SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1664

Implementacija i primjena algoritma stroja potpornih vektora u klasifikaciji i regresiji

Ivan Ćorić

Zagreb, lipanj 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 15. ožujka 2018.

DIPLOMSKI ZADATAK br. 1664

Pristupnik: **Ivan Ćorić (0036477124)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Implementacija i primjena algoritma stroja potpornih vektora u klasifikaciji i regresiji**

Opis zadatka:

Opisati algoritam stroja potpornih vektora i njegovu primjenu u problemima klasifikacije i regresije. Ispitati mogućnosti proširenja osnovne inačice algoritma s obzirom na optimizaciju hiperparametara i područje primjene. Razviti programsko ostvarenje algoritma stroja potpornih vektora za primjenu u klasifikaciji te regresiji u sklopu postojećeg programskog okvira za evolucijsko računanje. Primijeniti razvijeno rješenje na zadane probleme klasifikacije i regresije. Ispitati učinkovitost ostvarenog algoritma s obzirom na vrstu problema te različite metode optimizacije hiperparametara. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Zadatak uručen pristupniku: 16. ožujka 2018.
Rok za predaju rada: 29. lipnja 2018.

Mentor:

Prof. dr. sc. Domagoj Jakobović

Djelovođa:

Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za diplomski rad profila:

Prof. dr. sc. Siniša Srbljić

SADRŽAJ

1. Uvod

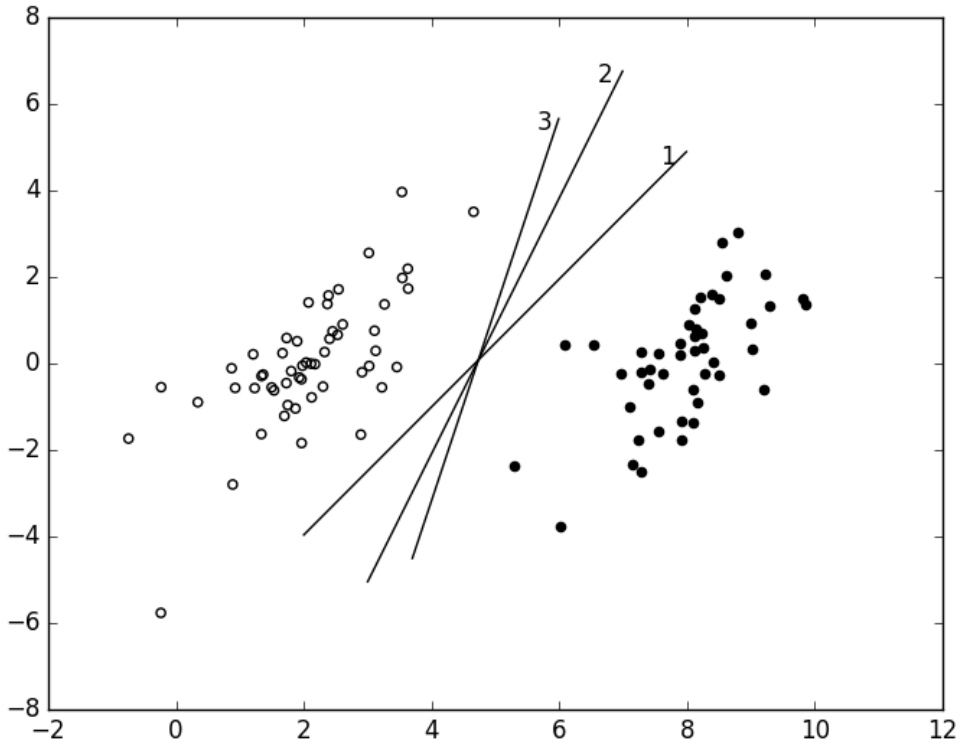
Algoritam stroja potpornih vektora (engl. Support Vector Machines (SVM)) je model nadziranog učenja koji se može koristiti za klasifikaciju i regresiju. Osmislili su ga V. N. Vapnik i A. Y. Chervonenkis 1963. godine. Originalno je SVM bio razvijen za klasifikaciju (SVC). SVC je linearni binarni klasifikator koji može rješavati nelinearne probleme pomoću jezgrenog trika (B. E. Boser, I. M. Guyon i V. N. Vapnik, 1992.) te se može proširiti na višeklasne probleme na standardan način. Regresija pomoću SVM-a (SVR) predložena je 1996. godine (V. N. Vapnik, H. Drucker, C. J. C. Burges, L. Kaufman and A. J. Smola.).

U ovom radu opisat će se algoritam stroja potpornih vektora posebno za klasifikaciju i regresiju. U oba slučaja, za optimizaciju bit će korišten Plattov algoritam Sekvencijalna minimalna optimizacija (engl. Sequential minimal optimization (SMO)) koji će također biti opisan. Usporedit će se primjena SVM-a na probleme klasifikacije i regresije s drugim implementacijama.

2. Klasifikacija potpornim vektorima

2.1. Motivacija

Neka je dan skup n - dimenzionalnih podataka, takvih da jedan dio podataka pripada jednoj klasi, a drugi pripada drugoj klasi. Zadatak je pronaći algoritam koji može odrediti kojoj klasi određeni podatak pripada. Ovo je uobičajen problem strojnog učenja. U ovom radu on je riješen koristeći SVM algoritam.



Slika 2.1: Prikaz skupa podataka koji se sastoji od dvije klase te mogućih pravaca koji ih razdjeljuju.

Ideja SVC-a je od svih mogućih $(n-1)$ - dimenzionalnih hiperravnina između dvije

klase, izabrati onu za koju je udaljenost do najbližeg primjera (podatka) sa svake strane maksimalna. To možemo vidjeti na slici (2.1) gdje se od svih pravaca koji razdjeljuju dvije skupine bira upravo pravac označen brojem 2, jer on maksimizira udaljenost do najbližih primjera.

2.2. Teorijska razmatranja

Neka je dan skup podataka:

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\},$$

gdje je $\mathbf{x}_i$ n - dimenzionalni vektor realnih brojeva koji predstavlja podatak, a y_i predstavlja klasu kojoj podatak pripada te poprima vrijednosti 1 ili -1. Tražimo hiperravninu čija je jednadžba dana kao:

$$h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = 0, \tag{2.1}$$

gdje su $\mathbf{w}$ i b zasad neodređeni parametri. $\mathbf{w}$ možemo promatrati i kao vektor normale na hiperravninu.

2.2.1. Tvrda margina

Pretpostavimo zasad linearnu odvojivost skupa $\mathcal{D}$. Tada imamo:

$$\forall \mathbf{x}_i \in \mathcal{D},\ y_i h(\mathbf{x}_i) \geq 0. \tag{2.2}$$

Udaljenost primjera $\mathbf{x}_i$ od hiperravnine iznosi $\frac{y_i h(\mathbf{x}_i)}{\|\mathbf{w}\|}$ pa formulaciju problema (tj. da tražimo hiperravninu koja je najdalja od najbližih primjera) možemo matematički zapisati kao:

$$max_{\mathbf{w},b}\left\{ \frac{1}{\|\mathbf{w}\|} min_i\{y_i(\mathbf{w} \cdot \mathbf{x}_i + b)\} \right\}. \tag{2.3}$$

Problem optimizacije (2.3) invarijantan je na skaliranje parametara $\mathbf{w}$ i b s realnim pozitivnim koeficijentom. Zbog toga imamo slobodu odabira skale te je postavljamo tako da je udaljenost hiperravnine do najbližih primjera jednaka točno 1 pa za sve primjere vrijedi:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, ..., N$$

Problem (2.3) se tada svodi na:

$$max_{\mathbf{w},b}\left\{ \frac{1}{\|\mathbf{w}\|} \right\},$$

uz ograničenja:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, ..., N.$$

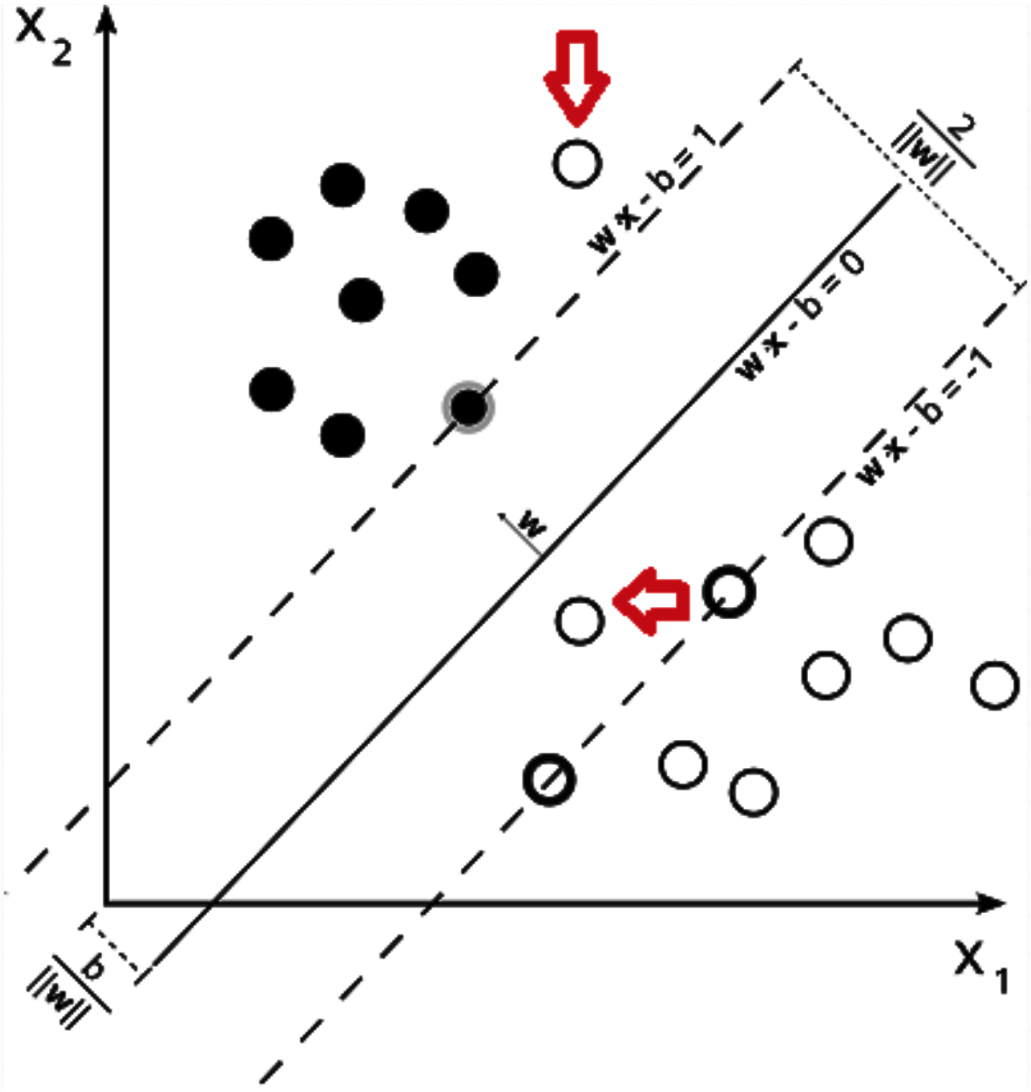
Ekvivalentno možemo pisati:

$$min_{\mathbf{w},b}\left\{\frac{1}{2}\|\mathbf{w}\|^2\right\}, \tag{2.4}$$

uz ograničenja:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, ..., N.$$

Time se dobiva problem kvadratnog programiranja, no kako zahtjev da primjeri ne smiju ulaziti u marginu (područje između najbližih primjera) vodi do prenaučenosti, doputit ćemo primjerima da uđu u marginu s određenom kaznom.



Slika 2.2: Prikaz ulaska primjera u marginu, slika posuđena iz [10]

2.2.2. Meka margina

Neka je sada dozvoljeno primjerima da ulaze u marginu. Tada ograničenja postaju:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \;\; i = 1, ..., N,$$

gdje ξ govori koliko je primjer unutar margine. Točnije, ξ ima ova svojstva:

- $0 < \xi < 1 \rightarrow$ primjer je ispravno klasificiran, ali unutar margine,
- $\xi > 1 \rightarrow$ primjer je pogrešno klasificiran.

Na slici 2.5 možemo vidjeti primjere označene sa strelicom koji imaju ξ različit od nule. Donji primjer označen sa strelicom ima $0 < \xi < 1$, dok gornji ima $\xi > 1$. Kako bismo uključili kaznu za ulaske u marginu, ciljnoj funkciji dodajemo sumu kazni tj. problem postaje:

$$min_{\mathbf{w},b}\left\{\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i\right\}, \tag{2.5}$$

uz ograničenja:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \;\; i = 1, ..., N,$$

gdje je C parametar takav da veći C predstavlja tvrđu marginu, a manji mekšu.

2.2.3. Dualni problem

Sada želimo problem (2.5) prebaciti u dualni problem [4] koji ima nekoliko prednosti koje ćemo komentirati kasnije. Prvo pišemo Lagrangeovu funkciju za problem (2.5):

$$\mathcal{L}\left(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}\right) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i) - \sum_{i=1}^{N}\beta_i\xi_i, \tag{2.6}$$

gdje su $\boldsymbol{\alpha}$ i $\boldsymbol{\beta}$ Lagrangeovi multiplikatori. Rješenje ovog problema mora zadovoljavati Karush–Kuhn–Tucker (KKT) uvjete [9]:

$$\left.\begin{aligned} y_i(\mathbf{w} \cdot \mathbf{x}_i + b) &\geq 1 - \xi_i \\ \xi_i &\geq 0 \\ \alpha_i &\geq 0 \\ \beta_i &\geq 0 \\ \alpha_i(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i) &= 0 \\ \beta_i\xi_i &= 0 \end{aligned}\right\} i = 1, ..., N.$$

Da bismo dobili dualnu Lagrangeovu funkciju, računamo minimum po primarnim parametrima:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = 0 \rightarrow \boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i,$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0,$$

$$\frac{\partial \mathcal{L}}{\partial \xi} = 0 \rightarrow \alpha_i = C - \beta_i.$$

Uvrštavanjem u (2.6) dobivamo dualnu Lagrangeovu funkciju:

$$L(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j.$$

Pripadni dualni optimizacijski problem dan je kao:

$$max_{\boldsymbol{\alpha}} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \right\}, \tag{2.7}$$

uz uvjete:

$$0 \leq \alpha_i \leq C, \ i = 1, ..., N$$

i

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

Kako riješiti ovaj problem bit će opisano za dva pododjeljka, za sada možemo napisati jednadžbu hiperravnine koja predstavlja granicu između dvije klase:

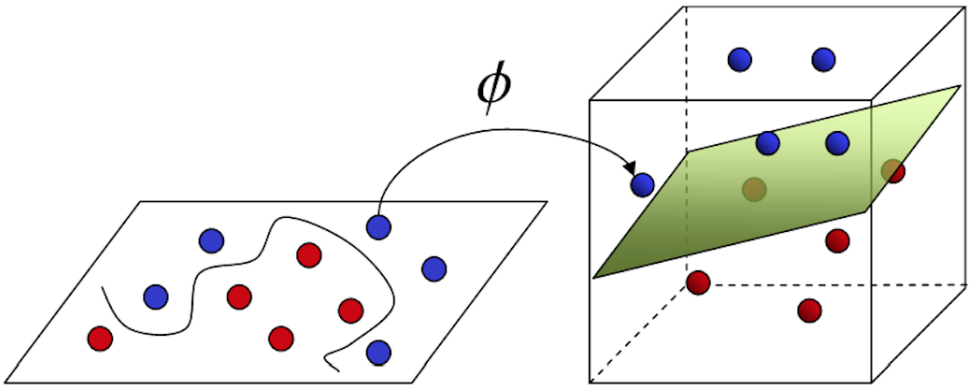
$$h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i \cdot \boldsymbol{x} + b = 0. \tag{2.8}$$

Sada možemo i objasniti i ime samog algoritma. Naime, za neke primjere vrijedit će $\alpha_i = 0$, takvi primjeri ne sudjeluju u klasifikaciji. Ostali primjeri, za koje je $\alpha_i \neq 0$ se nazivaju potporni vektori i kod klasifikacije dani primjer trebamo usporediti sa svakim potpornim vektorom (skalarni produkt $\boldsymbol{x}_i \cdot \boldsymbol{x}$ mjeri sličnost između danog podatka i primjera).

2.2.4. Jezgreni trik

Do sada SVC je bio linearan klasifikator. Kako bi mogao rješavati nelinearne probleme, ideja je da se osmisli preslikavanje $\Phi(\boldsymbol{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^m$ u višedimenzionalan prostor u kojem bi primjeri mogli biti linearno odvojivi. Ideju možemo vidjeti na slici

2.3. Time umjesto skalarnih produkta $\boldsymbol{x}_i \cdot \boldsymbol{x}$, u (2.9) imamo skalarne produkte oblika $\Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x})$.



Slika 2.3: Prikaz preslikavanja podataka u višedimenzionalan prostor radi postizanja linearne odvojivosti. Posuđeno iz [7].

Traženje preslikavanja za svaki poseban problem može predstavljati velik problem. Također, preslikavanje primjera u m - dimenzionalan prostor pa računanje skalarnog produkta može bit računski skupa operacija. To nas dovodi do jezgrenog trika. Umjesto da tražimo preslikavanje pa računamo sličnost u višedimenzionalnom prostoru, možemo računati sličnost pomoću određene jezgrine funkcije koja će implicitno odrediti i preslikavanje u višedimenzionalan prostor kao:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x'}) = \Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{x'}).$$

Time jednadžba (2.9) prelazi u:

$$h(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}_i) + b = 0, \tag{2.9}$$

a jednadžba (2.7) u:

$$max_\alpha \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) \right\}, \tag{2.10}$$

uz uvjete:

$$0 \leq \alpha_i \leq C, \ i = 1, ..., N$$

i

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

Jezgrene funkcije moraju zadovoljavati određena svojstva kao što je danu u [8].

2.2.5. SMO algoritam

Najpoznatiji pristup rješavanju optimizacijskog problema (2.10) jest J. C. Plattov SMO algoritam. SMO rastavlja problem u skup najmanjih mogućih potproblema koji se onda mogu riješiti analitički. Najmanji mogući potproblem je određen s dva Lagrangeova multiplikatora za koje se problem (2.10) svodi na (uz fiksiranje ostalih multiplikatora):

$$max_{\alpha_i,\alpha_j}\left\{\alpha_i+\alpha_j-\frac{1}{2}(\alpha_i y_i\sum_{l=1}^{N}\alpha_l y_l\mathcal{K}(\boldsymbol{x}_i,\boldsymbol{x}_l)+\alpha_j y_j\sum_{l=1}^{N}\alpha_l y_l\mathcal{K}(\boldsymbol{x}_j,\boldsymbol{x}_l))\right\}, \quad (2.11)$$

uz uvjete:

$$0\leq\alpha_i,\alpha_j\leq C$$

i

$$\alpha_i y_i+\alpha_j y_j=k_2,$$

gdje je k_1 konstanta u koju su stavljeni svi ostali fiksni članovi i gdje je $k_2=-\sum_{\substack{l=1\\l\neq i,j}}^{N}\alpha_l y_l$ konstanta. Ovaj problem se može riješiti analitički kao npr. u [13]. Pseudoalgoritam SMO algoritma dan je kao:

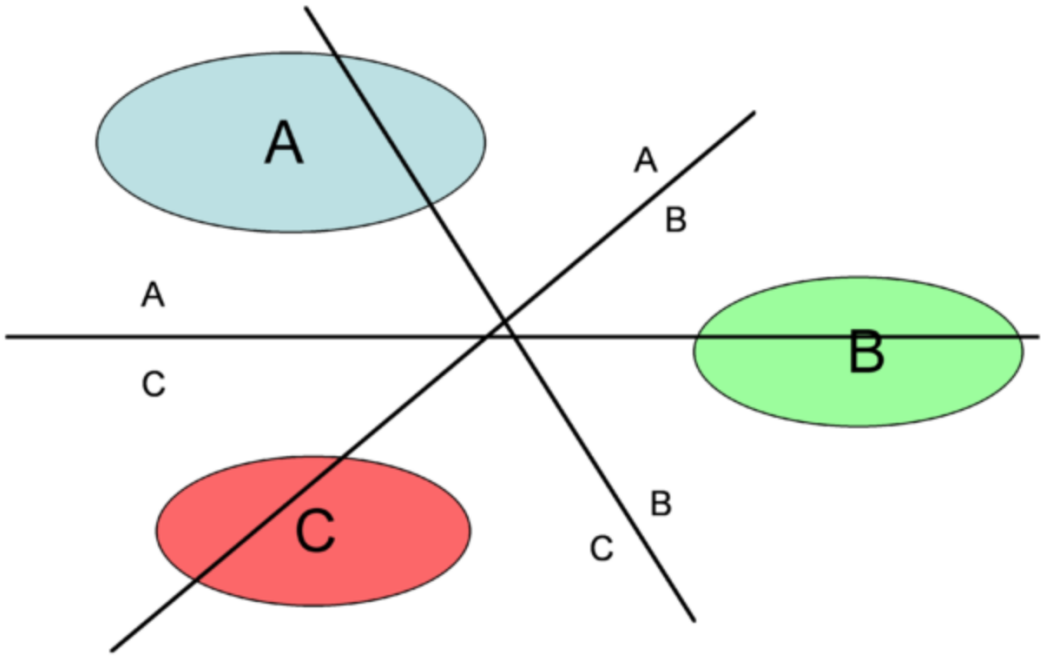
Algoritam 1 pseudokod SMO algoritma

Ponavljaj
izaberi α_i i α_j
optimiziraj(α_i, α_j)
dok nije postignuta konvergencija

Algoritam je završen kada svi multiplikatori zadovoljavaju KKT uvjete do na definiranu preciznost. Biranje multiplikatora se odvija u dvije petlje, vanjskoj i unutarnjoj. Vanjska petlja bira prvi multiplikator, a unutarnja drugi.

U vanjskoj petlji se izabire multiplikator koji narušava KKT uvjete. Petlja ne prolazi uvijek kroz sve primjere, već u prvoj iteraciji prolazi kroz sve, u sljedećoj iteraciji prolazi samo kroz primjere čiji $\alpha\neq 0$ i $\alpha\neq C$ tzv. negranične primjere. Petlja će prolaziti kroz negranične primjere toliko dugo dok u jednom prolazu ima promijenjenih multiplikatora. Kada u jednom prolazu kroz negranične primjere ne dođe ni do jedne promjene, petlja opet prolazi kroz sve primjere i to se ponavlja do konvergencije.

U unutarnjoj petlji se bira multiplikator koji maksimizira apsolutnu vrijednost razlike pogrešaka klasifikacije između primjera koji odgovaraju prvom i drugom multiplikatoru, jer se pokazuje u [13] da takav izbor najviše mijenja funkciju koju želimo



Slika 2.4: Prikaz sheme *jedan-naspram-jedan*, posuđeno iz [11].

optimizirati (2.11). Ako taj izbor multiplikatora ne da pomak, pokuša se izabrati negranični multiplikator počevši od nasumičnog. Ako ni takav izbor ne da pomak, izabire se bilo koji multiplikator, počevši od nasumičnog.

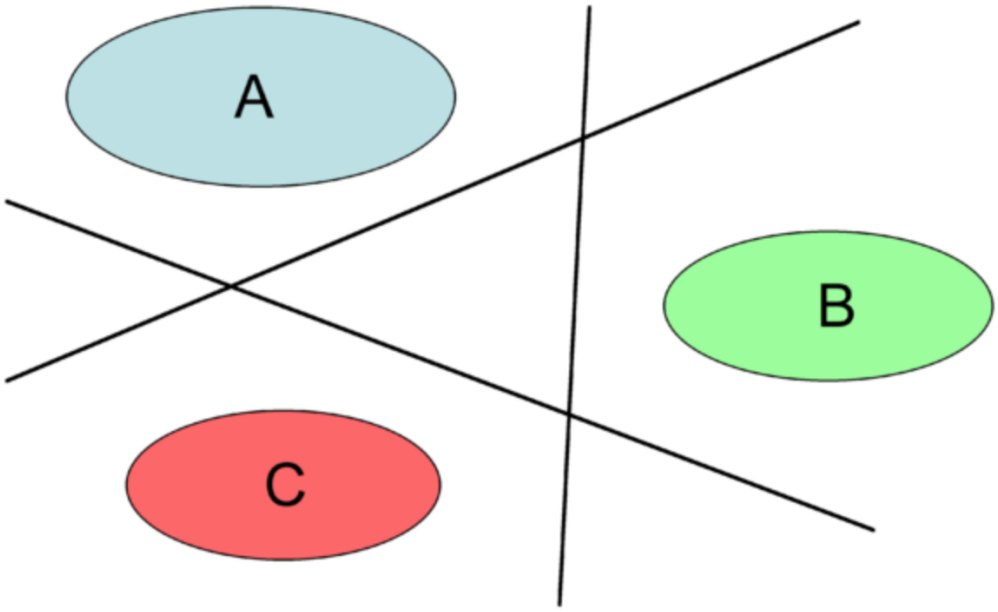
2.2.6. Problem više klasa

Zasad smo opisali kako koristiti SVC kao binarni klasifikator. Ako je broj klasa različiti od dva (neka je broj klasa označen s K), imamo dva moguća rješenja tzv. *jedan-naspram-jedan* (engl. *one-versus-one*) i *jedan-naspram-ostali* (engl. *one-versus-rest*).

Kod sheme *jedan-naspram-jedan* učimo $\binom{K}{2}$ klasifikatora tj. za svaki par klasa učimo po jedan klasifikator koji govori kojoj od dvije klase primjer pripada. Tada radimo tzv. glasovanje. Svaki klasifikator glasuje u koju klasu spada primjer. Primjer tada klasificiramo u klasu s najviše glasova. Primjer *jedan-naspram-jedan* sheme dan je na slici 2.4.

Kod sheme *jedan-naspram-ostali* učimo K klasifikatora tj. za svaku klasu imamo klasifikator koji govori pripada li primjer toj klasi ili pripada nekoj od ostalih. Primjer *jedan-naspram-jedan* sheme dan je na slici 2.5.

Kod obje sheme postoje određeni problemi. Shema *jedan-naspram-jedan* ima veći broj klasifikatora i može se dogoditi da postoji područje koje ne pripada ni jednoj klasi (trokut u sredini slike 2.4 ne odgovara ni jednoj klasi), dok shema *jedan-naspram-*



Slika 2.5: Prikaz sheme *jedan-naspram-ostali*, posuđeno iz [11].

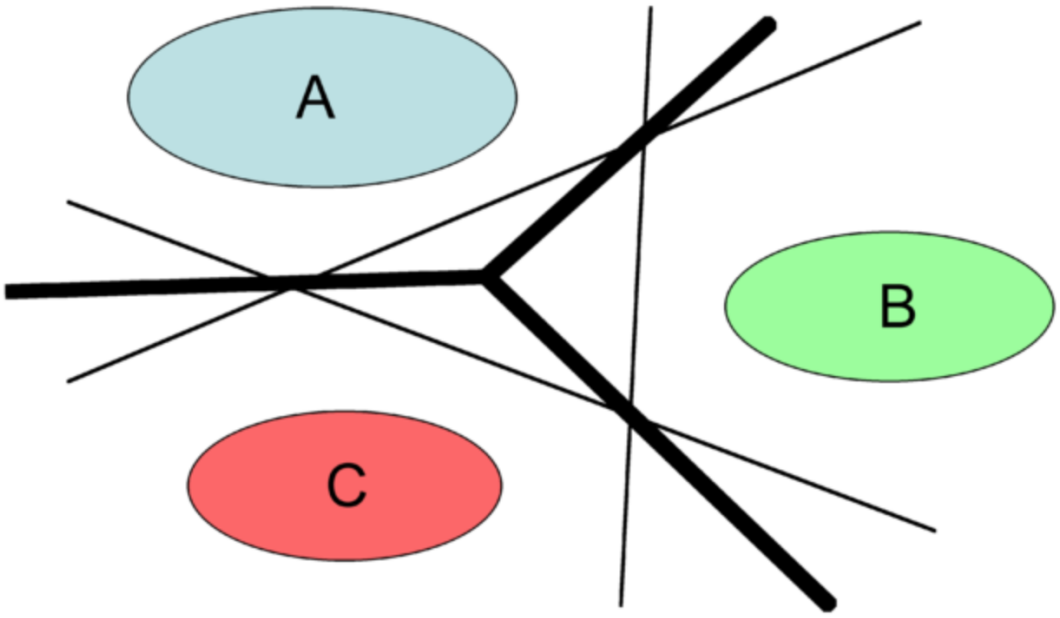
ostali ima manji broj klasifikatora, no može uzrokovati neuravnoteženost klasa te i ovdje može postojati područje koje ne pripada ni jednoj klasi (veliki trokut na slici 2.5). Kod sheme *jedan-naspram-ostali*, koristeći svojstva SVM-a, može se izbjeći područje koje ne pripada ni jednoj klasi. To možemo napraviti tako da se umjesto isključivo predznaka, za određivanje klase koristi kontinuirane vrijednosti naučenih funkcija. To možemo vidjeti na slici 2.6 Na primjer, kod *jedan-naspram-ostali* ukoliko jedna klasa ima značajno manji broj primjera od ostalih, klasifikatoru koji odgovara toj klasi bit će isplativije uvijek reći da primjer pripada ostalima jer će time imati jako veliku točnost.

2.3. Primjena SVC-a

Klasifikacija pomoću algoritma stroja potpornih vektora ima mnoštvo primjena. Samo neke od njih su: klasifikacija slika, kategorizacija teksta, primjene u bioinformatici, medicini, fizici i mnogim drugim područjima.

Kod klasifikacije slika, SVC se odlikuje visokom preciznosti [1]. Može se koristiti za npr. prepoznavanje lica tako da klasificira dijelove slike koji pripadaju licu u jednu grupu (1), a ostale dijelove u drugu (-1). Također, može se koristiti za prepoznavanje pisanog teksta i znamenaka. U medicini imamo primjene poput prepoznavanja zločudnih tumora.

Kod kategorizacije teksta, SVC se može naučiti na skupu za treniranje tako da



Slika 2.6: Prikaz sheme *jedan-naspram-ostali*, posuđeno iz [11].

svrstava tekst u različite klase poput web-stranica, članaka, e-pošte i drugih. Može se koristiti i da npr. razvrstava članke u kategorije poput sporta, politike, crne kronike, oglasa...

U bioinformatici, jedan od važnih problema je engl. *protein remote homology detection*. Upravo SVC postiže najbolje rezultate u rješavanju tog problema. Također, SVC ima i primjenu u problemima klasifikacije u fizici visokih energija tj. fizici elementarnih čestica.

2.4. Opis implementacije

SVC je implementiran te dodan u sustav za evolucijsko računarstvo (ECF) [5]. Unutar ECF-a SVC je implementiran u sklopu klase *Algorithm*. Implementirana je funkcija *advanceGenerations* gdje se u svakoj iteraciji traže dva Lagrangeova multiplikatora koja se onda optimiziraju. Napravljen je i umjetni genotip koji predstavlja poziciju jedinke u populaciji i kao takav koristi se kod izbora Lagrangeovih multiplikatora. Implementirani su i operatori selekcije (*SelectionOperator*) koji biraju prvi i drugi Lagrangeov multiplikator prema Plattovoj heuristici.

Implementirana je i metoda *registerParameters* klase *Algorithm*. Parametri koje prima SVC su tip jezgre kojom će se vršiti optimizacija, parametri korištene jezgre, maksimalni broj iteracija, parametar željene preciznosti, veličina cache-a te put do

datoteke s podacima. Većina parametara ima predodređene vrijednosti pa su jedini parametari koji se moraju implementirati put do datoteke s podacima te ECF-ov paramter *population.size* koji odgovara broju primjera.

Datoteka s podacima treba biti oblikovana tako da je u prvom retku napisan broj značajki, a svaki ostali red predstavlja jedan podatak, tako da su prvo napisane značajke, a onda klasa podatka. Sve vrijednosti u redu su odvojene zarezima kao u sljedećoj datoteci:

podaci.txt

```
10
1000025,5,1,1,1,2,1,3,1,1,2
1002945,5,4,4,5,7,10,3,2,1,2
...
```

Primjer kofiguracijske datoteka ECF-a dan je u sljedećoj datoteci:

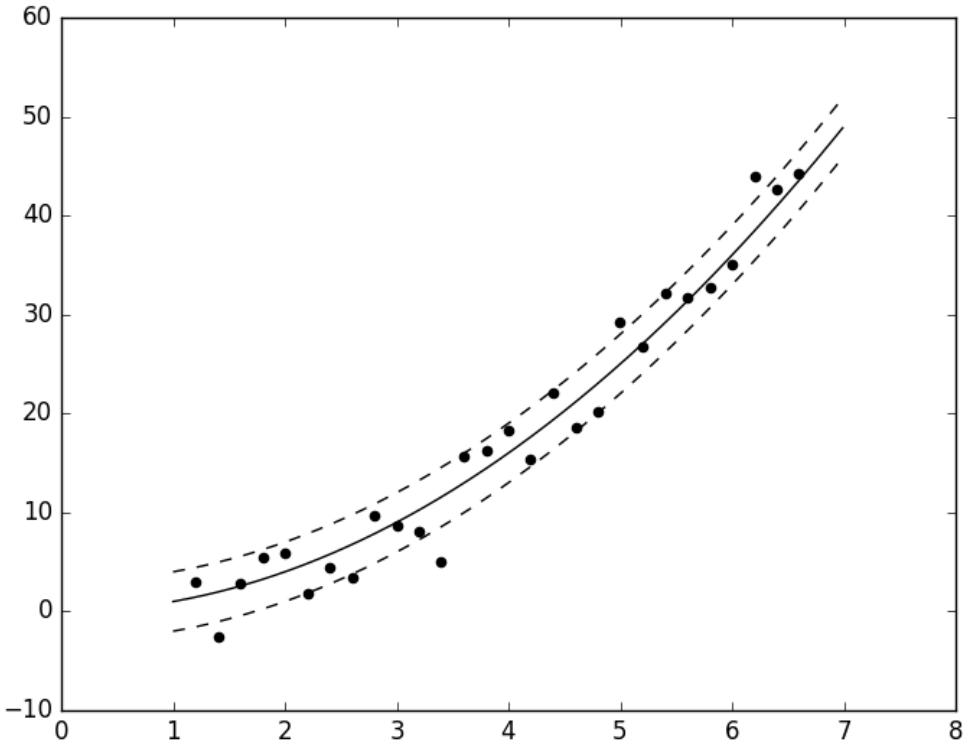
config.txt

```
<ECF>
  <Genotype>
    <Index>
    </Index>
  </Genotype>

  <Registry>
    <Entry key="population.size">100</Entry>
  </Registry>

  <Algorithm>
    <SVC>
      <Entry key="pathToFile">/Users/.../testSet.txt</Entry>
      <Entry key="kernel">linear</Entry>
    </SVC>
  </Algorithm>
</ECF>
```

3. Regresija potpornim vektorima

3.1. Motivacija

Neka je dan skup n - dimenzionalnih točaka i neka je svakoj točki pridružen realni broj koji predstavlja vrijednost neke nepoznate funkcije u toj točki (vrijednost može biti podvrgnuta šumu). Zadatak je pronaći funkciju $f(\mathbf{x})$ koja na neki način najbolje opisuje dani skup podataka. Ovo je primjer regresije koju ćemo u ovom radu rješavati algoritmom potpornih vektora (SVR).



Slika 3.1: Primjer rješavanja regresije SVR-om.

Konkretno uvest ćemo parametar ϵ i zahtijevat ćemo da funkcija $f(\mathbf{x})$ ima najviše ϵ

odstupanje od izmjerenih vrijednosti, a da je istovremeno što ravnija. Dakle, toleriramo pogreške koje su manje od ϵ. Primjer možemo vidjeti na slici (3.1). Točke predstavljaju podatke, puna linija naučenu funkciju, a iscrtkane linije granice unutar kojih su točke koje ne doprinose ukupnoj pogrešci.

3.2. Teorijska razmatranja

Neka je dan skup podataka:

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\},$$

gdje je $\mathbf{x}_i$ n - dimenzionalni vektor realnih brojeva koji predstavlja podatak, a y_i predstavlja vrijednost nepoznate funkcije u točki $\mathbf{x}_i$. Linearna funkcija bez šuma bila bi opisana jednandžbom:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b, \tag{3.1}$$

gdje su $\mathbf{w}$ i b zasad neodređeni parametri.

3.2.1. Formulacija problema

Ciljna funkcija koju minimiziramo ista je kao i u klasifikaciji. To će osigurati da su težine $\mathbf{w}$ što manje, odnosno da je funkcija $f(\mathbf{x})$ što ravnija. Problem optimizacije koji moramo riješiti dan je kao:

$$min_{\mathbf{w},b}\left\{\frac{1}{2}\|\mathbf{w}\|^2\right\}, \tag{3.2}$$

uz ograničenja:

$$\mathbf{w} \cdot \mathbf{x}_i + b - y_i \leq \epsilon, \;\; i = 1, ..., N.$$

i

$$y_i - \mathbf{w} \cdot \mathbf{x}_i - b \leq \epsilon, \;\; i = 1, ..., N,$$

Kao i u slučaju klasifikacije, želimo prijeći na meku marginu dopuštajući pogreške iznad i ispod funkcije ξ_i i ξ_i^*. ξ_i i ξ_i^* mjere koliko je odstupanje funkcije od danog primjera veće od ϵ tj. ako je sa $\bar{\xi_i}$ označeno odstupanje funkcije od danog primjera:

$$\xi_i = \begin{cases} 0, & \bar{\xi_i} \leq \epsilon \\ \bar{\xi_i} - \epsilon, & \bar{\xi_i} > \epsilon. \end{cases}$$

Vrijedi i analogan izraz za ξ_i^*. Primjetimo da je za dani i između ξ_i i ξ_i^* samo jedan od njih može biti različit od 0. Sada imamo optimizacijski problem koji je dan kao:

$$min_{\mathbf{w},b}\left\{\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)\right\}, \tag{3.3}$$

uz ograničenja:

$$\mathbf{w}\cdot\mathbf{x}_i + b - y_i \leq \epsilon + \xi_i, \;\; i = 1, ..., N,$$

$$y_i - \mathbf{w}\cdot\mathbf{x}_i - b \leq \epsilon + \xi_i^*, \;\; i = 1, ..., N,$$

i

$$\xi_i,\; \xi_i^* \geq 0, \;\; i = 1, ..., N,$$

gdje C predstavlja parametar kojim određujemo koliko jako se kažnjavaju pogreške veće od ϵ.

3.2.2. Dualni problem

Kao i kod klasifikacije, prelazimo na dualni problem. Lagrangeova funkcija za problem (3.3) dana je kao:

$$\begin{aligned}\mathcal{L}\left(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\beta}, \boldsymbol{\beta}^*\right) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) - \sum_{i=1}^{N}\alpha_i(\epsilon + \xi_i + y_i - \mathbf{w}\cdot\mathbf{x}_i - b) \\ - \sum_{i=1}^{N}\alpha_i^*(\epsilon + \xi_i^* - y_i + \mathbf{w}\cdot\mathbf{x}_i + b) - \sum_{i=1}^{N}(\beta_i\xi_i + \beta_i^*\xi_i^*),\end{aligned} \tag{3.4}$$

gdje su $\boldsymbol{\alpha}$, $\boldsymbol{\alpha}^*$, $\boldsymbol{\beta}$ i $\boldsymbol{\beta}^*$ Lagrangeovi multiplikatori. Pripadni KKT uvjeti su:

$$\left.\begin{aligned}
\mathbf{w}\cdot\mathbf{x}_i + b - y_i &\leq \epsilon + \xi_i \\
y_i - \mathbf{w}\cdot\mathbf{x}_i - b &\leq \epsilon + \xi_i^* \\
\xi_i &\geq 0 \\
\xi_i^* &\geq 0 \\
\alpha_i &\geq 0 \\
\alpha_i^* &\geq 0 \\
\beta_i &\geq 0 \\
\beta_i^* &\geq 0 \\
\alpha_i(\epsilon + \xi_i + y_i - \mathbf{w}\cdot\mathbf{x}_i - b) &= 0 \\
\alpha_i^*(\epsilon + \xi_i^* - y_i + \mathbf{w}\cdot\mathbf{x}_i + b) &= 0 \\
\beta_i\xi_i &= 0 \\
\beta_i^*\xi_i^* &= 0
\end{aligned}\right\} i = 1, ..., N.$$

Kao i kod klasifikacije, računamo minimum po primarnim parametrima:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = 0 \rightarrow \boldsymbol{w} = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)\boldsymbol{x}_i,$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0,$$

$$\frac{\partial \mathcal{L}}{\partial \xi} = 0 \rightarrow \alpha_i = C - \beta_i,$$

$$\frac{\partial \mathcal{L}}{\partial \xi^*} = 0 \rightarrow \alpha_i^* = C - \beta_i^*.$$

Uvrštavanjem u (3.4) dobivamo dualnu Lagrangeovu funkciju:

$$L(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = \epsilon\sum_{i=1}^{N}(\alpha_i^* + \alpha_i) - \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)y_i + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)\boldsymbol{x}_i \cdot \boldsymbol{x}_j. \tag{3.5}$$

Optimizacijski problem tada prelazi u:

$$max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*}\left\{\epsilon\sum_{i=1}^{N}(\alpha_i^* + \alpha_i) - \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)y_i + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)\boldsymbol{x}_i \cdot \boldsymbol{x}_j\right\}, \tag{3.6}$$

uz ograničenja:

$$\sum_{i=1}^{N}(\alpha_i^* - \alpha_i) = 0$$

i

$$0 \leq \alpha_i,\ \alpha_i^* \leq C,\ \ i = 1, ..., N.$$

Funkcija opisana ovim modelom tada izgleda kao:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) \boldsymbol{x} \cdot \boldsymbol{x}_i + b \tag{3.7}$$

Kao i kod klasifikacije uviđamo da se u jednadžbi (3.7) i (3.6) računa sličnost između primjera te možemo učiniti model nelinearnim zamjenom skalarnog produkta između primjera sa jezgrenom funkcijom $\mathcal{K}$. Time jednadžba (3.6) prelazi u:

$$max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \left\{ \epsilon \sum_{i=1}^{N} (\alpha_i^* + \alpha_i) - \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) y_i + \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) \right\} \tag{3.8}$$

uz ograničenja:

$$\sum_{i=1}^{N} (\alpha_i^* - \alpha_i) = 0$$

i

$$0 \leq \alpha_i,\ \alpha_i^* \leq C,\ \ i = 1, ..., N.$$

Dok jednadžba (3.7) prelazi u:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} (\alpha_i^* - \alpha_i) \mathcal{K}(\boldsymbol{x}, \boldsymbol{x_i}) + b \tag{3.9}$$

Problem (3.8) možemo kompaktnije zapisati prelaskom s varijabli α_i i α_i^* na λ_i i $|\lambda_i|$. Definirajmo $\lambda_i = \alpha_i^* - \alpha_i$. Tada vrijedi:

$$-C \leq \lambda_i \leq C,\ \ i = 1, ..., N.$$

Ako se primijeti da je za dani i samo jedna varijabla od α_i i α_i^* može biti različita od 0, tada je apsolutna vrijednost od λ_i jednaka $|\lambda_i| = \alpha_i^* + \alpha_i$. Prelaskom na nove varijable optimizacijski problem prelazi u:

$$max_{\boldsymbol{\lambda}} \left\{ \epsilon \sum_{i=1}^{N} |\lambda_i| - \sum_{i=1}^{N} \lambda_i y_i + \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) \right\} \tag{3.10}$$

uz ograničenja:

$$\sum_{i=1}^{N} \lambda_i = 0$$

i

$$-C \leq \lambda_i \leq C,\ \ i = 1, ..., N.$$

Dok naučena funkcija prelazi u:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \lambda_i \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}_i) + b. \tag{3.11}$$

3.2.3. SMO algoritam

Problem (3.10) sada možemo rješavati Plattovim SMO algoritmom. Kao i kod klasifikacije rastavljamo problem na najmanje moguće potprobleme koje onda rješavamo analitički. Analogno klasifikaciji, fiksiramo sve varijable $\boldsymbol{\lambda}$ osim dvije određene indeksima i i j. Tada se problem (3.10) svodi na:

$$max_{\lambda_i,\lambda_j} \left\{ \epsilon(|\lambda_i| + |\lambda_j|) - (\lambda_i y_i + \lambda_j y_j) + \frac{1}{2}(\lambda_i \sum_{l=1}^{N} \lambda_l \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x_l}) + \lambda_j \sum_{l=1}^{N} \lambda_l \mathcal{K}(\boldsymbol{x}_j, \boldsymbol{x}_l)) \right\} \tag{3.12}$$

uz ograničenja:

$$\lambda_i + \lambda_j = k_2$$

i

$$-C \leq \lambda_i, \lambda_j \leq C,$$

gdje su k_1 i k_2 konstante kao i kod klasifikacije. Ovaj problem se može riješiti analitički kao na primjer u [12]. Pseudokod algoritma isti je kao i kod klasifikacije u algoritmu (1) te se i selekcija indeksa i i j radi Plattovom heuristikom također kao i kod klasifikacije.

3.3. Primjena SVR-a

Kao i klasifikacija, regresija pomoću algoritma stroja potpornih vektora ima velik broj primjena. Neke od njih su: predviđanje temperature za sljedećih nekoliko dana, predviđanje trajanja puta, predviđanje kretanja financijskih vremenskih sljedova.

U farmaceutskoj industriji, SVR se može koristiti za na primjer procjenjivanje stabilnosti aktivne supstance lijeka kako bi se odredio rok trajanja lijeka. Osiguravajuće tvrtke mogu koristiti regresiju kod određivanja izglednosti da je stvarno došlo do problema kako bi smanjile broj lažnih tužbi. Uz ove, primjena ima još mnogo.

3.4. Opis implementacije

SVR je implementiran te dodan u sustav za evolucijsko računarstvo (ECF) [5]. Unutar ECF-a SVR je implementiran u sklopu klase *Algorithm*. Implementirana je funkcija

advanceGenerations gdje se u svakoj iteraciji traže dva Lagrangeova multiplikatora koja se onda optimiziraju. Implementiran je i operator evaluacije *EvaluateOp* tako da je ocjena uspješnosti u svakom trenutku iznos Lagrangeove funkcije koju i pokušavamo minimizirati. Napravljen je i umjetni genotip koji predstavlja poziciju jedinke u populaciji i kao takav koristi se kod izbora Lagrangeovih multiplikatora. Implementirani su i operatori selekcije (*SelectionOperator*) koji biraju prvi i drugi Lagrangeov multiplikator prema Plattovoj heuristici.

Implementirana je i metoda *registerParameters* klase *Algorithm*. Parametri koje prima SVR su tip jezgre kojom će se vršiti optimizacija, parametri korištene jezgre, maksimalni broj iteracija, parametar željene preciznosti, parametar ϵ, veličina cache-a te put do datoteke s podacima. Većina parametara ima predodređene vrijednosti pa je jedini parametar koji se mora implementirati put do datoteke s podacima te ECF-ov paramter *population.size* koji odgovara broju primjera.

Datoteka s podacima treba biti oblikovana tako da je u prvom retku napisan broj značajki, a svaki ostali red predstavlja jedan podatak, tako da su prvo napisane značajke, a onda vrijednost funkcije. Sve vrijednosti u redu su odvojene zarezima kao u sljedećoj datoteci:

podaci.txt

```
1
1.000000,97.62227
2.000000,97.80724
...
```

Primjer kofiguracijske datoteka ECF-a dan je u sljedećoj datoteci:

config.txt

```
<ECF>
  <Genotype>
    <Index>
    </Index>
  </Genotype>

  <Registry>
    <Entry key="population.size">100</Entry>
  </Registry>
```

```
  <Algorithm>
    <SVR>
      <Entry key="pathToFile">/Users/.../testSet.txt</Entry>
      <Entry key="kernel">linear</Entry>
    </SVR>
  </Algorithm>
</ECF>
```

---

4. Rezultati

U ovom su poglavlju opisani rezultati primjene razvijene implementacije algoritma stroja potpornih vektora na odabrane primjere klasifikacije i regresije.

4.1. Klasifikacija

U ovom odsječku opisat će se rezultati klasifikacije na skupove podataka: Jednostavni dvodimenzionalni skup podataka, Iris skup podataka, Wisconsin breast cancer i Cleveland hearth diesease. Sva mjerenja su napravljena s RBF jezgrom ([8]) te su se optimalni parametri C i γ tražili pretraživanjem po mreži (engl. *grid search*), osim za jednostavni dvodimenzionalni skup podataka koji je optimiziran s linearnom jezgrom.

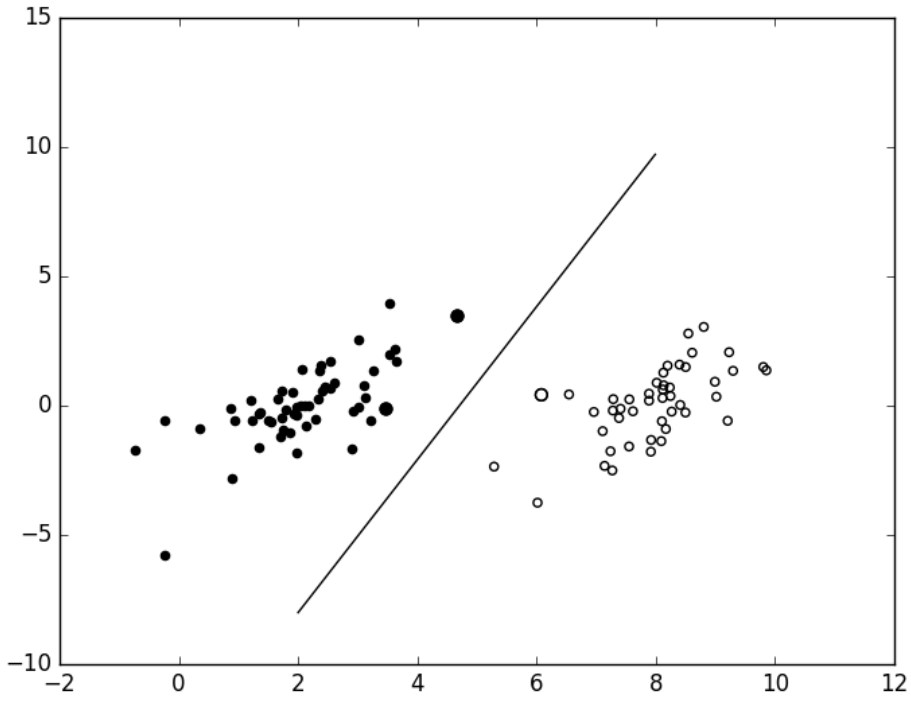
4.1.1. Jednostavni dvodimenzionalni skup podataka

Jednostavni dvodimenzionalni skup podataka sastoji se od dvije klase. Sastoji se od ukupno 100 primjera. Skup je linearno odvojiv pa su rezultati maksimalni. Mjereni su točnost, preciznost, odziv, pogreška te F1 ocjena (pogledati u [6]). Rezultati su dani u tablici 4.1.

Točnost (%)	Preciznost (%)	Odziv (%)	Pogreška (%)	F1 ocjena (%)
100	100	100	0	100

Tablica 4.1: Prikaz rezultata za jednostavan dvodimenzionalan skup podataka.

Na slici 4.1 je dan prikaz podataka. Ispunjeni kružići pripadaju jednoj klasi, dok prazni pripadaju drugoj. Pravac predstavlja naučenu decizijsku granicu, dok tri povećana primjera predstavljaju potporne vektore. Dakle za klasifikaciju bilo kojeg podatka, algoritam radi usporedbu s ta tri potporna vektora te donosi odluku kojoj klasi pripada podataka, što je izuzetno brzo te jedna od glavnih prednosti SVM algoritma.



Slika 4.1: Prikaz rezultata SVM algoritma na jednostavnom dvodimenzionalnom skupu podataka.

4.1.2. Iris

Skup podataka Iris sastoji se od tri klase. Svaka klasa odgovara jednom od tri tipa cvijeta Iris: Setosa, Versicolour i Virginica. Svaki primjer ima četiri značajke, duljini i širinu latice te duljinu i širinu čaške. Skup Iris se sastoji od ukupno 150 primjera. Rezultati su dobiveni k - strukom unakrsnom validacijom (pogledati u [3]) za $k = 10$, dok su mjereni bili točnost, preciznost, odziv, pogreška te F1 ocjena.

Točnost (%)	Preciznost (%)	Odziv (%)	Pogreška (%)	F1 ocjena (%)
98.22	97.50	97.71	1.78	97.49

Tablica 4.2: Prikaz rezultata za skup podataka Iris.

U tablici 4.2 dani su rezultati izvođenja klasifikacije na Iris skupu podataka. Iz rezultata možemo vidjeti da klasifikacija na ovom skupu ima iznimno dobre rezultate. U sljedeća dva pododjeljka opisano je testiranje napravljeno na nešto težim skupovima podataka.

4.1.3. Wisconsin breast cancer

Skup podataka Wisconsin breast cancer sastoji se od 699 primjera raspoređenih u dvije grupe malignih i benignih tumora. Svaki primjer se sastoji od 10 značajki: identifikacijskog broja, debljine kvržice, jednoličnosti stanične veličine, jednoličnosti staničnog oblika, marginalne adhezije, veličine jedne epitelne stanice, gole jezgre, blijedog kromatina, normalne jezgrice, mitoza. Svaka značajka poprima vrijednosti između 1 i 10, osim identifikacijskog broja. Od 699 primjera za njih 16 nisu poznate sve značajke. Dakle, klasifikaciju smo radili na preostalih 683 primjera uz mjerenje istih ocjena kao i u skupu Iris.

Točnost (%)	Preciznost (%)	Odziv (%)	Pogreška (%)	F1 ocjena (%)
96.67	95.91	96.98	3.33	96.39

Tablica 4.3: Prikaz rezultata za skup podataka Wisconsin breast cancer.

U početku nismo mogli dobiti zadovoljavajuću točnost na ovom skupu podataka (točnost je bila 63.48 %), što nas je navelo da pogledamo u podatke. Tada smo primijetili da je značajka identifikacijskog broja nekoliko redova veličine veća od ostalih te smo je odlučili normirati tako da smo od svih identifikacijski brojeva oduzeli najmanji te ih podijelili s razlikom najvećeg i najmanjeg identifikacijskog broja. Nakon normiranja značajke identifikacijskog broja dobivamo rezultate koji su dani u tablici 4.3. Rezultati su dobiveni k - strukom unakrsnom validacijom za $k = 10$. Dok značajka identifikacijskog broja nije bila normirana, dobivali smo manju točnost (oko 80 %), no nakon normiranja, kao što se vidi u tablici 4.3, rezultati su opet iznimno dobri. Ako pogledamo u [2], gdje su dani rezultati klasifikacije na različitim skupovima podataka te za različite algoritme, možemo vidjeti da naša implementacija postiže samo malo lošije rezultate (0.33 %).

4.1.4. Cleveland hearth diesease

Skup podataka Cleveland hearth diesease sastoji se od 303 primjera raspoređenih u dvije grupe koje označavaju prisutnost, odnosno odsutnost srčane bolesti. Svaki primjer se sastoji od 13 značajki: dob, spol, tip boli u prsima, krvni tlak, kolesterol, šećer u krvi, rezultati elektrokardiograma, maksimalan broj otkucaja srca, angina inducirana vježbom, razina ST segmenta elektorkardiograma ispod normalne razine, nagib vrha ST segmenta, rezultati fluoroskopije te stupanj talasemije.

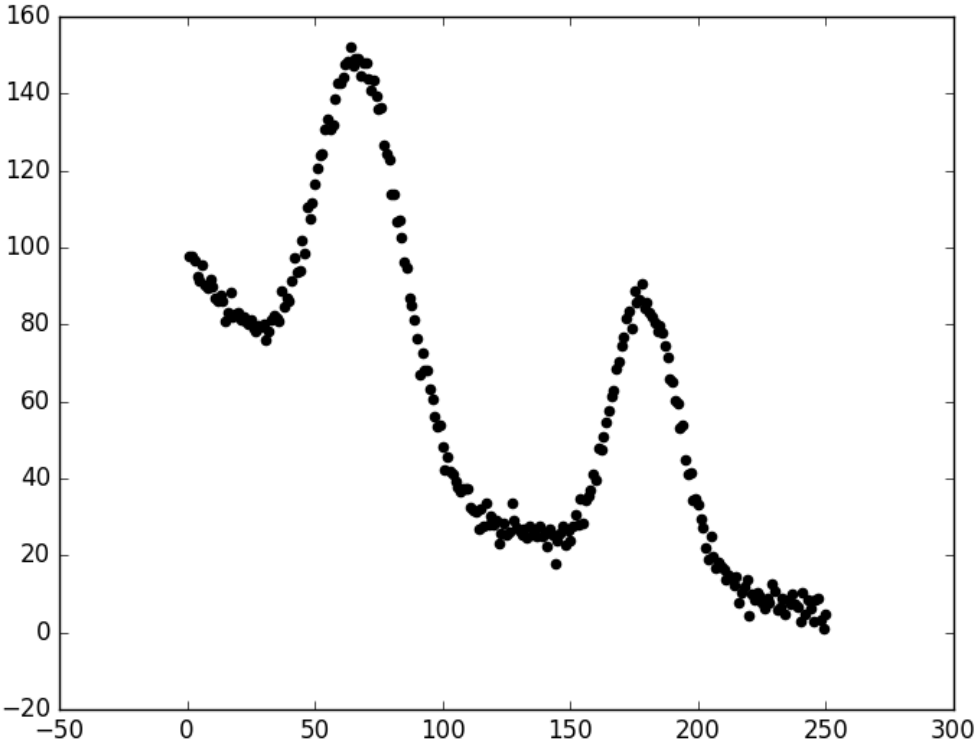
Točnost (%)	Preciznost (%)	Odziv (%)	Pogreška (%)	F1 ocjena (%)
83.00	84.82	81.83	17.00	81.74

Tablica 4.4: Prikaz rezultata za skup podataka Cleveland hearth diesease.

Rezultati su dani u tablici 4.4. Rezultati su dobiveni k - strukom unakrsnom validacijomza $k = 10$. U početku smo dobivali lošije rezultate (točnost je bila 65.67 %), zbog toga što su značajke dobi, krvnog tlaka, kolesterola te maksimalnog otkucaja srca bili otprilike red do dva reda veličine veći od ostalih značajki. Nakon normiranja (napravljenog na isti način kao i u skupu podataka Wisconsin breast cancer) su postignuti rezultati dani u tablici 4.4. Iz rezultata vidimo da je točnost relativno dobra, no ako rezultate naše implementacije SVM-a usporedimo [2], vidimo da su rezultati zapravo zadovoljavajući. Najbolja točnost (uspoređena s nekoliko algoritama, tj. ne samo SVM-om) postignuta na ovom skupu podataka iznosi (85.9 ± 5.5) %. Dakle, rezultati su zadovoljavajući.

4.2. Regresija

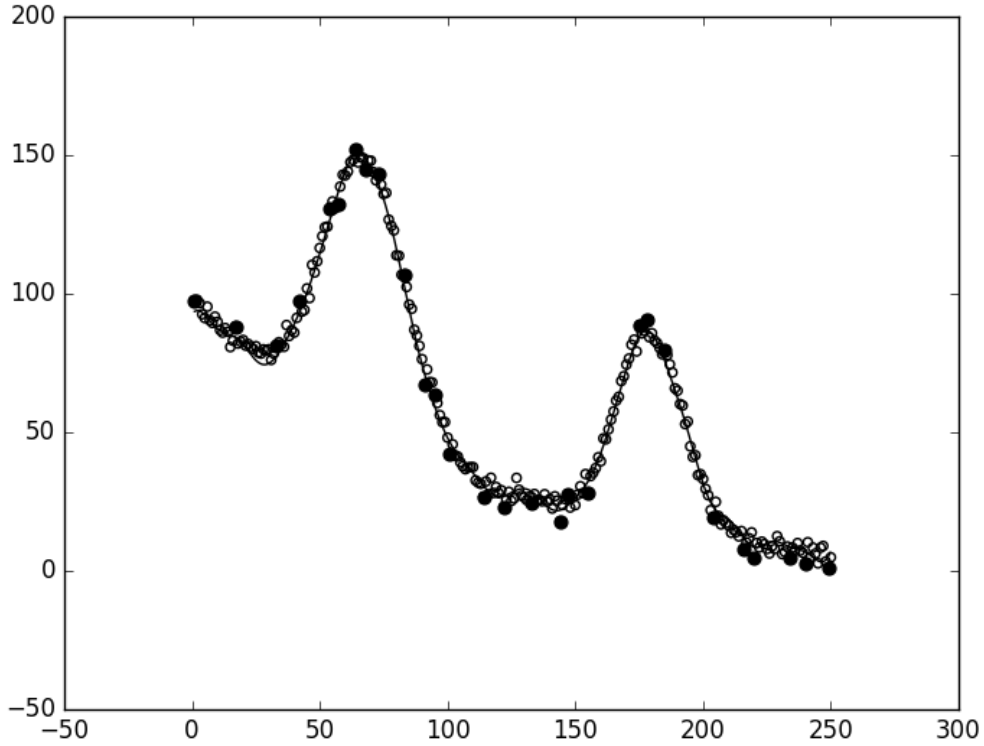
U ovom odsječku opisat će se rezultati regresije na skupove podataka: Jednostavni dvodimenzionalni skup podataka, Jednostavni trodimenzionalni skup podataka te Yacht Hydrodynamics. Na svim skupovima podataka, kao ocjena gledan je korijen iz srednje kvadratne pogreške.



Slika 4.2: Prikaz jednostavnog dvodimenzionalnog skupa podataka.

4.2.1. Jednostavni dvodimenzionalni skup podataka

Skup podataka se sastoji od 250 točaka napravljenih od dvije Gaussove razdiobe. Prikaz podataka dan je na slici 4.2. Naučena funkcija prikazana je na slici 4.3. Na slici 4.3 praznim kružićima su označeni svi podaci, punim kružićima su označeni potporni vektori, te je punom linijom označena naučena funkcija. Na ovom skupu podataka korištena je RBF jezgrena funkcija jer je i sam skup podataka napravljen od dvije Gaussove razdiobe (RBF ima istu ovisnost).

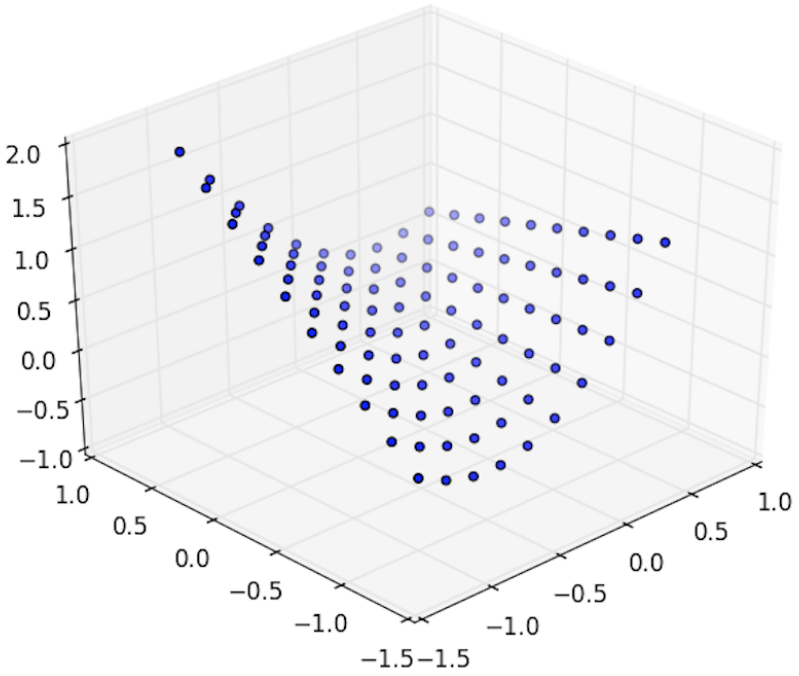


Slika 4.3: Prikaz rezultata SVR algoritma na jednostavnom dvodimenzionalnom skupu podataka.

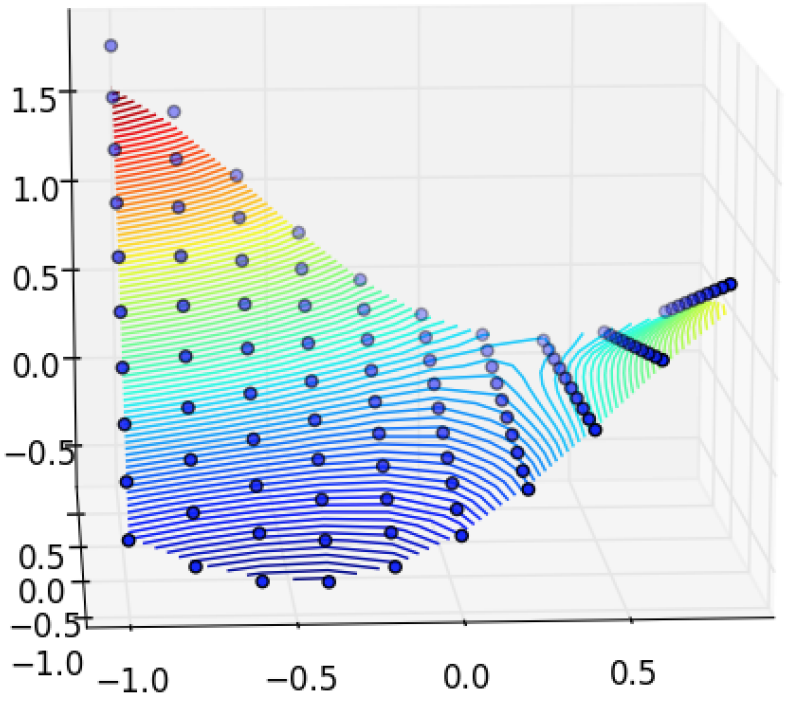
Na ovom skupu podataka korijen iz srednje kvadratne pogreške iznosi 119.2 što je i za očekivati jer su podaci dosta šumoviti, no sa slike 4.3 vidimo da je naučena funkcija relativno dobra.

4.2.2. Jednostavni trodimenzionalni skup podataka

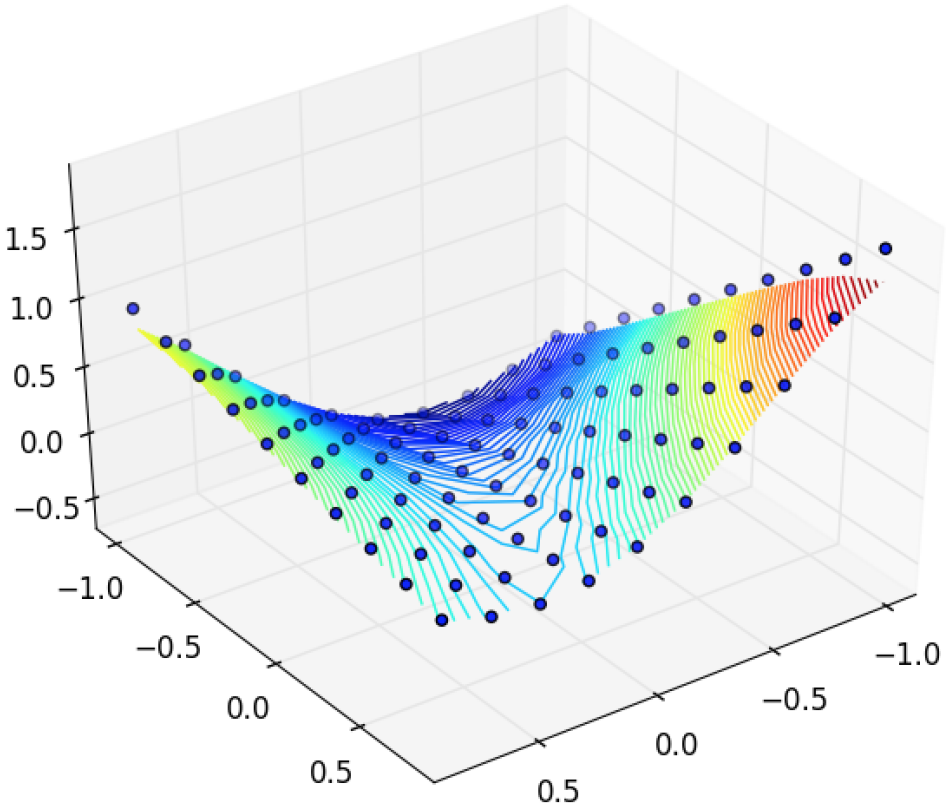
Skup podataka se sastoji od 100 točaka. Prikaz podataka dan je na slici 4.4. Naučena funkcija prikazana je na slikama 4.5 i 4.6. Za učenje je korištena RBF jezgra.



Slika 4.4: Prikaz jednostavnog trodimenzionalnog skupa podataka.



Slika 4.5: Prikaz rezultata SVR algoritma na jednostavnom trodimenzionalnom skupu podataka.



Slika 4.6: Prikaz rezultata SVR algoritma na jednostavnom trodimenzionalnom skupu podataka.

Na ovom skupu podataka korijen iz srednje kvadratne pogreške iznosi 1.06. Rezultati su dobiveni k - strukom unakrsnom validacijom za $k = 10$.

4.2.3. Yacht Hydrodynamics

Skup podataka se sastoji se od 308 6-dimenzionalnih točaka. Cilj mu je predvidjeti rezidualnu otpornost po jediničnoj veličini pomaka na temelju 6 atributa: logitudinalnom položaju središta uzgona, prizmatičkom koeficijentu, omjeru duljine i pomaka, omjera širine prema trupu, omjera duljine prema širini i Freudeova broja.

I ovaj je problem rješavan k - strukom unakrsnom provjerom gdje je $k = 10$ te je rađeno pretraživanje po mreži za parametre C i γ. Na ovom skupu podataka korijen iz srednje kvadratne pogreške iznosi 18.77.

5. Zaključak

U ovom je radu opisan algoritam stroja potpornih vektora primijenjen na klasifikaciju i regresiju. Opisana je teorijska pozadina algoritma kao i mnoštvo mogućih primjena algoritma. Za optimizaciju problema korišten je i opisan u radu algoritam sekvencijalne minimalne optimizacije. Napravljena je implementacija algoritma te je uspoređena s drugim implementacijama. Kod usporedbe dobiveni su dobri rezultati.

LITERATURA

[1] Real-life applications of SVM (support vector machines). URL https://data-flair.training/blogs/applications-of-svm/. [korišteno lipnja 2018].

[2] Datasets used for classification: comparison of results. URL http://www.is.umk.pl/~wduch/projects/projects/datasets.html. [korišteno lipnja 2018].

[3] Cross-validation (statistics) — Wikipedia, the free encyclopedia. URL https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation. [korišteno lipnja 2018].

[4] Duality (optimization) — Wikipedia, the free encyclopedia. URL https://en.wikipedia.org/wiki/Duality_(optimization). [korišteno lipnja 2018].

[5] ECF - Evolutionary Computation Framework. URL http://ecf.zemris.fer.hr. [korišteno lipnja 2018].

[6] F1 score — Wikipedia, the free encyclopedia. URL https://en.wikipedia.org/wiki/F1_score. [korišteno lipnja 2018].

[7] Kernel trick explanation. URL https://datascience.stackexchange.com/questions/17536/kernel-trick-explanation. [korišteno lipnja 2018].

[8] Kernel (statistics) — Wikipedia, the free encyclopedia. URL https://en.wikipedia.org/wiki/Kernel_(statistics). [korišteno lipnja 2018].

[9] Karush–Kuhn–Tucker conditions — Wikipedia, the free encyclopedia. URL https://en.wikipedia.org/wiki/Karush-Kuhn-Tucker_conditions. [korišteno lipnja 2018].

[10] SVM - non separable case/soft margin, where are the support vectors. URL https://stats.stackexchange.com/questions/191928/svm-non-separable-case-soft-margin-where-are-the-support-vectors. [korišteno lipnja 2018].

[11] Ben Aisen. A Comparison of Multiclass SVM Methods, 2006. URL http://courses.media.mit.edu/2006fall/mas622j/Projects/aisen-project/. [korišteno lipnja 2018].

[12] Gary William Flake. Support vector machines for regression problems with sequential minimal optimization. 1999.

[13] John C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

Implementacija i primjena algoritma stroja potpornih vektora u klasifikaciji i regresiji

Sažetak

Opisana je teorija iza algoritma stroja potpornih vektora, kako za klasifikaciju, tako i za regresiju. Opisan je algoritam sekvencijalne minimalne organizacije. Napravljena je implementacija algoritma i za klasifikaciju i za regresiju te su oba algoritma primijenjena na nekoliko skupova podataka. Napravljena je usporedba naše implementacije s drugim implementacijama te su rezultati zadovoljavajući.

Ključne riječi: Stroj potpornih vektora, klasifikacija, regresija, sekvencijalna minimalna optimizacija

Implementation and application of support vector machine algorithm in classification and regression

Abstract

In this paper, support vector machine (SVM) algorithm for classification and regression is described. Sequential minimal optimization (SMO) algorithm was also described and used for optimization. The implementation of SVM algorithm for classification and regression has been made. The implementation was tested and compared to other implementations and the results are satisfying.

Keywords: Support vector machine, SVM, SVC, SVR, classification, regression, Sequential minimal optimization, SMO