da točnost raste na 63,57%. Za samo tri razreda dobiva se rezultat 98,33% što je jako dobro. Ovaj rezultat je dobiven za razrede classical,

country i pop. Još je važno napomenuti da se za razred disco dobivaju iznimno loši rezultati.

5. Literatura

[1] Shigeo Abe, Support Vector Machines for Pattern Classification, Springer-Verlag London 2005

[2] John C. Platt, Fast Training of Support Vector Machines using Sequential Minimal Optimization,

[3] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, K. R. K. Murthy, Improvements to Platt's SMO Algorithm for SVM Classifier Design ,u Bernhard Scholkopf, Cristopher J. C. Burges, Sebastian Mika, Advances in Kernel Methods, MIT Press 1998

[4] Rong-En Fan, Pai-Hsuen Chen, Chih-Jen Lin, Working Set Selection Using Second Order Information for Training Support Vector Machines, Journal of Machine Learning Research 11/2005.

[5] Thorsten Joachims, Making Large-Scale SVM Learning Practical, u Bernhard Scholkopf, Cristopher J. C. Burges, Sebastian Mika, Advances in Kernel Methods, MIT Press 1998

[6] S.Theodoridis, K. Koutroumbas, Pattern Recognition, Elsevier (USA) 2006.

[7] Karin Kosina, Music Genre Recognition, Hagenberg, May 2002.

[8] B. Logan, Mel frequency cepstral coefficients for music modeling , In Proceedings of ISMIR, 2000., Cambrige ,MA

[9] Paolo Annesi, Roberto Basili, Raffael Gitto, Alessandro Moschitti – Audio Feature Engineering for Automatic Music Genre Classification, IT

[10] Md. Rashidul Hasan, Mustafa Jamil, Md. Golam Rabbani Md. Saifur Rahman – Speaker Identification Using Mel Frequency Cepstral Coefficients, Dhaka, Bangladesh

[11] Speaker Identification Using Mel Frequency Cepstral Coefficients, Dhaka, Bangladesh

Anders Meng, John Shawe –Taylor, An Investigation of Feature Models for Music Genre Classification Using the Support Vector Classifier, UK

[12] Jesper Hojvang Jensen, Mads Graesboll Christensen, Manohar N. Murthi, and Soren Holdt Jensen – Evaluation Of MFCC Estimation Techniques for Music Similarity, Denmark

[13] Peter Ahrendt, Music Genre Classification System - A Computational Approach, Kongens Lyngby, IMM-PHD-2006-164

[14] Fang Zheng, Guoliang Zhang, Zhanjiang Song, Comparison of Different Implementations of MFCC, Sept. 2001, Beijing,10084, P.R. China

[15] T. M. Mitchell. Machine Learning. Mc Graw Hill, 1997.