SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1045

PREDVIĐANJE MJESTA PROTEINSKIH INTERAKCIJA IZ SEKVENCE AMINOKISELINA

Darijo Šplihal

Zagreb, rujan 2006.

Zahvaljujem se mr. sc. Mile Šikiću
na poticanju i usmjeravanju,
dečkima sa LSS-a na svevremenskoj pomoći.

Zahvaljujem se svojoj obitelji: mami i tati na razumijevanju i potpori,
te svim svojim prijateljima na tome što su to što jesu.

SADRŽAJ

1 Uvod

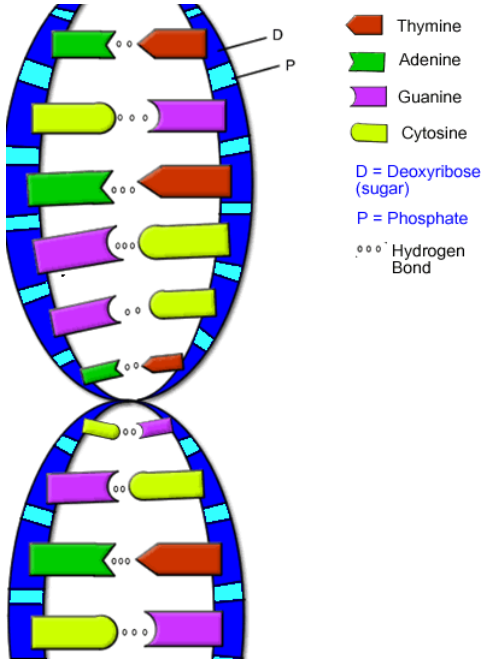
Pronalaženje mjesta proteinskih interakcija te detekcija specifičnih sekvenci aminokiselina koje su karakteristične za svako takvo mjesto je vrlo značajna tematika i u današnje vrijeme još uvijek predstavlja svojevrsni problem u istraživanjima. Ovaj problem posebno je značajan u razvoju novih lijekova i cjepiva, analizi metaboličkih reakcija, promatranju i praćenju razvoja organizma te u mnogim drugim aplikacijama. Zbog same činjenice da je broj eksperimentalno pronađenih i definiranih proteinskih kompleksa relativno mali, metode pronalaženja i prepoznavanja aminokiselina koje sudjeluju u proteinskim interakcijama postaje vrlo važno i značajno. Ovaj rad napisan je i osmišljen sa naglaskom na slijedeće pitanje: s obzirom da je poznato da protein sudjeluje u reakciji sa drugim proteinom, dali je moguće predvidjeti koje se aminokiseline nalaze na mjestima koja sudjeluju interakciji?

Rad se sastoji od više cjelina. Sustavno prolazeći kroz njih daje se opis implementiranih metoda za računalno učenje i predviđanje. Koriste se PARF, algoritam slučajnih šuma (engl. Random Forest), i LibSVM, algoritam potpornih vektora (engl. Support Vector Machine).

U poglavlju „2 Proteini“dana je teoretska osnova o samim proteinima, u poglavlju „3 Metode“ opisana je teoretska osnova metoda koje se primjenjuju. Unutar poglavlja „4 Podaci“ opisani su podaci te način na koji se generiraju, dok je u poglavlju „5 Rješenje“ dano rješenje implementiranog sustava. Unutar poglavlja „6 Rezultati“ dani su postupci i rezultati testiranja metoda. Sažetak rada te zaključak dani su u poglavljima „7 Sažetak“ i „8 Zaključak“. Poglavlje „9 Diskusija“ sadrži diskusiju dobivenih rezultata. Popis literature korištene pri izradi ovog rada dan je u poglavlju „10 Literatura“. Unutar poglavlja „11. Dodatak“ objašnjen je rad sa software-om.

2 Proteini

Molekula DNA sastoji se od četiri abecedna elementa, poimence A, T, C i G. Slova A, T, C, G predstavljaju molekule koje se zovu nekleotidi ili baze. Kombiniranjem ova četiri slova, slika 1, u DNA je zapisana genetička informacija bitna za sva živa bića.



Slika 1 DNA i nukleotidi

Informacija koja se prenosi u DNA bazama prevodi se u proteine. Molekula DNA kopira se u drugi oblik nukleinske kiseline – RNA ili ribonukleinska kiselina. RNA se dalje premješta u ribosome, strukture zadužene za izradu proteina. Svaki set od tri baze u RNA određuje koja će se aminokiselina dodati u molekulu proteina koja se trenutno izrađuje. Lanac RNA prolazi kroz ribosome sve dok protein ne bude kompletan.

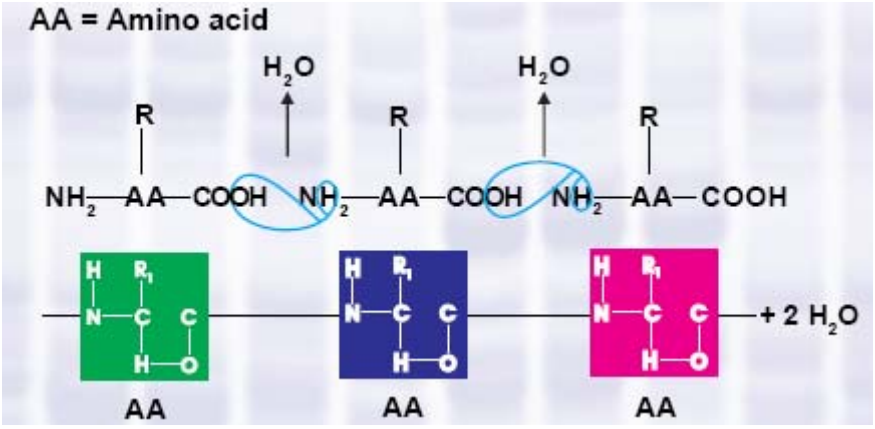
Ovaj proces se naziva „glavna dogma“ jer je to zapravo osnova biološkog života. Aminokiseline su bifunkcionalne organske čestice koje se sastoje od amino-skupine ($-NH_2$) i kiselinske karboksilne skupine (-COOH). Proteini, sastoje se obično od 20 različitih aminokiselina, variraju po svojoj funkciji i

svojstvima ovisno o prirodi njihove R-skupine. Primjerice, kod alanina, R-skupina je (-CH_3), dok je, kod cisteina, (-CH_2-SH).

2.1 Primarna struktura proteina

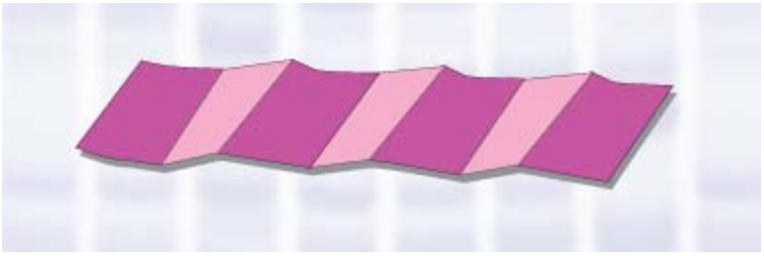
Unutar proteina aminokiseline su povezane peptidnim vezama koje tvore kralježnicu molekule. Peptidna veza stvara se između amino skupine (-NH_2) jedne aminokiseline i kiselinske karboksilne skupine (-COOH) druge aminokiseline. Općenita formula aminokiseline je NH_2-CHR-COOH. R skupina može biti bilo od atoma do kompleksne molekule. Termin polipeptid jednostavno označava dugačak lanac aminokiselina.

Jednom kada je proteinski lanac oformljen mora se svinuti na točno određeni način da bi mogao vršiti svoju zadaću. Struktura i način savijanja svakog proteina je posebna. Kako aminokiseline utječu na način savijanja proteina nije još potpuno razjašnjeno. Protein se može sastojati od jednog ili više polipeptida. Vidljivo je da su svojstva polipeptida i proteina jako ovisna o redoslijedu aminokiselina.



Slika 2 Primarna struktura proteina

2.2 Sekundarna struktura



Slika 3 Sekundarna struktura proteina

Sekundarna struktura rezultat je lokalnih hidrogenskih veza koje se stvaraju duž kralježnice polipeptida. Ove veze daju proteinu snagu i fleksibilnost. Najčešće strukture su:

- Alfa-helix, uzrokovan hidrogenskim vezama unutar polipetidnog lanca
- Beta-helix, uzrokovan hidrogenskim vezama između susjednih polipeptidnih lanaca

2.3 Tercijarna struktura

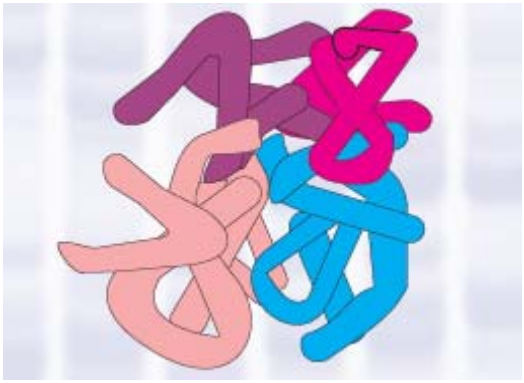
Slika 4 Tercijarna struktura proteina

Tercijarna struktura rezultat je vezivanja R-skupina unutar polipeptida, tipični su primjeri slabe nekovalentne veze (hidrogene, ionske, hidrofobne veze) i jake kovalentne veze (disulfidne veze između cisteinskih R-skupina).

2.4 Kvartarna struktura

Ponekad je dovoljan samo jedan polipeptid da bi protein mogao vršiti svoju ulogu, tada govorimo o proteinu koji se ponaša kao monomer. Međutim, često je slučaj da je potrebno dva ili više polipeptida da interagiraju kako bi protein mogao vršiti svoju ulogu. U ovom slučaju govorimo o dimeru, odnosno multimeru.

Kvartarna struktura je rezultat interakcija između dva ili više polipeptidnih lanaca koje stvaraju dimere, trimere, mutimere. Cijelu strukturu drže hidrogenske veze, iosnke veze i rjeđe hidrofobne veze i inter-lančane disulfidne veze.



Slika 5 Kvartarna struktura proteina

2.5 Funkcije proteina

Trodimenzionalna struktura proteina je direktno uzrokovana interakcijama unutar samog proteina. Ovo saznanje ima pak za posljedicu da nam struktura proteina govori puno o njegovoj ulozi u stanici.

Primjerice u vodenom okruženju hidrofobne R-skupine se okreću prema unutrašnjosti proteina. Promjene u temperaturi ili kiselosti okruženja mogu utjecati na nekovalentne veze, uzrokujući promjene u trodimenzionalnoj strukturi te konačno i prestanku odvijanja zadaće proteina. Ovaj proces se naziva denaturaizacija. Denaturirani proteini mogu se skupiti u grudice i postati netopivi u procesu koji se naziva koagulacija.

3 Metode

3.1 Metoda slučajne šume (Random Forest)

Slučajna šuma[7][8], kasnije RF, je općeniti naziv za skupinu metoda koje se koriste stablastim klasifikatorima $\{h(x,\Theta_k), k=1,....,\}$ gdje je $\{\Theta_k\}$ skup jednoliko distribuiranih, međusobno potpuno neovisnih, vektora, a x ulazni vektorski uzorak. Prilikom treniranja, RF algoritam stvara velik broj stabala, od kojih se svako trenira na određenom broju uzoraka originalnog trening seta, i vrši pretragu samo po slučajno generiranom podskupu ulaznih varijabli kako bi odredio mjesto na kojem će se razgranati. Za klasifikaciju svako stablo unutar RF daje glas jednoj od klasa unutar skupa x. Izlaz klasifikatora ovisi o broju glasova stabala svakoj pojedinoj klasi.

3.1.1 Kako algoritam slučajne šume radi

Trening set za svako stablo stvara se tako da se iz podataka za treniranje uzme određeni broj instanci ali na slučajan način. Iz tako stvorenog seta za treniranje stabla, jedna trećina instanci se odvaja. Ove instance se nazivaju oob instance (*out of bag*) i koriste se za dobivanje nepristrane procjene greške klasifikacije. Također se koriste i za procjenu važnosti pojedinih varijabli ulaznih instanci. Kako se stablo stvara sve instance se puštaju duž stabla te se računaju njihove međusobne sličnosti. Ako se dvije instance nađu u istom konačnom čvoru njihova se međusobna sličnost povećava za 1. Kada se sve instance provuku kroz stablo, sličnosti se normiraju tako da se podjele sa brojem stabala.

Kod slučajne šume nema potrebe za krosvalidacijom ili korištenjem posebnog seta za testiranje kako bi se dobila nepristrana procjena greške uzoraka za testiranje. Svako stablo se stvara tako da se koristi podskup iz početnih podataka za učenje koji se naziva *bootstrap* podskup. Otprilike jedna trećina instanci se izostavlja iz *bootstrap* podskupa i ne koriste se pri izradi k-tog stabla. Sada svaki uzorak izostavljen pri stvaranju k-tog stabla, oob instance, treba pustiti niz k-to stablo da bi se dobila klasifikacija. Na ovaj način dobiva se klasifikacija testnog seta za svaku instancu u jednoj trećini svih stabala. Nakon

završene obrade, neka je klasa j klasa koja je dobila najviše glasova svaki put kad je instanca n bila oob instanca. Omjer broja izlaza kada j nije bila jednaka pravoj klasi instance n s obzirom na sve instance naziva se procjena pogreške oob-a.

U svakom stablu stvorenom u šumi zanemaruju se oob instance i zbrajaju se glasovi koji su ispravno doneseni s obzirom na klasu. Sada se na slučajan način permutiraju vrijednosti varijable m u oob instancama i te se instance puštaju niz stablo. Oduzima se broj glasova za ispravnu klasu oob instanci sa permutiranom m varijablom od broja glasova za ispravnu klasu ne upotrijebljenih oob instanci. Srednja vrijednost dobivene razlike u svim stablima unutar šume naziva se važnost varijable m. Ako je broj varijabli jako velik, znači svaka instanca se sastoji od većeg broja varijabli, moguće je obaviti klasifikaciju sa svim varijablama, pa opet ponoviti postupak samo sa najbitnijim varijablama.

3.1.2 Slučajne šume konvergiraju

Uz skup klasifikatora $h_1(x), h_2(x),\ldots, h_K(x)$ te trening setom stvorenim od nasumce izabranih instanci iz distribucije slučajnih vektora *Y, X*, funkcija margine glasi:

$$mg(X,Y) = av_k I\left(h_k(X) = Y\right) - \max_{j \neq Y} av_k I\left(h_k(X) = j\right)$$

gdje I() je indikatorska funkcija. Margina opisuje broj za koliko glasova prosječan broja glasova za X,Y za ispravnu klasu nadmašuje prosječan broj glasova za bilo koju drugu klasu. Što je veća margina to je klasifikator točniji. Greška prilikom generalizacije zapisuje se kao

$$PE^* = P_{X,Y}\left(mg(X,Y) < 0\right)$$

Gdje ovi mali X i Y označavaju činjenicu da se vjerojatnost računa u X, Y prostoru. Kod slučajnih šuma $h_k(X) = h(X, \Theta_k)$. Kako se broj stabala povećava za gotovo sve sekvence $\Theta_1,\ldots$. PE^* konvergira prema

$$P_{X,Y}\left(P_{\Theta}\left(h\left(X,\Theta\right)=Y\right)-\max_{j\neq Y}P_{\Theta}\left(h\left(X,\Theta\right)=j\right)<0\right)$$

Ovaj rezultat objašnjava zašto slučajne šume ne generaliziraju što se više stabala dodaje šumi.

3.1.3 Snaga i korelacija

Kod slučajnih šuma postoje dva parametra koji definiraju gornju granicu greške generalizacije. Oni ujedno služe kao mjera točnosti klasifikatora te njihove međusobne ovisnosti.

Funkcija margine za slučajnu šumu glasi

$$mr\left(X,Y\right)=P_{\Theta}\left(h\left(X,\Theta\right)=Y\right)-\max_{j\neq Y}P_{\Theta}\left(h\left(X,\Theta\right)=j\right)$$

a snaga skupa klasifikatora $\left\{h\left(x,\Theta\right)\right\}$ je

$$s=E_{X,Y}mr\left(X,Y\right)$$

Uz pretpostavku da je s ≥ 0, Čebiševljeva nejednakost glasi

$$PE^{*}\leq\frac{\mathrm{var}\left(mr\right)}{s^{2}}$$

Izraz koji još bolje opisuje devijaciju *mr* može se izvesti iz slijedećeg. Neka je

$$\hat{j}\left(X,Y\right)=\arg\max_{j\neq Y}P_{\Theta}\left(h\left(X,\Theta\right)=j\right)$$

dalje možemo pisati

$$\begin{aligned}mr\left(X,Y\right)&=P_{\Theta}\left(h\left(X,\Theta\right)=Y\right)-P_{\Theta}\left(h\left(X,\Theta\right)=\hat{j}\left(X,Y\right)\right)\\&=E_{\Theta}\left[I\left(h\left(X,\Theta\right)=Y\right)-I\left(h\left(X,\Theta\right)\right)=\hat{j}\left(X,Y\right)\right]\end{aligned}$$

Funkcija neobrađene margine glasi

$$rmg(\Theta, X, Y) = I\left(h(X, \Theta) = Y\right) - I\left(h(X, \Theta) = \hat{j}(X, Y)\right)$$

Potrebno je primijetiti da je mr(X,Y) očekivanje rmg(Θ,X,Y) s obzirom na Θ. Za svaku funkciju f identitet

$$\left[E_{\Theta} f(\Theta)\right]^2 = E_{\Theta,\Theta'} f(\Theta) f(\Theta')$$

Postoji ako Θ i Θ' su međusobno neovisne ali sa istom distribucijom. Dalje slijedi

$$mr(X, Y)^2 = E_{\Theta,\Theta'} rmg(\Theta, X, Y) rmg(\Theta', X, Y)$$

Koristeći gornji izraz možemo pisati

$$\begin{aligned} \operatorname{var}(mr) &= E_{\Theta,\Theta'}\left(\operatorname{cov}_{X,Y} rmg(\Theta, X, Y) rmg(\Theta', X, Y)\right) \\ &= E_{\Theta,\Theta'}\left(\rho(\Theta, \Theta') sd(\Theta) sd(\Theta')\right) \end{aligned}$$

Gdje ρ(Θ, Θ') predstavlja korelaciju između rmg(Θ, X, Y) i rmg(Θ', X, Y) uz fiksne Θ i Θ', a sd(Θ) standardna devijacija rmg(Θ, X, Y) uz fiksni Θ. Dalje je

$$\begin{aligned} \operatorname{var}(mr) &= \bar{\rho}\left(E_{\Theta} sd(\Theta)\right)^2 \\ &\leq \bar{\rho} E_{\Theta} \operatorname{var}(\Theta) \end{aligned}$$

Gdje $\bar{\rho}$ predstavlja srednju vrijednost korelacije, i to

$$\bar{\rho} = \frac{E_{\Theta,\Theta'}\left(\rho(\Theta, \Theta') sd(\Theta) sd(\Theta')\right)}{E_{\Theta,\Theta'}\left(sd(\Theta) sd(\Theta')\right)}$$

Gornja granica generalizacijske pogreške glasi:

$$PE^* \le \frac{\bar{\rho}\left(1-s^2\right)}{s^2}$$

Ovaj izraz pokazuje kako dva osnovna elementa pogreške generalizacije slučajnih šuma su snaga pojedinog klasifikatora unutar šume i korelacija među njima. Izraz c/s^2 predstavlja omjer korelacije i kvadrata snage. Da bi se shvatio način na koji slučajne šume funkcioniraju ovaj omjer biti će jako koristan, što je on manji to bolje.

Omjer c/s^2 za slučajnu šumu definiran je kao

$$\frac{c}{s^2} = \frac{\bar{\rho}}{s^2}$$

U slučaju da ulazni podaci imaju samo dvije klase dolazimo pojednostavljenja. Funkcija margine glasila bi

$$mr\left(X,Y\right) = 2P_{\Theta}\left(h\left(X,\Theta\right)=Y\right)-1$$

Neobrađena margina izgleda $2I\left(h\left(X,\Theta\right)=Y\right)-1$ a korelacija $\bar{\rho}$ je između $I\left(h\left(X,\Theta\right)=Y\right)$ i $I\left(h\left(X,\Theta'\right)=Y\right)$. U slučaju da su vrijednosti Y +1 i -1 slijedi

$$\bar{\rho} = E_{\Theta,\Theta'}\left[\rho\left(h\left(\cdot,\Theta\right),h\left(\cdot,\Theta\right)\right)\right]$$

3.1.4 Korištenje oob instanci za praćenje pogreške, snage i korelacije

U slučajnim šumama koristi se odvajanje (*bagging*) u tandemu sa odabiranjem slučajne varijable. Svaki novi trening set stabla zapravo je podskup iz ulaznog trening seta slučajne šume, koji se vraća natrag te se uzima slijedeći. Stablo se grana na osnovu navedenog podskupa koristeći nasumično odabiranje varijable.

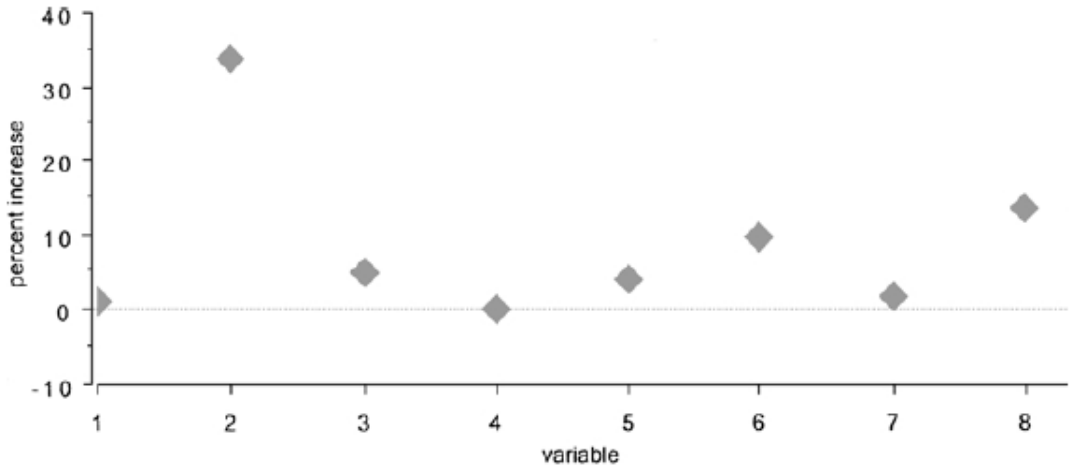
Postoje dva razloga zašto se upotrebljava odvajanje (*bagging*). Prvi je taj da odvajanje povećava točnost kada se varijable uzimaju nasumce. Drugi razlog je taj da odvajanje omogućuje prikazivanje pogreške generalizacije (PE*) skupa stabala, te ocjene snage i korelacije.

Ako imamo trening set T, konstruiramo klasifikatore $h(x, T_K)$ iz podskupa T_K trening seta T te ih puštamo da rade i donose glasove kako bi smo dobili odvajajući (*bagged*) prediktor. Za svaki y, x u trening setu pamte se samo glasovi klasifikatora T_K koji nisu vidjeli y, x. Ovo se naziva odvajajući (*out-of-bag*) klasifikator. Tada je oob ocjena generalizacijske greške zapravo greška oob klasifikatora na testnom setu.

U svakom podskupu trening seta (*bootstrap*) oko jedne trećine instanci se izostavlja. Na osnovu toga oob ocijene se temelje na zbroju ocjena jedne trećine svih klasifikatora. Treba primijetiti da za razliku od krosvalidacije kod koje ocjene u određenom postotku ovise jedna o drugoj, kod oob ocjena te međuovisnosti nema. Snaga i korelacija također mogu biti ocjenjene koristeći oob metode. Ovo daje interne ocjene koje mogu pomoći u razumijevanju klasifikacijske točnosti i kako ju poboljšati.

3.1.5 Mehanizam slučajnih šuma

Pretpostavimo da imamo M ulaznih varijabli. Nakon što je svako stablo konstruirano, vrijednosti m-te varijable u oob instancama se permutiraju na slučajan način i oob instance se puštaju niz dotična stabla. Rezultat klasifikacije za svaki x_n koji je oob se sprema. Ovaj postupak se ponavlja za m = 1, 2, …, M. Po završetku postupka uspoređuju se oob glasovi za klasu x_n sa unesenim šumom na m-toj varijabli sa stvarnim klasama x_n da bi se pronašla veličina greške klasifikacije.

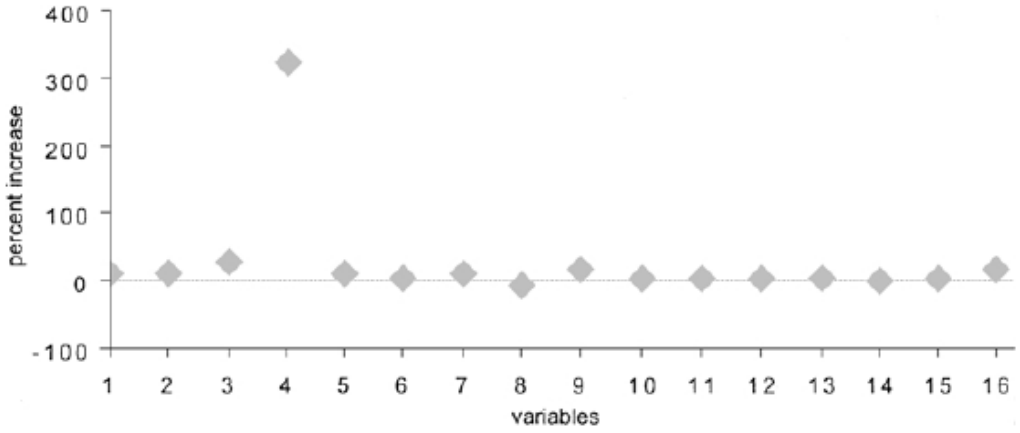


Slika 6 Mjera važnosti varijabli

Izlaz je postotno povećanje u pogrešnoj klasifikaciji. Ove ocjene dobivaju se na jednom stvaranju šume od 1000 stabala i bez testnog seta.

Može se primijetiti na slici 6 da kako je važnost pojedine varijable veća tako ona više odstupa od ostalih, odnosno unošenjem šuma u tu varijablu jako povećava pogrešku.

Vrlo je zanimljiv primjer glasovanja. U ovoj analizi ulazni set se sastoji od 435 instanci koje predstavljaju 435 kongresnika i 16 varijabli koje predstavljaju njihove glasove, da-ne, na 16 različitih pitanja. Varijabla klase može biti republikanac ili demokrat. Kako bi uvidjeli koja pitanja su najbitnija opet je generirano 1000 stabla sa šumom u varijablama. Najmanja pogreška na izvornim podacima, bez unosa šuma, dobivena je kada je za grananje upotrebljavano 5 varijabli. Rezultati su na slici 7.



Slika 7 Glasanje kongresnika

Može se primijetiti da varijabla 4 jako iskače, pogreška se utrostručuje ako se unese šum u varijablu 4. Šuma je ponovno konstruirana upotrebljavajući samo varijablu 4. Pogreška testa bila je 4.3%, gotovo ista kao da su sve varijable korištene. Rezultat glasovanja samo na 4. pitanje razdvaja republikance od demokrata gotovo jednako dobro kao i rezultat kada se promatraju svi glasovi.

3.2 Metoda potpornih vektora (Support Vector Machine)

3.2.1 Predstavljanje podataka i sličnosti

Jedan od osnovnih problema teorije učenja je slijedeći. Pretpostavimo da postoje dvije klase objekata. Dodijeljen je novi objekt i potrebno ga je staviti u jednu od dvije klase. Ovaj problem može se ovako formulirati: dani su osnovni podaci

$$(x_1, y_1), \ldots, (x_m, y_m) \in \mathrm{X} \times \{\pm 1\}$$

Ovdje je X ne prazan skup odakle su uzeti uzorci x_i (još se nazivaju i ulazi, slučajevi, elementi, uzorci) često se naziva i domena, a y_i su izlazi. Treba primijetiti da postoje samo dvije klase uzoraka. Zbog lakšeg matematičkog razumijevanja klase su obilježene sa 1 i -1. Ovo je ujedno i najjednostavniji slučaj koji ima naziv binarno prepoznavanje uzoraka.

Ulazi mogu biti bilo što pošto za *X* nije ništa definirano i poznato je samo da je to skup. Na primjer zadatak bi mogao glasiti da se ovce podjele u dvije skupine, a u tom slučaju ulazi x_i bile bi ovce.

Da bi se objasni problem učenja potrebno je uvesti još jedan element u do sad opisan set. Potrebno je razvrstati i elemente za koje se ne zna kojoj skupini ili klasi pripadaju. Kod prepoznavanju uzoraka to bi značilo da ako se uzme neki novi ulaz $x \in X$ potrebno je znati odrediti vrijednost njegova izlaza, odnosno pripadni $y \in \{\pm 1\}$. To u biti znači da se odabire *y* takav da se uređeni par (*x, y*) nekako može poistovjetiti sa primjerima koji su služili za trening (učenje).

Određivanje mjere sličnosti izlaza $\{\pm 1\}$ je lako jer u binarnoj klasifikaciji mogu biti samo dvije vrijednosti, ili je izlaz isti sa primjerima za treniranje ili nije.

Mjera sličnosti opisana je kao

$$k : X \times X \to \mathbb{R}$$
$$(x, x') \mapsto k(x, x')$$

funkcija koja na temelju ulaza *x* i *x′* kao izlaznu vrijednost vraća broj koji opisuje mjeru sličnosti ta dva ulaza. Pretpostavit će se da je *k* simetrična, odnosno *k(x, x')= k(x', x)* za sve $x, x' \in \mathrm{X}$. Funkcija *k* naziva se jezgrom (*kernel*).

Mjere sličnosti, gledano općenito, su dosta složene te će se krenuti od najjednostavnijeg primjera i postupno problem generalizirati za općeniti slučaj. Jedan od jednostavnijih primjera mjere sličnosti je skalarni produkt (*dot product*). Na primjer, ako postoje dva vektora $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^N$, skalarni produkt se definira kao:

$$\langle \mathbf{x}, \mathbf{x}' \rangle := \sum_{i=1}^{N} [\mathbf{x}]_i [\mathbf{x}']_i$$

gdje $[\mathbf{x}]_i$ označava i-tu komponentu vektora *x*. Treba napomenuti da se skalarni produkt još naziva i unutrašnji produkt (*inner product*) te da se negdje

upotrebljavaju i okrugle zagrade i znak za umnožak $(\mathbf{x} \cdot \mathbf{x}')$. Ovdje se sada može lako uočiti odakle naziv *dot product*.

Geometrijska interpretacija skalarnog produkta daje broj koji je jednak kosinusu kuta između vektora ***x*** i ***x'***, pod uvjetom da su normalizirani na duljinu 1. On također omogućava izračunavanje duljine vektora ***x*,** ili norme, kao:

$$\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$$

Na sličan način moguće je odrediti udaljenost između dva vektora. Može se primijetiti da korištenjem skalarnog produkta se mogu izvršiti sve geometrijske operacije koje koriste kutove, duljine i udaljenosti. No naravno postoje i složene operacije koje skalarnim produktom nije moguće izvesti.

Na početku se namjerno prešlo preko pretpostavke dali ulazi uopće mogu činiti skalarni produkt, mogli su biti bilo što. Stoga da bi se mogao uopće koristiti skalarni produkt kao mjera sličnosti potrebno je ulaze predstaviti kao vektore u nekom prostoru H u kojem je skalarni produkt definiran. Zato se koristi funkcija:

$$\Phi : \mathrm{X} \to \mathrm{H}$$
$$x \mapsto \mathbf{x} := \Phi(x)$$

Čak i ako početni uzorci postoje u prostoru u kojem je skalarni produkt definiran svejedno će biti potrebna još općenitija mjera sličnosti koja će se odrediti primjenjujući gornju funkciju (MAP). U tom slučaju Φ će biti nelinearna mapa.

Prostor *H* naziva se i *feature space*. Uočite da se koristi podebljani ***x*** koji označava vektorsku reprezentaciju *x* u *feature space*. Sve u svemu, prebacivanje podataka u *H* prostor funkcijom Φ ima slijedeća svojstva:

- omogućava određivanje mjere sličnosti na temelju skalarnog produkta u *H*,

$$k(x, x') := \langle \mathbf{x}, \mathbf{x}' \rangle = \langle \Phi(x), \Phi(x') \rangle$$

- omogućava da se ulazima bavi u geometrijskom smislu, da se bavi proučavanjem algoritama za učenje pomoću linearne algebre i analitičke geometrije.
- mogućnost odabira funkcije Φ omogućit će da se stvori velik broj različitih mjera sličnosti i algoritama za učenje. Ovo se odnosi i na situacije gdje ulaz x_i već postoji u prostoru gdje je skalarni produkt definiran. U tom slučaju može se skalarni produkt direktno koristiti kao mjera sličnosti. Međutim, ništa ne sprječava da se prvo ne primijeni funkcija Φ te se ulazi prebace u neki prostor u kojem će biti lakše riješiti problem.

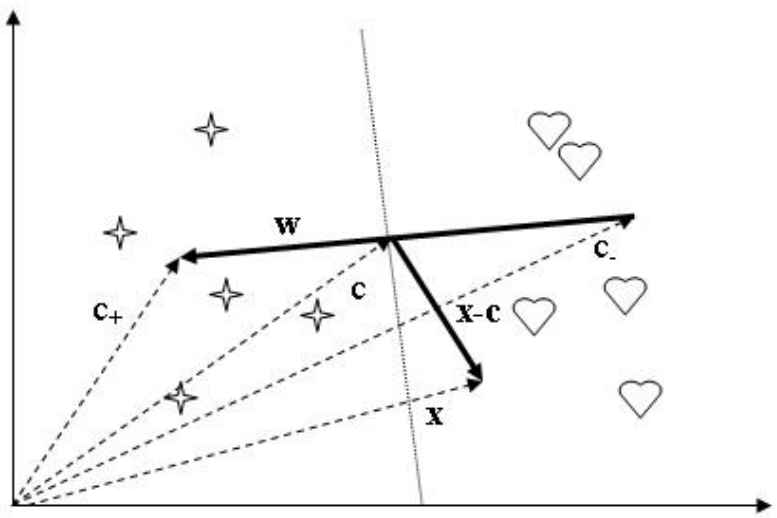
3.2.2 Jedan jednostavan algoritam za prepoznavanje uzoraka

Na osnovu do sada objašnjenog može se opisati jedan od najjednostavnijih algoritama za prepoznavanje uzoraka. Pretpostavit će se da su ulazi definirani u prostoru *H*, te korištenjem skalarnog produkta mogu se mjeriti međusobne udaljenosti. Osnovna ideja algoritma je da novi podatak na osnovu njegove srednje vrijednosti svrsta u klasu (razred) koji ima srednje udaljenosti jako bliske njegovoj. Neka se počne tako da se izračunaju srednje vrijednosti dviju klasa u prostoru *H*:

$$c_+ = \frac{1}{m_+} \sum_{\{i | y_i = +1\}} \mathbf{x}_i$$

$$c_- = \frac{1}{m_-} \sum_{\{i | y_i = -1\}} \mathbf{x}_i$$

gdje m_+ i m_- predstavljaju broj pozitivnih i negativnih primjera. Pretpostavlja se da su obje klase ne prazne te $m_+, m_- > 0$. Novi ulaz $\boldsymbol{x}$ dodjeljuje se klasi čija je srednja vrijednost najbliža srednjoj vrijednosti ulaza (Slika 8).



Slika 8 Jednostavan geometrijski algoritam: sadrži dvije klase uzoraka (srca i zvjezdice), računa srednju vrijednost klasa c_+ i c_- te testni uzorak x dodjeljuje jednoj od klasa na temelju njegove srednje vrijednosti. To se može napraviti ako promatramo skalarni produkt između x-c ($c=(c_+ + c_-)/2$) i $w:= c_+ - c_-$ koji mijenja predznak kako pripadni kut pređe preko π/2. Primijetite da pripadna granica podjele je hiperravnina (točkasta linija) okomita na w.

Ovo geometrijsko rješenje može biti formulirano u obliku skalarnog produkta $\langle \cdot, \cdot \rangle$. Između $\mathbf{c}_+$ i $\mathbf{c}_-$ nalazi se $\mathbf{c} := (\mathbf{c}_+ + \mathbf{c}_-)/2$. Klasa ulaza ***x*** određuje se tako da se gleda dali vektor ***x-c*** koji povezuje ***c*** i ***x*** zatvara kut manji od π/2 sa vektorom $\mathbf{w} := \mathbf{c}_+ - \mathbf{c}_-$ koji povezuje srednje vrijednosti klasa. Dalje slijedi

$$\begin{aligned} y &= \operatorname{sgn}\langle (\mathbf{x} - \mathbf{c}), \mathbf{w} \rangle \\ &= \operatorname{sgn}\langle (\mathbf{x} - (\mathbf{c}_+ + \mathbf{c}_-)/2), (\mathbf{c}_+ - \mathbf{c}_-) \rangle \\ &= \operatorname{sgn}(\langle \mathbf{x}, \mathbf{c}_+ \rangle - \langle \mathbf{x}, \mathbf{c}_- \rangle + b) \end{aligned}$$

gdje je $b = \frac{1}{2}\left(\|\mathbf{c}_-\|^2 - \|\mathbf{c}_+\|^2\right)$. Ako srednje vrijednosti klasa imaju jednake udaljenosti od izvora tada je b=0. Treba primijetiti da *y* stvara granicu koja je oblika hiperravnine koja je u biti skup točaka koje zadovoljavaju uvjet koji se može izraziti kao linearna jednadžba.

Sada se *y* zapiše sa ulazima x_i, upotrebljavajući jezgru *k* za izračunavanje skalarnog produkta. Treba imati na umu da jezgrena funkcija *k* govori samo kako izračunati skalarni produkt između vektorskih reprezentacija $\boldsymbol{x}_i$ ulaza x_i. Stoga je potrebno vektore $\boldsymbol{c}_i$ i $\boldsymbol{w}$ odrediti u ovisnosti o $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_m$. Uzimajući u obzir prijašnje napomene dobiva se decizijska funkcija (*decision function*):

$$y = \operatorname{sgn}\left(\frac{1}{m_+} \sum_{\{i|y_i=+1\}} \langle \mathbf{x}, \mathbf{x}_i \rangle - \frac{1}{m_-} \sum_{\{i|y_i=-1\}} \langle \mathbf{x}, \mathbf{x}_i \rangle + b \right)$$
$$= \operatorname{sgn}\left(\frac{1}{m_+} \sum_{\{i|y_i=+1\}} k(\mathbf{x}, \mathbf{x}_i) - \frac{1}{m_-} \sum_{\{i|y_i=-1\}} k(\mathbf{x}, \mathbf{x}_i) + b \right)$$

gdje je b:

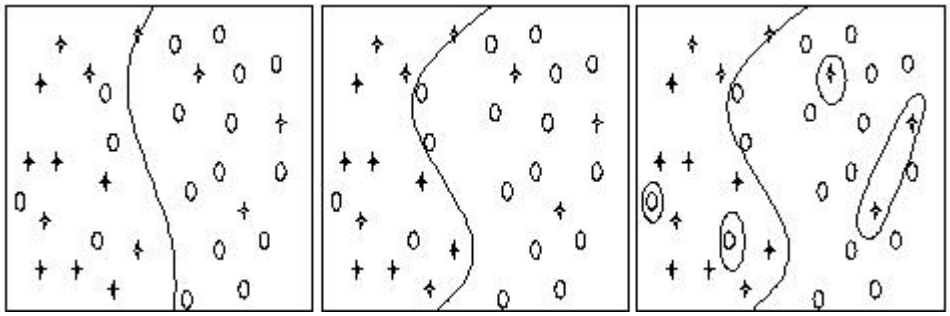
$$b := \frac{1}{2}\left(\frac{1}{m_-^2} \sum_{\{(i,j)|y_i=y_j=-1\}} k(\mathbf{x}_i, \mathbf{x}_j) - \frac{1}{m_+^2} \sum_{\{(i,j)|y_i=y_j=+1\}} k(\mathbf{x}_i, \mathbf{x}_j) \right)$$

3.2.3 Strogost podjele

Pri dvoklasnom prepoznavanju uzoraka tražila se funkcija

$$f : \mathrm{X} \to \{\pm 1\}$$

iz ulazno-izlaznih podataka. Podaci na kojima se trenira, uči, često se nazivaju i uzorci.



Slika 9 2D primjer binarne klasifikacije koji je riješen na tri različita načina. Modeli se razlikuju po kompleksnosti, od jednostavnog (lijevo), koji pogrešno klasificira velik broj uzoraka, do složenog (desno), koji uzima u obzir svaku točku i dolazi do rješenja koje je konzistentno sa svim uzorcima za učenje (postoji velika vjerojatnost da ovaj model ne radi ispravno sa novim ulazima).

Slika 9 prikazuje jednostavni 2D prikaz problema prepoznavanja uzoraka. Zadatak je razdvojiti kružiće od zvjezdica tako da se pronađe funkcija koja dodjeljuje vrijednost 1 za kružiće i -1 za zvjezdice. Treba uočiti da umjesto da se crta funkcija, mogu se nacrtati granice gdje se ona mijenja iz 1 u -1. Na desnoj slici može se vidjeti klasifikacijska funkcija koja točno dijeli uzorke. Međutim iz ove slike nije jasno dali bi dobro klasificirao nove uzorke. Lijeva slika prikazuje gotovo linearnu podjelu klasa. Međutim ovakva podjela ne samo da pogrešno klasificira uzorke koji su dublje u pojedinoj klasi a pripadaju suprotnoj, već i one koje su vrlo blizu granice. Središnja slika predstavlja kompromis, koristeći model srednje složenosti, koji dobro klasificira većinu točaka.

3.2.4 Hiperravninski klasifikatori

U ovom poglavlju biti će opisani algoritmi za učenje koji određuju hiperravnine, a mogu biti realizirani u prostoru u kojem je definiran skalarni produkt. Da bi se mogao napisati algoritam čija se statistička učinkovitost može kontrolirati potrebno je odrediti skup funkcija čiji se kapacitet može izračunati. Vapnik je zamislio skup hiperravnina u nekom prostoru *H* gdje je skalarni produkt definiran kao:

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0 \text{ za } \mathbf{w} \in \mathrm{H},\ b \in \mathbb{R}$$

što odgovara deciziskoj funkciji (*decision function*):

$$f(\mathbf{x}) = \operatorname{sgn}\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right)$$

i predložio algoritam za učenje za probleme koji mogu biti riješeni hiperravninom (odnosno oni koji su linearno separabilni), nazvao ga je *Generalized Portrait*, koji bi trebao odrediti *f* iz nekih osnovnih podataka. Ovdje su bitne dvije stvari. Prvo, među svim hiperravninama koje dijele uzorke postoji jedna koja se naziva optimalnom (*optimal hyperplane*). Optimalna hiperravnina određena je maksimalnim razmakom između hiperravnine i bilo kojeg uzorka (uvjetom maksimalne margine). Matematički zapisano:

$$\underset{\mathbf{w}\in \mathrm{H}, b\in\mathbb{R}}{\text{maximize}} \;\; \min\left\{ \left\| \mathbf{x} - \mathbf{x}_i \right\| \middle| \mathbf{x} \in \mathrm{H}, \langle \mathbf{w}, \mathbf{x} \rangle + b, i = 1, \ldots, m \right\}$$

Drugo, kapacitet skupa hiperravnina smanjuje se kako povećavamo razmak između hiperravnine i uzoraka.

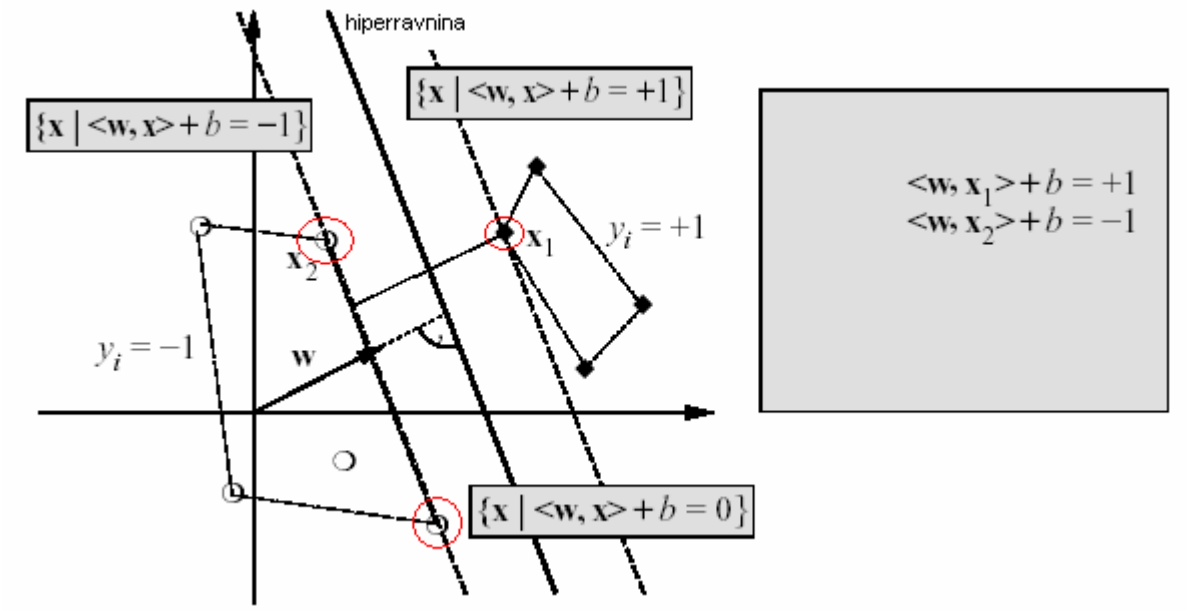
Treba primijetiti da je decizijska funkcija jako slična onoj spomenutoj u poglavlju 3.2.2. Međutim način na koji učimo algoritma je drugačiji. U primjeru iz poglavlja 3.2.2 vektor normale hiperravnine je bio jednostavno određen pomoću srednjih vrijednosti klasa (skupova) kao $\mathbf{w} = \mathbf{c}_+ - \mathbf{c}_-$. U ovom slučaju potrebno je obaviti nekoliko operacija da se dođe do vektora normale koji će zadovoljavati uvjet maksimalne margine. Da bi se odredila optimalnu hiperravninu potrebno je riješiti slijedeće:

$$\underset{\mathbf{w}\in \mathrm{H}, b\in\mathbb{R}}{\text{minimize}} \;\; \tau(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2,$$

s obzirom na $y_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) \geq 1$ za sve $i = 1, \ldots, m$

Treba primijetiti da gornja nejednadžba osigurava da $f(\mathbf{x}_i)$ bude 1 za y_i = 1, te -1 za y_i = -1. Potrebno je sada shvatiti zašto se želi minimizirati duljinu vektora ***w***. Kada bi $\|\mathbf{w}\|$ bio jednak 1 lijeva strana nejednadžbe bila bi jednaka

udaljenosti od uzorka $\boldsymbol{x}_i$ do hiperravnine. Općenito potrebno je $y_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right)$ podijeliti sa $\|\mathbf{w}\|$ kako bi se dobila ta udaljenost. Kada bi se uspjelo zadovoljiti nejednadžbu za sve i = 1,…, m sa minimalnom duljinom vektora $\boldsymbol{w}$ tada bi ukupna udaljenosti uzorka od hiperravnine bila maksimalna (slika 10).



Slika 10 Problem binarne klasifikacije ili podjele: razdvojiti lopte od dijamanta. Optimalna hiperravnina je označena punom crtom. Da bi bili separabilni mora postojati težinski vektor *w* i odstupanje b takvi da vrijedi $y_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) > 0$ (*i*=1,…,*m*). Mijenjajući *w* i b tako da točke najbliže hiperravnini zadovolje uvjet $\left|\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right| = 1$ dobivamo kanonski oblik rješenja (*w*, b) hiperravnine koje zadovoljava nejednadžbu $y_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) \geq 1$. Primijetite da je u ovom slučaju margina (udaljenost između najbliže točke hiperravnine i hiperravnine) jednaka $1/\|\mathbf{w}\|$. To se također može odrediti i tako da uzmemo dvije točke x_1 i x_2, sa suprotnih strana tako da vrijedi $\langle \mathbf{w}, \mathbf{x}_1 \rangle + b = 1, \langle \mathbf{w}, \mathbf{x}_2 \rangle + b = -1$, te ih projiciramo na vektor normale hiperravnine $\mathbf{w}/\|\mathbf{w}\|$.

Funkcija τ naziva se objektivnom funkcijom (*objective function)* dok se gornja nejednadžba naziva uvjet nejednakosti (*inequality constraints*). Oni zajedno tvore takozvani problem uvjetne optimizacije (*constrained optimization problem*). Ovakvi problemi se rješavaju uporabom Lagrangeovih koeficijenata (*Lagrange multipliers*) $\alpha_i \geq 0$ i Lagrangiana:

$$L(\mathbf{w},b,\alpha)=\frac{1}{2}\|\mathbf{w}\|^2-\sum_{i=1}^{m}\alpha_i\left(y_i\left(\langle\mathbf{w},\mathbf{x}_i\rangle+b\right)-1\right)$$

Lagrangian je potrebno minimizirati s obzirom na ***w*** i b i maksimizirati s obzirom na α_i, drugim riječima potrebno je naći neki kompromis. Primjećuje se da je uvjet nejednakosti sadržan u drugom dijelu Lagrangiana.

Kada bi uvjetna nejednadžba bila narušena, tada bi $y_i\left(\langle\mathbf{x}_i,\mathbf{w}\rangle+b\right)-1<0$, i povećanjem α_i povećali bi *L*. Istovremeno bi ***w*** i b trebali biti promijenjeni tako da se *L* smanji. Treba uočiti pojavu tzv. Lagrangeovih koeficijenata α_i koji, prema Lagrangeovoj teoriji, moraju biti veći ili jednaki nuli. Nužni uvjeti za ekstrem funkcije, u ovom slučaju minimum, je izjednačavanje parcijalnih derivacija s nulom.

$$\frac{\partial L(\mathbf{w},b,\vec{\alpha})}{\partial\mathbf{w}}=\mathbf{w}-\sum_{i=1}^{m}y_i\alpha_i\mathbf{x}_i=0$$

$$\frac{\partial L(\mathbf{w},b,\vec{\alpha})}{\partial b}=\sum_{i=1}^{n}y_i\alpha_i=0$$

$$\Rightarrow\qquad\mathbf{w}=\sum_{i=1}^{m}y_i\alpha_i\mathbf{x}_i$$

$$\Rightarrow\qquad 0=\sum_{i=1}^{m}y_i\alpha_i$$

Rezultantni vektor ***w*** sastoji se samo od podskupa iz skupa primjera za učenje, točnije samo od onih kod kojih je α_i različit od nule. Ti uzorci nazivaju se *Support Vectors* (SVs) i nose sve informacije potrebne za rješavanje problema (oni crveno zaokruženi). Prema $\alpha_i\left(y_i\left(\langle\mathbf{w},\mathbf{x}_i\rangle+b\right)-1\right)=0$ (za sve $i = 1,\ldots, m$) SV-i leže na margini (vidi sliku 8). Ostali trening uzorci ($\boldsymbol{x}_j$, y_j) su nebitni. Njihov uvjet $y_i\left(\langle\mathbf{w},\mathbf{x}_i\rangle+b\right)\geq 1$ se može i izostaviti te se oni ni ne pojavljuju u izrazu za ***w***. Ovo dovodi do osnovne ideje: hiperravnina je određena uzorcima koji su joj najbliži a ostali se zanemaruju.

Uvrste li se sada rješenja parcijalnih jednadžbi u početnu Lagrangeovu dobivamo:

$$\begin{aligned} L(\mathbf{w}, b, \vec{\alpha}) &= \tfrac{1}{2}(\mathbf{w}\cdot\mathbf{w}) - \sum_{i=1}^{m}\alpha_i[y_i((\mathbf{w}\cdot\mathbf{x}_i)+b)-1] \\ &= \frac{1}{2}\sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i\cdot\mathbf{x}_j) - \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i\cdot\mathbf{x}_j) + \sum_{i=1}^{m}\alpha_i \\ &= \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i\cdot\mathbf{x}_j) \end{aligned}$$

Ovime se eliminiraju varijable ***w*** i b i dobiva *dual optimization problem* koji se u praksi riješava:

$$\underset{\alpha\in\mathbb{R}^m}{\text{maximize}}\ W(\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i\cdot\mathbf{x}_j)$$

uz $\alpha_i \geq 0$ za sve *i* = 1,…, *m*, i $\sum_{i=1}^{m} y_i\alpha_i = 0$

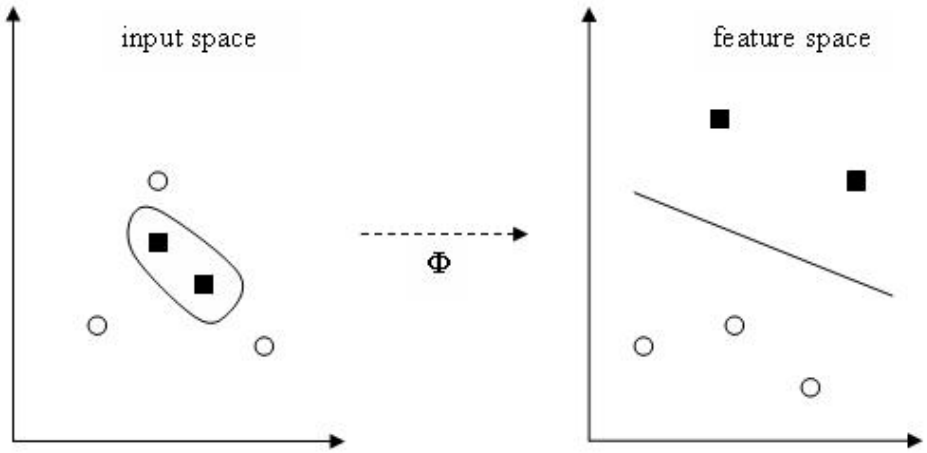
Decizijska funkcija (*decision function*) izgledala bi ovako:

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i \langle \mathbf{x}, \mathbf{x}_i \rangle + b\right)$$

Primjećuje se da su pojedini uvjeti nebitni i ne utječu na određivanje hiperravnine, to se jednostavno može objasniti mehanički. Neka se pretpostavi da je stavljena loptica u kutiju. Ona će se vrlo vjerojatno otkoturati u jedan od kutova. Stranice kutije predstavljaju uvjete, a iz tog izlazi da stranice koje loptica ne dotiče nisu bitni, odnosno ti su uvjeti nepotrebni.

3.2.5 Klasifikacija potpornim vektorima

Sada je poznato sve što je potrebno da se opiše SVM. U prošlim poglavljima sve je formulirano u prostoru *H* (*feature space*) gdje je definiran skalarni produkt (slika 11).



Slika 11 Ideja SVM-a: prebaciti uzorke za treniranje u visokodimenzionalni prostor sa Φ te tamo konstruirati hiperravninu koja će podijeliti uzorke uz uvjet maksimalne margine. To će nam dati nelinearnu granicu u ulaznom prostoru (*input space*). Korištenjem jezgrenih funkcija moguće je odrediti hiperravninu bez da se prebacujemo u visokodimenzionalni prostor.

Da bi se odredile formule koje vrijede za ulazni prostor (*input space*) *X*, treba se upotrijebiti jednadžba koja skalarni produkt vektora ***x, x'*** određuje uporabom jezgrene funkcije koja koristi ulazne parametre *x*, *x*':

$$k(x,x') = \langle \mathbf{x}, \mathbf{x}' \rangle$$

Ovu zamjenu, koja se ponekad i naziva *kernel trick*, koristili su Boser, Guyon i Vapnik kako bi proširili algoritam *Generalized portrait* na *Support Vector Machine* algoritam. *Kernel trick* se može primijeniti pošto su se svi vektori iz *feature space*-a pojavili samo u skalarnim produktima. Rezultantni vektor ***w*** se u *H* prostoru proširuje te stoga ne odgovara više preslikanom vektoru pomoću funkcije **Φ** iz prostora *X*. Dobiva se decizijska funkcija (*decision function*) u obliku:

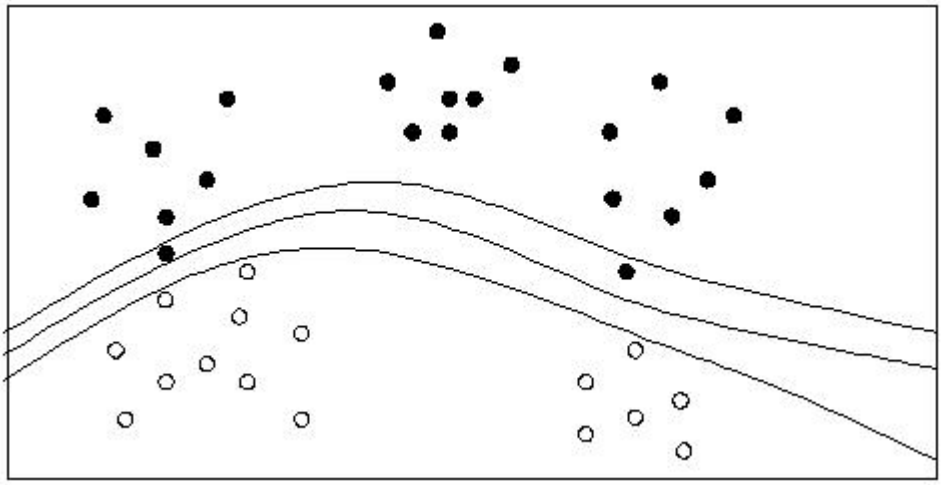
$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i \langle \Phi(x), \Phi(x_i) \rangle + b \right) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i k(x, x_i) + b \right)$$

te pripadni dualni problem:

$$\underset{\alpha\in\mathbb{R}^m}{\text{maximize}}\ W(\alpha)=\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{m}y_i y_j\alpha_i\alpha_j k(x_i,x_j)$$

uz $\alpha_i \geq 0$ za sve *i* = 1,…, *m*, i $\sum_{i=1}^{m} y_i\alpha_i = 0$.

Slika 12 pokazuje primjer ovakvog pristupa rješavanja problema koristeći Gaussovu radijalnu jezgru.



Slika 12 Primjer SV klasifikacije dobivene korištenjem radijalne funkcijske jezgre $k(x,x')=\exp\left(-\|x-x'\|^2\right)$ **(ovdje je ulazni prostor *X* =[-1, 1]^2). Crni i bijeli kružići predstavljaju dvije klase uzoraka za učenje. Srednja linija je decizijska ploha (*decision surface*), a dvije vanjske linije točno opisuju uvjet** $y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle+b)\geq 1$ za sve $i=1,\ldots,m$ **. Treba primijetiti da SV-i koji su određeni algoritmom (ovi koji se nalaze na linijama) ne predstavljaju središta nakupina primjera već su to uzorci koji su bitni za ovakvu klasifikaciju**.

Neka je ulazni prostor uzoraka dvodimenzionalan (d=2, $x, y \in R^2$). Ako se krene od nelinearnog preslikavanja iz dvodimenzionalnog u trodimenzionalni prostor zadanog sa:

$$\Phi(x)=\begin{bmatrix} x_1^{\,2} \\ \sqrt{2}\,x_1\ x_2 \\ x_2^{\,2} \end{bmatrix}$$

Računa se jezgrena funkcija:

$$k(x,y) = \begin{bmatrix} {x_1}^2 \\ \sqrt{2}\,x_1\,y_1 \\ {x_2}^2 \end{bmatrix}^T \cdot \begin{bmatrix} {y_1}^2 \\ \sqrt{2}\,y_1\,y_2 \\ {y_2}^2 \end{bmatrix} = {x_1}^2{y_1}^2 + 2x_1x_2y_1y_2 + {x_2}^2{y_2}^2 =$$

$$= (x_1y_1 + x_2y_2)^2 == \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \cdot \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = (x \cdot y)^2$$

Nakon kraćeg računa dobije se k(**x**,**y**) = (**x**·**y**)^2. Dakle, u ovom slučaju može se vrlo efikasno izračunati jezgrena funkcija bez da se izračunava preslikavanje Φ i skalarni produkt u *H*. Iako se tražila hiperravnina u prostoru *H*, nikad se nije efektivno prešlo u *H* – to je obavljeno implicitno kroz jezgrenu funkciju.

Više se ne traži odgovarajuće preslikavanje Φ, već samo odgovarajuća jezgrena funkcija koja će na efikasan način implicitno obaviti prelazak u *H*.

U praksi hiperravnina i ne mora postojati, a da bi se to moglo dozvoliti uvest će se dodatna varijabla $\xi_i \geq 0$ za sve *i* = 1,…, *m* kojom će se malo smanjiti težina uvjeta $y_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) \geq 1$ na:

$$y_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) \geq 1 - \xi_i \text{ uz } i=1,\ldots, m$$

Klasifikacijska funkcija koja dobro radi pronalazi se uz pomoć kapaciteta funkcije (preko $\|\mathbf{w}\|$) i sume novouvedenih varijabli $\sum_i \xi_i$. Može se pokazati da drugo navedeni član određuje gornju granicu broja grešaka pri učenju (*training errors*).

Jedna od mogućih realizacija malo blaže margine iliti *soft margin*-a se može dobiti minimiziranjem funkcije τ:

$$\tau(\mathbf{w},\xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i$$

gdje konstanta C > 0 pokazuje razliku između maksimiziranja margine i minimiziranja grešaka pri učenju. Ako se unese to sve u jezgru i sve napiše preko Lagrangeovih koeficijenata, opet se javlja problem maksimiziranja:

$$0 \le \alpha_i \le C \text{ za sve } i = 1,\ldots, m \text{ i } \sum_{i=1}^{m} y_i \alpha_i = 0$$

Na ovaj način utjecaj pojedinačnih uzoraka se ograničava. Odstupanje b, u *f(x),* se može odrediti ako se iskoristi činjenicu da za sve SV-e x_i sa $\alpha_i < C$ varijabla *ξ* je nula i vrijedi:

$$\sum_{j=1}^{m} y_j \alpha_j k\left(x_i, x_j\right) + b = y_j$$

Geometrijsko značenje ovog izraza je da mijenjajući veličinu b pomiče se hiperravninu i to sve dok SV-i koji imaju *ξ=0* ne nađu na ± 1 linijama (slika 10).

3.2.6 Primjeri jezgara

U dosadašnjim poglavljima opisane su osnove SVM-a stoga je potrebno sada to sve objediniti. Korištenjem jezgri optimalni klasifikator margine se transformirao u visokodimenzionalni klasifikator. Jezgre kao što su:

Polinominalna: $k\left(x, x'\right) = \left\langle x, x' \right\rangle^d$

Gaussova: $k\left(x, x'\right) = \exp\left(-\frac{\left\|x - x'\right\|^2}{2\sigma^2}\right)$

Sigmoidalna: $k\left(x, x'\right) = \tanh\left(\kappa \left\langle x, x' \right\rangle + \Theta\right)$

sa odabranim $d \in \mathbb{N}$ i $\sigma, \kappa, \Theta \in \mathbb{R}$ ($X \subset \mathbb{R}^N$) dale su rezultate s vrlo sličnom preciznošću i skupom SV-a. Ovo govori da skup SV-a rješava određeni problem gotovo neovisno o tipu jezgre koji je upotrijebljen, pod uvjetom da su parametri dobro namješteni.

4 Podaci

Unutar ovoga poglavlja ukratko će se opisati podaci koji su korišteni prilikom razvoja i testiranja sustava.

Kao ulazni set za učenje i testiranje je već ranije navedenih metoda korišten je podskup proteina izvučen iz proteinske baze podataka (*Protein Data Bank*). Već je ranije navedeno da se mjesta proteinskih interakcija dijele na hetero i homo, a također i na mjesta koja sudjeluju unutar interakcije te stvaraju trajne veze i one koje nisu trajne. Unutar ovog rada baviti ćemo se samo jednim tipom interakcije, odnosno tražit ćemo mjesta interakcije gdje se stvaraju ne trajne veze i to među dva ne identična lanca. Ovaj odabir dao nam je ovaj popis proteina.

1a0o	1bmf	1dio	1f2u	1g4y	1ixx	1qav	1tnr	3ygs
1a2k	1bml	1diq	1f34	1g6v	1jdb	1qbk	1tx4	4aah
1a2x	1bou	1dj7	1f3u	1g73	1jen	1qdl	1ubp	4fap
1a4y	1bp3	1dkg	1f3v	1g8k	1jrh	1qfu	1uea	4htc
1a9x	1bqh	1dm0	1f51	1gaq	1jsu	1qfw	1vcb	4sgb
1afv	1br4	1dml	1f5q	1gc1	1kac	1qgc	1viw	4ubp
1agr	1brs	1dn1	1f6m	1gle	1kb5	1qgk	1wdc	5sic
1ahj	1buh	1dow	1f93	1got	1kig	1qgw	1wej	6prc
1ahw	1bui	1dp5	1fak	1gp2	1kzu	1qkz	1xdt	7cei
1ai1	1bun	1dpj	1fbv	1gua	1lfd	1qla	1xtc	7prc
1aik	1bvn	1dtd	1fc2	1h2a	1lgb	1qle	1ycs	7req
1aip	1bx2	1du3	1fcd	1hcn	1lgh	1qo0	1yvn	8atc
1ak4	1bzq	1dv6	1fdl	1he1	1lod	1qo3	1zbd	8ruc
1al0	1c1y	1e0f	1fe8	1he8	1lpb	1qqp	2ahj	9atc
1am4	1c4z	1e1c	1fft	1hfe	1lts	1qs0	2bbk	
1an1	1c9t	1e79	1ffx	1hja	1luc	1qty	2btf	
1aok	1cc0	1e7p	1fg2	1hq3	1mah	1qun	2btv	
1aon	1cc1	1e96	1fg9	1hsb	1mcp	1quq	2ckb	
1aro	1ccw	1eai	1fi8	1hx1	1mda	1rcx	2frv	
1atn	1cd1	1eay	1fj1	1i1r	1mlc	1req	2min	
1aui	1cd3	1ebd	1fl7	1i2m	1mro	1rlb	2mll	
1ava	1cf7	1eer	1fle	1i4e	1mtn	1rrp	2mta	
1avg	1cg5	1efn	1flt	1i4t	1mty	1rvf	2occ	
1avw	1ci6	1efp	1fm6	1i50	1n2c	1sbb	2pcc	
1avx	1clv	1efv	1fma	1i6v	1ndo	1sbw	2scu	
1axi	1cmx	1efx	1fnt	1i7s	1nfi	1sct	2sic	
1ay7	1cn4	1egj	1foe	1i7x	1nmc	1seb	2siv	
1azs	1cse	1egv	1fq1	1i85	1noc	1sfc	2sni	
1b0n	1cul	1ehk	1fqj	1i8l	1nsg	1sgf	2trc	
1b33	1cxz	1ej6	1fqv	1iak	1nsn	1slu	2uug	
1b34	1cz8	1ejm	1fs0	1iar	1oak	1smp	2wsy	
1b35	1d0d	1ep3	1fs1	1ib1	1osp	1spp	3fru	
1b6c	1d2z	1es7	1fsk	1ibr	1pdk	1stf	3gtu	
1bab	1d3b	1eth	1ft1	1icf	1pgr	1taf	3hhr	
1bcp	1dan	1euv	1fxk	1iil	1phn	1tbg	3kin	
1bdj	1dce	1exb	1fzc	1iix	1pma	1tbr	3mon	
1bgx	1de4	1ezv	1g3i	1ijf	1poi	1tco	3pcg	
1bgy	1dee	1ezx	1g3j	1ikn	1pto	1tet	3pcn	
1bh9	1df9	1f0c	1g3n	1ira	1pys	1tii	3sgb	
1bkd	1dii	1f2t	1g4u	1itb	1pyt	1tmf	3tec	

Tablica 1 Popis proteina korištenih pri radu

Popis se sastoji od ukupno 333 proteina. Kako smo na početku ovog poglavlja naglasili da će nas zanimati samo jedan tip interakcije tako smo i iz ovog popisa proteina upotrijebili samo određene lance, točnije samo one lance za koje se 3D analizom proteina može reći da sadrže ovakav tip mjesta interakcije. Ovako odabrani set sastoji se od 1137 lanaca iz svih proteina nabrojenih u tablici 1.

Iz ovog konačnog popisa proteinskih lanaca potrebno je bilo naći mjesta proteinskih interakcija, odnosno potrebno je bilo stvoriti ulazni set koji će se kasnije upotrebljavati za učenje odnosno treniranje.

Kao mjesto proteinske interakcije definirali smo onu aminokiselinu koja je bila $\leq 6\,\mathring{A}$ udaljena od bilo koje druge aminokiseline drugog proteina. Uzeto je $6\,\mathring{A}$ jer se ova vrijednost pokazala kao kompromis. Kad se radilo sa manjim udaljenostima dobivao se prevelik broj mjesta interakcija, dakle i one aminokiseline koje nisu mjesta interakcije su prepoznate kao mjesta interakcije, a kad se uzimala veća vrijednost dobivalo se premalo. Definirao se prozor od 9 aminokiselina koji se sekvencijalno kretao po proteinskim lancima te su se izvlačile sekvence od 9 koje su označavane mjestima proteinskih interakcija ili mjestima koja to nisu. Prozor od 9 aminokiselina proglašavan je mjestom interakcije samo ako je njegova središnja aminokiselina bila u kontaktu sa nekom drugom iz drugog proteina.

Slika 13 Pozitivno definirana sekvenca

Zbog spoznaja i saznanja do kojih su došli Ofran i Rost [1], a to je da se mjesta proteinskih interakcija najčešće nalaze u skupinama, gornji uvjet još se malo pooštrio. Tražilo se da se oko središnje aminokiseline unutar prozora od 9 aminokiselina nalaze barem još četiri aminokiseline koje su također mjesta

proteinskih interakcija a nalaze se u krugu od tri najbliže aminokiseline središnjoj aminokiselini

Slika 14 Pozitivno definirana sekvenca i susjedne aminokiseline uzete u obzir

Na slici 14 žutom bojom su označeni susjedi unutar kojih se trebalo nalaziti još barem četiri aminokiseline koje su također sudjelovale u interakciji. Na ovaj način smo takvu sekvencu od 9 aminokiselina proglašavali mjestom proteinske interakcije. Kao negativne instance definirali smo one nizove od 9 aminokiselina koje na središnjem mjestu ne sadrže aminokiselinu koja interagira sa drugom aminokiselinom unutar drugog proteina, ali unutar cijele sekvence moglo se nalaziti do 2 aminokiseline koje su bile detektirane kao mjesta interakcije.

Na slijedećem primjeru može se vidjeti kako izgledaju instance korištene za učenje i trening.

NO,LEU,SER,ASP,ALA,ASP,LYS,LEU,ARG,LYS
YES,ASP,LYS,LEU,ARG,LYS,VAL,ILE,CYS,GLU

Može se primijetiti kako je se cijela instanca zapravo sastoji od 10 parametara, od toga prvi parametar predstavlja klasu, dvije klase YES i NO, odnosno dali je ili nije mjesto interakcije. Ostalih 9 parametara predstavljaju 9 aminokiselina u sekvenci kako je to već opisano u tekstu ranije.

5 Rješenje

U ovom će poglavlju biti opisan cjelokupni sustav koji rješava zadani zadatak upotrebom metoda i podataka opisanih u prethodna dva poglavlja.

5.1 Slučajna šuma (engl. Random Forest)

Za metodu slučajne šume (engl. Random Forest) koristi se program PARF [2]. To je zapravo paralelna realizacija RF (PArallel Random Forest). Ovaj program omogućuje puno bržu implementaciju i korištenje metode RF-a jer se algoritam može paralelno vršiti na više strojeva. Algoritam je potrebno prevesti u Fortran 90. jer je to strukturirani paralelni programski jezik.

Učenje i testiranje uporabom Parallel Random Forest algoritma vršeno je na određenom broju računala. Svako od tih računala je pod Linux operativnim sustavom. Stoga smo za prevođenje algoritma koristili Intelov kompajler i to Intel Fortran Compiler 9.1 za Linux. Da bi cijeli sustav mogao uopće raditi paralelno na više računala prije instaliranja PARF-a bilo je potrebno instalirati programski paket MPICH2 [3]. Ovaj programski paket omogućuje rad računala u paraleli.

5.1.1 Ulazni podaci

Potrebno je da ulazna datoteka za učenje bude u „*arff*“ formatu jer je to format s kojim barata program PARF. „*Arff*“ format datoteke je već opisan ranije u poglavlju 4 Podaci.

Trening set sastoji se od podataka dobivenih na osnovu ekstrakcije mjesta proteinskih interakcija iz 333 proteina ranije navedenih, ali smanjen za jednu trećinu ukupnog broja podataka. Za testni set iskoristili smo ovu jednu trećinu trening seta.

5.1.2 Izlazni podaci

Izlazni podaci koji nam govore o uspješnosti našeg testiranja mogu se mijenjati ovisno o želji korisnika. Postoje opcije unutar samog PARF-a kojima korisnik zadaje programu koje izlazne podatke želi da mu program vrati. Tako se kao izlazni podaci vraćaju:

- **Matrica zabune pri učenju** - svaki redak je jedna klasa kako je navedena u datoteci; svaka kolona je jedna klasa kako ju program klasificira. Redak obilježen s "NoTag" sadrži retke čija je klasa navedena kao nepoznata (?); kolona obilježena s "NotCl" sadrži instance koje nisu mogle biti klasificirane zato što su uzete u bootstrap za svako stablo (te tako nikad nisu bile out-of-bag).
- **Rezultati klasifikacije** - Ako je skup neobilježen (ispušten je atribut klase), ispisat će se klase svih instanci; ako je obilježen, prikazat će se samo neodgovarajuće klasificirane instance.
- **Matrica zabune pri klasifikaciji** - Svaki redak je jedna klasa kako je navedena u datoteci; svaka kolona je jedna klasa kako ju program klasificira. Redak obilježen s "NoTag" sadrži retke čija je klasa navedena kao nepoznata (?); "NotCl" kolona je uvijek jednaka nuli

5.2 Metoda potpornih vektora (Support Vector Machine)

Metoda potpornih vektora korištena je unutar programskog paketa LibSVM [4]. LibSVM je alat koji može vršiti klasifikaciju, regresiju i ocjenjivanje distribucije potpornih vektora. U ovom radu koristiti će se samo klasifikacija i to klasifikacija u 2 dvije klase. Da bi se učenje i treniranje moglo uopće izvršiti potrebno je prvo odrediti parametre. U ovom radu koristiti će se RBF jezgra, jer ona po analizama daje rezultate jednako dobre kao i neke složenije jezgre, a puno je jednostavnije pronaći njezine koeficijente. Kod ove jezgre bitni su parametri c i gama. Stoga je u tu svrhu napisana skripta koja na više strojeva paralelno vrši pretragu idealnih parametara.

5.2.1 Pronalaženje idealnih parametara

Skripta za traženje idealnih c i gama napisana je u programskom jeziku Python [5]. Sama skripta vrti se na više strojeva paralelno. Skripta zapravo radi jedan vrlo jednostavan posao, ali ga radi iterativno više puta. Dakle, skripta zapravo pokreće LibSVM sa zadanim parametrima c i gama. Ona vrši proces

krosvalidacije. To je proces kod kojeg se iz ulaznog seta podataka odabire određeni broj instanci koje se upotrebljavaju kao test set, a ostatak kao trening set. Nakon toga se vrši treniranje i klasifikacija. Kad se proces završi iz ukupnog seta se odabire drugi testni set iste veličine kao i prethodni ali potpuno različit. Ostatak se opet koristi kao trening set. Ovaj proces se ponavlja ovisno o broju ponavljanja krosvalidacije. Nakon završenog cijelog procesa krosvalidacije dobivaju se određeni rezultati koji se pamte. Zatim se mijenjaju parametri c i gama te se ponovo vrši krosvalidacija. Dobivaju se novi rezultati, te se uspoređuju sa starim. Ako je rezultat novog procesa krosvalidacije bolji od prošlog, parametri upotrijebljeni pri ovom procesu se uzimaju kao idealni. Sama metoda krosvalidacije nastoji naći idealne parametre, ali ne i najbolje. Razlika je u tome što učenje i treniranje sa idealnim parametrima možda neće dati bolje rezultate kao najbolji, ali pri testiranju naučenog modela sa potpuno novim ulaznim setom idealni parametri će dati puno bolji rezultat nego najbolji. Jednostavno rečeno najbolji rezultati se odlično ponašaju na poznatom skupu, a pri nepoznatom jako loše, dok je kod idealnih rezultat uvijek dobar.

5.2.2 Ulazni podaci

Nakon što su pronađeni idealni parametri c i gama za ulazni skup može se početi sa treniranjem računala. Valja naglasiti da je ulazni set jako velik a proces krosvalidacije dosta spor, stoga se traženje idealnih parametara vrši na podskupu ulaznog skupa. Ulazni set za LibSVM ponešto se razlikuje od ulaznog seta za PARF, ali zapravo se radi o istim podacima. Ulazni podaci se sastoje od 181 parametra. Prvi parametar predstavlja klasu, a ostalih 180 aminokiseline. Pošto se ulazni set sastoji od devet parametara koji predstavljaju jednu od 20 aminokiselina, svaka aminokiselina je predstavljena u binarnom obliku. To je ostvareno tako da se svaka aminokiselina sastoji od 19 nula i 1 jedinice, s tim da se ta jedinica za svaku aminokiselinu pojavljuje na drugom mjestu. Na taj način nastaje ulazni set od 181 parametra. U ulaznom setu LibSVM-a nije potrebno navoditi nul parametre stoga se ispisuju samo nule. Inače svaki od parametara

je numeriran pa se točno zna na kojoj poziciji se nalazi jedinica. LibSVM može raditi samo sa brojčanim podacima stoga su i klase zamijenjene iz YES u 1 i iz NO u -1. Da bi se što bolje pojasnilo dan je tipičan primjer:

```
1 3:1 39:1 51:1 64:1 100:1 119:1 123:1 143:1 172:1
-1 19:1 23:1 43:1 72:1 81:1 101:1 140:1 151:1 172:1
```

Može se također primijetiti da su parametri u ovom ulaznom setu odvojeni razmacima a ne kao u arff ulaznom setu, za PARF, zarezima.

5.2.3 Izlazni podaci

Izlaz iz LibSVM-a je jednostavan rezultat predikcije, odnosno klasifikacije u postocima i to u obliku preciznosti, te u obliku osnovnih pojmova točnosti predikcije.

6 Rezultati

U ovom poglavlju opisani su rezultati testiranja algoritma slučajne šume i potpornih vektora. Rezultati će biti opisani u potpoglavljima zasebno za svaku metodu.

6.1 Slučajne šume

Testiranje algoritma slučajne šume testiran je isprava na čistim ne obrađenim podacima. Pod ovim pojmom ne obrađeni podrazumijeva ju se podaci koji nisu težinski. Najme unutar samog trening i testnog seta podataka nema jednak broj instanci koje obilježavaju mjesta interakcije i onih koje ih ne obilježavaju. Ovih drugih ima puno veći broj što je zapravo sasvim prirodno. Nakon toga izvršeno je zadavanje težina svakoj klasi posebno. Da bi se pronašla idealna težina opet treba testirati, te je izvršeno par iteracija. Slijedeći rezultati to zorno prikazuju.

6.1.1 Neobrađeni podaci

U ovom poglavlju vršilo se treniranje šume na osnovnom trening setu, dakle sa instancama od 1137 lanaca.

U poglavlju 3.1 dana je teoretska osnova slučajne šume, te znamo da testni set za slučajnu šumu nam zapravo nije trebao jer slučajna šuma sama sebe testira na trećini ulaznih podataka kada stvara klasifikatore. Ulazni set nam se sastojao od 159333 instance. Ukupna greška kod klasifikacije iznosila je 12,21%. Znači naš klasifikator je naučio sa točnošću od 87,79%. Vrlo zoran primjer klasifikacijske točnosti je matrica zabune pri učenju (tablica 2). Redak predstavlja stvarnu klasu instanci, a stupac ono što je slučajna šuma klasificirala.

	YES	NO
YES	10400	17701
NO	1760	129472

Tablica 2 Matrica zabune pri učenju neobrađenih podataka

Možemo primijetiti da *true negative* ili TN imamo 129472, dakle klasifikator je ispravno pogodio 129472 negativne instance, dok je njih 1760 krivo klasificirao kao pozitivne, ove se instance još nazivaju i *false negative* ili FN. Pozitivne instance koje je klasifikator ispravno klasificirao kao pozitivne nazivamo i *true positive* ili TP i njih ima 10400. *False positive* ili FP imamo 17701 i to su instance koje su zapravo pozitivne no krivo su klasificirane kao negativne. Ovo su osnovni pojmovi točnosti predikcije i prikazani su u tablici 3.

Tag / Cl	YES	NO
YES	**TP**	**FN**
NO	**FP**	**TN**

Tablica 3 Osnovni pojmovi točnosti predikcije

Uspješnost klasifikatora vrednuje se iz složenijih pojmova koje je moguće izvesti iz ovih osnovnih. To su osjetljivost (eng. Recall), pozitivna predvidljivost (eng. Precision) i mjera F (eng. F-measure). Osjetljivost se izražava kao:

$$\text{Recall} = \frac{TP}{TP + FN}$$

,pozitivna predvidljivost izražava:

$$\text{Precision} = \frac{TP}{TP + FP}$$

, dok se mjera F izražava kao:

$$F = \frac{2TP}{2TP + FP + FN}$$

U našem testu rješenje je slijedeće:

Recall	37%
Precision	86%
F-measure	52%

Tablica 4 Uspješnost klasifikatora sa neobrađenim podacima

Iz ovih rezultata primjećuje se da je klasifikator dosta loše prepoznao pozitivne instance. Međutim to je i prirodno za očekivati pošto je negativnih instanci puno više nego pozitivnih. Da bi se riješio ovaj problem potrebno je uvesti težine klasa.

6.1.2 Podaci sa težinama klasa

Da bi smo odredili koja težina bi bila idealna potrebno je izvršiti par iteracija, odnosno zadavati različite težine pozitivnim klasama. Tablica 5 prikazuje kako su se mijenjali rezultati klasifikacije kako smo mijenjali težinu pozitivne klase.

Težina	10	5	3	2
Greška klasifikacije	42,61%	26,01%	16,17%	12,81%
TP	23753	18161	13342	11344
FP	63545	31501	11012	3660
TN	67687	99731	120220	127572
FN	4348	9940	14759	16757
Recall	85%	65%	47%	40%
Precision	27%	37%	55%	76%
F-measure	41%	47%	51%	53%

Tablica 5 Uporaba težinskih klasa i rezultati klasifikacije

6.1.3 Uporaba testnog seta

Ovdje će se sada vidjeti kako klasifikator slučajna šuma klasificira ako upotrebljavamo trening i test set. Naš trening set sastojat će se od 2/3 početnih ulaznih podataka, a trening set od ostale trećine. Treba primijetiti da ćemo u ovom slučaju klasifikator ispitivati na podacima koje nije vidio. Trening set sastoji se od 104102 instanci a testni od 51022. Rezultate prikazuje tablica 6.

Težina	10	5	3	2	1
Greška klasifikacije	62,41%	36,37%	16,02%	12,81%	12,67%
TP	6991	4734	2292	1700	1570
FP	31290	15751	2924	694	492
TN	12189	27728	40555	42785	42987
FN	549	2806	5248	5840	5970
Recall	93%	62%	31%	23%	21%
Precision	18%	23%	44%	71%	76%
F-measure	31%	34%	36%	34%	33%

Tablica 6 Rezultati klasifikacije testnog seta

6.2 Potporni vektori

U ovom poglavlju prikazani su rezultati testiranja algoritma potpornih vektora. Pri učenju i korištenju algoritma potpornih vektora koristila su se dva seta.

6.2.1 Prvi set

Kao što je ranije objašnjeno u poglavlju 5.2.1 potrebno je prije početka klasifikacije poznavati parametre klasifikatora. Pretraživanje parametara vršeno je pomoću skripte napisane u programskom jeziku python[5]. Pretraživanje za c i gamu vršeno je u rasponu od $2^{-10} \sim 2^{10}$ s tim da se broj u eksponentu mijenjao za jedan. Vrijednosti se mijenjaju eksponencijalno jer je su analizom utvrdili [6]

da je to najbolji način za pronalaženje dobrih parametara. Nakon završene pretrage dobiveni su ovi rezultati.

c	1
gama	64
krosvalidacija	82,17%

Tablica 7 Rezultati pretrage idealnih parametara prvog seta

Pretraga je vršena 10 strukom krosvalidacijom (eng. 10 fold cross-validation).

Nakon što su pronađeni idealni parametri pristupa se učenju i testiranju naučenog klasifikatora. Kod slučajne šume mogli smo upotrebljavati samo trening set i opet zapravo ne samo učiti već i testirati sam klasifikator. Kod potpornih vektora takav oblik učenja se vrši pomoću krosvalidacije. Drugi oblik učenja je korištenje trening i test seta Oni ne moraju nužno biti različiti ali mi ćemo uzeti različite setove. Trening set sastojao se od 104101 instance dok se testni set sastojao od 51021 instance. Prvo je bilo potrebno istrenirati klasifikator, odnosno naučiti ga kako da radi. Ovaj proces rezultira modelom koji se kasnije koristi za klasifikaciju. Ovdje se također pojavljuju težine klasa te su rezultati prikazani u slijedećoj tablici.

Težina	30	5	3	1
TP	1539	1539	1538	1511
FP	850	686	563	402
TN	42631	42795	42918	43079
FN	6001	6001	6002	6029
Recall	20%	20%	20%	20%
Precision	64%	73%	73%	79%
Accuracy	86,57%	86,89%	87,13%	87,39%
F-measure	31%	32%	32%	32%

Tablica 8 Rezultati klasifikacije prvog seta

6.2.2 Drugi set

Ulazni trening set sastoji se od 106501 instance dok se testni set sastoji od 48621 instance. Nakon traženja idealnih parametara za ovaj set dolazimo do ovih rezultata.

c	1
gama	64
krosvalidacija	83,75%

Tablica 9 Rezultati pretrage idealnih parametara drugog seta

Nadalje vrši se učenje i klasifikacija te se dobivene rezultate može vidjeti u tablici 10.

Težina	30	5	3	1
TP	702	702	696	614
FP	581	511	503	332
TN	39995	40065	40073	40244
FN	7343	7343	7349	7431
Recall	8%	8%	12%	7%
Precision	54%	58%	58%	65%
Accuracy	83,7%	83,85%	83,85%	84%
F-measure	15%	15%	15%	14%

Tablica 10 Rezultati klasifikacije za drugi set

7 Sažetak

Ovaj rad napisan je sa ciljem da se probaju predvidjeti mjesta proteinskih interakcija. Za analizu koristili su se samo oni proteini koji su stvarali trajne veze i to prema drugim proteinima, a ne unutar samog proteina, dakle hetero veze. To nas je dovelo do ulaznog seta od 333 proteina iz kojih smo izvukli 1137 lanaca koje smo obrađivali. Klasifikaciju smo vršili pomoću dva algoritma za strojno učenje i to algoritam slučajne šume (engl. Random Forest) i algoritam potpornih vektora (engl. Support Vector Machine). Ova dva algoritma implementirana su unutar programa PARF[2] (PArallel Random Forest) i LibSVM[4].

PARF se učio na dva i klasificirao na dva seta. Prvi set sastojao se od 159333 instance, on ujedno bio i testni set, te se dobila pozitivna predvidljivost od 86% i predvidljivost 37%. Ovi rezultati su dobiveni na neobrađenom ulaznom skupu, odnosno sve instance su imale jednaku važnost. Slijedeći testovi usredotočili su se na promjenu važnosti, težine, instanci. Pošto nema jednak broj pozitivnih i negativnih, prvih ima puno manje, mijenjala se težina pozitivnih. Ovaj postupak je donio slijedeće rezultate.

Težina	10	5	3	2
Recall	85%	65%	47%	40%
Precision	27%	37%	55%	76%
F-measure	41%	47%	51%	53%

Tablica 11 Rezultati nakon mijenjanja težina pozitivnih instanci

Daljnje testiranje vršilo se na različitim trening i test setovima. Trening set sastojao se od 104102 instanci a testni od 51022. Opet se tražila težina trening seta, a promatrala se pogreška pri klasifikaciji test seta. Klasifikacija ovog seta donijela je ove rezultate.

Težina	10	5	3	2	1
Recall	93%	62%	31%	23%	21%
Precision	18%	23%	44%	71%	76%
F-measure	31%	34%	36%	34%	33%

Tablica 12 Rezultati klasifikacije uporabom test seta

Drugi dio klasifikacije vršio se koristeći algoritam potpornih vektora (engl. support vector machine). Da bi se klasifikacija mogla provoditi bilo je potrebno naći parametre RBF jezgre koja je korištena. Nakon što su parametri pronađeni pristupilo se klasifikaciji. Korištena su 2 trening i 2 test seta. Prvi trening set imao je parametre c=1 i gama=64. Nakon završene klasifikacije pokazao se ovaj rezultat.

Težina	30	5	3	1
Recall	20%	20%	20%	20%
Precision	64%	73%	73%	79%
F-measure	31%	32%	32%	32%

Tablica 13 Rezultati klasifikacije prvog trening i prvog testnog seta

Drugi trening set te njegov pripadajući testni set dali su ove rezultate.

Težina	30	5	3	1
Recall	8%	8%	12%	7%
Precision	54%	58%	58%	65%
F-measure	15%	15%	15%	14%

Tablica 14 Rezultati klasifikacije drugog trening i drugog testnog seta

8 Zaključak

Sam naslov ove radnje zapravo postavlja pitanje, dali se uopće mogu predvidjeti mjesta proteinskih interakcija iz sekvence aminokiselina. Kroz teoretsku analizu nastojalo se prikazati da problem sam po sebi nije posve trivijalan, ali nije ni jako kompliciran. U ovom radu upotrebljene su metode i algoritmi koji kod nas a i u svijetu tek čekaju da postanu opće primjenjivi. Ako malo promotrimo rezultate iz poglavlja „7 Rezultati“ možemo primijetiti da je algoritam slučajnih šuma pokazao prilično zavidnu mogućnost prepoznavanja mjesta proteinskih interakcija. Rezultati algoritma potpornih vektora ne ističu se kao rezultati slučajnih šuma, ali postoji potencijal koji se može a i treba iskoristiti. Može se primijetiti da slučajne šume mogu sa visokom osjetljivošću od čak 93% i visokom preciznošću od čak 76% prepoznavati mjesta proteinskih interakcija. Ove dvije brojke međutim su obrnuto proporcionalne. To se može primijetiti u tablici 5 i 6. Izmjenom težina klasa dobili smo kretanja oba dva ova pokazatelja uspješnosti. Pronalaženjem idealne težine dolazimo do rezultata koji prilično zadovoljava i jedan i drugi pokazatelj točnosti. Ovim smo zapravo pokazali kako algoritmi za strojno učenje, slučajne šume i potporni vektori, mogu služiti za prepoznavanje mjesta proteinskih interakcija.

9 Diskusija

Proučavanjem rezultata može se primijetiti da mnoga mjesta proteinskih interakcija se nalaze u skupinama a ne na izoliranim mjestima. To nam daje novo saznanje o proteinima samim bitno za njihovu daljnju analizu

Da bi se mogli vrednovati ovi rezultati uspoređeni su sa već ranije objavljenim radom sa zadatkom o prepoznavanju mjesta proteinskih interakcija ali korištenjem neuronskih mreža[1]. Ofran i Rost su u svom radu došli do rezultata da čak 94% prepoznatih proteinskih interakcija je zapravo i zaista bilo mjesto interakcije, a slučajne šume su ta mjesta prepoznale sa čak 93% točnosti. Pri ovako visokoj osjetljivosti uspješno je detektirani čak 6991 mjesto interakcije iz 379 lanaca iz 110 proteina. Pri točnosti od 62% pronađeno je 4734 mjesta proteinske interakcije. Testiranja su na početku pokazala veliku osjetljivost međutim i sa malom preciznošću. Točno su detektirana 21% mjesta interakcije, dok su Ofran i Rost ostvarili čak 30% točnosti. Međutim reguliranjem težina i utjecanjem na podatke došlo se do puno boljih rezultata.

Moram naglasiti da je sama ideja o proučavanju proteina i uopće bavljenje bioinformatičkim istraživanjima polje koje je za mene dosta novo i jedan potpuno novi zaokret u pogledu na primjenu računala i znanja u proučavanju jedne tako kompleksne stvari kao što je organizam. Ne znam kako bih ušao u to polje da nije bilo mr. sc. Mile Šikića, koje nas je sve na jedan jednostavan i prijateljski način uveo u to područje.

Ideja cijelog ovog rada bila je dokazati da metode računalnog učenja mogu pomoći u radu na ovom i drugim područjima gdje ljudski um nije dovoljan.

10 Literatura

[1] Yanay Ofran, Burkhard Rost: „Predicted protein-protein interaction sites from local sequence information“, 13 May 2003

[2] www.irb.hr/hr/research/projects/it/2004/2004-111/

[3] www-unix.mcs.anl.gov/mpi/mpich2/

[4] www.csie.ntu.edu.tw/~cjlin/libsvm/

[5] www.python.org

[6] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin: „A Practical Guide to Support Vector Classification“, Taipei 106, Taiwan
www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[7] Leo Breiman: „Random Forests“, Machine Learning, 45, 5–32, 2001

[8] www.stat.berkeley.edu/users/breiman/RandomForests/

[9] IPGRI and Cornell University: “Protein-based technologies Protein basics”, 2003

[10] Chih-Chung Chang, Chih-Jen Lin: „LIBSVM: a Library for Support Vector Machines“, April 17, 2005

[11] Bernhard Scholkopf, Alexander J. Smola: „Learning with Kernels“, The MIT Press Cambridge, Massachusetts London, England

[12] Witten Frank: „Dana Mining“,

11 Dodatak

Ovdje će ukratko biti opisan software koji je napisan i koji je korišten.

11.1 PPDisstance03

Za dohvaćanje instanci, odnosno za stvaranje ulaznog seta za PARF korišten je program napisan u C++. Taj se program nalazi u direktoriju PPdistance03. Da bi ovaj program ispravno radio potreban je paket Visual Studio .NET 2003 i SQLserver4.1 ili noviji. Ovaj program na osnovu ulazne datoteke u kojoj su napisani lanci koji se proučavaju stvara ulazni set. Datoteka u kojoj se nalazi popis lanaca nalazi se u poddirektoriju pdb i zove lanci.txt. Ovaj program stvara FullSet.arff, FullSet.bnd, FullSet.convert datoteke. Prva datoteka predstavlja popis svih instanci i služi kao ulaz u PARF, druga datoteka služi za kontrolu obrade lanaca koji se koriste, a treća kao ulaz u converter.

11.2 Converter

Ovo je također program napisan u C++ a on konvertira FullSet.convert datoteke u FullSet.LibSVM, odnosno sprema ih za uporabu LibSVM-a. Imena ulazne i izlazne datoteke koje se koriste može se mijenja izravno u programu i to na samom početku.

11.3 LibSVM

LibSVM je program koji se koristi algoritmom potpornih vektora, te vrši učenje i klasifikaciju. Za njegovu upotrebu pogledati na stranicu www.csie.ntu.edu.tw/~cjlin/libsvm/.

11.4 PARF

PARF je program koji se koristi algoritmom slučajnih šuma i služi za učenje i klasifikaciju, a upute o njegovoj upotrebi mogu se pročitati na stranici www.irb.hr/hr/research/projects/it/2004/2004-111/