SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD, br. 901

PEDVIĐANJE KODIRAJUĆIH REGIJA U GENOMU KORIŠTENJEM POZNATIH KLASIFIKACIJSKIH METODA

Maja Bellotti

Zagreb, lipanj 2009.

Veliko hvala mentoru Damiru Seršiću na konstantnoj potpori i motivaciji, profesoru Mili Šikiću na strpljivim odgovorima, svim asistentima na ukazanoj pomoći i kolegicama Vedrani Baličević, Mirni Bokšić i Maji Štimac na razumijevanju i suradnji.

SADRŽAJ:

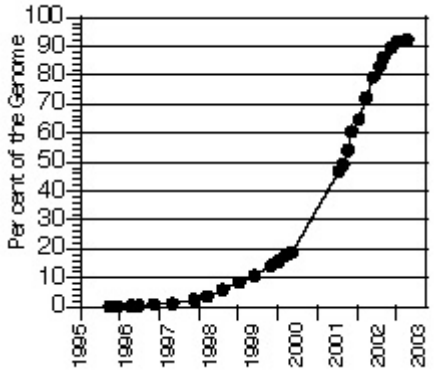
Uvod

Količina podataka koja se stvara i zahtijeva obradu raste s napretkom tehnologije eksponencijalno. Tu nastaje potreba za njihovom efikasnom obradom. Zbog obujma podataka znanstvenici i inženjeri se oslanjaju na snagu računala pri procesiranju i obradi podataka. U današnje vrijeme se sve više pozornosti usmjerava ka strojnom učenju koje nalazi svoju primjenu u širokom opusu zanimanja. Bitno je dobro odabrati metodu primijene. Brojne klasifikacijske metode daju rješenja, ali nameće se pitanje kojom brzinom i s kolikom točnošću.

Tema ovog završnog rada se bazira na rezultatima grupe znanstvenika sa njemačkog instituta - Friedrich Miescher Laboratory. Istraživanje su usmjerili na predikciju točnih mjesta izrezivanja u genomu koristeći poznatu klasifikacijsku metodu nadziranog učenja – SVM (eng. Support Vector Machines). Cilj rada bio je ponoviti njihove rezultate, time omogućujući daljnje istraživanje i unaprjeđenje njihovih rezultata dodatnim poravnavanjem i/ili izborom drugih klasifikacijskih metoda. Utvrđeno je da najbolje rezultate daje primjena SVM-a s implementiranim težinskim kernelom – (eng. weighted degree kernel - WD i weighted degree with shifts - WDS). Realizacija tih kernela je malo složenija od običnih koji se najčešće upotrebljavaju pri treniranju podataka (npr. polynomial, Radial Basis Function - RBF, Gausian radial basis function, hyperbolic tangent) te time i zahtijeva više vremena za obradu.

2. Bioinformatika

Bioinformatika je grana znanosti koja se bavi informatičkim osnovama, kao i problemom pohranjivanja, organizacije i analize bioloških podataka. Svoju primjenu prolazi u farmaceutskoj industriji i medicini. Istraživanja ljudskog genoma, kao i genoma drugih živih bića, su u zadnjih par desetljeća u stalnom porastu i interes za rezultatima na tom polju je velik. Svakodnevna istraživanja donose rezultate koji su ponekad i oprečni. Jedan dio znanstvenika se priklanja vjerovanju da ljuski genom još nije istražen do kraja i da cijela sekvenca još nije pronađena, dok drugi dio opovrgava tu teoriju. Istraživanja u sklopu projekta – *Human Genom Project*, donose grafički prikaz kako se kroz godine nadopunjavala sekvenca.

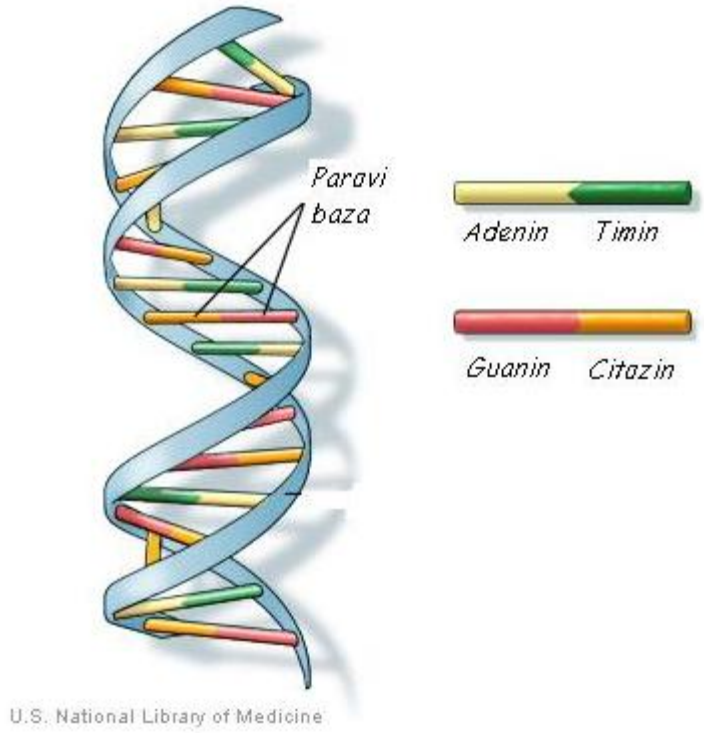


Sl.1.HGP- istraženost sekvenci ljudskog genoma u zadnjem desetljeću

Puno je stvari još neobjašnjivo čovjeku. I kad u potpunosti budemo znali sekvencu preostaje nam izučavanje kako mehanizmi rezanja DNK (Deoksiribonukleinske kiseline) funkcioniraju, na koji način se proteini stvaraju, kako dolazi do mutacije gena i time kako liječiti neke rijetke bolesti nastale njihovom mutacijom.

2.1. DNK i proteini

Deoksiribonukleinska kiselina je nukleinska kiselina u obliku dvostruke, spiralno zavijene zavojnice koja sadrži genetičke odrednice za specifični biološki razvoj staničnih oblika života i većine virusa. DNK je dugački polimer nukleotida koji kodira redoslijed aminokiselina u proteinima koristeći genetički kod tj. trostruki kod nukleotida.



Sl.2. DNK sa parovima baza(adenin na timin; guanin na citozin)

DNK nije jedinstvena molekula već par molekula međusobno povezanih vodikovim vezama i organiziranih na način da su njeni lanci međusobno komplementarni. Svaki se lanac sastoji od građevnih jedinica, nukleotida, koji se dijele na 4 skupine: adenin(A), guanin(G), citozin(C) i timin(T). Između dva lanca svaka baza jednog lanca može biti sparena s određenom bazom drugog lanca i to tako da se adenin spaja uvijek sa timinom i obrnuto, te citozin s guaninom na isti način. DNK sadrži genetičku informaciju koja se nasljeđuje potomcima. Ta

informacija određena je redoslijedom parova baza, a parovi baza određuju protein. RNK (Ribonukleinska kiselina) molekule su posrednici između DNK i njihovog konačnog produkta, proteina.

Proteini (bjelančevine) su kemijske stanice koje upravljaju svim životnim procesima stanice. Građeni su od dvadesetak različitih aminokiselina međusobno povezanih kao karike u lancu, a redoslijed i broj tih karika određuje pojedini protein. Aminokiseline su izgrađene od tripleta nukleotida (kodona) i broj takvih tripleta je 64,a postojećih aminokiselina 20.

AMINOKISELINE	TRIPLETI NUKLEOTIDA-KODONI
Lys	AAA AAG
Asn	AAT AAC
Arg	AGA AGG CGA CGG CGT CGC
Ser	AGT AGC TCA TCG TCT TCC
Ile	ATA ATT ATC
START/Met	ATG
Thr	ACA ACG ACT ACC
Glu	GAA GAG
Asp	GAT GAC
Gly	GGA GGG GGT GGC
Val	GTA GTG GTT GTC
Ala	GCA GCG GCT GCC
Tyr	TAT TAC
Cys	TGT TGC
Leu	TTA TTG CTA CTG CTT CTC
Phe	TTT TTC
Gln	CAA CAG
His	CAT CAC
Pro	CCA CCG CCT CCC
Trp	TGG
STOP	TAA TAG TGA

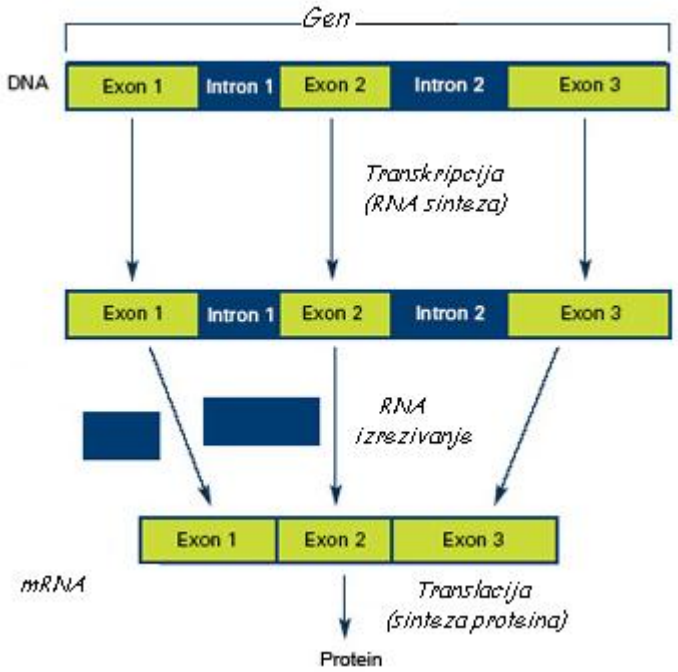
Tablica 1. Kombinacije nukleotida u aminokiselinama

2.2. Izrezivanje (eng. splicing)

Redoslijed baza u jednom od lanaca DNK se procesom transkripcije kopira na manju molekulu ribonukleinske kiseline (mRNK) koja tu informaciju prenosi do mjesta sinteze proteina. Pri kopiranju se timin zamijeni s uracilom. Informacije sadržane u kodonima prevode se u aminokiseline u procesu translacije (sinteze proteina).

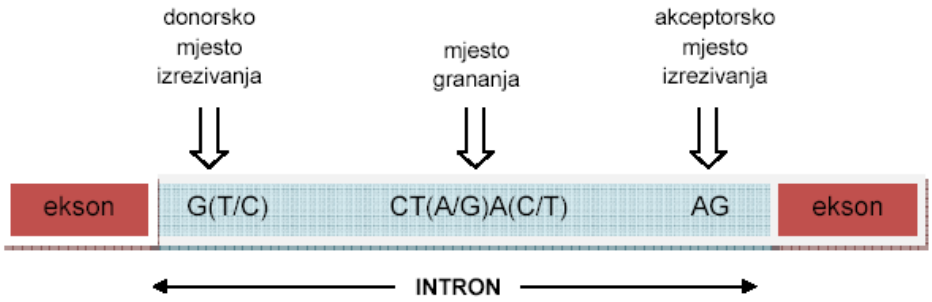
Sl.3. Proces sinteze proteina

Glasnička RNK (mRNK) se sastoji od kodirajućeg dijela – eksona i nekodirajućeg dijela – introna. U procesu izrezivanja se introni uklanjaju, a eksoni spajaju jedan na drugi. Ovaj cijeli proces se odvija kod eukariotskih organizama.



Sl.4. Izrezivanje introna i eksona

Usprkos poznavanju kompletnog genoma većine živih organizama, još se ne znaju točne pozicije mjesta izrezivanja, tj prijelaza između introna i eksona. Početak introna se naziva donorsko mjesto izrezivanja (eng. donor splice site), a eksona akceptorsko mjesto (eng. acceptor splice site). Oba mjesta imaju vrlo značajnu formu ponavljanja nukleotida, koja doduše varira od organizma do organizma. Golema većina svih mjesta izrezivanja su takozvana kanonička mjesta (eng. canonical splice sites) koja su okarakterizirana prisutnošću dimera AG za donorsko i GT za akceptorsko mjesto. GT se pojavljuje u 99% slučajeva, a onih 1% otpada na GC (nekanoničko mjesto). Primjera radi, ljudski genom ima 6 x 10^9 nukleotida, GT se može pronaći 400 milijuna puta, a gruba procjena kaže da su samo 0.1 % od tog broja prava mjesta izrezivanja. To će stvarati problem klasifikaciji zbog neizbalansiranog seta podataka.



Sl.5. Građa introna

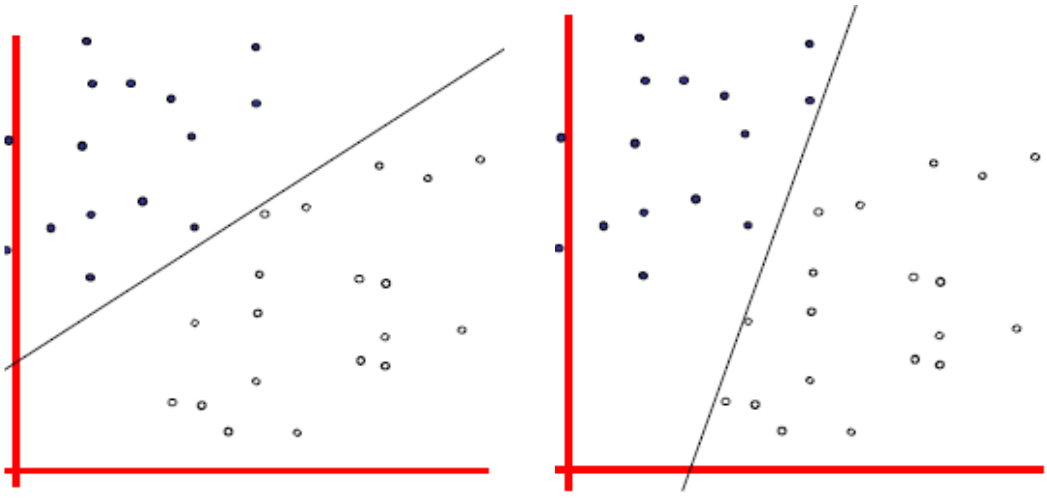
3. Statističke metode

Zbog već navedenog problema prevelike količine podataka za obradu u zadnjih par desetljeća znanstvenici su osmislili razne algoritme pojednostavljenja problema i donošenja odluka. Primjeri takvih klasifikatora su: Slučajne šume (eng. Parallel Random Forest - PARF), neuronske mreže, Bayesovi klasifikatori, Support Vector Machines (SVM) i mnogi drugi. To su mehanizmi za analiziranje, prepoznavanje i učenje iz ulaznih podataka. Još se nazivaju „strojevi za učenje“ (eng. learning machines) i dio su softvera koji implementira i algoritam za učenje i model čiji parametri moraju biti određeni dijelom softvera za učenje.

Ova tema je proširenje rada na kolegiju Projekt. Stari zadatak je bio pronalazak točnih mjesta izrezivanja eksona i introna koristeći PARF koji implementira klasifikacijski algoritam pod nazivom „Stablo odluke“ (eng. The Decision Tree). PARF funkcionira na način da iz ulaznog skupa podataka nasumično izabire određeni broj njih i tako stvara jedno stablo odluke. Korisnik određuje broj stabala, s time da preciznost rezultata (smanjivanje vjerojatnosti pogreške) raste s povećanjem broja stabala. Postoje dvije mogućnosti korištenja PARF-a. Prva je korištenje istih ulaznih podataka i za treniranje i za testiranje, a druga je podjela ulaznog seta podataka na navedene dijelove (proizvoljno od strane korisnika). U fazi treniranja klasifikatora PARF-u se serviraju poznati podatci da na njima uči kako donijeti odluku. Druga faza uključuje davanje nepoznatih podataka na temelju kojih klasifikator metodom „većine glasova“ donosi presudu. Klasifikatoru se zadaje prag. Stabla moraju glasati za neku odluku brojem većim ili jednakim tom pragu. Rezultati dobiveni na Projektu su bili zadovoljavajući, ali još nedovoljno dobri u usporedbi sa rezultatima dobivenim na njemačkom institutu. Stoga je cilj unaprijediti rezultate prvo ponavljanjem njihovih rezultata, kako bi dalje na istim podatcima mogli dobiti i bolja rješenja.

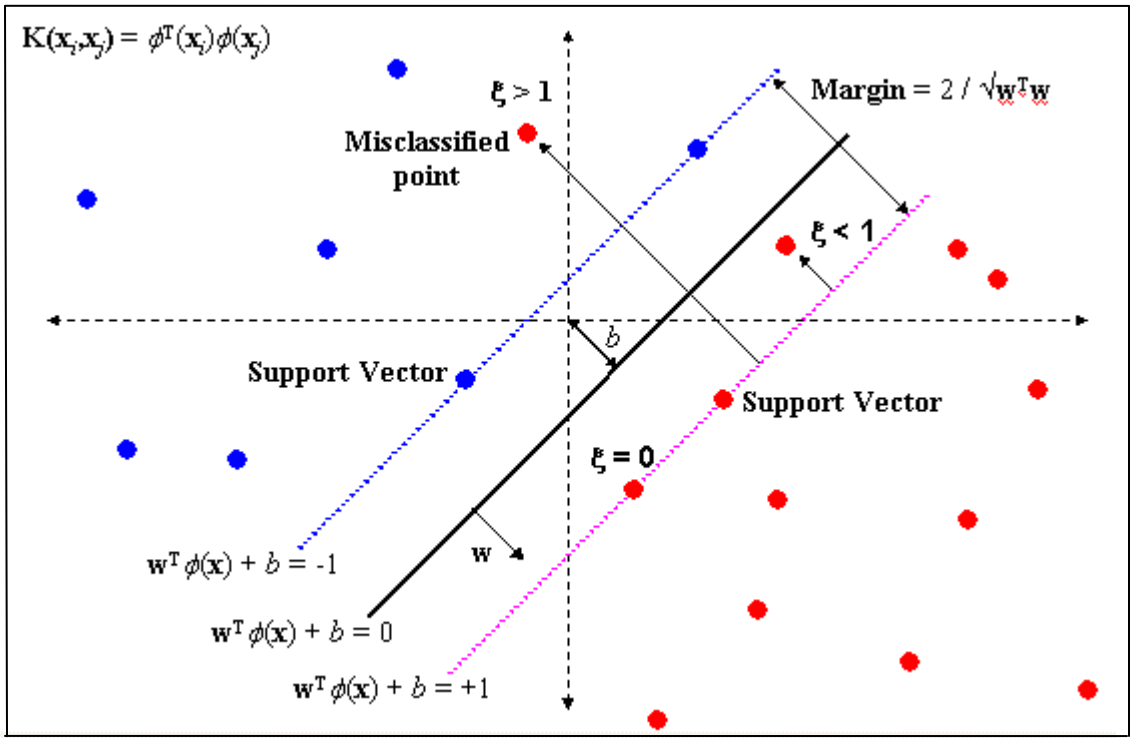
3.1. Support vector machines (SVM)

Spomenuto je da strojno učenje olakšava problem inženjerima u procesiranju podataka, međutim količina podataka koje senzori generiraju lako može izaći iz granica procesorske moći najnovijih kompjutera. Cilj je pronaći metodu strojnog učenja koja najbolje rješava taj problem. Izazov primjene SVM-a na velikim bazama podataka leži u činjenici da memorija potrebna za rješavanje optimizacijskog problema raste drastično s povećanjem seta za treniranje. Rješenje problema leži u primjeni raznih algoritama za učenje, kao što je LIBSVM (A LIbrary For Support Vector Machines), koji optimiziraju proces i omogućavaju dobivanje rezultata u razumnom vremenu. Postoje dvije vrste nadziranog učenja, klasifikacija (prepoznavanje uzoraka) i regresija (aproksimacija funkcije). Ovaj rad se temelji na problemu klasifikacije. Najjednostavniji takav problem je binarni, gdje se primjeri za treniranje nalaze u dva različita razreda. Ulazni vektori x_i se mogu razvrstati u razrede y_i € {1,-1}. Zadatak klasifikacije je napraviti klasifikator koji može rasporediti prije neviđene ulazne vektore x_j u definirane razrede. SVM metoda će razdijeliti nepoznate podatke pravcem, krivuljom odnosno plohom (eng. hyperplane) ovisno o tipu SVM-a koji se koristi te o prostoru u kojem dijelimo.



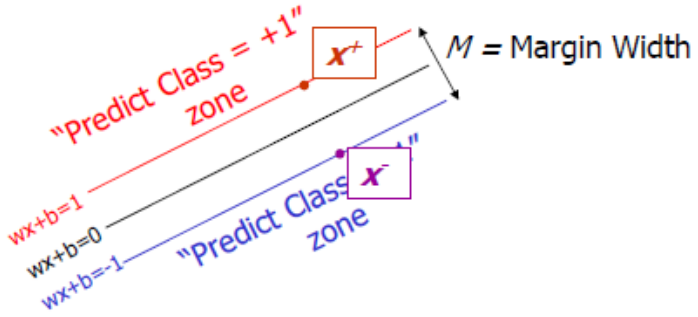
Sl.6. Odabir granice korištenjem linearne klasifikacije

Na slici 6. je primjer dva linearna klasifikatora. Bilo koji od njih zadovoljava početnu tezu dijeljenja ulaznih podataka na dva razreda, no pitanje je koji odabrati kao najbolji. Bitno je pronaći maksimalnu udaljenost dvaju paralelnih granica između razreda. Pojednostavljeno to znači pronaći takvu granicu čija je udaljenost od najbližeg ulaznog podatka maksimizirana. Postoji li kao takva naziva se maksimalna granica klasifikatora (eng. maximal margin).



Sl.7. Maksimalna granica kod SVM-a

Kao što je vidljivo na slici 7. ulazni uzorci koji su se našli na dvije krajnje granice (eng. plus and minus plane) se nazivaju pomoćni vektori (eng. support vectors). Iskustveno je dokazano da je ovo najbolji način klasifikacije elemenata i da je njegovom primjenom smanjena mogućnost pogrešne klasifikacije.



Sl.8. Podjela ulaznih podataka u razrede

Sa slike 8. je vidljivo da je udaljenost dvije krajnje granice označena kao M (širina granica, eng. margin width). M je dana formulom:

$$M = \frac{2}{||w||} \tag{1}$$

Bolji klasifikator teži pronalaženju što veće širine. Pravce definiramo funkcijom

$$d(x, \boldsymbol{w}, b) = \boldsymbol{w}^T x + b = \sum_{i=1}^{m} w_i \, x_i + b \tag{2}$$

gdje $\boldsymbol{w}$ označava vektor, a *b* nagib (eng. bias). Središnja linija na slici je opisana pravcem $d(x, w, b)$=0. Nakon uspješnog treniranja, koristeći dobivene težine, stroj za učenje na nepoznatom uzorku podataka x_p, proizvede izlaz *o* prema indikatorskoj funkciji:

$$i_F = o = sign\left(d(x_p, \boldsymbol{w}, b)\right). \tag{3}$$

gdje je *o* (eng. output) standardna notacija za izlaz. Drugim riječima odluka glasi:

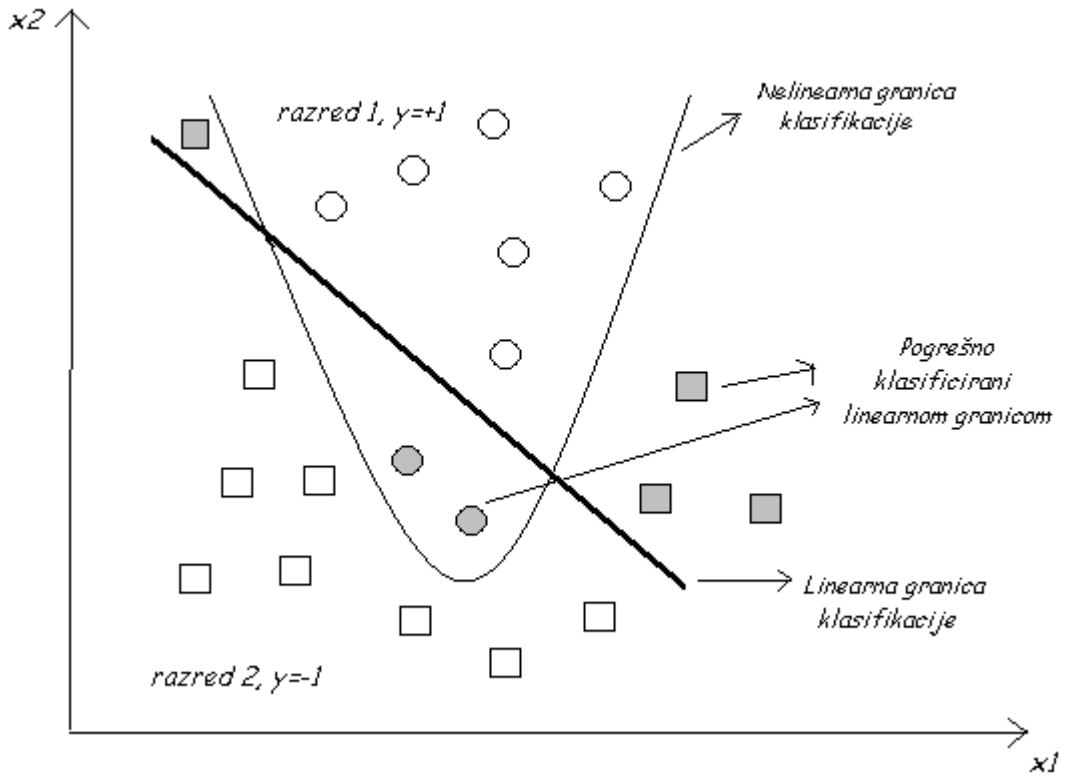
- ako je $d(x_p, \boldsymbol{w}, b) > 0$, uzorak x_p pripada razredu 1 $(o = y_p = +1)$
- te ako je $d(x_p, \boldsymbol{w}, b) < 0$, uzorak x_p pripada razredu -1 $(o = y_p = -1)$

Nakon što su pronađeni pomoćni vektori (SV) moguće je izračunati granicu na temelju očekivane vjerojatnosti pojavljivanja pogreške u skupu podataka za test na način:

$$E_n[P(pogreške)] \leq \frac{E[broj\ SVova]}{n} \qquad (4)$$

gdje E_n označava očekivanje na cijelom skupu za treniranje veličine n. Ova formula pokazuje kako je lako odrediti granicu koja je neovisna o dimenzijama ulaznog skupa podataka. Stoga SVM sa malim brojem pomoćnih vektora će imati dobru generalizaciju čak i u visoko dimenzionalnim sustavima.

Problem se javlja kada linearni klasifikatori ne mogu dobro podijeliti ulazne podatke. Tada se javljaju pogrešno klasificirani uzorci ulaznih podataka (eng. misclassified class points), tj. smješteni u pogrešan razred. U takvim slučajevima se uvodi proširenje na nelinearne klasifikatore.



Sl.9. Sivo označeni su pogrešno klasificirani koristi li se linearni klasifikator

Na slici 9. je vidljivo da je moguće pronaći granicu koja neće pogrešno klasificirati nijedan podatak. Glavna ideja nelinearnih klasifikatora je razvrstati ulazne vektore podataka $x_i \epsilon R^m$ u vektore $\boldsymbol{\phi}(x_i) \epsilon R^s$ višedimenzionalnog prostora S (gdje $\boldsymbol{\phi}$ predstavlja preslikavanje (eng. mapping): $R^m \rightarrow R^s$) i riješiti linearni klasifikacijski problem u ovom prostoru :

$$x \epsilon R^m \rightarrow \boldsymbol{\phi}(x) = [\, \boldsymbol{\phi}_1(x)\ \boldsymbol{\phi}_2(x) \ldots \boldsymbol{\phi}_s(x)]^T \epsilon R^s \qquad (5)$$

Preslikavanje $\boldsymbol{\phi}$ je odabrano unaprijed, fiksirana je funkcija. Očekivano ovakav način preslikavanja ponovo vodi do rješavanja optimizacijskog problema. Sada indikatorska funkcija glasi:

$$i_F\big(d(\boldsymbol{x})\big) = sign(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) + b) = sign(\textstyle\sum_{i=1}^{n} y_i\, \alpha_i K(\boldsymbol{x_i}, \boldsymbol{x}) + b) = \qquad (6)$$

$$= sign(\textstyle\sum_{i=1}^{n} v_i\ K(\boldsymbol{x_i}, \boldsymbol{x}) + b)$$

gdje v_i odgovara izlaznim težinama SVM-a, a $K(\boldsymbol{x_i}, \boldsymbol{x})$ označava vrijednost kernel funkcije. Bitno je primijetiti razliku težine $\boldsymbol{w}$ koja označava vektor dimenzija ulaznog prostora i v_i koji je skalar jednak $y_i \alpha_i$.

Javljaju se dva ključna problema kod korištenja ovakvog preslikavanja:

1. Izbor preslikavanja $\boldsymbol{\phi}$ koje rezultira u bogatom broju krivulja ili ploha kao rješenja.
2. Izračun skalarnog produkta (eng. dot product) $\boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\phi}(\boldsymbol{x})$ može biti računski vrlo izazovan problem ako je broj značajki velik.

Eksplozija u dimenzijama može se izbjeći korištenjem $K(\boldsymbol{x_i}, \boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\phi}(\boldsymbol{x})$. Treba primijetiti da je kernel funkcija K funkcija početnog ulaznog prostora. Uočava se glavna prednost korištenja kernel funkcija, koja leži u izbjegavanju izvođenja preslikavanja uopće. Umjesto toga skalarni produkt se računa direktno izračunom kernel funkcije za dani set za treniranje u ulaznom prostoru. Ovakvo

zaobilaženje problema dimenzionalnosti i smanjivanja vremena treniranja se popularno naziva „kernel trik“ (eng. kernel trick). U tome leži čar SVM-a u primjeni. Taj „trik“ je razlog zašto je SVM jedna od najpopularnijih metoda. Ne računajući sve umnoške štedi se puno na stvarnom vremenu i memorijskom prostoru.

4. Predviđanje kodirajućih regija korištenjem SVM-a

Njemački institut (Friedrich Miescher Laboratory) je 2007. objavio bitno istraživanje u području bioinformatike pod nazivom „Accurate splice site prediction using support vector machines“. U svom radu se referenciraju na brojna istraživanja u kojima su uspoređujući klasifikacijske metode na malom setu podataka (gen NN269 i DGSsplicer) te na 5 organizama zaključili da je optimalna metoda klasificiranja ipak SVM i to sa definiranim kernelom – težinskim (eng. weighted degree) i težinskim s pomakom (eng. weighted degree with shifts). Istraživanje su vršili, kao što je već spomenuto, na 5 organizama i to na:

- Caenorhabditis elegans
- Drosophila melanogaster
- Arabidopsis thaliana
- Danio rerio
- Homo sapiens

Rezultate su reprezentirali kroz Precision-Recall krivulje. Ova krivulja se crta koristeći četiri parametra:

- TP – True positives → detektor je rekao da je točno i bilo je točno (odluka donesena na temelju ulaznih podataka)
- TN – True negatives → detektor kaže da je netočno i njegova odluka je prema ulaznim podacima točna
- FP – False positives → detektor kaže da je točno i njegova odluka je prema ulaznim podacima netočna (bilo je netočno)
- FN – False negatives → detektor kaže da je netočno i njegova odluka je prema ulaznim podacima netočna (bilo je točno)

Formule za izračunavanje varijabli Precision i Recall su:

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN} \tag{7}$$

Posebnu je pažnju potrebno posvetiti shvaćanju ulaznog seta podataka. Postoji mogućnost pojave neizbalansiranog ulaznog seta (npr. najjednostavniji klasifikator koji predviđa -1 za sve primjere će imati sto postotnu točnost no takav rezultat nije relevantan). Također je bitno voditi računa kako urediti ulazne podatke. U radu na Projektu se potkrala greška kada su korišteni smanjeni ulazni podatci na način da se uzmu svi točni rezultati i svako deseti netočni(utvrđeno je da nema utjecaja na konačni rezultat zbog mnogo većeg broja netočnih od točnih). Tada i krivulju treba podesiti na način da u formuli za „Precision“ dodamo koeficijent 10 koji množi FP. Glavni problem je bio implementirati kernele u neki od programskih jezika i provući treniranje i testiranje podataka na SVM-u da bi dobili sumjerljive rezultate. Institut je naveo formule za računanje kernela koje je trebalo prevesti u funkcije i koristiti.

Težinski kernel (tako se zove jer pridodaje veću/manju važnost pojedinim elementima ispitivanja) uspoređuje dva vektora iste duljine koristeći prozor veličine δ. Broji mjesta na kojima dolazi do poklapanja podnizova (svi simboli podniza jednog vektora jednaki simbolima podniza drugog). Težinski kernel je definiran sa :

$$K(x, x') = \sum_{\delta=1}^{d} w_\delta \sum_{l=1}^{N-\delta+1} I(u_{\delta,l}(x) = u_{\delta,l}(x')) \tag{8}$$

gdje je izabrana težina $w_\delta = d - \delta + 1$, a d=22. Težina smanjuje važnost za poklapanja višeg reda (δ veći), koje ionako imaju veći doprinos zbog većeg broja poklapajućih nukleotida.

Težinski kernel s pomakom (WDS) ima kao jedinu razliku uvođenje parametra pomaka *s* (eng. shift), te se tako sada ne uspoređuju iste pozicije u oba vektora već se podniz jednog vektora uspoređuje s podnizom drugog pomaknutog za *s*. WDS je definiran sa:

$$K(x_i, x_j) = \sum_{\delta=1}^{d} w_\delta \sum_{l=1}^{N-\delta+1} \sum_{s=0}^{S(l)} \delta_s \ \mu_{\delta,l,s,x_i,x_j} \tag{9}$$

$$\mu_{\delta,l,s,x_i,x_j} = I\left(u_{\delta,l+s}(x_i) = u_{\delta,l}(x_j)\right) + I\left(u_{\delta,l}(x_i) = u_{\delta,l+s}(x_j)\right) \quad (10)$$

gdje je w_δ zadržao vrijednost kao i u prošlom slučaju, a $\delta_s = \frac{1}{2(s+1)}$ je težina dodijeljena pomacima dužine s u oba smjera. $S(l)$ određuje gornju granicu pomaka u za određeni l. Ovdje je izabrano da je $S(l) = \sigma|l - l_c|$, l_c je pozicija mjesta izrezivanja u vektoru podataka predanom SVM-u,a σ je odabran 0.5.

Za oba kernela je još potrebno provesti normizaciju :

$$\tilde{K}(x, x') = \frac{K(x,x')}{\sqrt{K(x,x)K(x',x')}} \quad (11)$$

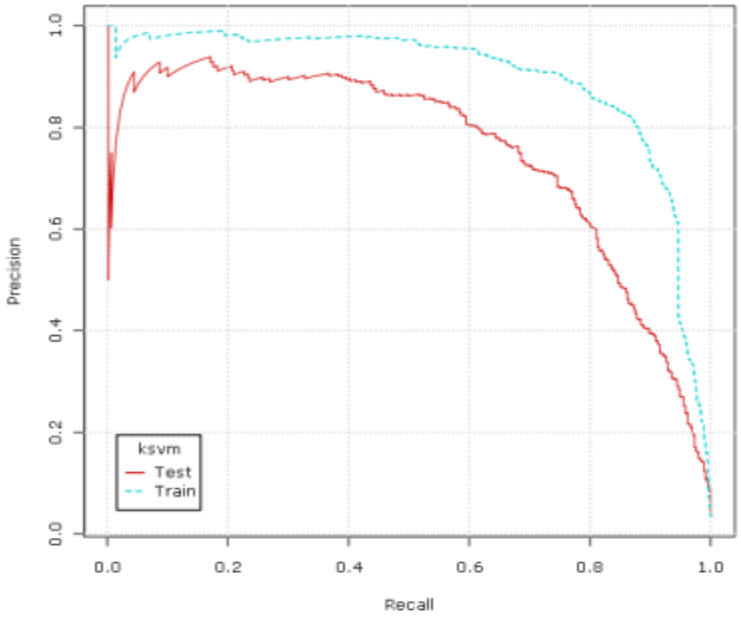
gdje je rezultat ove formule naša kernel funkcija. Implementacija u svim paketima na isti način radi sa kernelima. U statističkim paketima za SVM metodu je realiziran cijeli postupak treniranja i testiranja funkcijama i dozvoljen je pristup i izmjena korisniku. Moguće je ubaciti svoj kernel (eng. user defined kernel) i testirati podatke na njemu.

4.1. Tehnički podatci

Prvi cilj je bilo proučavanje istraživanja Njemačkog instituta da bi njihove rezultate bilo moguće implementirati i ponoviti. Istraživanje je vršeno na njihovim podatcima radi usporedivosti, već poravnatim i sortiranim, preuzetim sa: ftp://ftp.tuebingen.mpg.de/fml/behr/splicing/. Podatci su organizirani u vektore dužine 398 nukleotida. Rad je prvenstveno baziran na organizmu Caenorhabditis elegans i to na donorskim podatcima. Izbor je takav zbog optimalnog broja podataka u odnosu na ostale organizme i zbog ograničenosti snage računala za obradu drugih. Glavni problem je stvarala brzina izvođenja algoritma učenja klasifikatora, te uopće nedovoljno računalne memorije. Zbog toga je trebalo skratiti ulaznu količinu podataka, a da se time ne naruši konačni rezultat ili na drugačiji način organizirati podatke. Pri istraživanju su korišteni programski paketi MATLAB, R i C++ (Visual Studio). Statistički paket R ima implementirano grafičko sučelje

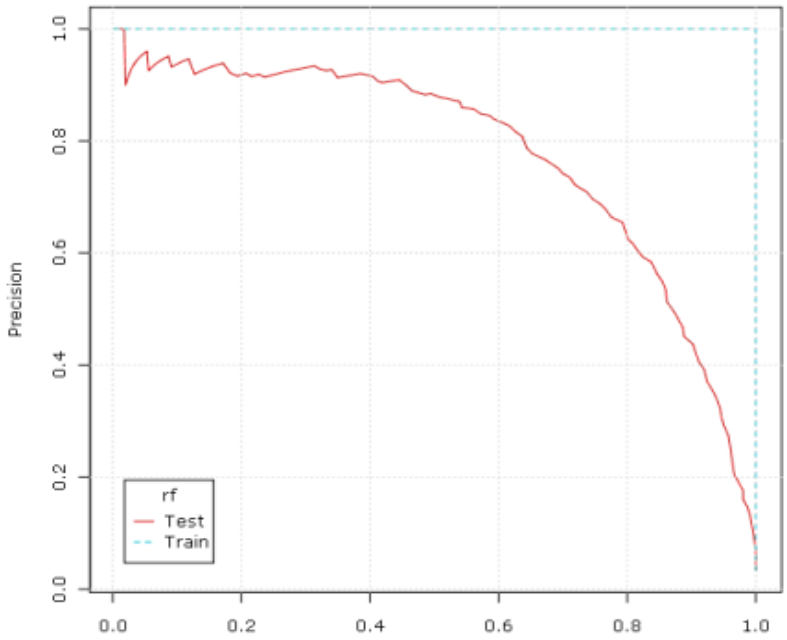
Rattle, za lakšu provedbu učenja. Za male količine podataka, R je dobar izbor, no pri povećanju podataka programi se izvode presporo, ako se uopće uspiju pokrenuti.

Rattle je „user friendly“ inačica R-a u kojoj odabirom ikona možemo odrediti kojom klasifikacijskom metodom se želimo koristiti. Unutar Rattle-a je realiziran i SVM i RF. U radu na projektu korišten je R za usporedbu točnosti SVM-a i RF-a. Na malom skupu podataka je pokazana nadmoć metode RF stoga je odlučeno ustanoviti tu nadmoć i na većem skupu podataka.



Sl.10. Rezultati dobiveni metodom SVM na skupu proba_48

Slika 10. pokazuje kakve rezultate daje SVM na testnom skupu podataka proba_48 koristeći njegov osnovni kernel – Radial Basis Function - RBF (najjednostavniji i sa najbržim vremenom izvođenja). Crvena krivulja na slici pokazuje rezultate testiranja, tj. kako je klasifikator presudio na neviđenim podatcima, a plava rezultate treniranja.



Sl.11. Rezultati dobiveni metodom RF na skupu proba_48

4.1.1. R- statistički paket

Već spomenuta inačica R-a, Rattle, kodira u sintaksi programskog jezika R odabrane opcije u njenom grafičkom sučelju. Tada se po potrebi mijenjaju kerneli u pozivu funkcije. Implemetacija kernela WD i WDS u bilo kojem statističkom paketu prima dva niza (vektora) i vraća skalarni produkt, tj. kernel. U privitku se nalaze funkcije kernela napisane U R-u, no nisu iskorištene za dobivanje rezultata zbog već spomenutog problema s pretjerano dugim izvođenjem programa. Stoga je istraživanje preusmjereno u već poodmaklom stadiju završnog rada na korištenje prihvatljivijeg paketa LIBSVM.

4.1.2. LIBSVM paket

LIBSVM je jedan od najbržih i popularnijih SVM rješenja. Za korištenje paketa u Matlabu potrebno je koristiti C++ prevodilac (eng. compiler). U Završnom radu je korištena verzija 2.89. Neke su funkcije tako napisane u Matlabu, a neke u jeziku C-a. Kernele je bilo potrebno reprogramirati da rade u jeziku C-a i definirati nova

dva kernela (WD i WDS) u već postojećim funkcijama. U README datoteci je detaljno opisano korištenje paketa te su navedeni primjeri pozivanja funkcija. Jednostavnim pozivom naredbe u sučelju Matlaba

matlab>> mex –setup

odabiremo kojim kompajlerom se želimo služiti, te prevodimo funkcije u direktoriju naredbom:

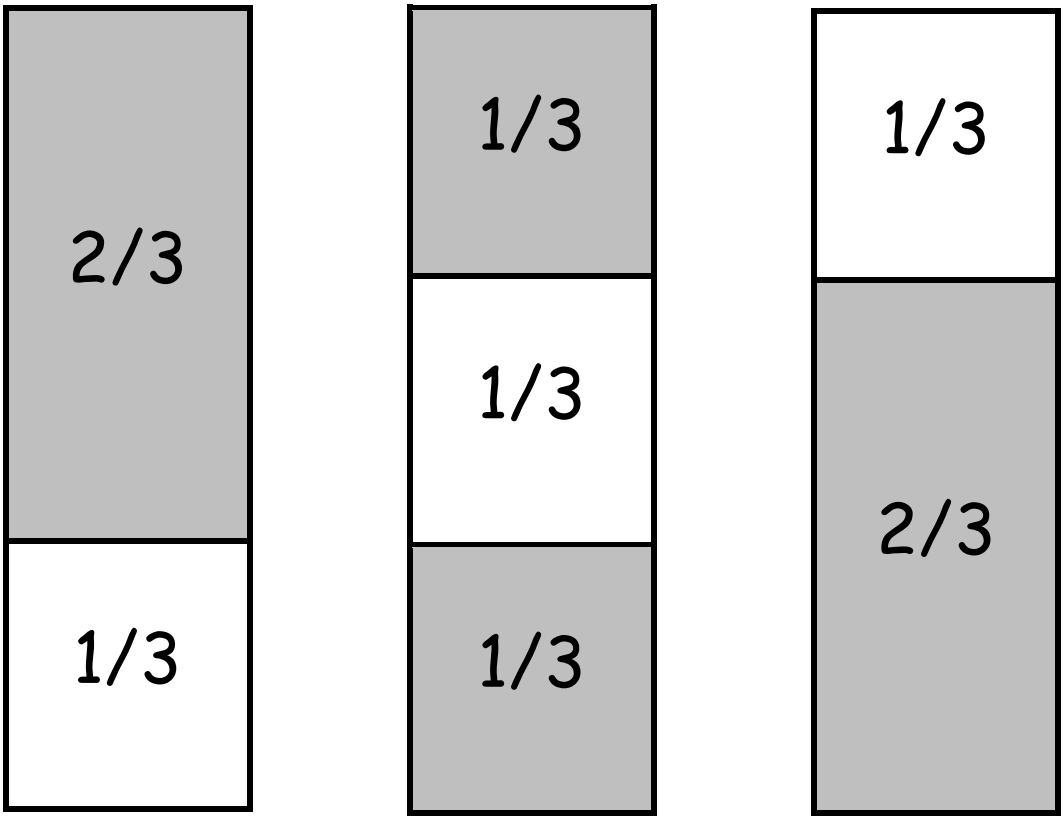
matlab>> make.

Kao rezultat javljaju se datoteke s ekstenzijom .mex32. Nakon toga slijedi treniranje podataka. Ulazni podatci preuzeti sa stranica njemačkog instituta su dodatno obrađeni programom u Matlabu koji se nalazi u privitku. U početku je postojao pokušaj pokretanja paketa na svim podatcima za C.elegansa (donorski podatci), ali tu se radi o 1700000 vektora dužine 398 i to nije bilo moguće pokrenuti u MATLABU pod Windowsima, a na Linuxu ne postoji instaliran potreban prevodilac. Pošto će to uvijek stvarati probleme idejno je riješen problem korištenjem paralelizacije podataka.

SVM-u problem stvara prevelik broj ulaznih podataka za treniranje stoga je u radu „Parallel Support Vector Machines:The Cascade SVM“ predložena podjela podataka u određeni broj vreća i treniranje svake vreće zasebno. Nakon toga je potrebno objedinjavanje rezultata dvije vreće te ponovno njihovo treniranje dok se ne dobije samo jedan izlaz. Na taj način bi bio riješen problem memorije koju MATLAB nema, no nije bilo vremena za realizaciju ideje.

Stoga su podatci obrađeni na način da se uzme svako deseti točni nasumičnim odabirom i svako stoti netočni. Time je dobivena matrica od 23037 vektora dužine 398 i matrica točnosti tih ulaznih podataka sa istim brojem vektora. Točnost može poprimiti vrijednosti 1 i -1 ovisno o tome je li odgovarajući vektor imao točno mjesto izrezivanja ili krivo. LIBSVM paket prima samo podatke tipa „double“ stoga je potrebno prebaciti zapis u datoteci koji je u obliku slova A, T, C i G – „char“ podatke u „double“.

Sada su podatci uređeni i spremni za treniranje. Sljedeći korak je podjela ulaznog seta podataka na dio za treniranje i za testiranje.



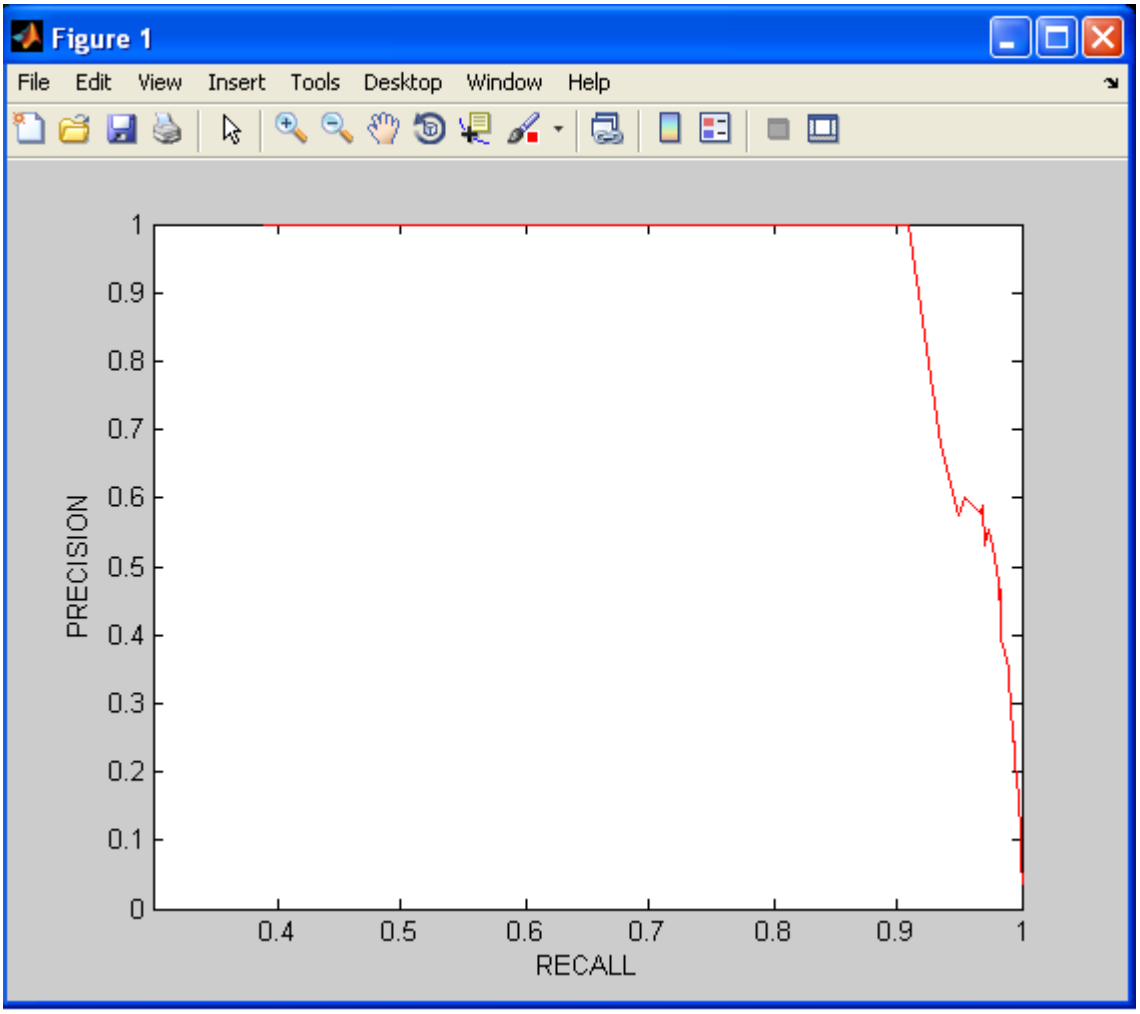
Sl.12. Metoda 3-kros validacije.

Slika 12. pokazuje na koji je način podijeljen ulazni set podataka. Tri puta je utreniran model: prvi put koristeći prve dvije trećine (sivom obojano) za treniranje, a zadnju trećinu za testiranje, drugi put koristeći prvu i zadnju trećinu za treniranje, a srednju za testiranje i zadnji put koristeći zadnje dvije trećine za treniranje i prvu za testiranje. Tako su pokrivene sve mogućnosti i rezultat daje stvarno stanje jer je sigurno korišten različit set podataka za treniranje i testiranje koji su još k tome i nasumično ispremetani.

SVM paket ima određene parametre koji mogu značajno utjecati na promjenu rezultata klasifikatora. Takav slučaj je i sa parametrom c (eng. cost). Najbolje

rješenje koje i izumitelji LIBSVM-a i znanstvenici njemačkog instituta preporučaju je pokrenuti treniranje i testiranje podataka na jako malom skupu i mijenjati c dok se ne nađe optimalno rješenje. Tada iskoristiti dobiveni c za treniranje i testiranje većeg broja podataka. Stoga je još dodatno smanjen broj ulaznih vektora sa svrhom bržeg izvođenja algoritma. Nasumično je uzet svako peti na tisuću točnih i svako peti na deset tisuća netočnih. Broj vektora je smanjen na 1145 i ponovo pokrenuta funkcija „svmtrain“ i „svmpredict“ (dio paketa LIBSVM, priložene u privitku) s postavljenim parametrom c na deset i kernelom WD ugrađenim ranije u njihovu funkciju „svm.cpp“. Rezultati su i više nego zadovoljavajući. Točnost za paketom ugrađeni prag (0.5) je 100%. Sada je bilo potrebno reprezentirati podatke grafički crtanjem P-ROC krivulje. Za druge pragove je klasifikator griješio stoga je krivulja crtana mijenjajući prag povećanjem za 0.01 od nula do jedan. Ideja P-ROC krivulje je ustanoviti koje su vrijednosti TP, FP, FN i TN, na temelju kojih bi skaliranjem i uvrštavanjem u formulu za Precision i Recall dobili vrijednosti krivulje u točkama pragova. Funkcija uspoređuje matricu točnosti sa vrijednosti koju vraća funkcija svmpredict u obliku vjerojatnosti (odnosno kojom je vjerojatnosti klasifikator glasao za,tj. da je to točno mjesto izrezivanja, te kojom je glasao protiv) . Krivulja na slici 13. je P-ROC krivulja dobivena iz 1145 ulaznih vektora dimenzije 398. Nagli padovi u funkciji pokazuju kako mijenjanjem praga točnost klasifikatora opada. Funkcija za crtanje P-ROC funkcije je dodatno skalirana jer je na početku izbačen određen broj točnih i netočnih. Kako je uzet svako peti točni na sto bilo je potrebno u formuli za Precision i za Recall pomnožiti TP (True Positives) sa 200, a pošto je uzet svako peti netočni na deset tisuća u formuli za Precision je bilo potrebno pomnožiti FP sa 2000.

$$Precision = \frac{200TP}{200TP + 2000FP} \qquad Recall = \frac{200TP}{200TP + FN} \qquad (12)$$



Sl.13. P-ROC krivulja dobivena za 1145 ulaznih vektora

5. Zaključak

Optimističan plan na početku rada na ovoj temi je bio u zadanom vremenu postići rezultate grupe znanstvenika koji su u svoj rad i istraživanje uložili godine truda i zanimanja. Taj cilj nije ostvaren, ali je napravljen dobar korak prema naprijed u odnosu na rad na Projektu i postavljeni temelji za daljnja istraživanja. Susrela sam se s problemima koji često nisu bili rješivi u kratkom vremenskom roku ili nije bilo odgovarajućih rješenja za njih. Rezultati dobiveni na kraju istraživanja pokazuju ipak značaj SVM-a te također otvaraju vrata za kombinacije raznih klasifikacijskih metoda u svrhu unaprjeđenja rezultata.

Bioinformatika je jedna od najbrže rastućih grana znanosti danas i posebnost ove teme je što je vrlo aktualna. Još ne postoje adekvatna rješenja i do napredaka na tom polju dolazi svaki dan. Upravo zbog te gladi za novim informacijama bitno je brzo djelovati. To znači pomno izabrati klasifikacijsku metodu koja daje rezultate u razumnom vremenskom roku i sa traženom točnošću. Bitno je uskladiti vlastite mogućnosti i želje.

Nakon dobivenih rezultata na malom skupu podataka smatram da metoda SVM još uvijek nije tako zastarjela iako je u zadnje vrijeme sve popularnija metoda Random Forest.

SVM se pokazuje kao optimalna metoda u odnosu na druge klasifikacijske metode zbog dva jednostavna razloga:

1. Elegantno zaobilazi problem (eng. overfitting) koji dovodi do pogrešno klasificiranih podataka izborom jedne od mnogo mogućih krivulja, na temelju maksimalne granice, koja neće dovesti do greške.
2. Također funkcija odluke sadrži samo skalarne produkte između točaka u prostoru značajki, a vektore u ulaznom prostoru. Zbog te činjenice pri izračunu granice se izbjegava reprezentiranje visoko dimenzionalnog prostora.

SVM sa definiranim kernelima pruža širok spektar mogućnosti za popravak rezultata. Neostvareni cilj je dobiti rezultate za težinski kernel sa pomakom (WDS) koji bi trebao biti bolji od običnog težinskog kernela (WD). No kako složenost kernela raste, proporcionalno raste i vrijeme izvođenja algoritma. Stoga to ostaje kao plan za „bolje“ dane.

6. Literatura

[1] Huang, Te-MIng; Kecman, Vojislav; Kopriva, Ivica: „ Kernel Based Algorithms for Mining Huge Dana Sets – Supervised, Semi-supervised, and Unsupervised Learning“,Springer, 2005.

[2] Sonnenburg, Sören; Schweikert, Gabriele; Philips, Petra; Behr, Jonas; Rätsch, Gunnar: *„Accurate splice site prediction using support vector machines“,* http://www.biomedcentral.com/content/pdf/1471-2105-8-S10-S7.pdf, 8.12.2006.

[3] Graf Peter, Hans; Cosatto, Eric; Bottou, Leon; Durdanovic, Igor; Vapnik, Vladimir: „Parallel Support Vector Machines-The Cascade SVM“ http://leon.bottou.org/publications/pdf/nips-2004c.pdf

[4] Parađina N., Analiza DNK metodama obrade signala u vremenskoj i frekvencijskoj domeni, Diplomski rad br. 1143, FER, 2008.

[5] http://en.wikipedia.org/wiki/DNA, 15.05..2009

[6] Penović M., diplomski rad, 2008.

7. Sažetak / Abstract

Predviđanje kodirajućih regija u genomu korištenjem poznatih klasifikacijskih metoda

Mjesta izrezivanja su pozicije u DNK koje odvajaju eksone, koji mogu kodirati protein, od nekodirajućih introna. U velikom broju slučajeva su sastavljeni od istih nukleotida, što omogućuje razvijanje preciznih detektora mjesta izrezivanja. Za predikciju mjesta izrezivanja klasifikator mora riješiti dva klasifikacijska problema: podjela mjesta izrezivanja na istinita i lažna i to za donore i akceptore odvojeno. U ovom radu ulazni podatci su kasificirani korištenjem metode Support Vector Machines. SVM je algoritam za nadzirano učenje razvijen u zadnjem desetljeću. Razvio ga je Vladimir Vapnik. Algoritam rješava problem podjele mjesta izrezivanja svrstavajući ih u pozitivne ili negativne razrede. Rezultati rada pokazuju nadmoć SVM-a u odnosu na druge klasifikacijske metode. Implementacija težinskog kernela (WD) je pokazala bolje rezultate u odnosu na druge kernele SVM-a.

Ključne riječi: klasifikator-nelinearni/linearni, preslikavanje, paralelizacija podataka, kros validacija, kanoničko mjesto, skalarni produkt, nadzirano učenje, P-ROC krivulja, predikcija

Prediction of coding regions in genome using known classification methods

Splice site are locations in the DNA which separate exons that may code for protein from non-coding introns. Their consensus signal is relatively strong and accurate splice site detectors thus form an important component of computational gene finders. For splice site recognition, classificator has to solve two classification problems: discriminating true from decoy splice sites for both acceptor and donor sites. In this work data sets were classified using Support Vector Machines. SVM is a supervised learning algorithm developed over the past decade by Vapnik. The algorithm describes the general problem of learning to discriminate between

positive and negative members of a given class of n-dimensional vectors. Results of this work show that SVM outperforms other classification methods. The implementation of weighted degree kernel turns out well suited for this task and shows better results in comparison to other kernels.

Key words: SVM, LIBSVM, splicing, kernel, hyperplane, overfitting, mapping, weighted degree, misclassified class point, learning machine

Privitak

Definirani težinski kerneli u programu R

WD.R

```
WD <- function(niz1,niz2)# funkcija prima dva ulazna niza-vektora
{
d<-22 # maksimalna duljina prozora
K<-784300 # izračunata konstanta za normizaciju
suma<-0

N<-length(niz1)

for (delta in 1:d) {
 zbroj<-0
 for(l in 1:(N-delta+1)) {
   zbroj <- zbroj + sum(niz2[l:l+delta-1]==niz1[l:l+delta-1]);
 }
 suma<-suma+(d-delta+1)*zbroj
}

return(suma/K)
}
```

WDS_pom.R i WDS.R

```
WDS_pom <- function(niz1,niz2)
{
d<-22
suma<-0
sigma<-0.5
lc<-201
N<-length(niz1)
for( delta in 1:d) {
   for (l in 1:(N-delta+1)) {
        for (s in 0:(sigma*abs(l-lc))) {
             if(s+l<=N && s+l+delta<=N) {
                if( niz2[l+s:(l+s+delta-1)]==niz1[l:(l+delta-1)])  {
                  suma<-suma+1
                 }
                if( niz2[l:(l+delta-1)]==niz1[l+s:(l+s+delta-1)])  {
                  suma<-suma+1
                 }
              suma<-suma/(2*(s+1))
              }
        }
    }
  tf<-d-delta+1;
  suma<-tf*suma
}
return(suma)
}
```

```
WDS <- function(niz1,niz2) #poziva se samo ova funkcija koja onda interno
poziva WDS_pom - rezultat normaliziran
{
suma<-WDS_pom(niz1,niz2)
k1<-WDS_pom(niz1,niz1)
k2<-WDS_pom(niz2,niz2)
k=suma/sqrt(k1*k2)
return(k)
}
```

Dio izmijenjenog koda funkcije svm.cpp (unutar LIBSVM paketa)

Implementacija kernela WD u C++

```
// ovaj dio je dodan - definiran je tezinski kernel - WD
double Kernel::wd(const svm_node *px, const svm_node *py)
{
    const svm_node *node_px;
    const svm_node *node_py;

    int N = 0;
    // Brojanje koliko clanova u nizu imamo
    node_px = px;
    node_py = py;
    while(node_px->index != -1 && node_py->index != -1)
    {
        ++N;
        ++node_px;
        ++node_py;
    }
    int d = 22;
    int K = 784300; // konstanta iskustveno dobivena
    long sum = 0;
    long tmp_sum = 0;// pomocna suma

    for(int delta = 0; delta < d; delta++)
    { // realizacija kernela u skladu s formulom kernela
        tmp_sum = 0;
        for(int l = 0; l < N - delta + 1; l++)
            {
                for(int index = l; index < l + delta + 1; index++)
                {
                    if((px + index)->value == (py + index)->value)
                    {
                    tmp_sum += 1;
                    }
                }
            }
        sum += (tmp_sum * (d - delta));
    }
    return (double)sum / K;
}
```

Implementacija kernela WDS u C++

```
// ovaj dio koda je dodan - definiran je tezinski kernel s pomakom - WDS
double Kernel::wds_pomocna(const svm_node *px, const svm_node *py)
{
    const svm_node *node_px;
    const svm_node *node_py;

    int N = 0;
    // Brojanje koliko clanova u nizu imamo
    node_px = px;
    node_py = py;
    while(node_px->index != -1 && node_py->index != -1)
    {
        ++N;
        ++node_px;
        ++node_py;
    }
    int d = 22;
    double sigma = 0.5;
    int lc = 201; //mjesto izrezivanja kod C.elegansa tj. pozicija prvog
člana - G tog mj. izr. - dimera GT
    double sum = 0;
    double tmp_sum = 0;
    for(int delta = 0; delta < d; delta++)
    {
        tmp_sum = 0;
        for(int l = 0; l < N - delta + 1; l++)
        {
            for (int s=0; s<= sigma*abs(l-lc);s++)
            {
              if(s+l<N)
              {
                for(int index = l; index < l+delta-1; index++)
                 {
                  if((px + index)->value != (py + index + s)->value)
                      tmp_sum+=1;

                  if((px + index + s)->value != (py + index)->value)
                      tmp_sum+=1;
                 }
                tmp_sum = tmp_sum / (2 * (s + 1));
              }
            }
        }
        sum += (tmp_sum * (d - delta));
    }
    return sum;
}
// dio koda koji obavlja normizaciju težiskog kernela s pomakom -WDS
double Kernel::wds(const svm_node *px, const svm_node *py)
{
    double suma = wds_pomocna(px, py);
    double k1 = wds_pomocna(px, px);
    double k2 = wds_pomocna(py, py);
```

```
    double k = suma / sqrt(k1 * k2);

    return k;
}
```

Gabriel don SVM 1145.m

```
 % Rezervacija prostora za:
    mat=zeros(50000,398);
    vektor=zeros(50000,1);
    index=0;

% Open file
fid = fopen('C_elegans_don_all_examples.fasta', 'r');
s=555;
rand('twister',s); % da bi svaki put bili nasumično izgenerirani isti
podatci - osigurava ponovljivost rezultata

% Read input data from the FASTA file
l=1;
for k=1:Inf % dio koda za provjeru rada - potreban zbog dužine izvođenja
programa
     if (mod(k,100000)==0)
        a={'gegam se'}
        k/100000
     end

    % Read header
    head = fgets(fid);
    if isnumeric(head), break, end % end of the file

     % Chromosome number
    ind = regexp(head, 'chr_num\s*=\s*[+-\d]', 'end');
    len = regexp(head(ind:end), '[+-]?\s*\d*', 'once', 'end');
    chr_num = str2double(head(ind:ind+len-1));

     % Strand
    ind = regexp(head, 'strand\s*=\s*[+-]', 'end');
    strand = head(ind);

    % Position
    ind = regexp(head, 'position\s*=\s*[+-\d]', 'end');
    len = regexp(head(ind:end), '[+-]?\s*\d*', 'once', 'end');
    position = str2double(head(ind:ind+len-1));

    % Label
    ind = regexp(head, 'label\s*=\s*[+-\d]', 'end');
    len = regexp(head(ind:end), '[+-]?\s*\d*', 'once', 'end');
    label = str2double(head(ind:ind+len-1));

    % Read data
    data = fgets(fid);
    if isnumeric(head), break, end % end of the file
```

```
    % Do something
    if (data(201) == 'G' && data(202) == 'T') % ispituje da li se na
predvidjenom mjestu izrezivanja nalaze G i T

        if (label==1 && rand(1,1)>0.995) % uzima svako peti točni na 1000
            index=index+1;
            mat(index,1:398)=double(data(1:398)); % LIBSVM paket radi samo
s podatcima tipa double pa char pretvaramo u double

            vektor(index,1)=1; % ako je bio točan u matricu labela
postavlja 1
        else
            if (rand(1,1)>0.9995) % uzima svako peti na 10000 netočnih
                index=index+1;
                mat(index,1:398)=double(data(1:398));
                vektor(index,1)=-1; % ako je bio netočan u matricu labela
postavlja -1
            end
        end
    end
end
fclose(fid);
% rješavanje viška početnog prostora rezerviranog
mat(index+1:end,:)=[];
vektor(index+1:end,:)=[];
save SVMdon_1145 mat vektor
```

Metoda 3-kros validacije realizirana u funkciji za 1145 vektora:

```
mat1_train=mat(1:381+382,:);%podjela na 3 dijela za treniranje i testiranje
od kojih se za treniranje uvijek uzima 2/3 od ukupnog broja matrice a za
testiranje 1/3
vektor1_train=vektor(1:381+382,:);
mat1_test=mat(381+382+1:end,:);
vektor1_test=vektor(381+382+1:end,:);

mat2_train=mat([1:381 381+382+1:end],:);
vektor2_train=vektor([1:381 381+382+1:end],:);
mat2_test=mat(1+381:381+382,:);
vektor2_test=vektor(1+381:381+382,:);

mat3_train=mat(382+1:end,:);
vektor3_train=vektor(382+1:end,:);
mat3_test=mat(1:382,:);
vektor3_test=vektor(1:382,:);


WD1= svmtrain(vektor1_train, mat1_train, '-c 10 -b 1 -t 5'); % bitno biti
pozicioniran unutar LIBSVM paketa. WD kernel je definiran na 5. mjestu zato
poziv -t 5
```

```
[predict_label_1, accuracy_1, prob_estimates_1] = svmpredict(vektor1_test,
mat1_test, WD1, '-b 1'); % bez poziva -b 1 ne bi bilo moguće kasnije
krivulju nacrtat - u njemu su nam sačuvane vjerojatnosti
save WD1 predict_label_1 accuracy_1 prob_estimates_1

WD2= svmtrain(vektor2_train, mat2_train, '-c 10 -b 1 -t 5');
[predict_label_2, accuracy_2, prob_estimates_2] = svmpredict(vektor2_test,
mat2_test, WD2, '-b 1');
save WD2 predict_label_2 accuracy_2 prob_estimates_2

WD3= svmtrain(vektor3_train, mat3_train, '-c 10 -b 1 -t 5');
[predict_label_3, accuracy_3, prob_estimates_3] = svmpredict(vektor3_test,
mat3_test, WD3, '-b 1');
save WD3 predict_label_3 accuracy_3 prob_estimates_3

prob_estimates=[prob_estimates_3;prob_estimates_2;prob_estimates1];%
ujedinjavanje svih vjeorojatnosti potrebnih za crtanje P-ROC krivulje
save WD vektor prob_estimates
```

Funkcija za crtanje P-ROC krivulje za 1145 vektora:

```
br_redova=length(vektor);
mat=zeros(br_redova,1);
prag=0; precision=0; recall=0;
for br=1:100 % mijenjanje praga od 0 do 1 s korakom od 0.01
    sumTP=0; sumTN=0; sumFP=0; sumFN=0;
    for i=1:br_redova
            if(prob_estimates(i,2)>prag) % 2. stupac matrice prob_estimates
sadrzi vjerojatnost da je to bilo tocno mjesto izrezivanja
                mat(i,1)=1;
            else
                mat(i,1)=-1;
            end
             if(mat(i,1)==1 && vektor(i,1)==1)% usporedjujemo stvarno stanje
i predikciju klasifikatora
                 sumTP=sumTP+1;
             end
             if(mat(i,1)==-1 && vektor(i,1)==1)
                 sumFN=sumFN+1;
             end
             if(mat(i,1)==1 && vektor(i,1)==-1)
                 sumFP=sumFP+1;
             end
             if(mat(i,1)==-1 && vektor(i,1)==-1)
                 sumTN=sumTN+1;
             end
    end
    precision(br)=(sumTP*200)/(sumTP*200+2000*sumFP); % skaliranje
    recall(br)=(sumTP*200)/(sumTP*200+sumFN);
    prag=prag+0.01;
end

% crtanje precision-recall krivulje
```

```
plot(recall,precision,'r')
xlabel('RECALL');
ylabel('PRECISION');
```