Univerzitet u Beogradu

Matematički fakultet

Master rad

OPTIMIZACIJA METAPARAMETARA MAŠINE PODRŽAVAJUĆIH VEKTORA

Student:
Eva Tuba
1086/2015

Mentor:
dr Zorica Stanimirović

Beograd, mart 2018.

Mentor: **prof. dr Zorica Stanimirović**
Matematički fakultet, Univerzitet u Beogradu

Članovi komisije: **doc. dr Mladen Nikolić**
Matematički fakultet, Univerzitet u Beogradu

dr Stefan Mišković
Matematički fakultet, Univerzitet u Beogradu

Datum odbrane: _____

Optimizacija metaparametara mašine podržavajućih vektora

Sažetak – Klasifikacija je deo brojnih aplikacija i predstavlja važan problem koji je aktuelna istraživačka tema. Mašina podržavajućih vektora je veoma moćan klasifikator koji je u širokoj upotrebi. Tačnost klasifikacije pomoću mašine podržavajućih vektora u velikoj meri zavisi od metaparametara učenja. Za određivanje optimalnih vrednosti metaparametara koriste se različte metode optimizacije, između ostalih, metode zasnovane na inteligenciji rojeva, koje su se pokazale efikasnim pri rešavanju ovog problema. U ovom radu, za podešavanje metaparametara mašine podržavajućih vektora korišćen je nedavno predložen algoritam naseljavanja slonova, koji je takođe zasnovan na inteligenciji rojeva. Elementi predloženog algoritma su prilagođeni razmatranom problemu i algoritam je testiran na standardnim test primerima iz literature. Rezultati dobijeni algoritmom naseljavanja slonova su upoređeni sa rezultatima postojećih metoda iz literature za rešavanje istog problema: genetskim algoritmom i mrežnom pretragom. Analizom i poređenjem rezultata zaključuje se da je predložen algoritam naseljavanja slonova superioran u odnosu na pomenute metode u pogledu kvaliteta dobijenih rešenja.

Ključne reči: mašine podržavajućih vektora, optimizacija metaparametara, inteligencija rojeva, algoritam naseljavanja slonova.

Support vector machine metaparameters optimization

Abstract – Classification is part of various applications and represents active research topic. Support vector machine is one of the widely used and powerful classifiers. The accuracy of support vector machine highly depends on the values of its learning metaparameters. There are numerous optimization methods that can be used for finding the optimal metaparameter values of support vector machine. Among the existing methods, swarm intelligence algorithms showed to be efficient for this problem. In this study, a recently proposed swarm intelligence algorithm, known as elephant herding optimization algorithm, is used for support vector machine metaparameter tuning. The elements of the applied algorithm are adapted to the considered problem and the algorithm was tested on standard datasets from the literature. Obtained results were compared with the results of existing approaches for the same problem from literature, genetic algorithm and grid search. By analyzing and comparing the obtained results, it can be concluded that the proposed elephant herding algorithm is superior over genetic algorithm and grid search in the sense of solution quality.

Keywords: support vector machine, metaparameter optimization, swarm intelligence, elephant herding algorithm.

Sadržaj

1 Uvod

Mašinsko učenja je oblast računarskih nauka koja je pronašla primenu u brojnim oblastima kao što su istraživanje podataka, bioinformatika, dijagnostici, ekonomiji i drugim. Danas postoje brojne metode mašinskog učenja i najgrublje se mogu podeliti na metode nadgledanog i nenadgledanog učenja. Nenadgledano učenje nema nikakvo predznanje o mogućim izlazima i zadatak algoritma ovog tipa mašinskog učenja je da pronađe veze među ulaznim podacima i prema tome ih grupiše. Nadgledano učenje podrazumeva da je za instance trening skupa poznat izlaz i neki od primera ovog tipa učenja su klasifikacija i regresija. Jedana od metoda nadgledanog mašinskog učenja je klasifikacija. Zadatak klasifikacije je pronaći obrazac u ulaznim podacima (instancama) i na osnovu tog obrasca odrediti klasu pripadnosti svakog novog podatka. Aplikacije klasifikacije su brojne i uključuju različite oblasti, kao što su: medicina [57], [35], procesiranje digitalnih slika za prepoznavanje lica i karaktera [18], [36], obrada signala [14], ekonomija [39] i mnoge druge.

Svaka instanca koja se koristi u procesu mašinskog učenja mora biti predstavljena istim skupom atributa, koji mogu biti: kontinualni, kategorički, redni, binarni, itd.

U mašinskom učenju, klasifikacija se koristi za rešavanje problema automatske indentifikacije klase nove instance na osnovu informacija prikuljenih od instanci iz trening skupa. Većina tehnika nadgledanog učenja se zasniva na veštačkoj inteligenciji i statistici. Neki od najpoznatijih algoritama mašinskog učenja za klasifikaciju su: k najbližih suseda, naivni Bajesov klasifikator, logistička regresija, veštačke neuronske mreže, stabla odlučivanja, mašine podržavajućih vektora i mnogi drugi.

Mašina podržavajućih vektora predstavlja efikasan binarni klasifikator koji nalazi primenu u različitim oblastima. U literaturi se mogu naći brojni primeri primene ovog algoritma mašinskog učenja: optičko prepoznavanje karaktera (engl. optical character recognition, OCR) [36], klasifikacija metastaza i nekroza na mozgu [31], prepoznavanja lica (engl. face recognition, FR) [18] i druge.

U cilju uspešne primene, neophodno je prilagoditi mašine podržavajućih vektora svakom konkretnom problemu koji se razmatra, pre svega, podesiti odgovarajuće vrednosti metaparametara ovog klasifikatora. Dva metaparametra su od posebnog značaja za dobre performanse mašine podržavajućih vektora: metaparametar meke margine (engl. soft margin metaparameter – C) i metaparametar funkcije jezgra (engl. kernel function metaparameter). Najčešće korišćena funkcija jezgra je Gausova radijalna bazna funkcija (engl. Gaussian radial basis function), koja uključuje metaparametar γ. Pronalaženje adekvatnih vrednosti za par metaparametara (C, γ) je ključno za uspešnu primenu mašine podržavajućih vektora na konkretan problem.

Određivanje vrednosti metaparametara (C, γ) mašina podržavajućih vektora se može formulisati kao problem optimizacije: potrebno je podesiti odgovarajuće vrednosti metaparametara ovog klasifikatora iz unapred zadatih intervala tako da greška klasifikacije bude minimalna. Prilikom rešavanja ovog problema optimizacije javljaju se dve poteškoće. Prvo, intervali dopustivih vrednosti za svaki od dva razmatrana metaparametra su najčešće veliki. Drugo, grafik funkcije cilja je obično površ sa mnogobrojnim lokalnim ekstremima. Iz navedenih razloga, tradicionalne metode, kao što je, na primer, mrežna pretraga (engl. grid search), su računski

veoma zahtevne i ne daju dobre rezultate u razumnom vremenu izvršavanja.

Heurističke metode su tehnike optimizacije koje u kratkom vremenu izvršavanja mogu dati visokokvalitetna rešenja (neretko i optimalna), pod uslovom da su adekvatno dizajnirane i prilagođene problemu koji se razmatra. Nedostatak heurističkih metoda u odnosu na egzaktne metode optimizacije je nemogućnost potvrde optimalnosti dobijenog rešenja, dok je njihova prednost brzina izvršavanja. Heurističke metode efikasno vraćaju rešenja visokog kvaliteta u slučajevima kada egzaktne metode rade neprihvatljivo dugo ili kada ne mogu dati čak ni dopustivo rešenje. U ovom radu, predložen je algoritam naseljavanja slonova (engl. elephant herding algorithm, EHO), koji spada u grupu heurističkih metoda zasnovanih na inteligenciji roja (engl. swarm intelligence), u cilju određivanja adekvatnih vrednosti metaparametara (C, γ) mašina podržavajućih vektora. Algoritam je testiran na standardnim test primerima za klasifikaciju i dobijeni rezultati su analizirani i uporedeđeni sa postojećim rezultatima iz literature.

Ostatak rada je organizovan na sledeći način. U drugom poglavlju je definisan problem optimizacije koji se razmatra. Dat je opis metoda za njihovo rešavanje sa posebnim naglaskom na opis metaheuristika inspirisanih prirodom i zasnovanih na populaciji jedinki. Dat je kratak pregled ovih metaheurisičkih metoda koje pripadaju dvema glavnim klasama: evolutivni algoritmi i algoritmi inteligencije rojeva.

U trećem poglavlju su opisane i precizno definisane mašine podržavajućih vektora kao jedan od najuspešnijih klasifikatora koji je relativno skoro predložen u literaturi. Izložen je i detaljno opisan matematički model mašina podržavajućih vektora, počevši od osnovnog modela do naprednijih varijanti koje uključuju meku marginu i linearizaciju kernel funkcijom.

Četvrto poglavlje sadrži opis problema optimizacije metaparametara mašine podržavajućih vektora. Dat je pregled postojećih metoda za rešavanje opisanog problema.

U petom poglavlju, detaljno je opisan algoritam naseljavanja slonova koji je korišćen u ovom radu za rešavanje problema optimizacije metaparametara mašine podržavajućih vektora.

U šestom poglavlju je opisana primena EHO algoritma na problem podešavanja metaparametara mašine podržavajućih vektora. Elementi predložene implementacije EHO algoritma su prilagođeni razmatranom problemu. Algoritam je implementiran je u Matlab-u a za mašinu podržavajućih vektora je korišćen paket LibSVM.

U sedmom poglavlju, predloženi EHO algoritam za optimizaciju metaparametara mašine podržavajućih vektora je testiran na standarnim primerima iz UCI biblioteke i LibSVM tool baze. Analizom dobijenih rezultata je potvrđeno da je adekvatan izbor vrednosti metaparametara mašine podržavajućih vektora veoma značajan za kvalitet klasifikacije. Odabir vrednosti metaparametra predloženom EHO heuristikom vodi ka kvalitetnijim rešenjima u odnosu na rešenja dobijena postojećim metodama optimizacije iz literature zasnovanim na evolutivnom principu i mrežom pretrage.

U zaključku je dat osvrt na naučni doprinos rada i kvalitet rezultata dobijenih predloženom EHO heuristikom. Ukazano je na pravce daljeg rada u cilju unapređivanja EHO heuristike i njene primene na slične probleme optimizacije.

2 Metaheuristike inspirisane prirodom zasnovane na populaciji jedinki

Optimizacija predstavlja oblast matematike i računarstva koja ima veliki praktični značaj, jer se brojni problemi iz realnog života (iz industrije, menadžmenta, ekonomije, medicine, itd.) mogu formulisati kao problemi optimizacije. Generalno, problem optimizacije se može definisati kao uređeni par (S, f), gde S predstavlja skup dopustivih rešenja (koji se još naziva i prostor pretrage), a $f : S \longrightarrow \mathbb{R}$ je funkcija cilja koja dodeljuje svakom rešenju iz prostora pretrage realnu vrednost koja označava njegov kvalitet. Cilj problema optimizacije je pronaći rešenje $s* \in S$ koje predstavlja globalni optimum (minimum ili maksimum) [48], odnosno rešenje koje ima bolju vrednost funkcije cilja od svih drugih rešenja prostora pretrage. U slučaju problema minimizacije, neophodno je naći rešenje $s^* \in S$ koje zadovoljava [48]:

$$(\forall s \in S)\ f(s^*) \leq f(s). \tag{1}$$

S druge strane, problemi tipa maksimizacije za cilj imaju pronalaženje rešenja $s^* \in S$ koje zadovoljava:

$$(\forall s \in S)\ f(s^*) \geq f(s). \tag{2}$$

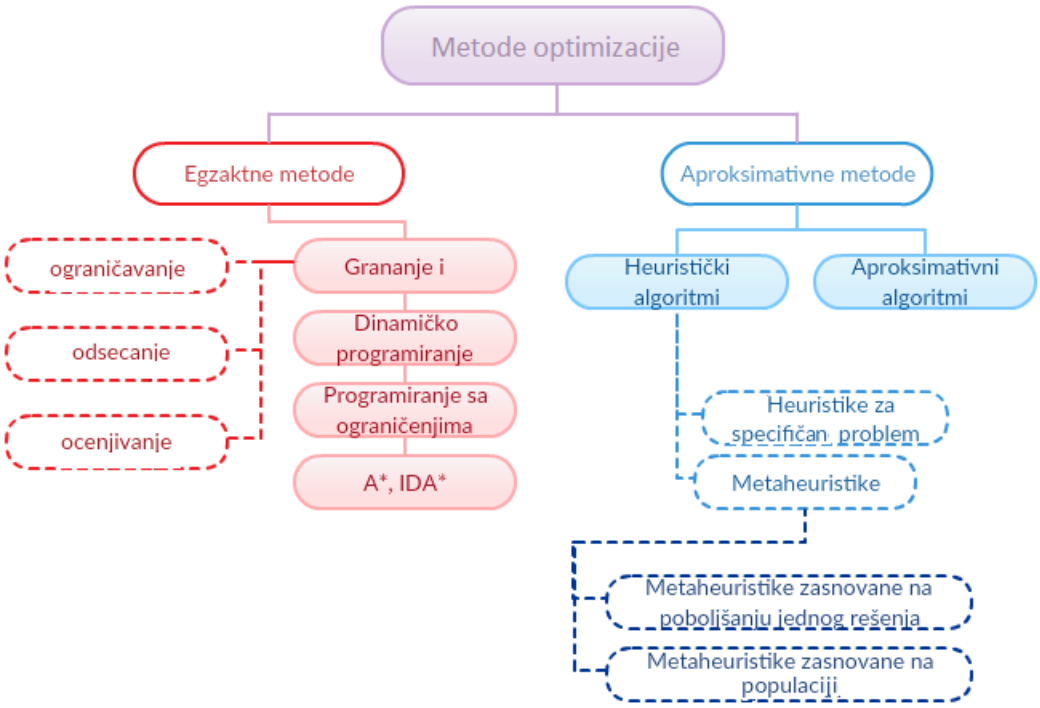
Mnogi problemi optimizacije imaju više globalnih optimuma, pa se definicija može preformulisati da je potrebno naći sve globalne minimume ili maksimume.

Do sada je u literaturi predložen veliki broj raznih metoda optimizacije, koje se najšire mogu podeliti u dve klase: egzaktne i aproksimativne meode (Slika 1) [48]. Egzaktne metode optimizacije su metode koje mogu da reše problem optimizacije u polinomijalnom vremenu. NP teški problemi se ne mogu rešiti u polinomijalnom vremenu (osim ako je P=NP, što do sada nije dokazano), pa se za njihovo rešavanje koriste aproksimativne metode koje ne garantuje pronalaženje globalnog optimuma, ali mogu da generišu veoma kvalitetno rešenje u razumnom vremenu za praktične upotrebe.

Među aproksimativnim metodama, najviše se koriste heuristički algoritmi, imajući u vidu da je pronalaženje optimalnog rešenja za većinu NP-teških problema veoma teško ili nemoguće u prihvatljivom vremenskom roku. Sa druge strane, u praksi je često dovoljno efikasno odrediti rešenje zadovoljavajućeg kvaliteta nekom heurističkom metodom.

Heuristike se mogu podeliti u dve grupe: heuristike koje su dizajnirane za specifičan problem i metaheuristike (Slika 1). Metaheuristike se mogu definisati kao viši nivo opštih metodologija koje se mogu koristiti kao strategije koje usmeravaju pravljenje heuristike za rešavanje konkretnog problema optimizacije. Izraz *metaheuristika* je prvo uveo F. Glover u jednom od svojih radova [16]. Metaheuristike se danas široko koriste za rešavanje raznih problema optimizacije, posebno u slučaju instanci problema velikih dimenzija koje ne mogu biti rešene egzaktnim metodama.

Najgrublja podela metaheuristika bi bila na metaheuristike zasnovane na populaciji (engl. population-based) i metaheuristike zasnovane na poboljšanju jednog rešenja (engl. single-based metaheuristics) [48]. U okviru ovih glavnih klasa metaheuristika postoje i grupe metaheurističkih metoda koje imaju neku zajedničku



Slika 1: *Klasifikacija metoda optimizacije*

karakteristiku. Na primer, postoje mataheuristike inspirisane prirodnim procesima ili pojavama, metaheuristike koje koriste memoriju i one bez memorije, determinističke, stohastičke, iterativne, pohlepne metaheuristike, itd.

Zajednička karakteristika svih metaheuristika su dva kontradiktorna kriterijuma pretraživanja koja su u izvesnoj meri uključena u svaku od njih: eksploracija prostora pretrage i ekplotacija najboljeg nađenog rešenja. Obećavajući regioni prostora pretrage su definisani ”dobrim” rešenjima koja su pronađena tokom rada algoritma. Ti regioni se istražuju detaljnije (eksploatacija) u cilju pronalaženja još boljeg rešenja, ako je moguće. Eksploracija je potrebna kako bi se osigurala pretraga celog prostora pretraga i time izbeglo zaglavljivanje u lokalnom optimumu. Eksploracija se obično obezbeđuje implementacijom neke metode zasnovane na principu nasumične pretrage (engl. random search), dok se ekspotacija implementira nekim algoritmom koji u osnovi ima lokalno pretraživanje [48]. Nasumična pretraga podrazumeva da se u svakoj iteraciji na slučajan način generiše neko rešenje iz prostora pretrage. Kod nasumične pretrage se generalno ne koriste prethodna ”znanja” iz procesa pretraživanja. Lokalno pretraživanje u svakoj iteraciji generiše rešenja u susedstvu ”dobrih” rešenja u cilju poboljšanja vrednosti funkcije cilja. Susedna rešenja se u zavisnosti od metaheuristike i problema mogu definisati na različite načine. Prilikom lokalnog pretaživanja poželjno je koristiti prethodno stečena znanja kako bi se izbegla konvergencija ka lokalnom optimumu.

Sve metaheuristike zasnovane na populaciji mogu se posmatrati kao iterativno popravljanje populacije rešenja. Prvi korak je inicijalizacija populacije nakon čega se generiše nova populacija. Nova populacija se integriše u trenutnu populaciju korišćenjem nekog procesa selekcije. Pretraga se završava kada se zadovolji neki uslov zaustavljanja, kao što je maksimalan broj iteracija, maksimalan broj evaluacija

funkcije cilja, itd. Posebnu grupu metaheuristika zasnovanih na populaciji su metaheuristike koje koriste principe evolucije (engl. evolutionary algorithms) i inteligenciju rojeva (engl. swarm intelligence algorithms).

Evolutivni algoritmi su inspirisani biološkim procesom evolucije. Neke od standardnih mehanizama koji se koriste u evolutivnim algoritmima su reprodukcija, odnosno mutacija i kombinovanje, i selekcija. Najpoznatija evolutivna metaheuristika je genetski algoritam [55]. Genetski algoritam je iterativna metoda optimizacije koja simulira proces evolucije. Kroz iteracije opstaju bolja rešenja koja se modifikuju operacijama mutacije i kombinovanja kako bi se pronašla još bolja rešenja. Genetski algoritam počinje generianjem populacije (rešenja) čija veličina zavisi od problema koji se rešava. Najčešće se početna populacija generiše na slučajan način. Nakon evaluacije generisanih rešenja, na osnovu kvaliteta istih dodeljuju se verovatnoće kojim će svako od njih biti izabran za roditelja nove populacije. Bolja rešenja se biraju sa većom verovatnoćom i obrnuto. Nova populacija se dobija primenom operatora mutacije i kombinovanja na odabrana rešenja. Operator mutacije menja deo jednog rešenja, dok operator kombinovanja pravi novo rešenje sklapajući delove od dva različita rešenja. Pored genetskog algoritma, neki od značajnijih predstavnika evolutivnih algoritama su genetsko programiranje [29], evolutivno programiranje [3], diferencijalna evolucija [47] i drugi.

Algoritmi inteligencije rojeva simuliraju kolektivno ponašanje više agenata koji prate jednostavna pravila i međusobno komuniciraju. Iako svaki agent pojedinačno se ne može smatrati inteligentnim, ceo roj može pokazati vrsnu kolektivnu inteligenciju. Insiracija za ove metaheuristike je pronađena u kolektivnom ponašanju insekata kao što su mravi, termiti, pčele, kao i u ponašanju različitih životinjskih vrsta poput jata ptica ili riba. Danas postoji na desetine algoritama inteligencije rojeva, a najstariji među njima su optimizacija rojevima čestica (engl. particle swarm optimization, PSO) [25] i optimizacija mravljim kolonijama (engl. ant colony optimization, ACO) [12].

Algoritam optimizacije rojevima čestica je jedan od najjednostavnijih populaciono zasnovanih metaheuristika inspirisanih prirodom. Ovaj algoritam predložili su 1995. godine Eberhart i Kenedi. PSO pronalazi rešenje na osnovu populacije kandidata rešenja, koja se nazivaju čestice. Ove čestice se pomeraju po prostoru pretrage jednostavnim formulama. Pomeranje svake čestice je usmereno sopstvenom najboljom pozicijom i najboljom pozicijom pronađene od strane cele populacije.

PSO algoritam se implementira na sledeći način. Pozicija svake čestice i u roju određena je vektorom x_i, dok je pomeraj određen brzinom (engl. velocity) v_i. Za svaku česticu pamti se njena najbolja pozicija p_i, dok se posebno pamti najbolja pozicija g među svim česticama roja. Nakon inicijalizacije pozicije i vektora brzina čestica nova pozicija čestice u svakoj iteraciji se računa po formuli:

$$x_{i,new} = x_{i,old} + v_i, \tag{3}$$

gde je v_i vektor brzine. Novi vektor brzine u svakoj iteraciji se određuje na sledeći način:

$$v_{i,new} = \omega v_{i,old} + \phi_p r_p(p_i - x_i) + \phi_g r_g(g - x_i), \tag{4}$$

gde su r_p i r_g slučajni brojevi izabrani uniformnom raspodelom iz $[0, 1]$, a ω, ϕ_p i ϕ_g su parametri algoritma koji se podešavaju u zavisnosti od problema koji se rešava primenom PSO. Ovi parametri kontrolišu, redom, uticaj prethodnog vektora brzine, uticaj najboljeg rešenja čestice i uticaj najboljeg rešenja celog roja.

Optimizacija mravljim kolonijama je inspirisana ponašanjem kolonija mrava prilikom pronalaženja najkraćeg puta od mravinjaka do izvora hrane ili neke druge tačke od interesa. ACO algoritam imitira jednostavan komunikacioni mehanizam koji omogućava mravima da pronađu najkraći put između dve tačke. Prolazeći određenim putem, mravi ostavljaju hemijski trag (feromone) na zemlji. Što više mrava prođe nekim putem, trag feromona na njemu je jači. Feromoni tokom vremena slabe, a mravi će birati put u skladu sa jačinom feromona. Na ovaj način, mravi kroz iterativan proces pronalaze najkraći put između dve tačke.

ACO algoritam ima dva osnovna koraka, konstrukciju rešenja i ažuriranje traga feromona. Ažuriranje feromona podrazumeva proces pojačavanja feromona, osvežavanje, i proces isparavanja feromona (evaporaciju). Pomoćni korak ACO algoritma je heuristika koja pomaže mravima pri odlučivanju.

Svaki mrav gradi rešenje koristeći trag feromona i informacije dobijene iz heuristike. Trag se menja dinamički tokom potrage za rešenjem i pamti karakteristike dobro izgrađenog rešenja, dok informacija iz heuristike pomaže tako što nagoveštava kako da se odlučuje prilikom izgradnje rešenja. Izbor heuristike je veoma bitan i mora se prilagoditi karakteristikama konkretnog problema koji se rešava. U svakoj iteraciji se trag feromona ažurira kroz faze evaporacije i faze osvežavanja. U fazi evaporacije, feromon opada po formuli:

$$\tau_{ij} = (1 - \rho)\tau_{ij}, \tag{5}$$

gde je τ_{ij} matrica feromona, a ρ konstantna stopa evaporacije i obično se bira na slučajan način iz ontervala $(0, 1]$.

Faza osvežavanja feromona može biti implementirana na više načina, što zavisi od karakteristika konkretnog problema koji se razmatra. Trag feromona se osvežava od strane svakog mrava ili dela populacije, tokom ili nakon konstrukcije kompletnog rešenja. Generalno, trag feromona se osvežava tako što se na elemente matrice feromona dodaju izvesne nenegativne vrednosti, čija veličina zavisi od kvaliteta komponente konstrisanog parcijalnog ili celog rešenja.

Još jedan algoritam inteligencije rojeva je optimizacija kolonijom pčela (engl. bee colony optimization, BCO). Ovaj algoritam su predložili Lučić i Teodorović 2001. godine [50], a inspirisana je ponašanjem pčela u potrazi za nektarom. Pčele izviđači kreću u potragu za hranom, a po povratku u košnicu svrstavaju se u jednu od tri grupe: *regruteri* obaveštavanjem drugih pčela o lokaciji, količini i kvalitetu pronađenog izvora, ostaju *lojalne* svom pronađenom izvoru i vraćaju se da nastave sa skupljanjem hrane ili odustaju od pronađenog izvora i postaju *neopredeljene*. Odluka po povratku u košnicu direktno zavisi od kvaliteta i količine pronađenog nektara.

U svakoj iteraciji BCO algoritma, svaka pčela (agent) gradi rešenje koisteći spostveno iskustvo i iskustvo ostalih pčela. Na osnovu razmene informacija o kvalitetu rešenja, u svakoj iteraciji pčela izvrši jednu od dve akcije: napušta svoje trenutno rešenje, postaje neopredeljena i preuzima rešenje druge pčele ili postaje (ili ostaje) regruter, nastavlja da modifikuje svoje trenutno rešenje i privlači (regrutuje)

neopredeljene pčele. Postoje dve osnovne varijante BCO algoritma, konstruktivna i metoda sa popravkom. Konstruktivna varijanta BSO algoritma podrazumeva da se pri izgradnji novog rešenja na parcijalno generisano rešenje dodaju nove komponente, dok u metodi sa popravkom se modifikuju neke komponente od prethodnog kompletnog rešenja.

Pored opisanih, u literaturi postoje i drugi algoritmi zasnovani na inteligenciji rojeva: algoritam veštačke kolonije pčela (engl. artificial bee colony, ABC) [24], algoritam slepog miša (engl. bat algorithm, BA) [60], algoritam vatrometa (engl. fireworks algorithm) [49], algoritam optimizacije pretragom svitaca (engl. firefly algorithm, FA) [59], i mnogi drugi.

3 Mašina podržavajućih vektora

Mašina podržavajućih vektora (MPV) je jedan od binarnih klasifikatora, predložen od strane Kortesa i Vapnika [10]. MPV spada u grupu algoritama nadgledanog učenja, što znači da je za svaku instancu iz trening skupa poznata i klasa kojoj pripada. MPV je uspešno primenjen na brojne probleme u različitim oblastima, a posebno su zapažene primene u medicini. Na primer, u [63] mašina podržavajućih vektora je zajedno sa metodom k-sredina korišćena za klasifikaciju raka dojke, dok je u [2] testiran kvalitet MPV algoritma za prepoznavanje raka na mamografskim slikama. MPV je takođe korišćen za ranu detekciju Alchajmerove bolesti na MRI snimcima, zajedno sa drugim algoritmima mašinskog učenja [27]. U literaturi nalazimo primene MPV i u drugim oblastima: MPV se koristi u ekonomiji [20], astronomiji [44], hidrologiji [11], biologiji [61] i mnogim drugim.

Osnovni zadatak mašine podržavajućih vektora je određivanje jednačine hiperravani koja razdvaja instance dve različite klase. Instance su predstavljene kao tačke u d-dimenzionalnom prostoru, gde je d broj atributa koja opisuje jednu instancu. Kako je MPV binarni klasifikator, svaka istanca može pripadati samo jednoj od dve klase.

U cilju opisa mašina podržavajućih vektora, uvodi se sledeća notacija. Instance trening skupa su označene kao vektori $x_i \in \mathbb{R}^d$ gde je $i = 1,\ 2,\ ...,\ N$ dok je N broj instanci u trening skupu. Za svaku od instanci x_i je poznato da pripada nekoj od dve klase y_i, gde $y_i \in \{-1, 1\}$. Koristeći ovu notaciju, podaci iz trening skupa mogu biti zapisani kao:

$$\{x_i, y_i\}, \quad y_i \in \{-1, 1\},\ x_i \in \mathbb{R}^d, i = 1, ..., N. \tag{6}$$

Hiperravan koja razdvaja instance dve klase može biti zapisana na sledeći način:

$$f_{w,w_0}(x) = w \cdot x + w_0, \quad w_0 \in \mathbb{R},\ w \in \mathbb{R}^d, \tag{7}$$

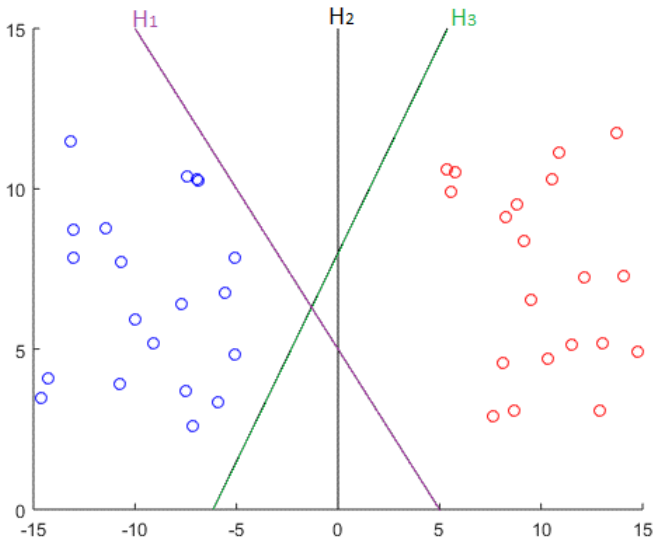
pri čemu je f_{w,w_0} funkcija koja slika R^d u R, a $w \cdot x$ skalarni proizvod definisan sa $w \cdot x = w^T x = w_1 x_1 + ... + w_d x_d$. Potrebno je odrediti koeficijente w i w_0 tako da su sve instance jedne klase sa iste strane, a instance iz raznih klasa sa različitih strana hiperravni. Preciznije, neophodno je da važi sledeći uslov:

$$y_i = \text{sign}(f_{w,w_0}(x_i)), \quad i = 1, ..., N. \tag{8}$$

Primetimo da w zapravo predstavlja vektor normale hiperravni, dok je udaljenost hipperravni od koordinatnog početka jednaka $\frac{w_0}{||w||}$, gde je $||.||$ proizvoljna norma vektora u R^d. Sa y_i je označena klasa (labela) instance x_i. Kako je klasa definisana znakom fukcije $f_{w,w_0}(x_i)$, može se reći da instanca pripada ili pozitivnoj ili negativnoj klasi. Sve tačke sa jedne strane hiperravni će imati pozitivnu vrednost funckije f_{w,w_0}, dok tačke koje se nalaze na suprotnoj strani imaju negativnu vrednost funkcije.

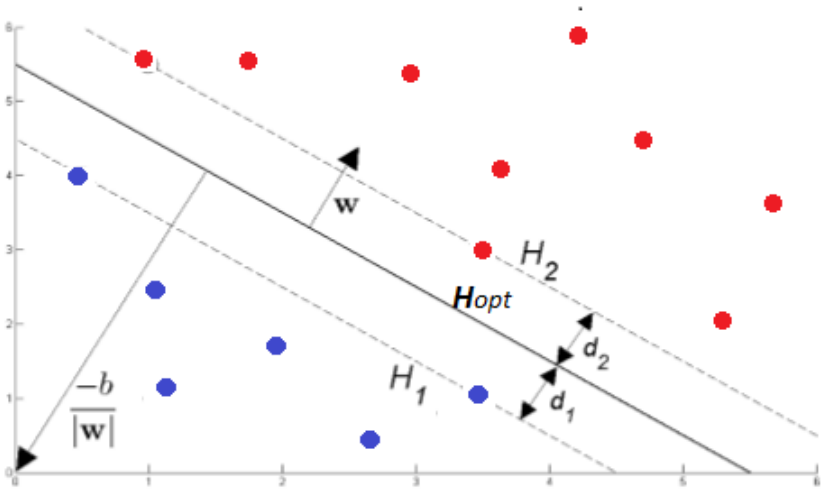
U opštem slučaju, hiperravan koja zadovoljava gornje uslove (razdvajajuća hiperavan) nije jedinstvena. Na Slici 2 prikazan je primer tri hiperravni $H_1,\ H_2$ i H_3 koje razdvajaju instance dve klase.

Među svim razdvajajućim hiperavnima, optimalna je ona hipperravan sa najvećom marginom. Margina razdvajajuće hiperravni se definiše kao minimalno rastojanje



Slika 2: *Primer hiperravni H_1, H_2, H_3 koje razdvajaju instance dve klase*

te hiperravni od neke tačke iz skupa instanci. Zadatak modela MPV je pronaći optimalnu hiperravan. Primer optimalne hiperravni i odgovarajuća margina prikazani su na Slici 3. Optimalna hiperravan (ravan na slici 3) je označena sa H_{opt}, prostor između pravih H_1 i H_2 je margina. Rastojanje H_{opt} od H_1 je označeno sa d_1 i jednako je rastojanju d_2, rastojanje prave H_{opt} od H_2. Udaljenost prave H_{opt} od koordinatnog početka je jednaka $\frac{-b}{||w||}$ i potrebna je za računanje rastojanja između granica margine.



Slika 3: *Optimalna hiperravan*

U cilju pronalaženja optimalne razdvajajuće hiperravani, potrebno je odrediti maksimalnu marginu, što se postiže postavljanjem nejednačina koje oslikavaju obezbeđuju da odgovarajući uslovi budu ispunjeni.

Neophodno je zadovoljiti uslov da svaka od instanci bude sa odgovarajuće strane

margine, u zavisnosti od klase kojoj instanca pripada dok je optimalna hiperravan na sredini te margine. Najpre, granice margine (prave H_1 i H_2 na slici 3) su dve paralelne hiperravni definisane sledećim jednačinama:

$$y_i(w \cdot x + w_0) = \delta \tag{9}$$
$$y_i(w \cdot x + w_0) = -\delta, \tag{10}$$

gde je $\delta = 1$ radi jednostavnosti računa. Rastojanje između te dve hiperravni je jednako $\frac{2}{||w||}$. Ovo rastojanje je funkcija po w koju treba maksimizovati, što je ekvivalentno minimizaciji vrednosti $||w||$. Kako bi bili ispunjeni uslovi da su sve istance van margine i sa odgovarajuće strane razdvajajuće hipperravni, umesto jednakosti (9)–(10) posmatramo:

$$w \cdot x_i + w_0 \geq 1, \quad za \quad y_i = 1 \tag{11}$$
$$w \cdot x_i + w_0 \leq -1, \quad za \quad y_i = -1. \tag{12}$$

Ove dve nejednakosti se mogu objediniti u:

$$y_i(w \cdot x_i + w_0) \geq 1, \quad \forall i = 1, 2, ..., N. \tag{13}$$

Uslovom (13) definisana je tzv. *tvrda margina.*

Konačno, problem optimizacije koji je potrebno rešiti u cilju određivanja jednačine optimalne hiperravni može biti formulisan kao problem kvadratnog programiranja:

$$\min_w \frac{||w||^2}{2} \tag{14}$$

pri uslovima

$$y_i \cdot (w \cdot x_i + w_0) \geq 1, \quad \forall i = 1, 2, ..., N, \tag{15}$$

Rešavanjem se dobijaju optimalni parametri mašine podržavajućih vektora w i w_0.

Problem optimizacije (14)–(15) se svodi na Lagranžov dualni problem. Lagranžova funkcija primalnog problema (14)–(15) je:

$$L(w, w_0, \alpha) = \frac{||w||^2}{2} - \sum_{i=1}^{N} \alpha_i \Big[y_i(x_i \cdot w + w_0) - 1 \Big], \tag{16}$$

gde je $\alpha \in R^N$ vektor sa koordinatama $\alpha_i,\ i = 1, .., N$. Dualni problem koje je potrebno rešiti je sledećeg oblika:

$$\min_w L(w, w_0, \alpha) \tag{17}$$

pri uslovima

$$y_i \cdot (w \cdot x_i + w_0) \geq 1, \quad \forall i = 1, 2, ..., N, \tag{18}$$

$$\alpha_i \geq 0. \tag{19}$$

Za fiksirane vrednosti α_i, tačka minimuma kvadratne funkcije L je upravo njena stacionarna tačka. Iz uslova stacionarnosti:

$$\frac{\partial L(w, w_0, \alpha)}{\partial w} = w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0, \tag{20}$$

dobija se vektor w izrazen preko α:

$$w(\alpha) = \sum_{i=1}^{N} \alpha_i y_i x_i. \tag{21}$$

Zamenom jednakosti (21) u izraz za funkciju L dobijamo:

$$L(w(\alpha), w_0, \alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i \alpha_i \alpha_j x_i^T x_j - w_0 \sum_{i=1}^{N} \alpha_i y_i. \tag{22}$$

Koeficijenti α_i se sada mogu dobiti kao rešenja problema konveksnog kvatratnog programiranja:

$$\max_{\alpha} \Big[\alpha^T \cdot e - \frac{1}{2} \alpha^T H \alpha \Big], \tag{23}$$

pri uslovima

$$y^T \cdot \alpha = 0, \tag{24}$$

$$\alpha_i \geq 0 \quad forall \ i = 1, 2, ..., N, \tag{25}$$

gde je $e \in R^N$ jedinični vektor, y je vektor koji sadrži sve labele $y = (y_1, y_2, ..., y_N)^T$ i H je simetrična pozitivno semidefinitna matrica sa elementima $H_{ij} = y_i y_j x_i^T \cdot x_j, \ i, j = 1, ..., N$.

Rešavanjem problema (23)–(25) dobijaju se optimalne koordinate vektora w koje se zatim uvrštavaju u problem (17)–(19). Rešenja ovog problema su tražene koordinate vektora w. Konačno, parametar w_0 se dobija iz nejednakosti (11) na sledeći način:

$$w_0 = -\frac{1}{2} \Big(\min_{i \in I^+} x_i^T w + \max_{i \in I^-} x_i^T w \Big), \tag{26}$$

gde I^+ i I^- predstavljaju skupove indeksa za koje su instance redom u pozitivnoj, odnosno negativnoj klasi.

Može se primetiti da jednačinu optimalne hiperravni definišu samo one instance koje su njoj najbliže, odnosno one koje leže na granicama margine je definišu. Ove instance se nazivaju podržavajući vektori i za njih važi:

$$y_i(w \cdot x_i + w_0) = 1. \tag{27}$$

Konačno, model mašine podržavajućih vektora ima oblik:

$$f_{w,w_0}(x) = \sum_{i=1}^{N} \alpha_i y_i x_i \cdot x + w_0, \tag{28}$$

gde su podržavajući vektori one instance za koje je $\alpha_i > 0$. Vrednost labele, a samim tim i klase je određena znakom funkcije f_{w,w_0}:

$$y = \text{sgn}(f_{w,w_0}(x)). \tag{29}$$

Metaparametar meke margine

Prethodno opisan model (28)–(29) mašine podržavajućih vektora je zasnovan na pretpostavci da skup instanci može biti razdvojen hiperravnima, odnosno da je linearno razdvojiv. Međutim, ovo je redak slučaj u praksi. Naime, često se dešava da postoje instance koje se značajno razlikuju od ostalih instanci iste klase (engl. outliers), a takođe neretko dolazi i do grešaka prilikom unosa podataka, greške prilikom merenja i slično. U ovakvim situacijama, koristeći model (28)–(29) nije moguće odrediti razdvajajuću hiperravan što bi vodilo ka praktičnoj neupotrebljivosti mašine podržavajućih vektora. U cilju prevazilaženja ovog problema, dozvoljava se izvesna greška, ali se teži da se veličina greške minimizuje. U tu svrhu, uvode se nenegativne promenljive ζ_i koje obezbeđuju da instance upadnu unutar margine. Preciznije, uvođenjem nenegativnih promenljivih ζ_i u nejednakosti (13) koje definišu tvrdu marginu transformišu se u:

$$y_i \cdot (w \cdot x_i + w_0) \geq 1 - \zeta_i, \quad \zeta_i \geq 0, \quad i = 1, 2, ..., N. \tag{30}$$

Problem optimizacije koji je potrebno rešiti je sledeći:

$$\min_{w} \frac{||w||^2}{2} + C\Big(\sum_{i=1}^{N} \zeta_i\Big) \tag{31}$$

pri uslovima

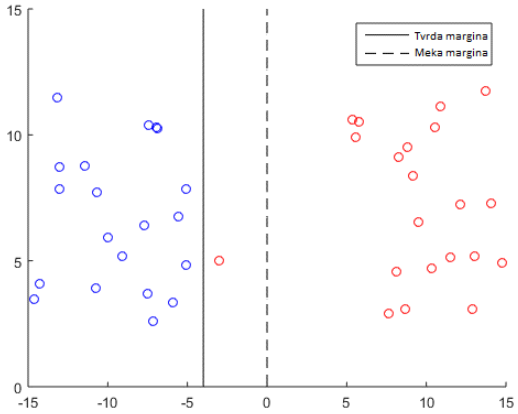
$$y_i \cdot (w \cdot x_i + w_0) \geq 1 - \zeta_i, \quad i = 1, 2, ..., N, \tag{32}$$

$$\zeta_i \geq 0, \quad i = 1, 2, ..., N, \tag{33}$$

gde C predstavlja metaparametar koji određuje uticaj greške na model.

U primeru prikazanom na Slici 4 tvrda margina pronalazi razdvajajuću hiperravan koja nije optimalna za dati skup podataka, dok meka margina pronalazi hiperravan koja dopušta da jedna instanca bude sa pogrešne strane (pogrešno klasifikovana), ali ostale instance deli pravednije u odnosu na tvrdu marginu.

Sa porastom vrednosti metaparametra C raste i uticaj greške, odnosno raste cena instance unutar ili sa pogrešne strane margine. Kada C teži beskonačnosti model sa mekom marginom (31)−(33) se svodi na model sa tvrdom marginom dat jednačinom (28). S druge strane, ako se vrednost metaparametra C smanjuje, i greška se takođe smanjuje i postaje nula u graničnom slučaju. Za veoma male vrednosti C, greška se



Slika 4: *Tvrda i meka margina*

praktično može ignorisati, što vodi ka neadekvatnom modelu. Za svaki konkretan skup podataka potrebno je odrediti optimalnu vrednost metaparametra C.

Model mašine podržavajućih vektora i dalje ima oblik (28), uz dodatni uslov da važi da vrednost Lagranžovih množioca mora biti manja ili jednaka od vrednosti metaparametra C, odnosno $0 \leq \alpha_i \leq C$.

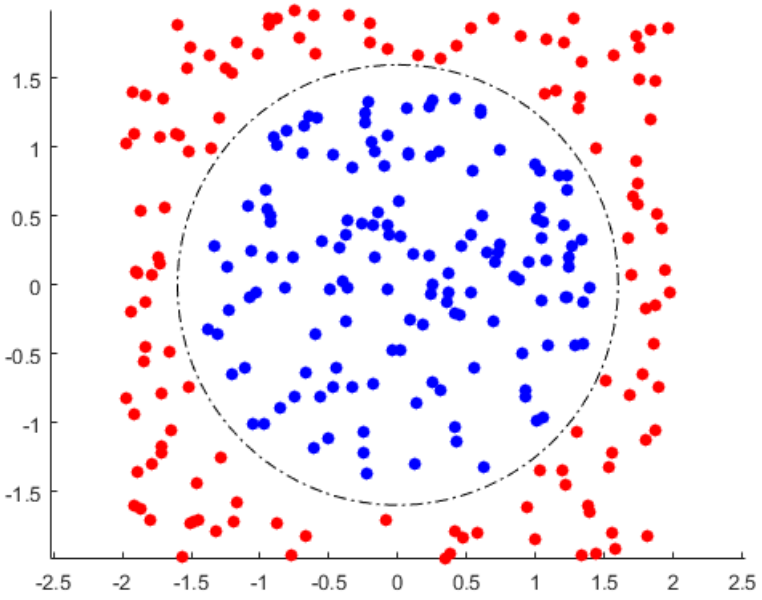
Metaparametar kernel funkcije

Uvođenjem meke margine u modele mašine podržavajućih vektora dozvoljeno je instancama da budu pogrešno klasifikovane. Za rešavanje MPV sa mekom marginom, dizajniran je algoritam nadgledanog mašinskog učenja koji može da definiše model za proizvoljni skup podataka. Prirodno se postavlja pitanje ocene kvaliteta rezultujućeg modela.

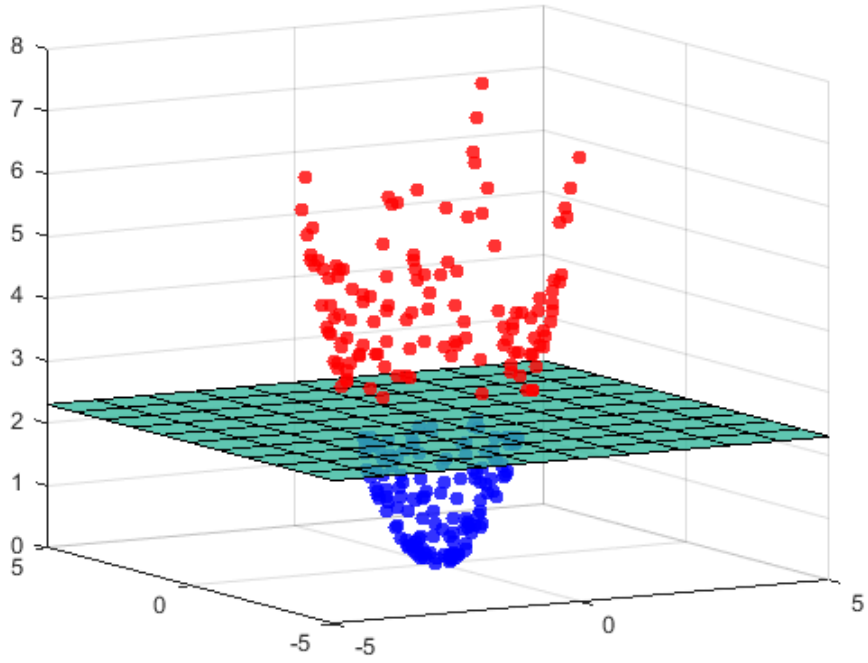
Jedan od nedostataka algoritma nadgledanog učenja je činjenica da koristi podelu instanci pomoću hiperravni. U praksi su česti slučajevi da hiperravan nije najpogodniji tip površi za razdvajanje instanci. Na Slici 5 je ilustrovan primer kruga kao razdvajajuće krive za instance koje se nalaze u ravni. U ovom primeru, instance jedne klase su unutar kruga dok su instance druge klase izvan tog kruga. Za ovaj primer ne postoji hiperravan (u ovom slučaju prava) koja će podeliti prostor (u ovom slučaju ravan) ako da se svaka instanca nalazi sa odgovarajuće strane prave. Pored ilustrovanog primera, nedostatak linearne separabilnosti postoji i kada je razdvajajuća granica zakrivljena površ.

Da bi se u opštem slučaju prevazišao ovaj problem nepostojanja razdvajajuće hiperravni u datom prostoru u kojem se nalaze instance, koristi se sledeća ideja. Instance koje nisu razdvojive pomoću hiperravni u nekom prostoru trebalo bi preslikati u prostor veće dimenzije u kojem je moguće razdvojiti instance modelom (31)–(33). U primeru prikazanom na Slici 5, preslikavanjem instanci iz dvodimenzionog u trodimenzionalni prostor, instance postaju razdvojive. Dakle, u trodimenzionom prostoru postoji ravan koja razdvaja instance različitih klasa, što je prikazano na Slici 6. U ovom slučaju je korišćeno preslikavanje:

$$(x_i, x_j) \rightarrow (x_i, x_j, x_i^2 + x_j^2). \tag{34}$$



Slika 5: *Preslikavanje podataka pomoću kernel funkcije*



Slika 6: *Preslikavanje podataka pomoću kernel funkcije*

Koristeći takozvane kernel funkcije i kernel trik (engl. kernel trick), može se izbeći eksplicitno preslikavanje instanci u visokodimenzionalni prostor. Naime, za sve instance koje pripadaju polaznom prostoru, vrednosti određenih funkcija mogu biti predstavljene u vidu skalarnog proizvoda definisanog na nekom drugom prostoru. Takve funkcije se nazivaju kerneli ili kernel funkcije. U metodi podržavajućih vektora, implicitno preslikavanje instanci se realizuje zamenom skalarnog proizvoda u definiciji

modela kernel funkcijom koja zadovoljava određene uslove.

Neka je $k : \chi \times \chi \to \mathbb{R}$ funkcija koja preslikava Dekartov proizvod nepraznog skupa χ sa samim sobom u skup realnih brojeva. Da bi k bila kernel funkcija, neophodno je da zadovoljava sledeći uslov: za svako $n \in \mathbb{N}$ i proizvoljan skup različitih tačaka $x_1, x_2, \ldots, x_n \in \chi$, matrica M dimenzije $n \times n$ sa elementima $k(x_i, x_j)$ je pozitivno semidefinitna. Preciznije, za svako $u \in \mathbb{R}^N$ važi $u^T M u \geq 0$.

Instance se implicitno preslikavaju u prostor veće dimenzije zamenom skalarnog proizvoda u (28) kernel funkcijom, pa novi model mašine podržavajućih vektora ima oblik:

$$f_{w,w_0}(x) = \sum_{i=1}^{N} \alpha_i y_i k(x_i, x) + w_0, \tag{35}$$

$$y = \operatorname{sgn}(f_{w,w_0}(x)). \tag{36}$$

U praksi najčešće se koristi nekoliko kernel funkcija kao što su linearna (odnosno skalarni proizvod), polinomska, Gausova (engl. Gauss Radial Basis Function, RBF), sigmoid i druge. Definicije nekih kernel funkcija koje se najčešće koriste su prikazane u Tabeli 1.

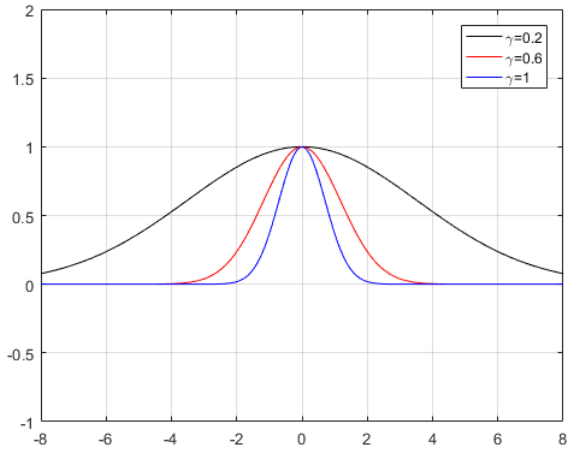
Tabela 1: *Definicije kernel funkcija*

Naziv funkcije	**definicija**
Linearna	$k(x_i, x_j) = x_i \cdot x_j + c,\ c \in R$
Polinomska	$k(x_i, x_j) = (\gamma(x_i \cdot x_j) + r)^d,\ r, d \in R$
Gausova	$k(x_i, x_j) = \exp(-\gamma\|x_i - x_j\|^2),\ \gamma \in R$
Sigmoid	$k(x_i, x_j) = \tanh(x_i \cdot x_j + r),\ r \in R$

Izbor odgovarajuće kernel funkcije najviše zavisi od problema odnosno podataka i cilja. Najbolji model mašine podržavajućih vektora se dobija pažljivim odabirom kernel funkcije i podešavanjem njenih metaparametara.

Pre izbora kernel funkcije, najpre se razmatra da li postoji potreba za kernel funkcijom, što je uglavnom slučaj. Često je prvi izbor Gausova (RBF) funkcija zato što uključuje i linearnu i polinomsku kernel funkciju, odnosno linearna i polinomska kernel funkcija predstavljaju specijalne slučajeve Gausove funkcije. Vrednost RBF funkcije primenjene na par vektora x i y odslikava njihovu sličnost i zavisi od proizvoda njihovog rastojanja izraženog preko norme razlike vektora i pozitivnog metaparametra γ (videti definiciju u Tabeli 1). Pri korišćenju RBF funkcije, neophodno je odabrati i odgovarajuću vrednost metaparametra γ. Sa smanjivanjem rastojanja $\|x - y\|$ i vrednost $\gamma\|x - y\|^2$ se smanjuje, pa se vrednost RBF funkcije povećava. Drugim rečima, dva bliska vektora imaju veću vrednost RBF funkcije od onih koji su udaljeni. Gausova funkcija ima oblik zvona, a metaparametar γ određuje "širinu zvona" (Slika 7). Manje vrednosti metaparametra γ vode ka širem zvonu i obrnuto.

U slučaju korišćenja RBF kao kernel funkcije za mašine podržavajućih vektora, od velikog je značaja podešavanje metaparametara γ. Veće vrednosti ovog metaparametra (usko Gausovo zvono) mogu voditi ka lošijem modelu, jer bi rezultujući



Slika 7: *Grafik Gausove funkcije za različite vrednosti metaparametra* γ

model mogao biti preprilagođen ulaznim podacima. S druge strane, suviše male vrednosti metaparametra γ vode do suviše grubog modela koji ne može da sagleda kompleksnost (ili oblik) podataka.

Multiklasifikacija pomoću binarnog klasifikatora

Mašina podržavajućih vektora je binarni klasifikator, međutim, prilikom rešavanja praktičnih problema, često se javlja potreba za podelu podataka u više od dve klase. Da bi se izvršila multiklasifikacija pomoću binarnog klasifikatora predložene su različite metode. U praksi se najčešće koriste metode *jedan protiv jednog* i *jedan protiv svih* [37]. Obe metode su zasnovane na ideji konstrukcije sistema više modela MPV za isti skup podataka. Svaka nova instanca se propušta kroz sve modele i na kraju, na osnovu dobijenih klasifikacija određuje se konačna klasa za datu instancu.

Metoda *jedan protiv jednog* pravi $\frac{n(n-1)}{2}$ modela za razdvajanje instanci u n klasa. Naziv metode potiče od činjenice da se za svaki par klasa konstruiše model. Nova instanca se klasifikuje od strane svih konstruisanih modela i na kraju se, na osnovu dobijenih rezultata, određuje konačna klasa. Najjednostavnija metoda dodeljivanja konačne klase instanci je zasnovana na principu glasanja: instanca pripada onoj klasi u koju je najviše puta klasifikovana. Na primer, u slučaju da postoji 10 klasa, svaka instanca može biti najviše 9 puta dodeljena istoj klasi. U slučaju izjednačenog rezultata između dve ili više klasa, uvode se dodatni mehanizmi odlučivanja.

Metoda *jedan protiv svih* za raspoređivanje podataka u n klasa pravi modele koji razlikuju svaku klasu od preostalih, što znači da ova metoda pravi n različitih modela. Na primer, u slučaju tri klase, ova metoda daje model koji razdvaja instance prve klase od instanci druge i treće, model za razdvajanje druge klase od prve i treće i, na kraju, model koji razlikuje instance treće od instanci prve i druge klase. U idealnoj situaciji, konačna klasa nove instance je određena tako što u odnosu na jedan model pripada jednoj (samostalnoj) klasi, dok u odnosu na ostala dva modela pripada zajedničkoj klasi, različitoj od predstavnika samostalne klase u tom modelu. U slučaju kada je instanca dva puta klasifikovana u samostalne klase, mogu biti primenjeni različiti dodatni mehanizmi odlučivanja. Na primer, može se izračunati verovatnoća pripadanja klasi ili dodatno napraviti modele *jedan protiv jednog* kako bi se rešili nejasni slučajevi.

4 Optimizacija metaparametara mašine podržavajućih vektora

U ovom poglavlju, dat je pregled metoda iz literature za određivanje metaparametara mašine podržavajućih vektora. Kako MPV predstavlja jedan od najrasprostranjenijih klasifikatora i primenjuje se u brojnim oblastima, konstrukcija dobrog modela je predmet istraživanja brojnih studija. Određivanje metaparametara ima veliki uticiaj na kvalitet modela mašine podržavajućih vektora.

U cilju evaluacije vrednosti metaparametara, za svaku od razmatranih vrednosti neophodno je odrediti tačnost odgovarajućeg modela. Jedan od pristupa se sastoji u podeli skupa instanci na skup za trening i skup za testiranje. Na osnovu instanci iz trening skupa, kreira se model, koji se zatim evaluira koristeći instance iz test skupa, odnosno upoređujući klasifikaciju dobijenu modelom i sa unapred poznatom klasifikacijom test instanci. Podela skupa instanci na test i trening skup se vrši u zavisnosti od problema, broja instanci u početnom skupu i drugih faktora. Podela na slučajan način nije najbolje rešenje, jer se može desiti da neki podskup sadrži većinu podataka neke klase, što vodi ka lošijem modelu. Iz tog razloga, često se koristi strategija podele koja obezbeđuje da u svakom podskupu svaka klasa bude podjednako zastupljena. Ova strategija u zavisnosti od broja instanci, može da bude veoma složena. Najčešće se podela realizuje tako da trećina, četvrtina ili petina podataka bude izdvojena za testiranje, a ostatak instanci se koristi za pravljenje modela. Ova metoda ocene modela MPV je relativno jednostavna, ali je njen nedostatak gubitak određenog broja važnih podataka, jer podaci u test instancama mogu biti značajni za konstrukciju modela, a oni su izostavljani tokom konstrukcije modela. Problem postaje još izraženiji ukoliko ostaje relativno mali skup instanci za treniranje, jer se tada ocena greške korišćenog modela može dovesti u pitanje.

U cilju ocene greške modela korišćenjem svih raspoloživih podataka, može se koristiti višeslojna unakrsna validacija (engl. k-fold cross validation) [17, 28]. Kod ove metode ocene kvaliteta modela, skup podataka se podeli na k podskupova (slojeva) koji sadrže približno jednake brojeve instanci. Proces ocene greške ima k iteracija. U svakoj iteraciji, po jedan od podskupova se koristi kao skup za testiranje modela. Model koji se testira na nekom podskupu je napravljen korišćenjem podataka iz preostalih $k-1$ podskupova koji predstavljaju trening skup. Ova metoda ocene greške kombinuje sve dostupne podatke i samim tim je pouzdanija od prethodno opisane metode koja koristi samo jedan trening i jedan test skup. Za svaku od instanci iz polaznog skupa podataka, ova metoda određuje njenu klasu nekim od modela tačno jednom ovom metodom, pa je rezultat unakrsne validacije procenat tačno klasifikovanih instanci. Korišćenjem unakrsne validacije smanjuje se mogućnost konstruisanja preprilagođenog modela (engl. overfitting) [21]. Preprilagođen model postiže visoku tačnost pri klasifikaciji instanci iz trening skupa, ali je tačnost klasifikacije novih instanci mala.

U praksi, najčešće se koristi unakrsna validacija sa pet ili deset podskupova (slojeva) [42, 13, 62, 40]. Za manji broj slojeva, unakrsna validacija je manje pouzdana, ali se koristi ukoliko je broj instanci u početnom skupu relativno mali, pa deljenjem na više slojeva podskupovi postaju suviše mali da bi se konstruisao upotrebljiv model. Sa druge strane, ukoliko se početni skup instanci deli na veći broj podskupova, metoda

višeslojne unakrsne validacije postaje računski zahtevna. Iz navedenih razloga, kod metode višeslojne unakrsne validacije, parametar v koji predstavlja broj slojeva najčešće ima vrednost pet ili deset prilikom konstrukcije modela.

Pored odabira metode, za procenu kvaliteta modela, neophodno je odrediti vrednosti metaparametara (C, γ) modela koje vode ka najmanjoj grešci. Kvalitet dobijenog modela se tada procenjuje ugnježđenom višeslojnom unakrsnom validacijom (engl. nested k-fold cross validation). Kod ove metode, jedna unakrsna validacija služi za pronalaženje metaparametara: par vrednosti za (C, γ) sa kojim se dobija najveća tačnost višeslojnom unakrsnom validacijom se bira za metaparametre konačnog modela. Ovo je takozvana unutrašnja validacija. Kada se pronađu vrednosti metaparametara (C, γ), tada se model koji će se nadalje koristiti dobija treniranjem nad svim raspoloživim instancama. Spoljašnja validacija služi za procenu kvaliteta tog modela. Postupak ugnježđene višeslojne unakrsne validacije je prikazan algoritmom 1.

Algorithm 1 Pseudo kod ugnežđene višeslojne unakrsne validacije

1: Skup podataka se deli na K podskupova (slojeva)
2: **for** i=1 to K **do**
3: Vrši se ocena greške svih konfiguracija na preostalih K-1 slojeva višelojnom unakrsnom validacijom
4: Bira se konfiguracija sa najmanjom greškom
5: Trenira se model pomoću te konfiguracije na preostalih K-1 slojeva
6: Vrši se predviđanje dobijenim modelom na izabranom sloju
7: **end for**
8: Računa se ocena greške

U literaturi postoji više metoda za određivanje vrednosti metaparametara (C, γ). Jedan od najjednostavnijih načina za pronalaženje para vrednosti (C, γ) je mrežna pretraga (engl. grid search) [22, 46]. Ideja ove metode je jednostavna: potrebno je napraviti mrežu unapred određenih vrednosti metaparametara i za njih konstruisati modele koji se zatim evaluiraju. Par vrednosti za (C, γ) za koji odgovarajući model postiže najveću tačnost se uzima za vrednosti metaparametara. Skup modela kod mrežne pretrage se konstruiše u okviru dvostruke petlje, od kojih jedna prolazi kroz različite vrednosti metaparametra C a druga kroz vrednosti metaparametra γ.

Na početku primene mrežne pretrage neophodno je odrediti opseg vrednosti za svaki od metaparametara C i γ, kao i korak sa kojim se pravi mreže vrednosti. Koraci za C i γ ne moraju biti jednaki. U praksi se za pronalaženje metaparametara mašine podržavajućih vektora C i γ metodom mrežne pretrage obično koristi eksponencijalna rastuća sekvenca. Vrednosti za metaparametar C su obično iz skupa $C = \{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$, a γ vrednosti iz skupa $\gamma = \{2^{-15}, 2^{-13}, \ldots, 2^{5}\}$ [22]. Iako je idejno jednostavna, nedostatak metode mrežne pretrage je njena računska složenost. Za svaki par metaparametara potrebno je napraviti k modela, gde je k broj slojeva u unakrsnoj validaciji. Drugi problem je određivanje koraka pretrage. Gušća mreža pretrage omogućava pronalaženje kvalitetnijeg modela mašine podržavajućih vektora, ali je proces pretrage računski zahtevniji. Računska složenost mrežne pretrage raste stepeno sa smanjivanjem koraka pretrage i dok sa povećanjem broja metaparametara raste eksponencijalno [4]. S druge strane, ukoliko je korak pretrage mali, algoritam

će brže završiti ali dobijene vrednosti metaparametara mogu voditi ka modelu lošeg kvalieta.

Dobra strana metode mrežne pretrage je što može lako da se paralelizuje. Parovi vrednosti (C, γ) su nezavisni, tako da je paralelizacija jednostavna. U cilju ubrzanja metode pretrage mreže, na početku se može uzeti veći korak i raditi gruba pretraga. Na početku postaviti veliki korak pretrage, odnosno C i γ uzimaju samo par vrednosti iz prostora pretrage. Kada se grubom pretragom pronađu bolje vrednosti metaparametara, može se primeniti finija pretraga na manjem intervalu. Ovaj postupak se može iterativno ponavljati više puta.

Za pronalaženje optimalnih metaparametara mogu se koristiti i neke numeričke metode optimizacije koje su računski efikasnije od mrežne pretrage i imaju bržu konvergenciju. Problem sa ovim metodama je osetljivost na lokalne minimume i velika zavisnost tačnosti rezultujućeg modela od od izbora početnog rešenja.

Imajući u vidu gore navedene teškoće prilikom određivanja vrednosti metaparametara mašine podržavajućih vektora, u literaturi se poslednjih godina sve više koriste metaheuristički algoritimi za optimizaciju metaparametara MPV, ali i za odabir atributa prilikom klasifikacije. Najčešće su to metaheuristicke metode inspirisane prirodom, kao što su evolutivni algoritmi (engl. evolutionary algorithms) [3],[9] i algoritmi inteligencije rojeva (engl. swarm intelligence algorithms) [5], [26], [6].

U radu [9] modifikovani genetski algoritam je korišćen za rešavanje optimizacije metaparametara (C, γ) mašine podržavajućih vektora koja se koristi za rano predviđanje spora projekata javno-privatnog partnerstva. Preciznije, autori su predložili takozvani brzi, neuredni genetski algoritam (engl. fast messy genetic algorithm). Hromozomi algoritma predstavljaju kodirana dopustiva rešenja, odnosno potencijalne parove vrednosti prametara (C, γ). U svakoj iteraciji algoritma, kroz operatore selekcije, ukrštanja i mutacije kreira se nova populacija rešenja (skup hromozoma). Nakon ispunjenja kriterijuma zaustavljanja, algoritam vraća najbolje pronađene vrednosti metaparametara. Analizom rezultata u radu [9] je zaključeno da je predložena varijanta evolutivnog algoritma dala vrednosti metaparametara (C, γ) koje vode ka modelu veće tačnosti u poređenju sa drugim osnovnim metodama koje se koriste za ovaj problem (CART, CHAID, QUEST i C5.0).

Algoritam optimizacije rojevima čestica (PSO) je prilagođen za optimizaciju metaparametara i odabir podskupa atributa instanci u radu [33]. PSO algoritam je posebno prilagođen za odabir metaparametara i za odabir podskupa atributa. Predloženi PSO algoritam prilagođen za optimizaciju metaparametara je upoređen sa mrežnom pretragom, neoptimizovanom MPV, kao i Njutnovom i Lagranžovom MPV predloženim u radu [15] i bio je superioran u odnosu na sve njih. U drugom slučaju, predloženi PSO metod prilagođen za odabir podskupa atributa i metaparametara je upoređen sa mašinom podržavajućih vektora optimizovanom genetskim algoritmom koji je takođe prliagođen za odabir i atributa i metaparametara. Na osnovu rezultata ovog poređenja se došlo do zaključka da MPV konstruisana pomoću PSO algoritma i genetskim algoritmom imaju slične performanse, na pet od devet test skupa bolje rezultate daje optimizacija PSO algoritmom.

U radu [4], dizajnirana je hibridizacija PSO algoritma i metode pretrage obrazaca (engl. pattern search) za optimizaciju metaparametara (C, γ) MPV. U predloženoj metodi, PSO algoritam je korišćen za eksploraciju prostora i pronalaženje obećavaju-

ćih regiona prostora pretrage, dok je pretraga obrazaca korišćena za lokalnu pretragu pronađenih regiona odnosno za eksploataciju. Implementirana je i probabilistička strategija odabira koja bira odgovarajuće jedinke iz populacije za lokalnu pretragu i na taj način održava balans između eksploracije i ekploatacije. Predloženi hibridni metod je upoređen sa drugim metodama iz literature i na osnovu dobijenih rezultata je zaključeno da je postignuto značajno poboljšanje kvaliteta klasifikacije primera iz standardnih test skupova.

Algoritam svitaca (engl. firefly algorithm, FA) predložen u [58] pronalazi adekvatne vrednosti metaparametara za mašina podržavajućih vektora korišćenu za regresiju sa više izlaza za prognoziranje cena akcija. Za ocenu kvaliteta predloženog FA algoritma, korišćene su tri standardne mere: statistička mera, ekonomska mera i cena izračunavanja. Dobijeni rezultati pokazuju da je predložena MPV optimizovana FA algoritmom obećavajuća alternativa za predviđanje intervalnih vrednosti finansijskih vremenskih serija.

U radu [56], autori su predložili PSO algoritam koji koristi teoriju haosa za preslikavenje, za podešavanje metaparametara varijante mašine podržavajućih vektora (engl. wavelet ν-support vector machine). Slučajne vrednosti r_p i r_g koje se koriste za računanje brzine čestica u jednačini (4) su zamenjene vrednostima iz haotičnih mapa (engl. chaotic maps). Ova modifikacija je omogućila bržu konvergenciju i povećanje tačnosti MPV.

U radu [34], koristi se slična ideja modifikacije PSO algoritma implemeniranjem haotičnih mapa. Predloženi modifikovani PSO je korišćen za pronalaženje metaparametara mašine podržavajućih vektora najmanjih kvadrata (engl. least square support vector machine). Eksperimentalni rezultati su pokazali da predložena metoda optimizacije smanjuje vreme izračunavanja i da se može uspešno primeniti čak i na test skupovima sa malim brojem instanci.

Algoritam slepog miša (engl. bat algorithm, BA) je korišćen za pronalaženje adekvatnih vrednosti metaparametara (C, γ) mašine podržavajućih vektore u radu [51]. Predložena metoda je testirana na devet standardnih test skupova i upoređena sa metodom mrežne pretrage, genetskim algoritmom i PSO algoritmom. Pokazano je da predloženi algoritam slepog miša uspešno pronalazi vrednosti za metaparametre MPV sa manjom greškom klasifikacije u poređenju sa ostalim metodama.

U literaturi se mogu naći i drugi radovi koji se bave primenom algoritama inteligencije rojeva i evolutivnih algoritama za rešavanje problema optimizacije metaparametara mašine podržavajućih vektora [38, 23, 30, 41]. U ovom radu, za podešavanje metaparametara mašine podržavajućih vektora korišćen je algoritam naseljavanja slonova, koji je zasnovan na inteligenciji rojeva. Elementi predloženog algoritma su prilagođeni razmatranom problemu i algoritam je testiran na standardnim test primerima iz literature, o čemu ce biti reči u narednim poglavljima.

5 Algoritam naseljavanja slonova

Algoritam naseljavanja slonova (engl. elephant herding optimization algorithm, EHO) predstavlja jedan od novijih metaheurističkih algoritama inteligencije rojeva. Ovaj algoritam je predložen 2015. godine od strane Wanga, Deba i Celha u radu [53], a inspirisan je društvenim ponašanjem slonova. Kompleksno ponašanje slonova u prirodi je uzeto u pojednostavljenom obliku za potrebe algoritma. U principu, populacija slonova živi podeljena u klanove. Slonovi u jednom klanu žive u matrijarhatu, dok odrasli mužjaci napuštaju klan čim odrastu i žive osamljeno.

U EHO algoritmu, jedinke su slonovi od kojih je svaki predstavljen svojim vektorom pozicije. Ponašanje jedinki (slonova) je idealizovano i svedeno na tri pravila:

- Populacija od n slonova se sastoji od k klanova (c_i) gde svaki klan ima konstantan broj članova n_{c_i}, $i = 1, 2, ..., k$.
- U svakoj generaciji, konstantan broj mužjaka napušta klan i živi u izolovanosti daleko od grupe.
- U svakom klanu c_i, svaki član j se pomera prema matrijarhatu, članu klana c_i, odnosno slonu sa najboljom vrednošću funkcije cilja.

Pozicija slona je određena vektorom x u D-dimenzionom prostoru pretrage ($\mathbb{R}^D$). Dimenzija D predstavlja dimenziju problema, odnosno broj parametara koje je potrebno odrediti u problemu optimizacije koji se razmatra. EHO algoritam uključuje dva operatora: ažuriranja pozicije na osnovu trenutne pozicije i pozicije glavne ženke u klanu i operator razdvajanja gde se najgori član zamenjuje.

Operator ažuriranja odnosno pomeranje članova klana je definisano sledećom formulom [54]:

$$x_{new,ci,j} = x_{c_i,j} + \alpha(x_{best,c_i} - x_{c_i,j})r, \tag{37}$$

gde $x_{new,c_i,j}$ označava novu poziciju slona j u klanu i, dok je $x_{c_i,j}$ njegova stara pozicija, x_{best,c_i} je trenutno najbolje rešenje unutar klana c_i. Parametar $\alpha \in [0, 1]$ je parametar algoritma koji određuje uticaj najboljeg člana na pomeranje, r je slučajan broj iz uniformne raspodele koji poboljšava raznovrsnost populacije u kasnijim fazama algoritma.

Pozicija najboljeg rešenja x_{best,c_i} u klanu c_i, se ažurira prema formuli [54]:

$$x_{new,c_i} = \beta x_{center,c_i}, \tag{38}$$

gde je $\beta \in [0, 1]$ parametar algoritma koji kontroliše uticaj centra klana. Centar klana c_i, u oznaci x_{center,c_i}, definisan je na sledeći način:

$$x_{center,c_i,d} = \frac{1}{n_{c_i}} * \sum_{l=1}^{n_{c_i}} x_{l,d}, \tag{39}$$

gde $1 \leq d \leq D$ predstavlja d-tu dimenziju u D-dimenzionom prostoru pretrage dok je n_{c_i} broj slonova u klanu c_i.

Mužjaci koji odlaze da žive daleko od klana odgovaraju komponenti eksploracije EHO algoritma. U svakom klanu c_i, tačno k mužjaka se pomera na novu poziciju, daleko od ostalih slonova u klanu. U EHO algoritmu, k mužjaka koji odlaze daleko od klana imaju najlošiju vrednosti funkcije cilja. Nova pozicija ovih slonova se računa prema sledećoj formuli:

$$x_{worst,c_i} = x_{min} + (x_{max} - x_{min} + 1)rand, \tag{40}$$

gde x_{min} i x_{max} predstavljaju redom donju i gornju granicu prostora pretrage, odnosno najmanju i najveću moguću vrednost vektora položaja jedinke x. Parametar $rand$ je slučajan broj izabran na osnovu uniformne raspodele. Ova formula postavlja jedinku nasumično u prostoru pretrage i koristi se i za određivanje početne pozicije jedinki.

Algoritmom 2 prikazan je pseudokod algoritma naseljavanja slonova.

Algorithm 2 Pseudo kod EHO algoritma

Inicijalizacija
Postavi jedinke nasumično po formuli 40
repeat
    Sortiraj sve jedinke prema vrednosti funkcije cilja
    Ažuriraj pozicije slonova u svakom klanu
    Ažuriraj najbolju jedinku unutar svakog klana
    U svakom klanu zameni pozicije k slonova sa najgorim vrednostima funkcije cilja
    Izvrši evaluaciju jedinki u populaciji sa ažuriranim pozicijama
until kriterijum zaustavljanja je zadovoljen
return Rešenje sa najboljom vrednošću funkcije cilja

U radu [53], algoritam naseljavanja slonova je primenjen na 15 standardnih benčmark funkcija i dodatno je kvalitet EHO algoritma potvrđen u [54]. gde je primenjen na 20 različitih benčmark funkcija i dva inženjerska benčmark problema, a dobijeni rezultati su upoređeni sa rezltatima primene drugih heuristika na iste probleme. Iako EHO predstavlja relativno nov algoritam, u literaturi već postoje brojni primeri primene EHO algoritma na razne probleme optimizacije. U [19], EHO algoritam je primenjen na problem kontrole nivoa tečnosti sistema tri rezervoara pri optimizaciji proportional-integral-derivative (PID) kontrolera. Algoritam naseljavanja slonova je predložen i za detekciju zajednica u složenim društvenim mrežama u [1]. U [45] pravila dobijenja primenom algoritma učenja pravilom asocijacija su optimizovana korišćenjem EHO algoritma.

6 Predloženi EHO algoritam za optimizaciju metaprametara mašina podržavajućih vektora

U ovom poglavlju opisana je primena EHO algoritma na optimizaciju metaparametara C i γ mašine podržavajućih vektora.

U EHO algoritmu svaka jedinka odgovara jednom dopustivom rešenju, odnosno paru vrednosti metaparametara (C, γ) za model mašine podržavajućih vektora. Predloženi algoritam koristi pretragu u logaritamskom prostoru, pa generisana rešenja predstavljaju stepen odnosno ukoliko je generisano rešenje (x_c, x_g) tada su metaparametri koji se koriste za pravljenje modela mašine potpornih vektora $C = 2^{x_c}$ i $\gamma = 2^{x_g}$. Ovakav prostor pretrage se pokazao kao praktičan za pronalaženje metaparametara mašine podržavajućih vektora. Očigledno, dimenzija prostora pretrage za ovaj problem je 2. U fazi inicijalizacije na slučajan način se generiše skup od n jedinki na sledeći način:

$$x_{id} = x_{min,d} + (x_{max,d} - x_{min,d})rand, \tag{41}$$

gde je i indeks jedinke, $d = 1, 2$ dimenzija, $x_{min,d}$ i $x_{max,d}$ predstavljaju redom donju i gornju granicu prostora pretrage za dimenziju d, a $rand$ je slučajan broj izabran uniformnom raspodelom iz segmenta $[0, 1]$.

Jedinke se dele u c klanova sa jednakim brojem članova. Prilikom podele jedinke su naizmenično raspoređivane u klanove, što je ekvivalentno nasumičnoj podeli. Na ovaj način se postiže raznovrsnost rešenja u svakom klanu. Podrazumeva se da je broj jedinki deljiv brojem klanova.

Nakon generisanja početne populacije potrebno je izvršiti evaluaciju dobijenih rešenja. Za optimizaciju metaparametara mašine podržavajućih vektora, vrednost funkcije cilja odgovara tačnosti modela MPV sa vrednostima metaparametara iz rešenja. Tačnost modela se može odrediti ili primenom istog na test skup instanci ili unakrsnom validacijom sa v preklapanja. U ovom radu je korišćena 10-slojna unakrsna validacija za ocenu tačnosti modela.

Nakon inicijalizacije i evaluacije generisanih rešenja započinje iterativni ciklus tokom koga se u svakoj iteraciji kreira nova populacija. Svaka nova populacija odgovara jednoj generaciji jedinki. U svakoj generaciji nove pozicije jedinki se računaju po formuli (37), najbolja jedinka u klanu se pomera po formuli (39), dok se k najgorih jedinki pomera po formuli (40). Dodatno je implementiran mehanizam koji sprečava da rešenje budu dopustiva, odnosno da ne budu van granica prostora pretrage. Ukoliko se pomeranjem pomoću formula EHO algoritma generiše rešenje van granica prostora pretrage ono se vraća na najbližu granicu.

Formule za ažuriranje pozicije slonova uključuju ograničavajući faktor, nasumičan broj u intervalu $[0, 1]$. U širokom prostoru pretrage pomeranje jedinki bi bilo zanemarljivo malo u odnosu na kompletan prostor. Iz ovog razloga standardna praksa je da se rešenja generišu u manjem intervalu a da se prilikom evaluacije funkcije cilja rešenja transformišu na posmatranom intervalu. U ovom radu rešenja se generišu u intervalu $[-1, 1]$ i nad tako generisanim rešenjima se primenjuju operacije ažuriranja i separacije. Pre evaluacije rešenja se transformišu formulom:

$$x_{trans} = x_{gener}\frac{x_{max} - x_{min}}{2} + x_{min}. \tag{42}$$

U predloženom EHO algoritmu kriterijum zaustavljanja je dostignut maksimalan broj iteracija. Osnovna struktura predložene metode je prikazana Algoritmom 3.

Algorithm 3 Pseudo kod predoženog EHO algoritma

Inicijalizacija
Postavi vrednosti parametara α i β EHO algoritma
Postavi broj jedinki n
Postavi broj klanova c
Postavi brojač generacija *t=1*
Postavi maksimalan broj generacija $MaxGen$
Inicijalizacija populacije
Generisanje slučajnih vrednosti za svaku jedinku (uređene parove (C, γ))
repeat
    Za svaku jedinku generisati model MPV sa odgovarajućim vrednostima metaparametara (C, γ)
    Izvrši evaluaciju svake jedinke na osnovu vrednosti funkcije cilja
    Sortiraj sve jedinke u nerastućem poretku prema vrednosti funkcije cilja
    Nasumično raspodeli jedinke u klanove jednake veličine
    for all klan c_i u populaciji **do**
        for all jedinke j u klanu c_i **do**
            Ažuriraj $x_{ci,j}$ i generiši $x_{new,ci,j}$ po formuli (37)
            if $x_{ci,j}$=$x_{best,ci}$ **then**
                Ažuriraj $x_{ci,j}$ i generiši $x_{new,ci,j}$ po formuli (38)
            end if
        end for
    end for
    for all klan c_i u populaciji **do**
        Zameni poziciju jedinki unutar klana c_i po formuli (40)
    end for
until $t < MaxGen$
return jedinka sa najboljom vrednošću funkcije cilja

7 Eksperimentalni rezultati

Predložena EHO metaheuristika za podešavanje metaparametara C i γ mašine podržavajućih vektora je implementirana u programskom jeziku Matlab R2016a a za MPV je korišćen paket LIBSVM (Verzija 3.22) [8]. Sva testiranja su izvršena na računaru sa procesorom Intel ® Core^{TM} i7-3770K CPU at 4GHz, 8GB RAM memorije i Windows 10 Professional operativnim sistemom.

U eksperimentalnoj analizi korišćeno je pet standardnih test skupova podataka: *iris*, *vowel*, *glass* i *wine* iz UCI kolekcije test primera [32] i *svmguide2* skup dostupan na internet strani [7]. Vrednosti parametara skupova podataka korišćenih u testiranju su prikazani u tabeli 2.

Tabela 2: *Pregled parametara korišćenih skupova podataka*

Skup	**Broj klasa**	**Broj atributa**	**Broj instanci**
Iris	3	4	150
Vowel	11	10	528
Glass	6	9	214
Wine	3	13	178
svmguide2	3	20	391

Parametri EHO algoritma su određeni empirijski izvođenjem nekoliko preliminearnih eksperimenata sa različitim veličinama populacije i vrednostima parametara α i β. Predloženi EHO algoritam je testiran sa različitim kombinacijama veličine populacije i vrednostima parametara. Na osnovu dobijenih rezultata, zaključeno je da algoritam radi najbolje sa sledećom konfiguracijom. Broj slonova u popoulaciji je postavljen na 50 dok se populacija deli u dva klana. Maksimalan broj evaluacije funkcije cilja je postavljen na 1500. Vrednost parametra α koji reflektuje uticaj najboljeg člana u klanu je fiksiran na 0.7, dok je parametar β koji kontroliše uticaj centra klana postavljen na vrednost 0.1. Granice pretrage su postavljene na sledeći način: $C \in [2^{-5}, 2^{15}]$ i $\gamma \in [2^{-15}, 2^{5}]$. Ove granice su standardne u literaturi za podešavanje vrednosti ova dva metaparametra MPV [22]. Tačnost modela je određena izvršavanjem 10-slojne unakrsne validacije i uzeta je za vrednost funkcije cilja.

U cilju ocene kvaliteta rešenja predloženog EHO algoritma, rezultati dobijeni korišćenjem EHO algoritma su upoređeni sa rezulatima prikazanim u radu [43]. Autori rada [43] su predložili genetski algoritam za rešavanje istog problema a rezultati upoređeni sa rezultatima metode mrežne pretrage.

U radu [43] je dokazano da za test primere iz tabele 2 metoda *jedan protiv jednog* za multiklasifikaciju daje bolje rezultate od metode *jedan protiv svih*. Iz tog razloga, prva tehnika je korišćena u ekseperimentima sa EHO algoritmom. Metoda *jedan protiv jednog* je implementirana u paketu LibSVM gde se klasa dodeljuje principom glasanja. U slučaju izjednačenog rezultata, konačna klasa instance je određena najmanjim indeksom klase među svim klasama sa izjednačenim brojem glasova.

U tabeli 3 je prikazano poređenje rezultata dobijenih predloženim EHO algoritmom, mrežnom pretragom i genetskim algoritmom predloženim u [43]. Tabela 3 je

organizovana na sledeći način. U prvoj koloni dat je naziv test skupa podataka, dok su u narednim kolonama date procentualne uspešnosti klasifikacije (procenat tačno klasifikovanih instanci) na skupu podataka iz odgovarajuće klase primenom metoda mrežne pretrage, genetskog algoritma i predloženog EHO algoritma, respektivno. Najbolji rezultati na svakom od razmatranih test skupova su istaknuti. Kako je predložena EHO metoda stohastički algoritam, očekivano je da se dobijaju različiti rezultati u svakom pokretanju algoritma. U poslednjoj koloni je prikazana porosečna tačnost modela MPV dobijenih u 5 različitih pokretanja predloženog EHO algoritma.

Tabela 3: *Poređenje rezultata predložene metode i rezultata prijavljenih u [43] (uspeh klasifikacije izražen u %)*

Skup	**Grid**	**GA**	**EHO**	**EHO prosek**
Iris	92.00	97.00	**98.67**	98.53
Vowel	93.00	97.00	**99.78**	99.62
Glass	70.00	**79.00**	76.83	75.15
Wine	92.00	96.00	**96.63**	96.63
svmguide2	83.00	86.00	**87.21**	86.42

Iz rezultata prikazanih u tabeli 3 može se videti da je predloženi EHO algoritam postigao bolje rezultate od mrežne pretrage na svim skupovima podataka, dok je samo za skup podataka *glass* genetski algoritam nadmašio rezultate dobijene EHO algoritmom. Najveće poboljšanje rezultata se uočava kod skupa *vowel*. Tačnost modela pronađenog kao optimalan mrežnom pretragom je 93.00%, genetskim algoritmom 97.00%, dok je predloženi EHO algoritam pronašao model koji klasifikaciju realizuje sa 99.62% tačnosti. Sa druge strane, najmanje poboljšanje koje je postignuto predloženim EHO algoritmom u odnosu na mrežnu pretragu i genetski algoritam je u slučaju skupa *wine*. Za ovaj skup podataka, genetskim algoritmom je pronađen model sa 96.00%, dok je predloženi EHO algoritam uspeo da poboljša kvalitet klasifikacije za samo 0.63%. Kao što se moglo i očekivati, genetski algoritam i predloženi EHO algoritam su postigli značajno bolje rezultate u odnosu na mrežnu pretragu, što potvrđuje da je optimizacija metaparametara korišćenjem metaheuristika pogodniji pristup.

U tabeli 4 su prikazane detaljnije informacije o rešenjima dobijenim EHO algoritmom. Prva kolona sadrži ime test skupa, dok druga i treća kolona sadrže vrednosti metaparametara koji su pronađeni predloženim algoritmom za koji su dobijene prijavljene tačnosti iz tabele 3. U poslednje dve kolone data su prosečna vremena izvršavanja algoritma i prosečno vreme izvršenja u zavisnosti od broja instanci (prosečno vreme podeljeno brojem instanci u test skupu) u sekundama.

Na osnovu rezultata prikazanih u tabeli 4, može se primetiti da se vrednosti metaparametara C i γ dobijene EHO algoritmom značajno razlikuju za različite skupove podataka. Ovakav ishod testiranja dodatno dokazuje važnost i potrebu podešavanja metaparametara mašine podržavajućih vektora za svaki konkretan problem koji se razmatra, kao i skupove podataka za isti problem koji su različite prirode.

Vreme izvršavanja EHO algoritma takođe varira od skupa do skupa podataka. Najduže vreme izvršavanja je za skup *vowel*. Ovakav ishod je očekivan, imajući u

Tabela 4: *Najbolje C i γ vrednosti i vremena izvršavanja dobijeni predloženim EHO algoritmom*

Skup	$\log C$	$\log \gamma$	vreme (s)	prosečno vreme po instanci (s)
Iris	5.6053	-1.5084	5.78	0.04
Vowel	6.6547	0.3382	181.62	0.34
Glass	12.5141	3.4296	135.67	0.63
Wine	14.1525	1.1490	92.06	0.50
svmguide2	2.3022	1.5267	123.28	0.32

vidu da ovaj skup podataka ima više instanci u poređenju sa ostalim. Prilikom svake konstrukcije modela se moraju odrediti α_i i w_0 iz (35), odnosno za svaki par (C, γ) je potrebno odrediti ove parametre čiji broj direktno zavisi od broja instanci u trening skupu. Vreme izvršenja na skupu *vowel* je 181.62 sekunde. Sa druge strane, za skup *iris* je potrebno manje od 6 sekundi da bi se pronašli parametri mašine podržavajućih vektora. U principu, vreme izvršavanja se povećava sa porastom veličine ulaznog skupa podataka, kao i sa konstrukcijom modela sa vrednostima metaparametara koji predstavljaju veoma loš izbor za konkretan skup. Rezultati testiranja pokazuju da predloženi EHO algoritam postiže rezultate viskog kvaliteta u relativno kratkom vremenu izvršavanja, ali postoji prostor za dodatno unapređivanje uvođenjem nekih mehanizama za bržu konvergenciju.

Tačnosti modela prijavljene u tabeli 3 predstavljaju tačnosti dobijene 10-slojnom unakrsnom validacijom. Kao što je opisano ranije, ovo ne predstavlja objektivnu procenu kvaliteta modela. Da bi se dobila objektivna ocena modela dobijenog sa metaparametarima prijavljenih u tabeli 4, korišćena je ugnježđena unakrsna validacija. U tabeli 5 su prikazane dobijene procene kvaliteta.

Tabela 5: *Procena kvaliteta modela korišćenjem ugnježđene unakrsne validacije*

Skup	Tačnost (%)
Iris	97.33
Vowel	99.05
Glass	73.36
Wine	94.94
svmguide2	82.31

8 Zaključak

U radu je razmatran problem podešavanja metaparametara mašine podržavajućih vektora. Implementiran je metaheuristički algoritam naseljavanja slonova (EHO) koji za cilj ima poboljšanje kvaliteta klasifikatora prilagođavanjem elemenata algoritma karakteristikama razmatranog problema.

U cilju testiranja implementiranog EHO algoritma korišćeno je pet standardnih skupova podataka za proveru kvaliteta klasifikatora. Četiri baze su preuzete iz UCI repozitorijuma mašinskog učenja dok je jedna baza preuzeta sa internet stranice paketa za mašine podržavajućih vektora, LibSVM tool. Izvršena je analiza i poređenje rezultata EHO algoritma sa rezultatima postojećih metoda iz literature za rešavanje istog problema – mrežnom pretragom i genetskim algoritmom.

Analizom implementiranih eksperimenata i dobijenih rezultata može se doći do zaključka da je predložena EHO metoda pogodna za podešavanje metaparametara mašine podržavajućih vektora. Za sve testirane skupove podataka pronađeni su metaparametri za koje model postiže visoku tačnost u razumnom vremenu. Predloženi EHO algoritam je za sve skupove podataka dao vrednosti metaparametara koji vode ka modelu veće tačnosti u odnosu na model dobijen pomoću mrežne pretrage. Za četiri od ukupno pet test skupova, predloženi EHO algoritam je u odnosu na optimizaciju metaparametara genetskim algoritmom pronašao vrednosti metaparametara za bolji model MPV.

Razmatrani problem je prvi put rešavan algoritmom naseljavanja slonova, što predstavlja osnovni dopirnos ovog rada. Predstavljeni rezultati u ovom radu ukazuju na kvalitet i potencijal predloženog algoritma za podešavanje metaparametara mašine podržavajućih vektora. Rezultati opisani u ovom rukopisu saopšteni su na međunarodnoj IEEE konferenciji i objavljeni u radu [52] indeksiranom u bazama Thompson-Reuters WoS i IEEExplore.

Postoje mogućnosti za unapređenja predloženog EHO algoritama. Dalje istraživanje može da obuhvati sledeće:

- Testiranje na većem broju skupova podataka za klasifikaciju,
- Poboljšavanje kvaliteta klasifikacije izborom podskupa atributa,
- Hibridizacija predloženog algoritma sa drugim metodama u cilju postizanja boljeg kvaliteta rešenja,
- Paralelilzacija predloženog algoritma u cilju bržeg izvršavanja.

9 Literatura

[1] Ahmed, K., Hassanien, A.E., Ezzat, E.: An efficient approach for community detection in complex social networks based on elephant swarm optimization algorithm. In: Handbook of Research on Machine Learning Innovations and Trends, pp. 1062–1075. IGI Global (2017)

[2] Azar, A.T., El-Said, S.A.: Performance analysis of support vector machines classifiers in breast cancer mammography recognition. Neural Computing and Applications 24(5), 1163–1177 (2014)

[3] Back, T.: Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford university press (1996)

[4] Bao, Y., Hu, Z., Xiong, T.: A PSO and pattern search based memetic algorithm for SVMs parameters optimization. Neurocomputing 117, 98–106 (2013)

[5] Blum, C., Li, X.: Swarm intelligence in optimization. In: Swarm Intelligence, pp. 43–85. Springer (2008)

[6] Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm intelligence: from natural to artificial systems. No. 1, Oxford University Press (1999)

[7] Chang, C., Lin, C.: LIBSVM Data: Classification (Multi-class) (2001), https://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/multiclass.html

[8] Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2(3), 27:1–27:27 (May 2011)

[9] Chou, J.S., Cheng, M.Y., Wu, Y.W., Pham, A.D.: Optimizing parameters of support vector machine using fast messy genetic algorithm for dispute classification. Expert Systems with Applications 41(8), 3955–3964 (2014)

[10] Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20(3), 273–297 (1995)

[11] Deka, P.C., et al.: Support vector machine applications in the field of hydrology: a review. Applied Soft Computing 19, 372–386 (2014)

[12] Dorigo, M., Gambardella, L.M.: Ant colonies for the travelling salesman problem. Biosystems 43(2), 73–81 (1997)

[13] Duarte, E., Wainer, J.: Empirical comparison of cross-validation and internal metrics for tuning svm hyperparameters. Pattern Recognition Letters 88, 6–11 (2017)

[14] Esmaili, I., Dabanloo, N.J., Vali, M.: Automatic classification of speech dysfluencies in continuous speech based on similarity measures and morphological image processing tools. Biomedical Signal Processing and Control 23, 104 – 114 (2016)

[15] Fung, G., Mangasarian, O.L.: Finite Newton method for Lagrangian support vector machine classification. Neurocomputing 55(1), 39–55 (2003)

[16] Glover, F.: Future paths for integer programming and links to artificial intelligence. Computers & operations research 13(5), 533–549 (1986)

[17] Golub, G.H., Heath, M., Wahba, G.: Generalized cross-validation as a method for choosing a good ridge parameter. Technometrics 21(2), 215–223 (1979)

[18] Gumus, E., Kilic, N., Sertbas, A., Ucan, O.N.: Evaluation of face recognition techniques using PCA, wavelets, and SVM. Expert Systems with Applications 37(9), 6404–6408 (2010)

[19] Gupta, S., Singh, V., Singh, S., Prakash, T., Rathore, N.: Elephant herding optimization based PID controller tuning. International Journal of Advanced Technology and Engineering Exploration 3(24), 194–198 (2016)

[20] Harris, T.: Credit scoring using the clustered support vector machine. Expert Systems with Applications 42(2), 741–750 (2015)

[21] Hawkins, D.M.: The problem of overfitting. Journal of Chemical Information and Computer Sciences 44(1), 1–12 (2004)

[22] Hsu, C.W., Chang, C.C., Lin, C.J.: A practical guide to support vector classification. Tech. rep., National Taiwan University (2010)

[23] Kanimozhi, T., Latha, K.: An integrated approach to region based image retrieval using firefly algorithm and support vector machine. Neurocomputing 151, Part 3, 1099–1111 (2015)

[24] Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical Report - TR06 pp. 1–10 (2005)

[25] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks. vol. 4, pp. 1942–1948 vol.4 (Nov 1995)

[26] Kennedy, J.: Swarm intelligence. In: Handbook of Nature-Inspired and Innovative Computing, pp. 187–219. Springer (2006)

[27] Khedher, L., Ramírez, J., Górriz, J.M., Brahim, A., Segovia, F., et al.: Early diagnosis of Alzheimer's disease based on partial least squares, principal component analysis and support vector machine using segmented MRI images. Neurocomputing 151, 139–150 (2015)

[28] Kohavi, R., et al.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the International Joint Conference on Artificial Intelligence, (IJCAI). vol. 14, pp. 1137–1145. Stanford, CA (1995)

[29] Koza, J.R.: Genetic programming: on the programming of computers by means of natural selection, vol. 1. MIT press (1992)

[30] Kuo, R., Huang, S.L., Zulvia, F., Liao, T.: Artificial bee colony-based support vector machines with feature selection and parameter optimization for rule extraction. Knowledge and Information Systems pp. 1–22 (2014)

[31] Larroza, A., Moratal, D., Paredes-Sanchez, A., Soria-Olivas, E., Chust, M.L., Arribas, L.A., Arana, E.: Support vector machine classification of brain metastasis and radiation necrosis based on texture analysis in MRI. Journal of Magnetic Resonance Imaging 42(5), 1362–1368 (Nov 2015)

[32] Lichman, M.: UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences (2013), http://archive.ics.uci.edu/ml

[33] Lin, S., Ying, K., Chen, S., Lee, Z.: Particle swarm optimization for parameter determination and feature selection of support vector machines. Expert Systems with Application 35(4), 1817–1824 (2008)

[34] Liu, F., Zhou, Z.: A new data classification method based on chaotic particle swarm optimization and least square-support vector machine. Chemometrics and Intelligent Laboratory Systems 147, 147–156 (2015)

[35] Liu, H., Liu, L., Zhang, H.: Ensemble gene selection for cancer classification. Pattern Recognition 43(8), 2763–2772 (2010)

[36] Malon, C., Uchida, S., Suzuki, M.: Mathematical symbol recognition with support vector machines. Pattern Recognition Letters 29(9), 1326 – 1332 (2008)

[37] Milgram, J., Cheriet, M., Sabourin, R.: "one against one"or "one against all": Which one is better for handwriting recognition with svms? In: Tenth international workshop on frontiers in handwriting recognition. Suvisoft (2006)

[38] Mustaffa, Z., Yusof, Y., Kamaruddin, S.S.: Enhanced artificial bee colony for training least squares support vector machines in commodity price forecasting. Journal of Computational Science 5(2), 196 – 205 (2014)

[39] Pai, P.F., Hsu, M.F., Lin, L.: Enhancing decisions with life cycle analysis for risk management. Neural Computing and Applications 24(7-8), 1717–1724 (2014)

[40] Palaniappan, R., Sundaraj, K., Sundaraj, S.: A comparative study of the svm and k-nn machine learning algorithms for the diagnosis of respiratory pathologies using pulmonary acoustic signals. BMC Bioinformatics 15(1), 223 (2014)

[41] Pereira, D.R., Pazoti, M.A., Pereira, L.A., Papa, J.P.: A social-spider optimization approach for support vector machines parameters tuning. In: IEEE Symposium on Swarm Intelligence (SIS). pp. 1–6. IEEE (2014)

[42] Rodriguez, J.D., Perez, A., Lozano, J.A.: Sensitivity analysis of k-fold cross validation in prediction error estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(3), 569–575 (2010)

[43] Samadzadegan, F., Soleymani, A., Abbaspour, R.A.: Evaluation of genetic algorithms for tuning SVM parameters in multi-class problems. In: Proceedings of the IEEE 11th International Symposium on Computational Intelligence and Informatics (CINTI). pp. 323–328 (2010)

[44] Shi, F., Liu, Y.Y., Sun, G.L., Li, P.Y., Lei, Y.M., Wang, J.: A support vector machine for spectral classification of emission-line galaxies from the sloan digital sky survey. Monthly Notices of the Royal Astronomical Society 453(1), 122–127 (2015)

[45] Sowkarthika, B., Tiwari, A., Singh, U.P.: Elephant herding optimization based vague association rule mining algorithm. International Journal of Computer Applications 164(5), 15–23 (2017)

[46] Staelin, C.: Parameter selection for support vector machines. Hewlett-Packard Company, Technical Report HPL-2002-354R1, http://www.hpl.hp.com/techreports/2002/HPL-2002-354R1.pdf (2003)

[47] Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization 11(4), 341–359 (1997)

[48] Talbi, E.G.: Metaheuristics: from design to implementation, vol. 74. John Wiley & Sons (2009)

[49] Tan, Y., Zhu, Y.: Fireworks algorithm for optimization. In: Tan, Y., Shi, Y., Tan, K.C. (eds.) Advances in Swarm Intelligence: Proceedings of the First International Conference (ICSI 2010), Part I. pp. 355–364. Springer, Berlin Heidelberg (2010)

[50] Teodorović, D.: Bee colony optimization (BCO). Innovations in swarm intelligence pp. 39–60 (2009)

[51] Tharwat, A., Hassanien, A.E., Elnaghi, B.E.: A ba-based algorithm for parameter optimization of support vector machine. Pattern Recognition Letters 93, 13–22 (2016)

[52] Tuba, E., Stanimirovic, Z.: Elephant herding optimization algorithm for support vector machine parameters tuning. In: Proceedings of the 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI). pp. 1–4. Targoviste, Romania, IEEE (2017), DOI 10.1109/ECAI.2017.8166464

[53] Wang, G.G., Deb, S., Coelho, L.d.S.: Elephant herding optimization. In: Proceedings of the IEEE International Symposium on Computational and Business Intelligence (ISCBI). pp. 1–5 (2015)

[54] Wang, G.G., Deb, S., Gao, X.Z., Coelho, L.D.S.: A new metaheuristic optimisation algorithm motivated by elephant herding behaviour. International Journal of Bio-Inspired Computation 8(6), 394–409 (2016)

[55] Whitley, D.: A genetic algorithm tutorial. Statistics and computing 4(2), 65–85 (1994)

[56] Wu, Q.: A self-adaptive embedded chaotic particle swarm optimization for parameters selection of Wv-SVM. Expert Systems with Applications 38(1), 184–192 (2011)

[57] Xian, G.: An identification method of malignant and benign liver tumors from ultrasonography based on GLCM texture features and fuzzy SVM. Expert Systems with Applications 37(10), 6737–6741 (2010)

[58] Xiong, T., Bao, Y., Hu, Z.: Multiple-output support vector regression with a firefly algorithm for interval-valued stock price index forecasting. Knowledge-Based Systems 55, 87–100 (2014)

[59] Yang, X.S.: Firefly algorithms for multimodal optimization. Stochastic Algorithms: Foundations and Applications, Lecture Notes in Computer Science 5792, 169–178 (2009)

[60] Yang, X.S.: A new metaheuristic bat-inspired algorithm. Studies in Computational Intelligence 284, 65–74 (November 2010)

[61] Yu, J.S., Pertusi, D.A., Adeniran, A.V., Tyo, K.E.: Cellsort: a support vector machine tool for optimizing fluorescence-activated cell sorting and reducing experimental effort. Bioinformatics 33(6), 909–916 (2016)

[62] Zhang, Y.D., Yang, Z.J., Lu, H.M., Zhou, X.X., Phillips, P., Liu, Q.M., Wang, S.H.: Facial emotion recognition based on biorthogonal wavelet entropy, fuzzy support vector machine, and stratified cross validation. IEEE Access 4, 8375–8385 (2016)

[63] Zheng, B., Yoon, S.W., Lam, S.S.: Breast cancer diagnosis based on feature extraction using a hybrid of k-means and support vector machine algorithms. Expert Systems with Applications 41(4), 1476–1482 (2014)