Algoritmi učenja bazirani na jezgrama

Beti, Domagoj

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Domagoj Beti

ALGORITMI UČENJA BAZIRANI NA JEZGRAMA

Diplomski rad

Voditelj rada:
doc. dr. sc. Goranka Nogo

Zagreb, 2017.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Uvod

Analiza uzoraka sastavni je dio znanstvenih disciplina kao što su strojno učenje, rudarenje podataka, statistika i bioinformatika. U ovom radu promatrat ćemo kako primijeniti jezgrine funkcije u različitim algoritmima učenja te kako takvim pristupom pronaći uzorke među podacima. Vidjet ćemo da je takav pristup modularan, odnosno da se svaka jezgra može koristiti u svakom algoritmu učenja i obratno. Modularnost te činjenica da je pristup u stanju premostiti razlike koje postoje u različitim disciplinama u kojima se analiza uzoraka koristi, pružaju jedinstven i efikasan alat za rad nad podacima svih tipova, bili oni vektori, skupovi, stringovi ili čak neki složeni oblik podatka. Rad se sastoji od četiri poglavlja.

U prvom poglavlju opisat ćemo problem te ćemo navesti matematičke definicije pojmova koje ćemo koristiti u daljnjem tekstu radi njegovog lakšeg razumijevanja.

U drugom poglavlju ćemo najprije kroz konkretan primjer uvesti pojam jezgrinih funkcija. Vidjet ćemo kako pomoću njih možemo pronaći linearne relacije u nekom konačnom skupu podataka te ćemo saznati koje su prednosti njihovog korištenja. Zatim ćemo napravit klasifikaciju jezgrinih funkcija prema tipu podataka za koji su predviđene.

U trećem poglavlju opisat ćemo efikasne algoritme učenja za probleme otkrivanja noviteta, klasifikacije i regresije koji su bazirani na potpornim vektorima.

U četvrtom poglavlju implementirat ćemo dva algoritma za klasifikaciju s dvije različite jezgre, te ćemo ih primijeniti za problem kategorizacije teksta. Analizirat ćemo dobivene rezultate.

Poglavlje 1

Opis problema i uvodne definicije

1.1 Opis problema

Proučavanje uzoraka među podacima seže daleko u prošlost. Tako primjerice možemo reći da su tri zakona gibanja planeta oko sunca Johannesa Keplera posljedice relacija koje je uočio u velikom skupu opservacijskih podataka koje je sakupio Tycho Brahe.

Isto tako, želja da se traženje uzoraka automatizira sigurno je stara koliko i samo računarstvo. Problem automatiziranja prisutan je u mnogim područjima prirodnih znanosti poput statistike, strojnog učenja ili rudarenja podataka.

Analiziranje uzoraka bavi se problemom automatskog prepoznavanja relacija, pravilnosti ili struktura u nekom skupu podataka. Ukoliko uočimo značajan broj uzoraka u dostupnim podacima, moći ćemo napraviti model s kojim će biti moguće raditi predikcije na novim podacima koji dolaze iz istog skupa.

Razvoj automatiziranih algoritama za analizu uzoraka možemo podijeliti u tri razdoblja. Prvo razdoblje su šezdesete godine 20-og stoljeća kada su predstavljeni algoritmi za detekciju linearnih relacija unutar skupova vektora. U tom razdoblju pitanje detekcije nelinearnih relacija bilo je postavljeno kao glavni cilj istraživanja.

Sredinom osamdesetih godina 20-og stoljeća bilo je moguće otkriti nelinearne uzorke novootkrivenim algoritmima, ali mana im je bila što su koristili heuristike te se nije razumjelo njihovo statističko ponašanje. Kako su bili temeljeni na metodama najbržeg silaska i pohlepnim heuristikama patili su od zaustavljanja u lokalnom minimumu.

Treće razdoblje razvoja odvija se sredinom devedesetih godina s pojavom novog pristupa znanim kao algoritmi učenja bazirani na jezgrama koji je konačno omogućio istraživačima analiziranje nelinearnih relacija s efikasnošću koja je bila prije bila rezervirana samo za algoritme koji su uočavali linearne relacije. Njihova statistička analiza omogućila je da se algoritmi primjenjuju na prostore velikih dimenzija.

1.2 Uvodne definicije

Radi lakšeg razumjevanja daljnjeg teksta ovdje ćemo definirati i objasniti osnovne pojmove koje ćemo koristiti. Reći ćemo da je **uzorak** relacija, pravilnost ili struktura u nekom skupu podataka, te da je **značajka** konkretna vrijednost koja predstavlja jedno svojstvo danog podatka. Nadalje, pretpostavljat ćemo da je **prostor značajki** unitaran prostor.

Definicija 1.2.1. *Za skup S kažemo da je **konveksan**, ako je za svake dvije točke $x, y \in S$ i njihova spojnica sadržana u S. **Konveksna ljuska** skupa S je najmanji konveksan skup koji ga sadrži.*

Definicija 1.2.2. *Neka su $f, g_1, \ldots, g_l$ i $h_1, \ldots, h_m$ dovoljno glatke funkcije definirane nad nekom domenom $X \subseteq \mathbb{R}^n$. Za problem*

$$\begin{aligned} \text{minimiziraj} \quad & f(x), x \in X \\ \text{uz ograničenja} \quad & h_i(x) \leq 0, i = 1, \ldots, m \\ & g_j(x) = 0, j = 1, \ldots, l \end{aligned}$$

definiramo Lagrangeovu funkciju s

$$L(x, u, v) = f(x) + \sum_{i=1}^{m} u_i h_i(x) + \sum_{j=1}^{l} v_j g_j(x),$$

pri čemu za vrijednosti u_i, $i \in \{1, \ldots, m\}$ i v_j, $j \in \{1, \ldots, l\}$ kažemo da su Lagrangeovi multiplikatori. Dualnu Lagrangeovu funkciju definiramo s

$$g(u, v) = \min_{x \in X} L(x, u, v),$$

a za

$$\begin{aligned} \nabla f(x) = \sum_{i=1}^{m} u_i h_i(x) + \sum_{j=1}^{l} v_j g_j(x) \\ u_i h_i(x) = 0, \quad i \in \{1, \ldots, m\} \\ h_i(x) \leq 0, g_j(x) = 0, \quad i \in \{1, \ldots, m\}, j \in \{1, \ldots, l\} \\ u_i \geq 0, \quad i \in \{1, \ldots, m\} \end{aligned}$$

kažemo da su Karush-Kuhn-Tuckerovi (KKT) uvjeti.

Poglavlje 2

Jezgrine funkcije

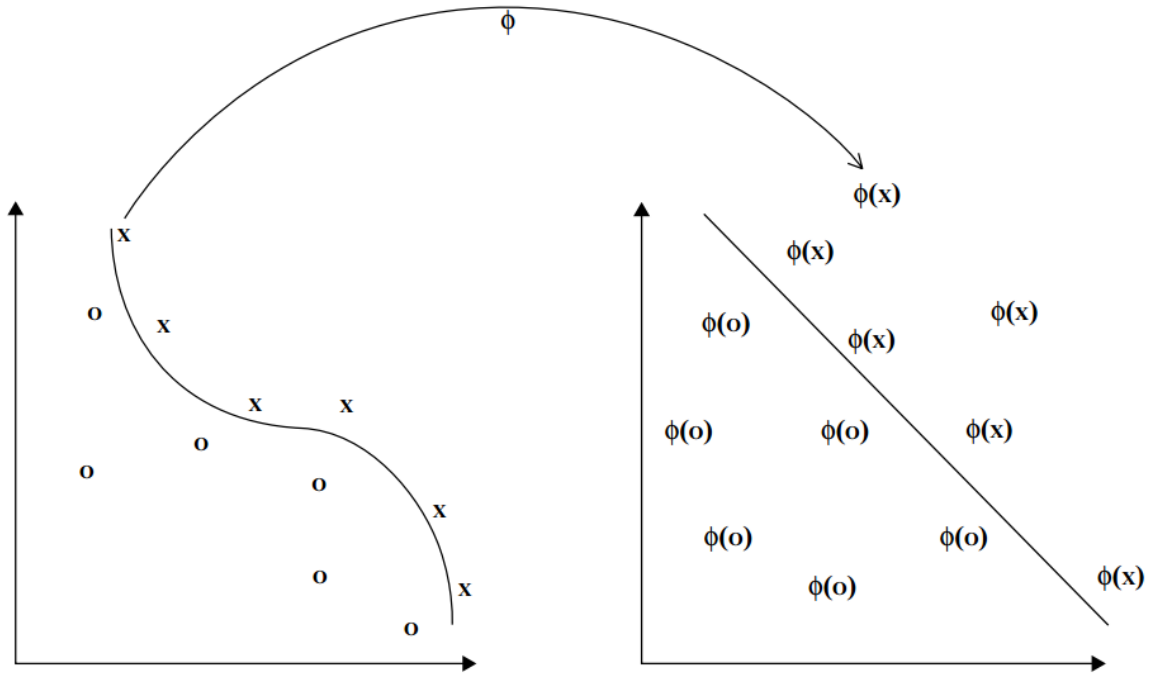
2.1 Uvod

U ovom poglavlju ćemo kroz konkretni primjer linearne regresije pokazati kako jezgrine funkcije robusno i efikasno pronalaze uzorke u konačnom skupu podataka. Ideja je da se dani podaci preslikaju u prostor gdje će se moći uočiti linearne relacije. Ta ideja se može razložiti na dva dijela. Prvi dio je implicitno preslikavanje podataka u prostor značajki pomoću jezgrine funkcije. Odabir preslikavanja ovisit će o specifičnosti podataka i znanju domene promatranog problema, odnosno znanju koje uzorke možemo očekivati među tim podacima. Drugi dio je otkrivanje uzoraka u prostoru značajki pomoću algoritama opće namjene koji se temelje na linearnoj algebri, geometriji i statistici. Takva modularnost povlači da se bilo koja jezgra može kombinirati s bilo kojim algoritmom i obratno.

Unatoč tome što ćemo se primjerom ograničiti na specijalan slučaj učenja s podrškom, vidjet ćemo kako se svaka primjena jezgrinih funkcija razlaže na dva dijela od kojih svaki ima dva koraka.

1.
   - Podaci se ugrađuju u prostor značajki.
   - Uočavaju se linearne relacije između točaka u prostoru značajki.
2.
   - Algoritmi se implementiraju tako da koordinate ugrađenih točaka nisu potrebne, već samo njihov skalarni produkt.
   - Skalarni produkt se izračunava efikasno, direktno iz originalnih točaka pomoću jezgrine funkcije.

Koraci su ilustrirani slikom 2.1. Kasnije ćemo vidjeti da unatoč tome što smo se ograničili na algoritme koji optimiziraju linearne funkcije, jezgrine funkcije omogućavaju razvoj široke lepeze efikasnih i dobro definiranih metoda za otkrivanje nelinearnih relacija između podataka koristeći nelinearna preslikavanja.



Slika 2.1: Ilustracija djelovanja jezgrine funkcije

Promotrimo funkciju

$$\phi : x \in \mathbb{R}^2 \longmapsto \phi(x) \in F \subseteq \mathbb{R}^2.$$

Funkcija ϕ preslikava podatke u prostor značajki gdje se nelinarni uzorak sada čini linearnim. Jezgra zatim računa skalarni produkt u prostoru značajki direktno iz početnih podataka.

Ridge[1] regresija

Neka je $S = \{(x_1, y_1), \ldots, (x_l, y_l)\}$, $\forall l \in \mathbb{N}$ skup točaka pri čemu su x_i iz $X \subseteq \mathbb{R}^n$ sa odgovarajućim oznakama y_i iz $Y \subseteq \mathbb{R}$, $\forall i \in \{1, 2, \ldots, l\}$ i $n \in \mathbb{N}$. Sa $x = (x_1, x_2, \ldots, x_n)$ označavat ćemo n-dimenzionalane ulazne vektore. Nadalje, neka je $g : \mathbb{R}^n \to \mathbb{R}$ realna linearna funkcija definirana kao

$$g(x) = \langle w, x \rangle = wx' = \sum_{i=1}^{n} w_i x_i.$$

Ridge regresija je problem

$$\min_w \mathcal{L}_\lambda(w, S) = \min_w \lambda \|w\|^2 + \sum_{i=1}^{l} (y_i - g(x_i))^2 \tag{2.1}$$

[1] eng. Greben

pri čemu je λ pozitivan realan broj koji predstavlja kompromis između točnosti i jednostavnosti modela. Problem učenja je sveden na rješavanje problema optimizacije nad $\mathbb{R}^n$. Iz nužnog uvjeta postojanja ekstrema deriviranjem po w dobivamo

$$X'Xw + \lambda w = (X'X + \lambda I_n)w = X'y \tag{2.2}$$

pri čemu smo sa X označili matricu čiji su retci vektori $x_1, \ldots, x_l$, sa y vektor $(y_1, \ldots, y_l)'$ a s I_n $n \times n$ jediničnu matricu. U ovom slučaju za matricu $(X'X + \lambda I_n)$ postoji inverz ako je $\lambda > 0$ tako da je rješenje dano s

$$w = (X'X + \lambda I_n)^{-1}X'y. \tag{2.3}$$

w dobivamo rješavanjem sustava 2.2, a vremenska složenost tog rješenja je $O(n^3)$. Rezultirajuća prediktivna funkcija dana je s

$$g(x) = \langle w, x \rangle = y'X(X'X + \lambda I_n)^{-1}x.$$

Primijetimo da iz 2.2 slijedi

$$w = \lambda^{-1}X'(y - Xw) = X'\alpha,$$

čime smo pokazali da se w može zapisati kao linearna kombinacija točaka za učenje, $w = \sum_{i=1}^{l} \alpha_i x_i$ i $\alpha = \lambda^{-1}(y - Xw)$. Iz toga slijedi:

$$\begin{aligned} \alpha &= \lambda^{-1}(y - Xw) \\ &\Rightarrow \lambda\alpha = y - XX'\alpha \\ &\Rightarrow (XX' + \lambda I_l)\alpha = y \\ &\Rightarrow \alpha = (G + \lambda I_l)^{-1}y, \end{aligned} \tag{2.4}$$

pri čemu je $G = XX'$, odnosno $G_{ij} = \langle X_i, X_j \rangle$. Izračunati α znači riješiti l linearnih jednadžbi sa l nepoznanica čija je vremenska kompleksnost $O(l^3)$. Prediktivna funkcija dana je sa

$$g(x) = \langle w, x \rangle = \langle \sum_{i=1}^{l} \alpha_i x_i, x \rangle = \sum_{i=1}^{l} \alpha_i \langle x_i, x \rangle = y'(G + \lambda I_l)^{-1}k,$$

pri čemu je $k = \langle x_i, x \rangle$. Pokazali smo da optimizacijski problem 2.1 možemo riješiti na dva načina. Prvi način dan izrazom 2.3 eksplicitno računa težinski vektor i nazivamo ga *primarno rješenje*, dok nam izraz 2.4 daje rješenje kao linarnu kombinaciju točaka za učenje. Njega nazivamo *dualno rješenje*, a parametre α_i nazivamo *dualni parametri*. Bitno je primijetiti da se u dualnom rješenju potrebne informacije točaka za učenje nalaze u Grammovoj matrici $G = XX'$. Grammova matrica i matrica $G + \lambda I_l$ su dimenzije $l \times l$. Ako

prostor značajki ima dimenziju n mnogo veću od broja točaka l nad kojima učimo onda je efikasnije riješiti izraz 2.4 nego izraz 2.3 koji u sebi ima matricu $X'X + \lambda I_n$ dimenzija $n \times n$. Mogli bismo sada kritizirati to što evaluacija dualnog rješenja zahtijeva više operacija od primarnog, ali unatoč tome što je ta evaluacija vremenske složenosti $O(nl)$, a one primarnog rješenja $O(n)$, kasnije ćemo vidjeti da dualno rješenje ima određene prednosti nad primarnim. Da bismo utvrdili te prednosti promotrimo funkciju

$$\phi : x \in \mathbb{R}^n \longmapsto \phi(x) \in F \subseteq \mathbb{R}^N$$

gdje su $n, N \in \mathbb{N}$. Cilj je izabrati takvu funkciju ϕ koja će pretvoriti nelinearne relacije u linearne. Primjenom funkcije ϕ naš će se početni skup ulaznih podataka S preslikati u $\{(\phi(x_i), y_i), \cdots, (\phi(x_l), y_l)\}$. Promotrimo sada funkciju

$$f(x, y) = |y - g(x)| = |u - \langle w, \phi(x)\rangle| = |\xi|.$$

Iako bi se ovdje moglo primijeniti primarno rješenje, mogli bismo naići na probleme ukoliko je N jako veliki što će učiniti rješenje $N \times N$ sustava jednadžbi skupim za izračunavanje. S druge strane, ukoliko se odlučimo za dualno rješenje, pokazali smo da su algoritmu potrebni samo skalarni produkti točaka $\langle \phi(X), \phi(z)\rangle$ u prostoru značajki F. Specijalno, prediktivna funkcija $g(x) = y'(G + \lambda I_l)^{-1}k$ sadrži Grammovu matricu $G = XX'$ s elementima

$$G_{ij} = \langle \phi(x_i), \phi(x_j)\rangle, \tag{2.5}$$

gdje su retci matrice X vektori $\phi(x_1)', \cdots, \phi(x_l)'$ i vektor k sadrži vrijednosti

$$k_i = \langle \phi(x_i), \phi(x)\rangle. \tag{2.6}$$

Dakle, u slučaju prevelikog N razumno je izbjeći izračunavanje $N \times N$ sustava koristeći dualno rješenje. Ukoliko optimistično pretpostavimo da je vremenska složenost izračunavanja funkcije ϕ jednaka $O(N)$ te da je vremenska kompleksnost izračunavanja skalarnih produkta 2.5 i 2.6 isto $O(N)$ onda se izračunavanje vektora α odvija u vremenu

$$O(l^3 + l^2 N), \tag{2.7}$$

dok se izračunavanje g na novoj točki odvija u vremenu

$$O(l^N). \tag{2.8}$$

Vidjeli smo da u dualnom rješenju koristimo skalarni produkt u prostoru značajki. U gornjoj analizi pretpostavili smo da je kompleksnost izračunavanja svakog skalarnog produkta proporcionalna dimenziji prostora značajki. No, skalarni produkti se ponekad mogu izračunati direktno koristeći početne točke bez eksplicitnog izračunavanja funkcije ϕ. Funkcija koja obavlja takvo direktno izračunavanje naziva se *jezgrina funkcija*.

Definicija 2.1.1. *Za funkciju k takvu da vrijedi*

$$\kappa(x, z) = \langle \phi(x), \phi(z) \rangle, \forall x, z \in X$$

kažemo da je jezgra pri čemu je ϕ preslikavanje iz X u prostor značajki F

$$\phi : x \longmapsto \phi(x) \in F.$$

U sljedećem primjeru dat ćemo jednu generaliziranu jezgrinu funkciju čija je složenost manja od dimenzije prostora značajki te ćemo opravdati njezino korištenje u algoritmu *Ridge* regresije tako što će smanjiti njegovu složenost.

Primjer 2.1.2. *Neka je $X \subseteq \mathbb{R}^n$ neki n-dimenzionalan prostor s preslikavanjem*

$$\phi : x \longmapsto \phi(x) = (x_i x_j)_{i,j=1}^n \in F = \mathbb{R}^{n^2}.$$

Pokažimo da je tada $\kappa(x, z) = \langle x, z \rangle^2$ odgovarajuća jezgrina funkcija.

$$\begin{aligned} \langle \phi(x), \phi(z) \rangle &= \langle (x_i x_j)_{i,j=1}^n, (z_i z_j)_{i,j=1}^n \rangle \\ &= \sum_{i,j=1}^n x_i x_j z_i z_j = \sum_{i=1}^n x_i z_i \sum_{j=1}^n x_j z_j \\ &= \langle x, z \rangle^2. \end{aligned}$$

Takvu jezgrinu funkciju nazivamo polinomna jezgra. Ukoliko sada iskoristimo tu jezgru u dualnom rješenju algoritma Ridge regresije, složenost izračunavanja vektora α je $O(l^3 + l^2 n)$ umjesto $O(l^3 + l^2 n^2)$ koliko smo izračunali u 2.7 i 2.8.

Primjerom smo pokazali kako jezgrine funkcije mogu poboljšati vremensku složenost izračunavanja skalarnog produkta u prostoru značajki što čini algoritme u kojima se mogu primijeniti efikasnijima u višedimenzionalnim prostorima značajki. Primijetimo da nismo morali odabrati točno tu jezgru za promatrani algoritam. Jasno je da smo mogli odabrati bilo koju koja bi odgovarala promatranim podacima. Slično, jezgru smo mogli primijeniti i u nekom drugom algoritmu koji od podataka za učenje očekuje samo njihov skalarni produkt. Takva modularnost nam omogućuje da odvojimo oblikovanje i analizu algoritama od jezgrinih funkcija.

Prije nego što započnemo klasifikaciju jezgrinih funkcija navest ćemo teorem koji kaže kada je jezgra dobro definirana.

Teorem 2.1.3 (Mercerov uvjet)**.** *Kažemo da je jezgrina funkcija κ dobro definirana ako vrijedi:*

- *postoji Hilbertov prostor F u kojem je κ njegov skalarni produkt*
- *κ je pozitivno definitna funkcija za koju vrijedi*

$$\iint f(x)\kappa(x, z)f(z)dxdz > 0 \quad (\forall f \in L_2).$$

2.2 Osnovni tipovi

Polinomne jezgre

Već smo se susreli s polinomnom jezgrom u primjeru 2.1.2. Sada ćemo definirati izvedenu polinomnu jezgru jer će biti poveznica s onima koje će slijediti.

Definicija 2.2.1. *Izvedena polinomna jezgra za jezgru κ_1 je definirana kao*

$$\kappa(x, z) = p(\kappa_1(x, z)), \tag{2.9}$$

pri čemu je $p(\cdot)$ bilo koji polinom s pozitivnim koeficijentima. Često se referira i na specijalan slučaj

$$\kappa_d(x, z) = (\langle x, z\rangle + R)^d, \tag{2.10}$$

definiran nad vektorskim prostorom X dimenzije n pri čemu su R i d parametri.

Partitivne jezgre

Pretpostavimo da su nam ulazni podaci elementi partitivnog skupa $\mathcal{P}(\{1, 2, \ldots, n\})$. Tada naš prostor značajki čine značajke ϕ_A za svaki $A \subseteq \mathcal{P}(\{1, 2, \ldots, n\})$. Možemo ih i prikazati kao što smo napravili i u polinomnim jezgrama na sljedeći način:

$$\phi_i(x) = x_i^{i_1} x_2^{i_2} \ldots x_n^{i_n},$$

pri čemu je $i = (i_1, \ldots, i_n) \in \{0, 1\}^n$. Značajka ϕ_A je dana množenjem svih elemenata skupa A

$$\phi_A(X) = \prod_{i \in A} x_i.$$

Definicija 2.2.2. *Partitivna jezgra definirana je preslikavanjem*

$$\phi : x \longmapsto (\phi_A(x))_{A \in 1,\ldots,n},$$

s pripadajućom jezgrom $\kappa_\subseteq(x, z)$

$$\kappa_\subseteq(x, z) = \langle \phi(x), \phi(z)\rangle.$$

Sljedeći primjer daje nam odgovor na to kako se ova jezgra izračunava.

Primjer 2.2.3. *Partitivna jezgra se izračunava kao*

$$\kappa_\subseteq(x, z) = \prod_{i=1}^{n} (1 + x_i z_i).$$

$$\kappa_\subseteq(x, z) = \langle \phi(x), \phi(z)\rangle = \sum_{A \subseteq \{1,\ldots,n\}} = \phi_A(x), \phi_A(z) = \sum_{A \subseteq \{1,\ldots,n\}} \prod_{i \in A} x_i z_i = \prod_{i=1}^{n} (1 + x_i z_i).$$

Gaussove jezgre

Gaussove jezgre su najraširanije i opsežno se proučavaju i u ostalim srodnim područjima.

Definicija 2.2.4. *Za $\sigma > 0$ Gaussova jezgra definirana je kao*

$$\kappa(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right).$$

Primijetimo da ne trebamo koristiti samo Euklidsku normu. Na primjer, neka je $\kappa(x, z)$ jezgra s odgovarajućim preslikavanjem ϕ_1 u prostoru značajki F_1. Ukoliko vrijedi

$$\|\phi_1(x) - \phi_1(z)\|^2 = \kappa_1(x, x) - 2\kappa_1(x, z) + \kappa_1(z, z),$$

možemo konstruirati deriviranu Gaussovu jezgru kao

$$\kappa(x, z) = \exp\left(-\frac{\kappa_1(x, x) - 2\kappa_1(x, z) + \kappa_1(z, z)}{2\sigma^2}\right).$$

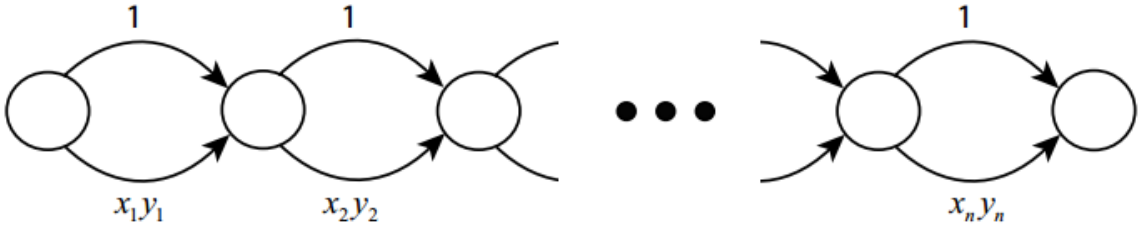
Parametar σ kontrolira na koliko će ulaznih podataka jezgra djelovati. Male vrijednosti σ omogućuju algoritmima u kojima se koriste da odrede više vrijednosti čime naravno riskiraju *overfitting*. U takvim slučajevima jezgrina matrica je slična jediničnoj matrici. S druge strane, velike vrijednosti σ postupno približavaju jezgru konstantnoj funkciji čime će učiniti učenje nemogućim bilo kojem algoritmu.

Jezgre grafova

Za jezgre koje smo dosada spomenuli nikako ne možemo reći da ih možemo primijeniti i u generalnom slučaju, odnosno, njihovo djelovanje je usko vezano uz strukturu podataka nad kojima djeluju. U idealnom slučaju, željeli bismo proširiti prostor djelovanja jezgre bez utjecanja na njezino efikasno izračunavanje. Prije nego pokažemo kako bismo to napravili, prvo ćemo pokazati kako se izračunavanje partitivne jezgre može prikazati pomoću grafa. Ilustrirajmo izračunavanje

$$\kappa_{\subseteq}(x, z) = \prod_{i=1}^{n}(1 + x_i z_i)$$

grafom na slici 2.2. Jasno se vidi kako su značajke jezgre prikazane kao put kojem je početni vrh prvi s lijeva, a završni je prvi s desna. Svaki takav put mora proći kroz vrh koristeći njegov gornji ili donji brid. Prolazak kroz donji brid označava uključivanje značajke, dok prolazak kroz gornji znači da se ona izostavlja. Jasno je da takvih puteva ima 2^n, pri čemu je n broj promatranih značajki. Promotrimo sada usmjereni graf $G = (V, E)$ gdje je V skup vrhova u kojem je sa s označen početni vrh, a s t završni, a E skup usmjerenih bridova označenih osnovnim jezgrama. Za partitivnu jezgru to su ili konstante ili $x_i z_i$ za neki i. Sada možemo dati formalnu definiciju:



Slika 2.2: Graf partitivne jezgre

Definicija 2.2.5. *Graf jezgre je usmjereni graf $G = (V, E)$ s početnim vrhom s i završnim vrhom t. Nadalje, svaki vrh e je označen jezgrom κ_e. Neka je P_{st} skup puteva iz vrha s u vrh t i neka je za put $p = (u_0u_1 \dots u_d)$*

$$\kappa_p(x, z) = \prod_{i=1}^{d} \kappa_{(u_{i-1} \to u_i)}(x, z)$$

produkt jezgri bridova iz p. Sada imamo sve što je potrebno da bismo definirali jezgru grafa G

$$\kappa_G(x, z) = \sum_{p \in P_{st}} \kappa_p(x, z) = \sum_{p \in P_{st}} \prod_{i=1}^{d} \kappa_{(u_{i-1} \to u_i)}(x, z).$$

Radi jednostavnosti, pretpostavljamo da niti jedan vrh nema brid u samog sebe.

Jezgre na skupovima

Promotrimo prostor partitivnog skupa $\mathcal{P}(D)$ fiksne domene D. Jasno je da su značajke u ovom slučaju skupovi. Neka je

$$X \subseteq \mathcal{P}(D)$$

i $A_1, A_2 \subseteq X$. Primjer jezgre definirane na A_1, A_2 dat ćemo sa

$$\kappa(A_1, A_2) = 2^{|A_1 \cap A_2|}.$$

Pretpostavimo da je nad skupom D definirana mjera ili funkcija gustoće μ. Ako je D konačan to je jednostavno preslikavanje iz D u $\mathbb{R}^+$. Za beskonačan D trebali bismo znati tehničke detalje koji uključuju poznavanje σ-algebre izmjerivih skupova. U svakom slučaju μ se definira kao

$$\mu(f) = \int_D f(a) d\mu(a).$$

Definicija 2.2.6. *Jezgru presjeka definiramo kao*

$$\kappa_{\cap}(A_1, A_2) = \mu(A_1 \cap A_2), \quad A_1, A_2 \subseteq D.$$

Kako bismo vizualizirali kakva je to jezgra promotrimo prostor značajki svih izmjerivih funkcija sa skalarnim produktom definiranim kao

$$\langle f_1, f_2 \rangle = \int_D f_1(a) f_2(a) d\mu(a).$$

Preslikavanje značajki je dano sa

$$\phi : A \longmapsto I_A,$$

što implicira sljedeće

$$\begin{aligned}\kappa_{\cap}(A_1, A_2) = \mu(A_1 \cap A_2) &= \int_D I_{A_1 \cap A_2}(a) d\mu(a) \\ &= \int_D I_{A_1}(a) I_{A_2}(a) d\mu(a) = \langle I_{A_1}, I_{A_2} \rangle \\ &= \langle \phi(A_1), \phi(A_2) \rangle.\end{aligned}$$

Definicija 2.2.7. *Za $\mu(D) = 1$, definiramo jezgru unije komplementa kao*

$$\tilde{\kappa}(A_1, A_2) = \mu((D \backslash A_1) \cap (D \backslash A_2)) = 1 - \mu(A_1 \cup A_2).$$

Postoji mnogo slučajeva u kojima bismo htjeli prikazati podatke kao skupove. Potencijalno možemo pronaći dualnost, budući da objekte prikazujemo elementima koje sadržava, dok ti elementi ujedno opet mogu biti prezentirani skupom elemenata. Takvu dualnost ćemo iskoristiti pri opisivanju jezgri za tekst, a sada ćemo dati konkretan primjer kako nestrukturirane podatke prezentirati skupovima.

Primjer 2.2.8. *Promotrimo neki dokument kao skup riječi rječnika D. Sličnost između dva dokumenta d_1 i d_2 će biti definirana kao mjera njihovih presjeka. Važnost neke riječi također može biti uzeta u obzir pri izračuvanju mjere presjeka. Također možemo uspoređivati riječi tako da definiramo funkciju nad dokumentima koja govori u kojim dokumentima se ona pojavljuje. Tako smo dobili dualnost koju možemo iskoristi na više načina, a detalje tog pristupa ćemo vidjeti u sekciji o jezgrama za tekst.*

2.3 Jezgre za tekst

Od prošlog desetljeća svjedočimo sve većem broju dostupnih digitalnih tekstova što je dovelo do toga da je njegovo ručno analiziranje i klasificiranje postalo gotovo nemoguće. Stoga je u znanstvenim poljima poput strojnog učenja u fokusu proučavanja postalo automatsko procesiranje tekstualnih dokumenata.

Semantički prostor

Iako zadatak automatskog analiziranja značenja dokumenta spada pod procesiranje prirodnog jezika, zajednica koja se bavila izvlačenjem podataka (IR[2]) razvila je relativno jednostavnu reprezentaciju dokumenta koja je danas najraširenija kada se želi odrediti njegova tema. Ta reprezentacija poznata je kao model vektorskog prostora (VSM[3]) ili jednostavnije reprezentacija *vrećom riječi*. Tom tehnikom se tvori baza jezgrinih funkcija kojima ćemo se baviti u ovom poglavlju. Pokazat ćemo kako koristeći VSM konstruirati jezgre s kojima će osim izvlačenja podataka biti moguće rješavati probleme poput analize sličnosti, klasifikacije i klasteriranja.

Reprezentacija teksta

Najjednostavnija moguća verzija VSM-a je reprezentacija dokumenta *vrećom riječi*. Vreća je zapravo multiskup tako da se ne uzima u obzir samo pristutnost neke riječi već i njezina frekvencija pojavljivanja. Dakle, dokument određuju riječi od kojih se sastoji pri čemu se ignorira njihov poredak i interpunkcija. To implicira da se dio informacija o gramatici gubi.

Definicija 2.3.1. *Riječ ili term je niz slova osnovnog alfabeta. Za skup dokumenata kažemo da čine zbirku, a za skup termova koji se pojavljuju u zbirci kažemo da čine riječnik.*

Kako smo već rekli, možemo promatrati dokument kao *vreću* termova odnosno *vreću riječi*. U sljedećoj definiciji ćemo reći kako prikazati *vreću* kao vektor.

Definicija 2.3.2. *Neka je d neki dokument i $t_1, t_2, \ldots, t_N$ termi koji se pojavaljuju u dokumentu d. Preslikavanje dokumenta u prostor dimenzije N definiramo kao:*

$$\phi : d \longmapsto \phi(d) = (tf(t_1, d), tf(t_2, d), \ldots, tf(t_N, d)) \in \mathbb{R}^N,$$

pri čemu je $tf(t_i, d)$ frekvencija pojavljivanja terma t_i u dokumentu d za svaki $i \in 1, 2, \ldots, N$ i $N \in \mathbb{N}$ veličina rječnika.

Primijetimo da će N uglavnom biti jako veliki broj, ali unatoč velikoj dimenziji rječnika vektor pridružen dokumentu je rijedak, odnosno većina elemenata u njemu će biti jednaka 0 jer će se riječi pojavljivati neznačajno mnogo puta.

Definicija 2.3.3. *Matricu*

$$D = \begin{pmatrix} tf(t_1, d_1) & \cdots & tf(t_N, d_1) \\ \vdots & \cdots & \vdots \\ tf(t_1, d_l) & \cdots & tf(t_N, d_l) \end{pmatrix}$$

[2] *Information retrieval*
[3] *Vector space model*

nazivamo dokument-term matrica, pri čemu je (i, j)*-ti element matrice frekvencija pojavljivanja terma* t_j *u dokumentu* d_i*,* $i \in \{1, 2, \ldots, l\}$ i $j \in \{1, 2, \ldots, N\}$*. Ukoliko transponiramo matricu D dobivamo term-dokument matricu. Term-term matrica je dana sa* $D'D$*, a dokument-dokument matrica sa* DD'*.*

Zanimljivo svojstvo VSM-a je dodatna interpretacija dualnosti između reprezentacija jezgrinih algoritama. U ovom slučaju, dualna reprezentacija odgovara tome da problem promatramo kroz dokument, dok primarna znači da ga promatramo kroz termove. Dokument možemo promatrati kao broj termova koji se u njemu pojavljuju što odgovara retcima dokument-term matrice, ali isto tako term možemo promatrati kao broj dokumenata u kojima se pojavljuje što odgovara retcima term-dokument matrice. Dimenzija zbirke će uglavnom biti manja od dimenzije rječnika, pa će ponekad biti korisno izračunavanje u dualnoj reprezentaciji, dok je za svrhu promatranja interpretacije bolje raditi s primarnom reprezentacijom.

Semantika teksta

Primijetili smo da reprezentacija dokumenta VSM-om ignorira semantičke veze između riječi. Kako bismo unaprijedili naš prostor vektora potrebno je osigurati da se dokumenti koji sadržavaju semantički ekvivalentne riječi preslikavaju u slične vektore značajki. Na primjer, riječi koje su sinonimi nalaze se u vektoru kao dva različita elementa i takvu situaciju bismo htjeli izbjeći. U ovoj podsekciji ćemo izložiti tehnike kojima je cilj riješiti takve i slične probleme.

Prva tehnika je pridruživanje različitih težina w_i promatranim termima. Jednostavan primjer toga je pridruživanje veznicima koji nemaju važno značenje poput *a, ali, i, ...* težine 0.

Druga tehnika je normaliziranje vektora značajki. Naime, duljine promatranih dokumenata mogu utjecati na dobivene rezultate jer što je neki dokument dulji to sadrži više riječi te je norma pridruženog vektora veća. Ako duljina dokumenta nije relevantna za rješavanje problema, na primjer u problemu određivanja teme dokumenta, smisleno je normalizirati pridruženi vektor. Primjer jezgre koja uklanja problem duljine dokumenta dana je sa

$$\tilde{\kappa}(x, y) = \left\langle \frac{\phi(x)}{\|\phi(x)\|}, \frac{\phi(y)}{\|\phi(y)\|} \right\rangle = \frac{\kappa(x, y)}{\sqrt{\kappa(x, x)\kappa(y, y)}}.$$

Jezgre vektorskog prostora

Definicija 2.3.4. *Unutar VSM-a možemo definirati matricu jezgre*

$$K = DD'$$

koja odgovara jezgri vektorskog prostora

$$\kappa(d_1, d_2) = \langle \phi(d_1), \phi(d_2) \rangle = \sum_{j=1}^{N} tf(t_j, d_1) tf(t_j, d2).$$

Kako bismo izračunali jezgru κ prvo moramo pretvoriti dokument u listu termova koje sadrži. Taj proces nazivamo *tokenizacija*. To je fundamentalna tehnika procesiranja u računarstvu još od implementacije prvih kompajlera. Svakom termu iz zbirke se pridružuje jedinstveni prirodni broj što nam omogućuje sortiranje termova zajedno s pridruženim brojem pojavljivanja. Tako smo dokument pretvorili u listu $L(d)$ što je praktičnije nego promatrati vektor $\phi(d)$. Sada možemo jednostavno i efikasno izračunati

$$\kappa(d_1, d_2) = A(L(d_1), L(d_2)),$$

koristeći liste $L(d_1)$ i $L(d_2)$ kao ulazne podatke algoritma $A(\cdot, \cdot)$ koji iterira listama i računa umnoške frekvencija kad god broj termova odgovara. Ukratko, izračunavanje jezgre ne uključuje eksplicitno izračunavanje vektora značajki $\phi(d)$, ali koristi posrednu listu $L(d)$. Iako se nalazimo u prostoru koji je tipično velike dimenzije, izračunavanje slika i odgovarajućih skalarnih produkata se može implementirati u vremenu proporcionalnom sumi duljina dva dokumenta

$$O(|d_1| + |d_2|).$$

U ovom poglavlju ograničit ćemo se na promatranje linearnih transformacija osnovnog VSM-a. Moguće je uzeti u obzir nelinearna preslikavanja koristeći standardne jezgrine konstrukcije. Na primjer, polinomna jezgra nad reprezentacijom normalizirane vreće riječi

$$\tilde{\kappa}(d_1, d_2) = (\kappa(d_1, d_2) + 1)^d = (\langle \phi(d_1), \phi(d_2) \rangle + 1)^d,$$

koristi sve n-torke riječi za $0 \leq n \leq d$ kao značajke. Isti pristup bi se mogao iskoristiti za bilo koju jezgru vektorskog prostora koristeći polinomne jezgre ili čak primjerice Gaussove jezgre.

Oblikovanje semantičkih jezgri

Reprezentacija vrećom riječi nije idealna uglavnom jer se zanemaruju poredak i semantika riječi. Kako bi se riješila druga mana razmotrit ćemo transformaciju vektora dokumenta $\phi(d)$. VSM uzima u obzir samo jednostavan slučaj linearne transformacije $\tilde{\phi}(d) = \phi(d)S$, gdje je S dijagonalna, kvadratna ili općenito bilo koja $N \times k$ matrica, $N, k \in \mathbb{N}$. Koristeći ovu transformaciju odgovarajuća jezgra sada ima oblik

$$\tilde{\kappa}(d_1, d_2) = \phi(d_1) S S' \phi(d_2)' = \tilde{\phi}(d_1) \phi(\tilde{d}_2)'.$$

Matricu S nazivamo još i *semantička matrica*. Različiti odabiri matrice S vode različitim oblicima VSM-a. Na primjer, jedan mogući odabir matrice je

$$S = RP,$$

gdje je R dijagonalna matrica koja daje termima težine, dok je P *matrica bliskosti* koja definira semantičku vezu između termova zbirke.

Već smo spomenuli da nemaju sve riječi jednaku važnost pri određivanju teme nekog dokumenta. U problemima koji se bave izvlačenjem podataka, riječi poput veznika se odmah u startu eliminiraju iz analize. Entropija riječi u zbirci može se iskoristiti kako bi se odredila važnost riječi. To je "apsolutna" mjera jer ne uzima u obzir problem kojim se netko bavi. Primjerice, za kategorizaciju dokumenata, mogli bismo definirati relativnu mjeru važnosti neke riječi s obzirom na danu temu. Ovdje ćemo razmotriti apsolutnu mjeru poznatu kao *idf*[4] koja daje težinu termima funkcijom njihove *inverzne frekvencije dokumenta*. Pretpostavimo da imamo l dokumenata u nekoj zbirci i neka je $df(t)$ broj dokumenata koji sadrže term t. Mjera kojom se izračunava *inverzna frekvencija dokumenta* terma t dana je sa

$$w(t) = \ln\left(\frac{l}{df(t)}\right).$$

Primijetimo da se *idf*-om efikasno eliminaraju nevažne riječi poput veznika. Ukoliko se term pojavljuje u svakom dokumentu, tada je $df(t) = l$, odnosno $w(t) = 0$. No unatoč tome, preporučljivo je napraviti eksplicitnu listu nevažnih terma koji će se *apriori* eliminirati iz rječnika i time smanjiti prostor značajki.

Kako sada imamo alat s kojim možemo efikasno dati težine termima, možemo definirati novi VSM koristeći dijagonalnu matricu R s elementima

$$R_{tt} = w(t).$$

Pripadajuća jezgra tada izračunava skalarni produkt

$$\tilde{\kappa}(d_1, d_2) = \phi(d_1)RR'\phi(d_2)' = \sum_t w(t)^2 tf(t, d_1)tf(t, d_2),$$

pri čemu opet možemo primijeniti težinsku verziju A_w algoritma A:

$$\tilde{\kappa}(d_1, d_2) = A_w(L(d_1), L(d_2)).$$

Izračunavanje ove jezgre uključuje frekvenciju termova i inverznu frekvenciju dokumenta, stoga se često naziva *tf-idf* reprezentacija.

[4] *Inverse document frequency*

Tf-idf reprezentacija implementira eliminaciju irelevantih termova, kao i pronalazak potencijalno problematičnih, ali još uvijek nije u stanju otkriti kada su dva terma u semantičkoj vezi. Znači da ona nije u stanju uspostaviti vezu između dva dokumenta koji nemaju zajedničkih termova, iako primjerice obrađuju istu temu koristeći sinonime. Jedini način na koji se to može postići jest definiranjem semantičke bliskosti između termova. Unutar VSM-a to znači da zahtijevamo da matrica bliskosti P ima nenegativne nedijagonalne elemente $P_{ij} > 0$ kada je term i u semantičkoj vezi sa termom j. Tada imamo jezgru vektorskog prostora

$$\tilde{\kappa}(d_1, d_2) = \phi(d_1)PP'\phi(d_2)' \tag{2.11}$$

koja odgovara reprezentaciji dokumenta s manje rjeđim vektorom $\phi(d)P$ koji ima nenegativne elemente za sve termove koji su semantički slični onima u dokumentu $\dot{d}$. Alternativno, možemo promatrati matricu PP' kao enkodiranje semantičke snage između termova. Ako stavimo da je $Q = PP'$ i proširimo jednadžbu 2.11

$$\tilde{\kappa}(d_1, d_2) = \sum_{i,j} \phi(d_1)_i Q_{ij} \phi(d_2)_j,$$

možemo reći da Q_{ij} predstavlja enkodiranje količine semantičke veze između termova i i j.

2.4 Jezgre za strukturirane podatke

U ovoj sekciji pokazat ćemo kakve sve jezgre postoje za važan tip podatka - stringove. Oni svoje mjesto nalaze u primjenama u bioinformatici jer se njima mogu prezentirati proteini kao niz aminokiselina ili DNK kao niz nukleida.

Jezgre su razvijene kako bi izračunavale skalarne produkte slika stringova u prostorima značajki velikih dimenzija koristeći tehnike dinamičkog programiranja. Iako su nizovi specijalan slučaj opće klase strukturiranih podataka za koju su jezgre razvijene, dat ćemo im posebnu pažnju kako bismo naglasili njihovu važnost u primjenama.

Bavit ćemo se problemom preslikavanja dva niza u prostor velike dimenzije, ali na način da njihova relativna udaljenost u tom prostoru reflektira njihovu sličnost te da se njihov skalarni produkt može efikasno izračunati. Prva odluka koju ćemo morati donijeti je kako odrediti sličnost nakon preslikavanja. Hoćemo li pokušati grupirati nizove po duljini, kompoziciji ili nekom drugom svojstvu? U daljnjem tekstu vidjet ćemo više takvih preslikavanja koja se mogu koristiti ili individualno ili u kombinacijama kako bi zadovoljile potrebe primjene.

Definicija 2.4.1. *Alfabet je konačan skup* Σ *od* $|\Sigma|$ *simbola. Kažemo da je string*

$$s = s_1 \dots s_{|s|},$$

svaki niz simbola iz Σ*, uključujući i prazan simbol* ε*, jedini simbol duljine 0. Sa* Σ^n *označavamo skup svih stringova duljine n, a s* Σ^* *skup svih stringova*

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n.$$

Za stringove s, t *sa* $|s|$ *označavamo duljinu stringa s, a sa st označavamo string koji je nastao konkatenacijom stringova s i t. Kažemo da je string t podstring stringa s, ako postoje stringovi u i v takvi da*

$$s = utv.$$

Sve jezgre predstavljene u ovoj sekciji bit će definirane kao eksplicitna preslikavanja iz prostora svih konačnih nizova alfabeta Σ u vektorski prostor F. Vektori iz F su indeksirani podskupom $I \subseteq \Sigma$. U nekim slučajevima, I će biti skup stringova duljine p, odnosno Σ^p, pri čemu će dimenzija prostora biti $|\Sigma|^p$. U ostalim slučajevima I će biti beskonačan podskup od Σ^*. To formalnije možemo zapisati kao

$$\phi : s \longmapsto (\phi_u(s))_{u \in I} \in F.$$

Za svaki prostor F postojat će više funkcija ϕ između kojih ćemo moći izabrati onu koja nam je potrebna. Primjerice, vrijednost koordinate indeksirane stringom u $\phi_u(s)$ može biti broj pojavljivanja stringa u u stringu s kao podstring, dok je drugi izbor da prebrojimo koliko se elemenata stringa u pojavljuje u s.

Jezgre sprektra

Vjerojatno najprirodniji način usporedbe dva niza u mnogim primjenama je jednostavno prebrojavanje koliko imaju zajedničkih podstringova duljine p.

Definicija 2.4.2. *Kažemo da je spektar poretka p ili p-spektar stringa s histogram frekvencija svih njegovih podstringova duljine p.*

Usporedba dva *p-spektra* dva stringa nam može dati važne informacije o njihovoj sličnosti. Sada možemo definirati jezgru kao skalarni produkt *p-spektra*.

Definicija 2.4.3. *Neka je F prostor značajki indeksiran skupom* $I = \Sigma^p$ *i neka je dano preslikavanje*

$$\phi_u^p(S) = |\{(v_1, v_2) : s = v_1 u v_2\}|, u \in \Sigma^p.$$

Tada jezgru p-spektra definiramo kao

$$\kappa_p(s, t) = \langle \phi^p(s), \phi^p(t) \rangle = \sum_{u \in \Sigma^p} \phi_u^p(s) \phi_u^p(t).$$

Primjer 2.4.4. *Promotrimo dva stringa*

$$s = \text{"statistika"}, t = \text{"matematika"}.$$

Oni sadrže sljedeće podstringove duljine 3:

"sta", "tat", "ati", "tis", "ist", "sti", "tik", "ika"

"mat", "ate", "tem", "ema", "mat", "ati", "tik", "ika"

i vidimo da su im zajednički "ati", "tik", "ika"*, stoga je* $\kappa(s,t) = 3$.

Jezgru *p-spektra* možemo izračunati na više načina, svaki od njih ima različitu vremensku složenost. Iako ćemo sada pokazati primjer izračunavanja složenosti $O(p|s||t|)$, kasnije ćemo vidjeti kako se ta složenost može reducirati do $O(p(|s| + |t|)) = O(p\max(|s|, |t|))$.

Definirajmo najprije pomoćnu jezgru koja se naziva *k-sufiks* jezgra koju ćemo koristiti pri izračunavanju jezgre *p-spektra*.

Definicija 2.4.5. *k-sufiks jezgru definiramo kao*

$$\kappa_k^S(s,t) = \begin{cases} 1, & ako\ s = s_1 u, t = t_1 u, \text{za}\ \ u \in \Sigma^k \\ 0, & inače. \end{cases}$$

Jasno je da izračunavanje $\kappa_k^S(s,t)$ zahtijeva $O(k)$ usporedbi, stoga se jezgra *p-spektra* može izračunati pomoću

$$\kappa_p(s,t) = \sum_{i=1}^{|s|-p+1} \sum_{j=1}^{|t|-p+1} \kappa_p^S(s(i:i+p), t(j:j+p))$$

u $O(p|s||t|)$ operacija.

Jezgre svih podnizova

Definicija 2.4.6. *Neka je F prostor značajki indeksiran skupom* $I = \Sigma^*$ *i neka je dano preslikavanje*

$$\phi_u(s) = |\{i : u = s(i)\}|, u \in I,$$

koje nam govori koliko se puta string u pojavljuje kao poniz stringa s. Za

$$\kappa(s,t) = \langle \phi(s), \phi(t) \rangle = \sum_{u \in \Sigma^*} \phi_u(s)\phi_u(t)$$

kažemo da je jezgra svih podnizova.

Eksplicitno izračunavanje prethodno definiranog preslikavanja je neisplativo čak i ako prikažemo $\phi_u(s)$ kao listu nenegativnih elemenata jer ako je $|s| = m$ onda možemo očekivati

$$\min\left(\binom{m}{k}, |\Sigma|^k\right)$$

različitih podnizova duljine k. Stoga moramo razmotriti kako izračunati jezgru κ efikasnije od eksplicitnog izračunavanja vektora značajki. U tu svrhu, promotrimo značajku indeksiranu stringom u. Njezin doprinos u krajnjem skalarnom produktu se može izraziti sa

$$\phi_u(s)\phi_u(t) = \sum_{i:u=s(i)} 1 \sum_{(i,j):u=s(i)=t(j)} 1 = \sum_{(i,j):u=s(i)=t(j)} 1.$$

Iz toga slijedi da se krajnji skalarni produkt može zapisati kao

$$\kappa(s,t) = \langle \phi(s), \phi(t) \rangle = \sum_{u \in I} \sum_{(i,j):u=s(i)=t(j)} 1 = \sum_{(i,j):s(i)=t(j)} 1. \tag{2.12}$$

Kako bismo definirali rekurziju s kojom ćemo efikasno izračunati jezgru trebamo dodati simbol a na jedan od stringova s. U krajnjoj sumi izraza 2.12 dvije su mogućnosti za podniz i: ili ga s sadrži u potpunosti ili njegov zadnji indeks odgovara zadnjem a simbolu. Stoga možemo razdvojiti sumu na dva dijela

$$\sum_{(i,j):sa(i)=t(j)} 1 = \sum_{(i,j):s(i)=t(j)} 1 + \sum_{u:t=uav} \sum_{(i,j):s(i)=u(j)} 1, \tag{2.13}$$

pri čemu smo u drugom dijelu iskoristili činjenicu da je zadnji simbol podniza simbol a koji se stoga mora pojaviti na nekom mjestu u t. Sada možemo rekurzivo definirati efikasniju jezgru

$$\begin{aligned} \kappa(s,\varepsilon) &= 1, \\ \kappa(sa,t) &= \kappa(s,t) + \Sigma_{k:t_k=a}\kappa(s,t(1:k-1)). \end{aligned} \tag{2.14}$$

2.5 Jezgre iz generativnih modela

Pri proučavanju nekog problema često znamo nešto o tome kako se njegovi podaci stvaraju. Na primjer, DNK niz djeteta je kombinacija DNK nizova njegovih roditelja, a za tekstualni dokument možemo reći da se sastoji od riječi koje su važne za njegovu temu.

Modeli podataka mogu biti deterministički ili vjerojatnosni i više je načina s kojima ih možemo pretvoriti u jezgre. U ovoj sekciji naglasak će ipak biti na generativnim modelima podataka koji već postoje te ćemo definirati popularnu *P-jezgru*.

P-jezgre

Opći tip jezgre može se dobiti ukoliko definiramo zajedničku razdiobu $P(x, z)$ na parovima podataka (x, z) iz prostora $X \times X$ pri čemu je X konačan ili prebrojiv skup ulaznih podataka. Pretpostavljamo da zajednička razdioba $P(x, z)$ ne ovisi o poretku elemenata, odnosno P je simetrična, ali da bi bila jezgra

$$\kappa(x, z) = P(x, z),$$

mora zadovoljiti svojstvo pozitivne semi-definitnosti.

Definicija 2.5.1. *Za vjerojatnosnu razdiobu $P(x, z)$ nad prostorom $X \times X$ koja zadovoljava svojstvo konačne pozitivne semi-definitnosti kažemo da je P-jezgra nad skupom X.*

Neka je dana jezgra κ nad prebrojivo beskonačnim skupom X. Tada možemo definirati *P-jezgru* koja je jednaka κ

$$\sum_{x \in X} \sum_{z \in X} \kappa(x, z) = M < \infty,$$

budući da razdioba

$$P(x, z) = \frac{1}{M} \kappa(x, z)$$

zadovoljava definiciju *P-jezgre*.

S druge strane, neće sve razdiobe biti *P-jezgre*. Kako bismo to pokazali definirajmo skup $X = \{x_1, x_2\}$, s vjerojatnostima

$$P(x_1, x_1) = P(x_2, x_2) = 0$$
$$P(x_1, x_2) = P(x_2, x_1) = 0.5.$$

Matrica

$$\begin{pmatrix} 0 & 0.5 \\ 0.5 & 0 \end{pmatrix}$$

ima svojstvene vrijednosti $\lambda = \pm 0.5$ te je jasno da nije pozitivno semi-definitna.

Ukoliko promatramo *P-jezgre* nad neprebrojivim skupovima tada će one biti definirane vjerojatnosnom funkcijom gustoće $p(x, z)$ koje opet moraju zadovoljavati svojstvo konačne pozitivno semi-definitnosti i biti simetrične. Jezgra κ ekvivalentna *P-jezgri* bila bi

$$\int_X \int_X \kappa(x, z) dx dz = M < \infty.$$

Primjer 2.5.2. *Neka je P vjerojatnosna razdioba nad konačnim ili prebrojivim skupom X. Definirajmo 1-dimenzionalnu P-jezgru sa*

$$\kappa(x, z) = P(x)P(z).$$

Iz toga odmah slijedi da ta jezgra odgovara preslikavanju

$$\phi : x \longmapsto P(x).$$

Poglavlje 3

Algoritmi bazirani na jezgrama

U ovom poglavlju opisat ćemo algoritme koji se mogu primijeniti pri rješavanju klasifikacije, regresije te otkrivanja noviteta. Bit će definirani u okvirima konveksne optimizacije stoga neće imati problem zaglavljivanja u lokalnom minimumu. Opisat ćemo metodu potpornih vektora jer će ona imati glavnu ulogu u algoritmima za klasifikaciju i regresiju.

Na početku ćemo opisati kako pronaći najmanju hipersferu koja sadrži sve podatke za učenje u prostoru značajki te pripadajući algoritam za otkrivanje noviteta. Zatim ćemo vidjeti kako laganom prilagodbom prethodno opisanog možemo pronaći maksimalnu hiperravninu koja razdvaja dva skupa podataka što pak vodi algoritmima za regresiju.

3.1 Otkrivanje noviteta

Otkrivanje noviteta je identifikacija novih, odnosno nepoznatih podataka na temelju podataka za učenje. U ovoj sekciji opisat ćemo kako pomoću hipersfere odlučiti je li neki podatak nepoznat. Intuitivno je da će nepoznati podaci biti izvan hipersfere, dok će podaci za učenje i njima slični biti unutar nje.

Jasno je da što je manja hipersfera, to je točnija detekcija nepoznatih podataka. Stoga će naš cilj biti pronaći najmanju hipersferu za koju s velikom vjerojatnošću možemo garantirati da sadrži većinu podataka za učenje. Problem pronalaska najmanje hipersfere koja sadrži određeni dio podataka za učenje je nažalost NP težak, što znači da nisu poznati efikasni algoritmi koji ga rješavaju u potpunosti točno. No, može biti riješen u potpunosti ako zahtjevamo da hipersfera sadrži sve podatke. Stoga ćemo se prvo baviti tim problemom.

Najmanja hipersfera koja sadrži sve podatke

Neka je $S = \{x_1, \ldots, x_l\}$ pri čemu su $x_i \in \mathbb{R}^n \ \forall i \in \{1, \ldots, l\}$, $l \in \mathbb{N}$, $n \in \mathbb{N}$ skup točaka za učenje, te F prostor značajki pripadajućim preslikavanjem $\phi : X \to F$ te jezgrom κ

$$\kappa(x, z) = \langle \phi(x), \phi(z) \rangle .$$

Središte najmanje hipersfere koja sadrži S je točka c koja minimizira udaljenost r od najdalje točke iz S odnosno

$$c^* = \underset{c}{\operatorname{argmin}} \max_{1 \le i \le l} \|\phi(x_i) - c\|.$$

Sada možemo formalno definirati najmanju hipersferu koja sadrži sve točke iz S.

Definicija 3.1.1. *Za skup vektora*

$$S = \{x_1, \ldots, x_l\}$$

hipersfera (c,r) koja rješava problem

$$\begin{aligned} &\text{minimiziraj} && r^2 \\ &\text{uz ograničenja} && \|\phi(x_i) - c\|^2 = (\phi(x_i) - c)'(\phi(x_i) - c) \le r^2 \\ & && i = 1, \ldots, l \end{aligned} \tag{3.1}$$

je hipersfera koja sadrži S s najmanjim radijusom r.

Probleme optimizacije uz ograničenja ovog tipa možemo riješiti definirajući *Lagrangeovu* funkciju pri čemu uvodimo *Lagrangeove* multiplikatore α_i za svako ograničenje $i \in \{1, \ldots, l\}$

$$L(c, r, \alpha) = r^2 + \sum_{i=1}^{l} \alpha_i \left[\|\phi(x_i) - c\|^2 - r^2 \right].$$

Iz nužnog uvjeta postojanja ekstrema deriviranjem po c i r dobivamo

$$\frac{\partial L(c, r, \alpha)}{\partial c} = 2 \sum_{i=1}^{l} \alpha_i (\phi(x_i) - c) = 0,$$

$$\frac{\partial L(c, r, \alpha)}{\partial r} = 2r \left(1 - \sum_{i=1}^{l} \alpha_i \right) = 0,$$

iz čega onda slijedi

$$\sum_{i=1}^{l} \alpha_i = 1 \quad \text{i} \quad c = \sum_{i=1}^{l} \alpha_i \phi(x_i).$$

Druga jednakost implicira da središte najmanje hipersfere koja sadržava sve točke iz S uvijek leži u njihovom rasponu. To nam pokazuje da se središte može izraziti u dualnoj reprezentaciji. Nadalje, prva jednakost implicira da središte leži na konveksnoj ljusci točaka iz S. Njihovim uvrštavanjem u *Lagrangeovu* funkciju dobivamo

$$\begin{aligned}
L(c, r, \alpha) &= r^2 + \sum_{i=1}^{l} \alpha_i \left[\|\phi(x_i) - c\|^2 - r^2 \right] \\
&= \sum_{i=1}^{l} \alpha_i \langle \phi(x_i) - c, \phi(x_i) - c \rangle \\
&= \sum_{i=1}^{l} \alpha_i \left(\kappa(x_i, x_i) + \sum_{k,j=1}^{l} \alpha_j \alpha_k \kappa(x_j, x_k) - 2 \sum_{j=1}^{l} \alpha_j \kappa(x_i, x_j) \right) \\
&= \sum_{i=1}^{l} \alpha_i \kappa(x_i, x_i) + \sum_{k,j}^{l} \alpha_k \alpha_j \kappa(x_j, x_k) - 2 \sum_{i,j=1}^{l} \alpha_i \alpha_j \kappa(x_i, x_j) \\
&= \sum_{i=1}^{l} \alpha_i \kappa(x_i, x_i) - \sum_{i,j=1}^{l} \alpha_i \alpha_j \kappa(x_i, x_k).
\end{aligned}$$

Izrazili smo *Lagrangeovu* funkciju u potpunosti pomoću *Lagrangeovih* multiplikatora, nju nazivamo dualna *Lagrangeova* funkcija. Rješenje dobivamo njezinim maksimiziranjem.

U nastavku slijedi pseudokod algoritma 3.1 za pronalazak najmanje hipersfere pri čemu sa $\mathcal{H}$ označavamo *Heavsidovu* funkciju

$$\mathcal{H}(x) = \begin{cases} 1, & \text{ako} \quad x \geq 0, \\ 0, & \text{inače.} \end{cases}$$

Iz *Karush-Kuhn-Tuckerovih* uvjeta slijede jednakosti

$$\alpha_i^* \left[\|\phi(x_i) - c^*\|^2 - {r^*}^2 \right] = 0, \quad i = 1, \ldots, l.$$

To implicira da samo vektori x_i iz skupa S koji leže na optimalnoj hipersferi imaju nenegativne odgovarajuće *Lagrangeove* multiplikatore α_i^*. Za ostale vektore vrijedi da je njihov odgovarajući multiplikator jednak nuli. Znači da su u izrazu za optimalni centar hipersfere važni samo vektori na njezinoj površini. To je razlog zašto ih se naziva potpornim vektorima.

Skup indeksa potpornih vektora označavat ćemo sa *sv*. Tako funkciju predikcije dobivenu iz 3.1 možemo zapisati kao

$$f(x) = \mathcal{H} \left[\kappa(x, x) - 2 \sum_{i \in sv} \alpha_i^* \kappa(x, x_i) + D \right]$$

Algoritam 3.1: Pseudokod za izračunavanje najmanje hipersfere

Ulaz: Skup za učenje $S = \{x_1, \ldots, x_l\}$

Izlaz: centar sfere c^* i prediktivna funkcija f

1 Nađi α^* kao rješenje problema:

$$\text{maksimiziraj} \qquad W(\alpha) = \sum_{i=1}^{l} \alpha_i \kappa(x_i, x_i) - \sum_{i,j=1}^{l} \alpha_i \alpha_j \kappa(x_i, x_j)$$

$$\text{uz ograničenja} \qquad \sum_{i=1}^{l} \alpha_i = 1 \text{ i } \alpha_i \geq 0, i = 1, \ldots, l.$$

2 $r^* = \sqrt{W(\alpha^*)}$

3 $D = \sum_{i,j=1}^{l} \alpha_i^* \alpha_j^* \kappa(x_i, x_j) - r^{*^2}$

4 $f(x) = \mathcal{H}\left[\kappa(x, x) - 2\sum_{i=1}^{l} \alpha_i^* \kappa(x_i, x) + D\right]$

5 $c^* = \sum_{i=1}^{l} \alpha_i^* \phi(x_i)$

Primijetimo da funkcija f vraća 1 ako se vektor x nalazi izvan pronađene hipersfere (smatramo ga novitetom), a 0 inače.

Hipersfere koje sadržavaju većinu podataka

Kako bismo pokazali robusniji algoritam otkrivanja noviteta definirajmo prvo *slack* varijable.

Definicija 3.1.2. *Neka je* $S = \{x_1, \ldots, x_l\}, x_i \in \mathbb{R}^n, n \in \mathbb{N}, i \in \{1, \ldots, l\}$ *skup vektora za učenje, te neka je* (c, r) *neka hipersfera sa središtem* $c \in \mathbb{R}^n$ *i radijsom* r*. Za* $\forall i \in \{1, \ldots, l\}$ *definiramo slack varijablu*

$$\xi_i = \left(\|c - \phi(x_i)\|^2 - r^2\right)_+,$$

pri čemu je $(x)_+ = x$ *ako je* $x \geq 0$*, a* 0 *inače. Sa* ξ *ćemo označavati vektor koji sadrži slack varijable.*

Za vektore koji se nalaze unutar hipersfere, *slack* varijabla bit će jednaka nuli, a za ostale će mjeriti koliko je kvadrat udaljenosti od središta veći od kvadrata radijusa. Ideja je da pomoću njih "olabavimo" ograničenja optimizacije i dopustimo da se neki vektori iz skupa za učenje nalaze izvan hipersfere. Sada možemo formalno definirati problem na kojem će se temeljiti sljedeći algoritam.

Definicija 3.1.3. *Za skup vektora*

$$S = \{x_1, \ldots, x_l\}$$

i parametar $C \in \mathbb{R}$ koji predstavlja kompromis između točnosti i radijusa sfere, hipersferu (c,r) koja rješava problem

$$\begin{aligned} &\text{minimiziraj} && r^2 + C\|\xi\| \\ &\text{uz ograničenja} && \|\phi(x_i) - c\|^2 = (\phi(x_i) - c)(\phi(x_i) - c)' \leq r^2 + \xi_i \\ & && \xi_i \geq 0, i = 1, \ldots, l \end{aligned} \tag{3.2}$$

nazivamo najmanja mekana hipersfera.

Kao i u prethodnoj sekciji uvođenjem *Lagrangeovih* multiplikatora dobivamo funkciju

$$L(c, r, \alpha, \xi) = r^2 + C \sum_{i=1}^{l} \xi_i + \sum_{i=1}^{l} \alpha_i \left[\|\phi(x_i) - c\|^2 - r^2 - \xi_i \right] - \sum_{i=1}^{l} \beta_i \xi_i,$$

te analogno kao i prije dobivamo

$$\begin{aligned} \frac{\partial L(c, r, \alpha, \xi)}{\partial c} &= 2 \sum_{i=1}^{l} \alpha_i (\phi(x_i) - c) = 0 \\ \frac{\partial L(c, r, \alpha, \xi)}{\partial r} &= 2r \left(1 - \sum_{i=1}^{l} \alpha_i \right) = 0 \\ \frac{\partial L(c, r, \alpha, \xi)}{\partial \xi_i} &= C - \alpha_i - \beta_i = 0. \end{aligned}$$

Uvrštavanjem u početnu funkciju dobivamo

$$\begin{aligned} L(c, r, \alpha, \xi) &= r^2 + C \sum_{i=1}^{l} \xi_i + \sum_{i=1}^{l} \left[\|\phi(x_i) - c\|^2 - r^2 - \xi_i \right] \\ &= \sum_{i=1}^{l} \alpha_i \langle \phi(x_i) - c, \phi(x_i) - c \rangle \\ &= \sum_{i=1}^{l} \alpha_i \kappa(x_i, x_i) - \sum_{i,j=1}^{l} \alpha_i \alpha_j \kappa(x_i, x_j) \end{aligned}$$

dualnu *Lagrangeovu* funkciju. Sada ćemo predstaviti algoritam za pronalazak najmanje mekane hipersfere.

Prediktivna funkcija f dobivena algoritmom 3.2 kao izlaz daje 1 za vektore koji se nalaze izvan sfere. Budući da smo smanjili njezinu veličinu bit će lakše otkriti vekotore koje su noviteti.

Algoritam 3.2: Pseudokod za izračunavanje najmanje mekane hipersfere

Ulaz: Skup za učenje $S = \{x_1, \ldots, x_l\}, \gamma > 0, C > 0$

Izlaz: centar sfere c^*, prediktivna funkcija f, radijus r^*, suma *slack* varijabli $\|\xi^*\|$

1 Nađi α^* kao rješenje problema:

$$\text{maksimiziraj} \qquad W(\alpha) = \sum_{i=1}^{l} \alpha_i \kappa(x_i, x_i) - \sum_{i,j=1}^{l} \alpha_i \alpha_j \kappa(x_i, x_j)$$

$$\text{uz ograničenja} \qquad \sum_{i=1}^{l} \alpha_i = 1 \text{ i } 0 \leq \alpha_i \leq C, i = 1, \ldots, l.$$

2 izaberi i tako da je $0 < \alpha_i^* < C$

3 $r^* = \sqrt{\kappa(x_i, x_i) - 2\sum_{j=1}^{l} \alpha_j^* \kappa(x_j, x_i) + \sum_{i,j=1}^{l} \alpha_i^* \alpha_j^* \kappa(x_i, x_j)}$

4 $D = \sum_{i,j=1}^{l} \alpha_i^* \alpha_j^* \kappa(x_i, x_j) - {r^*}^2 - \gamma$

5 $f(x) = \mathcal{H}\left[\kappa(x, x) - 2\sum_{i=1}^{l} \alpha_i^* \kappa(x_i, x) + D\right]$

6 $\|\xi^*\|_1 = (W(\alpha^*) - {r^*}^2)/C$

7 $c^* = \sum_{i=1}^{l} \alpha_i^* \phi(x_i)$

3.2 Klasifikacija pomoću metode potpornih vektora

Sada ćemo se posvetiti problemu klasifikacije. Za razliku od prethodne sekcije, u kojoj smo novitete tražili pomoću hipersfere, u ovoj ćemo klasifikaciju raditi pomoću hiperravnine. Prvo ćemo reći da klasifikatorima nazivamo algoritme koji implementiraju klasifikaciju podataka te ćemo definirati što je to *margina*.

Definicija 3.2.1. *Neka je* $S = \{(x_1, y_1), \ldots, (x_l, y_l)\}, x_i \in X \subseteq \mathbb{R}^n, y_i \in \{-1, 1\}, \forall i \in \{1, \ldots, l\}$ *skup podataka za učenje te neka je* (x, y) *neki element tog skupa. Za funkciju* $g : X \to \mathbb{R}$ *definiramo marginu na primjeru* (x, y) *kao vrijednost* $yg(x)$*. Funkcionalnu marginu skupa* S *definiramo kao*

$$m(S, g) = \min_{1 \leq i \leq l} y_i g(x_i).$$

Neka je $\gamma \in \mathbb{R}$ *neka margina. Tada sa* $\xi_i = \xi((x_i, y_i), \gamma, g)$ *označavamo vrijednost za koju funkcija* g *ne postiže marginu* γ *na nekom* $(x_i, y_i) \in S$*. Tu vrijednost još nazivamo i slack varijabla.*

Maksimalna margina

Pretpostavimo da za dani skup podataka

$$S = \{(x_1, y_1), \ldots, (x_l, y_l)\}, x_i \in \mathbb{R}^n, y_i \in \{-1, 1\}, \forall i \in \{1, \ldots, l\}$$

postoji linearna funkcija

$$g(x) = \langle w, \phi(x_i)\rangle + b$$

određena vektorom w i brojem b te da postoji $\gamma > 0$ tako da vrijedi $\xi_i = (\gamma - y_i g(x_i))_+$ za svaki $i \in \{1, \ldots, l\}$. To znači da za marginu $m(S, g)$ skupa S vrijedi

$$m(S, g) = \min_{1 \leq i \leq l} y_i g(x_i) \geq \gamma.$$

Dakle, postoje dvije klase podataka koje su odvojive hiperravninom koja ima marginu veličine γ što možemo vidjeti na slici 3.1. Za takve skupove ćemo reći da su *odvojivi* ili preciznije *linearno odvojivi* s marginom γ.

Budući da vektor w ima normu 1, izraz $\langle w, \phi(x_i)\rangle$ mjeri udaljenost projekcije $\phi(x_i)$ na vektor određen vektorom w stoga izraz

$$y_i g(x_i) = y_i(\langle w, \phi(x_i)\rangle + b)$$

mjeri koliko je daleko vektor $\phi(x_i)$ od granične hiperravnine. Iz tog razloga takvu funkcionalnu marginu linearne funkcije nazivamo *geometrijskom* marginom pripadnog klasifikatora. Stoga $m(S, g) \geq \gamma$ implicira da se S točno klasificira funkcijom g geometrijskom marginom koja je veća ili jednaka od γ.

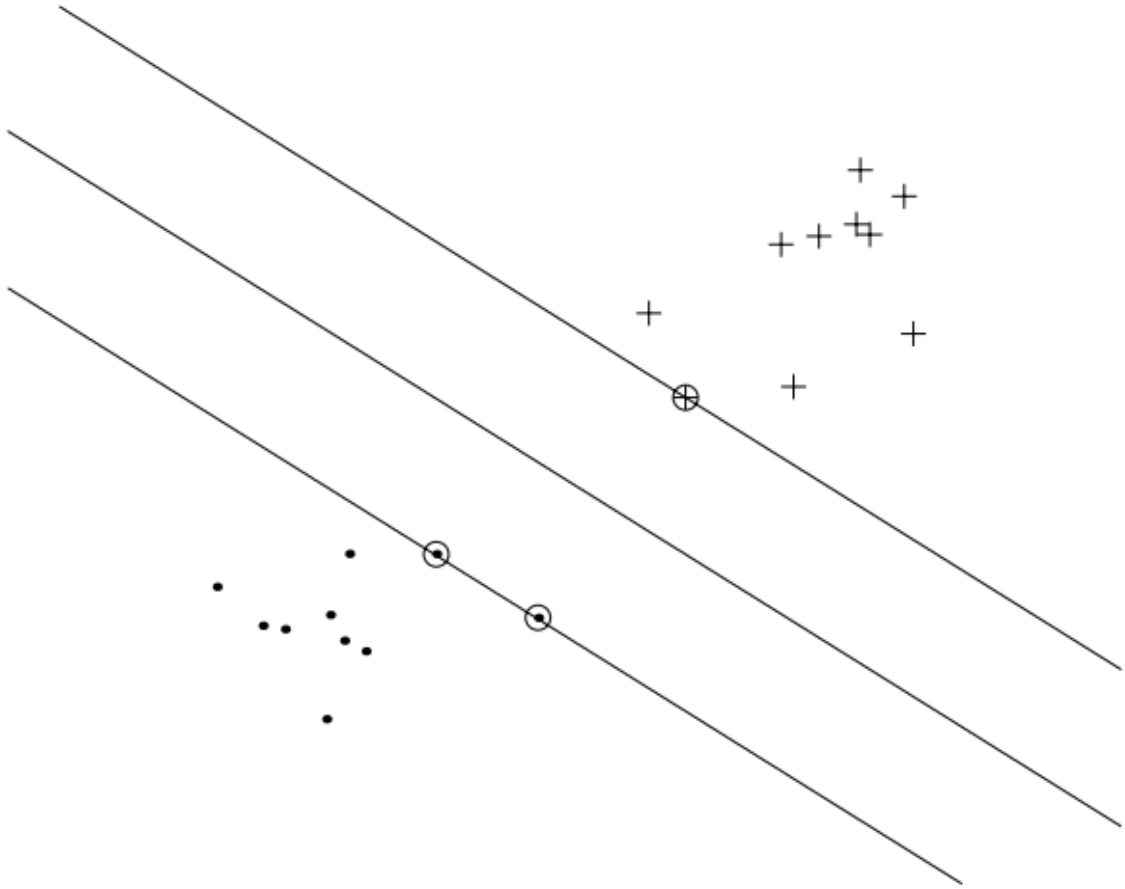
Napomenimo da hiperravnina s maksimalnom marginom γ ne daje robusno rješenje. Primjerice, samo jedan novi podatak u skupu za učenje može znatno smanjiti vrijednost margine i tako učiniti klasifikator lošijim.

Imajući na umu prethodno navedeno, reći ćemo da je naš cilj pronaći linearnu funkciju koja maksimizira geometrijsku ravninu.

Definicija 3.2.2. *Problem pronalaska maksimalne margine γ*

$$\begin{aligned} &\text{maksimiziraj} && \gamma \\ &\text{uz ograničenja} && y_i(\langle w, \phi(x_i)\rangle + b) \geq \gamma, i = 1, \ldots, l \\ & && \|w\|^2 = 1 \end{aligned}$$

nazivamo stroj potpornih vektora s tvrdom marginom.



Slika 3.1: Hiperravnina s marginom i potpornim vektorima koji su zaokruženi

Kao i u prethodnim sekcijama sada ćemo prikazati dualnu reprezentaciju problema. Radi toga, bolje je gledati optimizaciju pronalaska najmanje $-\gamma$ margine. Uvođenjem *Lagrangeovih* multiplikatora imamo

$$L(w, b, \gamma, \alpha, \lambda) = -\gamma - \sum_{i=1}^{l} \alpha_i \left[y_i(\langle w, \phi(x_i) \rangle + b) - \gamma \right] + \lambda(\|w\|^2 - 1).$$

Deriviranjem po primarnim varijablama zatim dobivamo

$$\begin{aligned}
\frac{\partial L(w,b,\gamma,\alpha,\lambda)}{\partial w} &= -\sum_{i=1}^{l} \alpha_i y_i \phi(x_i) + 2\lambda w = 0 \\
\frac{\partial L(w,b,\gamma,\alpha,\lambda)}{\partial \gamma} &= -1 + \sum_{i=1}^{l} \alpha_i = 0 \\
\frac{\partial L(w,b,\gamma,\alpha,\lambda)}{\partial b} &= -\sum_{i=1}^{l} \alpha_i y_i = 0.
\end{aligned} \tag{3.3}$$

Uvrštavanjem u početnu funkciju dobivamo

$$\begin{aligned}
L(w,b,\gamma,\alpha,\lambda) &= -\sum_{i=1}^{l} \alpha_i y_i \langle w, \phi(x_i)\rangle + \lambda \|w\|^2 - \lambda \\
&= \left(-\frac{1}{2\lambda} + \frac{1}{4\lambda}\right) \sum_{i,j=1}^{l} \alpha_i y_i \alpha_j y_j \langle \phi(x_i), \phi(x_j)\rangle - \lambda \\
&= -\frac{1}{4\lambda} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \kappa(x_i, x_j) - \lambda.
\end{aligned} \tag{3.4}$$

Slično kao i u 3.3 parcijalnim deriviranjem prethodnog izraza po λ imamo

$$\lambda = \frac{1}{2}\left(\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \kappa(x_i, x_j)\right)^{1/2},$$

te ponovnim uvrštavanjem u 3.4 konačno dobivamo

$$L(\alpha) = -\left(\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \kappa(x_i, x_j)\right)^{1/2}. \tag{3.5}$$

Sada predstavljamo pseudokod za implementaciju stroja potpornih vektora s tvrdom marginom. *Karush-Kuhn-Tuckerovi* uvjeti kažu kako dobiveni parametri $\alpha^*, (w^*, b^*)$ iz algoritma 3.3 moraju zadovoljavati

$$\alpha_i^*[y_i(\langle w^*, \phi(x_i)\rangle + b^*) - \gamma^*] = 0, i = 1, \ldots, l.$$

To implicira da svi vektori x_i koji imaju geometrijsku marginu jednaku γ^*, te su samim time najbliži hiperravnini, imaju nenegativne odgovarajuće multiplikatore α_i^*. Ukoliko pogledamo opet sliku 3.1 to bi bile zaokružene oznake. Svi ostali multiplikatori α_i^* su jednaki nuli. Vektore koji imaju nenegativne multiplikatore α_i^* nazivamo *potpornim vektorima*. Skup indeksa potornih vektora označavat ćemo sa *sv*.

Algoritam 3.3: Pseudokod za implementaciju stroja potpornih vektora s tvrdom marginom

Ulaz: Skup za učenje $S = \{(x_1, y_1), \ldots, (x_l, y_l)\}, \delta > 0, \gamma > 0$

Izlaz: težinski vektor w, dualno rješenje α^*, margina γ^*, funkcija predikcije f

1 Nađi α^* kao rješenje problema:

$$\begin{aligned} &\text{maksimiziraj} && W(\alpha) = -\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \kappa(x_i, x_j) \\ &\text{uz ograničenja} && \sum_{i=1}^{l} y_i \alpha_i = 0, \sum_{i=1}^{l} \alpha_i = 1 \text{ i } \alpha_i \geq 0, i = 1, \ldots, l. \end{aligned}$$

2 $\gamma^* = \sqrt{-W(\alpha^*)}$

3 izaberi i tako da vrijedi $0 < \alpha_i^*$

4 $b = y_i \gamma^{*^2} - \sum_{j=1}^{l} \alpha_j^* y_j \kappa(x_j, x_i)$

5 $f(x) = \text{sgn}\left(\sum_{j=1}^{l} \alpha_j^* y_j \kappa(x_j, x) + b\right)$

6 $w = \sum_{j=1}^{l} y_j \alpha_j^* \phi(x_j)$

Klasifikatori s mekanom marginom

Stroj potpornih vektora s tvrdom marginom je važan koncept, ali je primijenjiv samo ako su podaci odvojivi. Iz tog razloga nije primijenjiv u mnogim problemima iz stvarnog svijeta gdje podaci gotovo uvijek imaju određen šum. Ukoliko bismo htjeli osigurati linearnu odvojivost u takvim slučajevima, morali bismo koristiti kompleksne jezgre koje bi dovele do pretreniranosti, odnosno, klasifikator bi se prilagodio šumu i bio bi efikasan samo na sličnim skupovima podataka.

To nas motivira da dođemo do robusnijeg rješenja, koje će moći tolerirati šum i outliere u podacima.

Isto kao i kod otkrivanja noviteta, želimo dopustiti da se neka ograničenja pri optimizaciji mogu prekršiti. To možemo napraviti uvođenjem *slack* varijabli $\xi = \xi((y_i, x_i), \gamma, g) = (\gamma - y_i g(x_i))_+$.

Definicija 3.2.3. *Problem*

$$\begin{aligned} &\text{minimiziraj} && -\gamma + C \sum_{i=1}^{l} \xi_i \\ &\text{uz ograničenja} && y_i(\langle w, \phi(x_i) \rangle + b) \geq \gamma - \xi_i, \xi_i \geq 0, i = 1, \ldots, l \\ & && \|w\|^2 = 1 \end{aligned}$$

nazivamo stroj potpornih vektora s mekom marginom, pri čemu parametar C predstavlja kompromis između veličine margine i slack varijabli.

Kao i prije, uvodimo *Lagrangeovu* funkciju

$$L(w,b,\gamma,\xi,\alpha,\beta,\lambda) = -\gamma + C\sum_{i=1}^{l}\xi_i - \sum_{i=1}^{l}\alpha_i[y_i(\langle\phi(x_i),w\rangle + b) - \gamma + \xi_i] - \sum_{i=1}^{l}\beta_i\xi_i + \lambda(\|w\|^2 - 1)$$

gdje su $\alpha_i \geq 0$ i $\beta_i \geq 0$. Odgovarajuću dualnu funkciju zatim pronalazimo pomoću nužnog uvjeta postojanja ekstrema.

$$\begin{aligned}
\frac{\partial L(w,b,\gamma,\xi,\alpha,\beta,\lambda)}{\partial w} &= 2\lambda w - \sum_{i=1}^{l} y_i\alpha_i\phi(x_i) = 0\\
\frac{\partial L(w,b,\gamma,\xi,\alpha,\beta,\lambda)}{\partial \xi_i} &= C - \alpha_i - \beta_i = 0\\
\frac{\partial L(w,b,\gamma,\xi,\alpha,\beta,\lambda)}{\partial b} &= \sum_{i=1}^{l} y_i\alpha_i = 0\\
\frac{\partial L(w,b,\gamma,\xi,\alpha,\beta,\lambda)}{\partial b} &= 1 - \sum_{i=1}^{l}\alpha_i = 0
\end{aligned}$$

Uvrštavanjem u početnu funkciju dobivamo

$$L(\alpha,\lambda) = -\frac{1}{4\lambda}\sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \kappa(x_i,x_j) - \lambda,$$

iz čega slijedi

$$L(\alpha) = -\left(\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j \kappa(x_i,x_j)\right)^{1/2}$$

što je identično kao i u slučaju maksimalne margine, s razlikom u tome što izraz $C-\alpha_i-\beta_i = 0$ skupa sa $\beta_i \geq 0$ osigurava da je $\alpha_i \leq C$ za $i = 1,\ldots,l$. Slijedi da su KKT uvjeti

$$\begin{aligned}
\alpha_i[y_i(\langle\phi(x_i),w\rangle + b) - \gamma - \xi_i] = 0, &\quad i \in \{1,\ldots,l\}\\
\xi_i(\alpha_i - C) = 0, &\quad i \in \{1,\ldots,l\}.
\end{aligned}$$

Sada ćemo predstaviti algoritam za stroj potpornih vektora s mekom marginom.

Prirodno se nameće pitanje odabira parametra C. Za različite skupove podataka on ima različite vrijednosti i u praksi se algoritam pokreće za više vrijednosti te se odabire onaj

Algoritam 3.4: Pseudokod za implementaciju stroja potpornih vektora s mekom marginom

Ulaz: Skup za učenje $S = \{x_1, \ldots, x_l\}, C \in [1/l, \infty\rangle$

Izlaz: težinski vektor w, dualno rješenje α^*, margina γ^*, funkcija predikcije f

1 Nađi α^* kao rješenje problema:

$$\text{maksimiziraj} \qquad W(\alpha) = -\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \kappa(x_i, x_j)$$

$$\text{uz ograničenja} \qquad \sum_{i=1}^{l} y_i \alpha_i = 0, \sum_{i=1}^{l} \alpha_i = 1 \text{ i } 0 \leq \alpha_i \leq C, i = 1, \ldots, l.$$

2 $\lambda^* = \frac{1}{2}\left(\sum_{i,j=1}^{l} y_i y_j \alpha_i^* \alpha_j^* \kappa(x_i, x_j)\right)^{1/2}$

3 izaberi i, j takve da vrijedi $-C < \alpha_i^* y_i < \alpha_j^* y_j < C$

4 $b^* = -\lambda^*(\sum_{k=1}^{l} \alpha_k^* y_k^* \kappa(x_k, x_i) + \sum_{k=1}^{l} \alpha_k^* y_k \kappa(x_k, x_j))$

5 $\gamma^* = 2\lambda^* \sum_{k=1}^{l} \alpha_k^* y_k \kappa(x_k, x_j) + b^*$

6 $f(x) = \text{sgn}\left(\sum_{j=1}^{l} \alpha_j^* y_j \kappa(x_j, x) + b^*\right)$

7 $w = \sum_{j=1}^{l} y_j \alpha_j^* \phi(x_j)$

koji daje najbolje razultate. Prema teoremu 7.9 kojeg možemo pronaći u [7] parametar C mora biti veći od $1/l$, gdje je l veličina skupa za učenje, inače ograničenje

$$\sum_{i=1}^{l} \alpha_i = 1$$

ne može biti zadovoljeno. To znači da ukoliko koristimo

$$C = \frac{1}{\nu l}, \quad \nu \in \langle 0, 1],$$

dobivamo mnogo robusniji algoritam 3.5 iako je gotovo identičan algoritmu 3.4.

3.3 Regresija pomoću metode potpornih vektora

Koncept jezgri uveli smo pomoću *Ridge* regresije. U ovoj sekciji predstavit ćemo algoritam za njezino rješavanje pomoću potpornih vektora, no prije nego ga predstavimo definirajmo funkciju gubitka koja ignorira greške manje od nekog praga $\varepsilon > 0$.

Algoritam 3.5: Pseudokod za implementaciju stroja potpornih vektora s mekom marginom i parametrom ν

Ulaz: Skup za učenje $S = \{x_1, \ldots, x_l\}, \nu \in \langle 0, 1]$

Izlaz: težinski vektor w, dualno rješenje α^*, margina γ^*, funkcija predikcije f

1 Nađi α^* kao rješenje problema:

$$\begin{aligned} &\text{maksimiziraj} && W(\alpha) = -\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \kappa(x_i, x_j) \\ &\text{uz ograničenja} && \sum_{i=1}^{l} y_i \alpha_i = 0, \sum_{i=1}^{l} \alpha_i = 1 \text{ i } 0 \leq \alpha_i \leq 1/(\nu l), i = 1, \ldots, l. \end{aligned}$$

2 $\lambda^* = \frac{1}{2}\left(\sum_{i,j=1}^{l} y_i y_j \alpha_i^* \alpha_j^* \kappa(x_i, x_j)\right)^{1/2}$

3 izaberi i, j takve da vrijedi $-1/(\nu l) < \alpha_i^* y_i < \alpha_j^* y_j < 1/(\nu l)$

4 $b^* = -\lambda^*(\sum_{k=1}^{l} \alpha_k^* y_k^* \kappa(x_k, x_i) + \sum_{k=1}^{l} \alpha_k^* y_k \kappa(x_k, x_j))$

5 $\gamma *^{=} 2\lambda^* \sum_{k=1}^{l} \alpha_k^* y_k \kappa(x_k, x_j) + b^*$

6 $f(x) = \operatorname{sgn}\left(\sum_{j=1}^{l} \alpha_j^* y_j \kappa(x_j, x) + b^*\right)$

7 $w = \sum_{j=1}^{l} y_j \alpha_j^* \phi(x_j)$

Definicija 3.3.1. *Neka je $g : X \subseteq \mathbb{R}^n \to \mathbb{R}, n \in \mathbb{N}$. Za $x \in X$ i $y \in \mathbb{R}$ definiramo ε-neosjetljivu funkciju gubitka kao*

$$\mathcal{L}^{\varepsilon}(x, y, g) = |y - g(x)|_{\varepsilon} = \max(0, |y - g(x)| - \varepsilon).$$

Slično, kvadratna ε-neosjetljiva funkcija gubitka dana je s

$$\mathcal{L}_2^{\varepsilon}(x, y, g) = |y - g(x)|_{\varepsilon}^2.$$

Sumu kvadratnih ε-neosjetljivih gubitaka možemo optimizirati uz ograničenje norme težinskog vektora. No, ukoliko uvedemo *slack* varijable za slučajeve u kojima je izlaz premali (ξ_i) ili preveliki ($\hat{\xi}_i$), $i \in \{1, \ldots, l\}, l \in \mathbb{N}$ možemo definirati sljedeći problem.

Definicija 3.3.2. *Problem*

$$\begin{aligned} &\text{minimiziraj} && \|w\|^2 + C \sum_{i=1}^{l} (\xi_i^2 + \hat{\xi}_i^{\,2}) \\ &\text{uz ograničenja} && (\langle w, \phi(x_i) \rangle + b) - y_i \leq \varepsilon + \xi_i, \quad i \in \{1, \ldots, l\} \\ & && y_i - (\langle w, \phi(x_i) \rangle + b) \leq \varepsilon + \hat{\xi}_i, \quad i \in \{1, \ldots, l\} \end{aligned}$$

nazivamo ε-neosjetljiva regresija potpornih vektora, pri čemu parametar C predstavlja kompromis između norme i gubitaka.

Kao i prije možemo doći do dualnog problema koristeći *Lagrangeovu* funkciju, ali sada ćemo preskočiti njegovo dobivanje te ćemo ga samo navesti.

$$\begin{aligned} \text{maksimiziraj} \quad & \sum_{i=1}^{l} y_i(\hat{\alpha}_i - \alpha_i) - \varepsilon \sum_{i=1}^{l}(\hat{\alpha}_i + \alpha_i) - \\ & \frac{1}{2}\sum_{i,j=1}^{l}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_i - \alpha_j)(\kappa(x_i, x_j) + \frac{1}{C}) \\ \text{uz ograničenja} \quad & \sum_{i=1}^{l}(\hat{\alpha}_i - \alpha_i) = 0 \\ & \hat{\alpha}_i \geq 0, \alpha_i \geq 0, i \in 1, \ldots, l. \end{aligned}$$

Odgovarajući KKT uvjeti su

$$\begin{aligned} \alpha_i(\langle w, \phi(x_i)\rangle + b - y_i - \varepsilon - \xi_i) = 0, \quad & i = 1, \ldots, l \\ \hat{\alpha}_i\left(y_i - \langle w, \phi(x_i)\rangle - b - \varepsilon - \hat{\xi}_i\right) = 0, \quad & i = 1, \ldots, l \\ \xi_i\hat{\xi}_i =, \alpha_i\hat{\alpha}_i = 0, \quad & i = 1, \ldots, l. \end{aligned}$$

U nastavku slijedi pseudokod algoritma.

Algoritam 3.6: Pseudokod za regresiju potpornih vektora

Ulaz: Skup za učenje $S = \{x_1, \ldots, x_l\}, C > 0$

Izlaz: težinski vektor w, dualno rješenje α^*, b^*, funkcija predikcije f

1 Nađi α^* kao rješenje problema:

$$\begin{aligned} \text{maksimiziraj} \quad & W(\alpha) = \sum_{i=1}^{l} y_i\alpha_i - \varepsilon\sum_{i=1}^{l}\|\alpha_i\| - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j(\kappa(x_i, x_j) + \frac{1}{C} \\ \text{uz ograničenje} \quad & \sum_{i=1}^{l}\alpha_i = 0 \end{aligned}$$

2 $w = \sum_{j=1}^{l}\alpha_j^*\phi(x_j)$

3 izaberi i takav da $\alpha_i^* > 0$

4 $b^* = -\epsilon - (\alpha_i^*/C) + y_i - \sum_{j=1}^{l}\alpha_j^*\alpha_j^*\kappa(x_j, x_i)$

5 $f(x) = \sum_{j=1}^{l}\alpha_j^*\kappa(x_j, x) + b^*$

Poglavlje 4

Studijski primjer klasifikacije teksta

U ovom poglavlju opisat ćemo primjer kojem je cilj kategorizirati novinske članke u predefinirane kategorije. Implementirat ćemo stroj potpornih vektora (SVM) za klasifikaciju i analizirati dobivene rezultate. Vidjet ćemo kako je lagano primijeniti različite jezgre u algoritmima, te kako daju različite rezultate. Također, uvjerit ćemo se kako je SVM s mekanom marginom robusniji od onoga s tvrdom. No, za početak, recimo nešto o podacima za učenje.

Podaci za učenje

Kako bismo bili konzistentni s prethodnom teorijom, novinske članke u daljnjem teksu zvat ćemo dokumenti. Svaki dokument može biti u više kategorija odjednom, u točno jednoj ili biti nesvrstan. Naš skup podataka je *Reuters-21578* kojeg je sastavio *David Lewis* koji sadrži novinske članke razvrstane po kategorijama. Iako ih izvorni skup ima više, mi ćemo se usredotočiti na one kategorije koje imaju značajan broj dokumenata s kojima možemo učiti te testirati. U tablici 4.1 možemo vidjeti distribuciju dokumenata po kategorijama.

Kako je vokabular korpusa 14598 termova, tako će svaki dokument biti predstavljen vektorom dimenzije 14598 pri čemu će svaka komponenta vektora biti umnožak frekvencije terma u dokumentu te inverzne frekvencije dokumenta. Te pojmove smo opisali već u sekciji o oblikovanju semantičkih jezgri za tekstualne dokumente.

Evaluacijska mjera

Kao evaluacijsku mjeru koristit ćemo F-mjeru, harmonijsku sredinu preciznosti i odziva, pri čemu je preciznost udio točno klasificiranih primjera u skupu svih primjera, a odziv udio točno klasificiranih primjera u skupu svih pozitivnih primjera. Detaljnije se može pogledati u [6].

Kategorija	Broj dokumenata za učenje	Broj dokumenata za testiranje	Ukupno
acq	1596	696	2292
crude	253	121	374
earn	2840	1083	3923
grain	41	10	51
interest	190	81	271
money-fx	206	87	29
ship	108	36	144
trade	251	75	326
Ukupno	5485	2189	7674

Tablica 4.1: Distribucija dokumenata po kategorijama

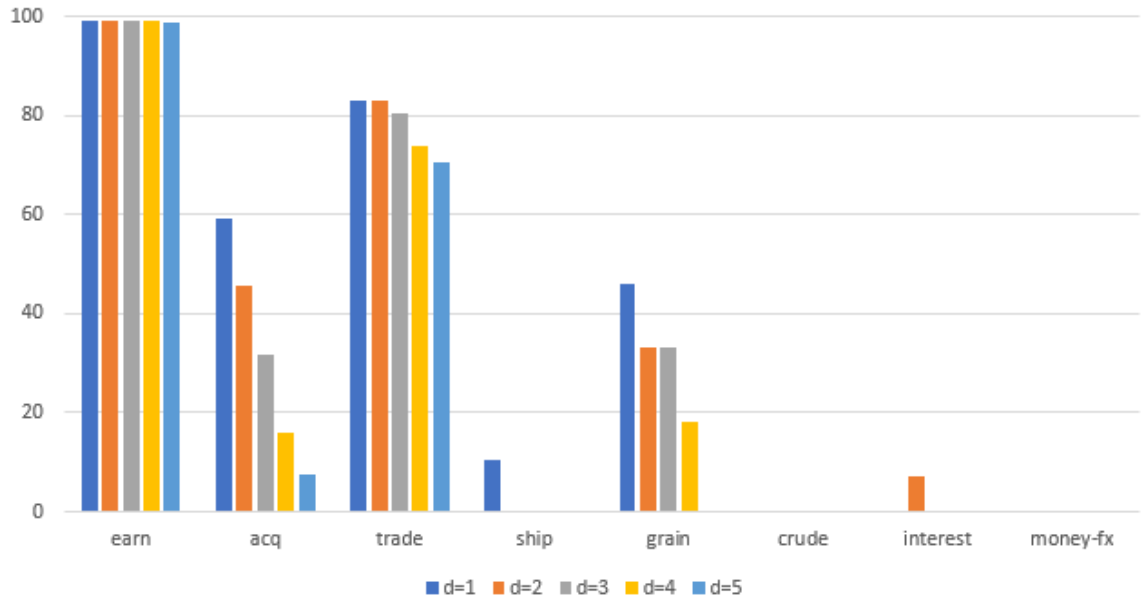
Detalji implementacije

Kako bismo vidjeli da SVM s tvrdom marginom daje loše rezultate na podacima iz stvarnog života, a da je SVM s mekom marginom i parametrom ν stvarno uspješniji odlučili smo se za njihove implementacije. Odabrani programski jezik implementacije je C#. Kako svaki algoritam u prvom koraku rješava optimizacijski problem, koristili smo vanjsku biblioteku koja u sebi ima metode za njegovo rješavanje. Radi se o *ALGLIB* biblioteci [1]. Svaki algoritam kombinirat ćemo s polinomnim i *Gaussovim* jezgrama te ćemo ih pokretati s različitim parametrima. Napomenimo još da smo parametre jezgara zadavali eksplicitno, dok smo parametar ν SVM-a s mekom marginom postupno smanjivali dok nismo mogli pronaći rješenje optimizacijskog problema u prvom koraku.

Dobiveni rezultati

Na slici 4.1 su prikazani rezultati SVM-a s tvrdom marginom u kojem smo koristili polinomnu jezgru. Možemo primijetiti kako smo uspješno klasificirali dokumente iz samo dvije kategorije *earn* i *trade*. (Sve iznad 50% više nije pogađanje.) Zanimljivo je primi-

jetiti kako povećanjem stupnja polinoma nismo povećavali uspješnost te kako veći broj dokumenata za učenje ne garantira veću točnost.

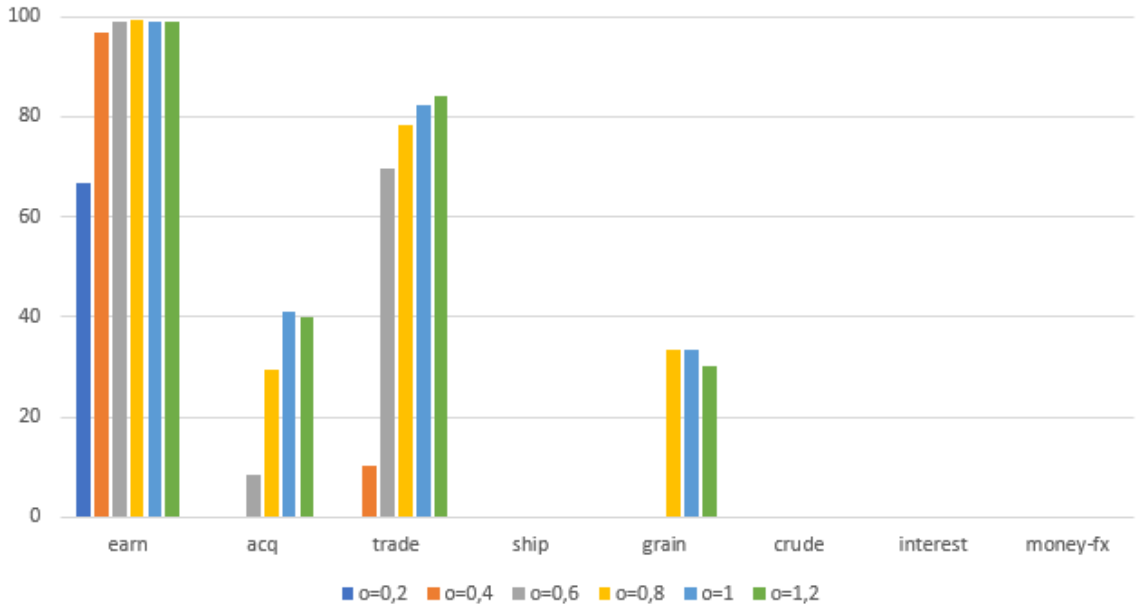


Slika 4.1: Rezultati SVM-a s tvrdom marginom i polinomnom jezgrom

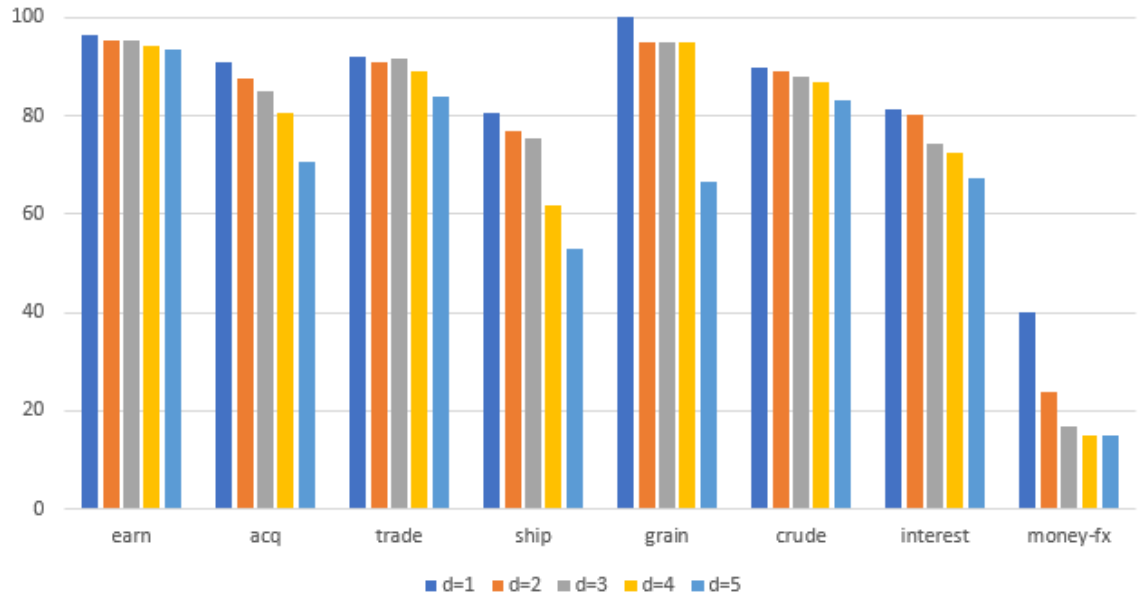
Na 4.2 vidimo rezultate SVM-a s tvrdom marginom u kojem smo koristili *Gaussovu* jezgru. Primijetimo da smo povećanjem parametra σ povećavali točnost klasifikatora, ali smo opet uspješno klasificirali iste dvije kategorije kao i prije. Štoviše, za kategorije *acq* i *grain* smo dobili za oko 20% manju uspješnost od SVM-a s tvrdom marginom koji je koristio polinomu jezgru. No, takve loše rezultate smo i očekivali. Očito je da klasifikatoru smetaju šumovi u podacima bez obzira kakvu jezgru koristili.

Na slici 4.3 vidimo rezultate SVM-a s mekom marginom i polinomnom jezgrom. Slično kao i u slučaju SVM-a s tvrdom marginom, povećanjem stupnja polinoma nismo povećavali uspješnost klasifikatora, ali smo zato uspješno klasificirali čak sedam kategorija. Na slici 4.4 vidimo da smo primjenom *Gaussove* jezgre uspješno klasificirali svih osam kategorija, ali tek za $\sigma \geq 0.8$.

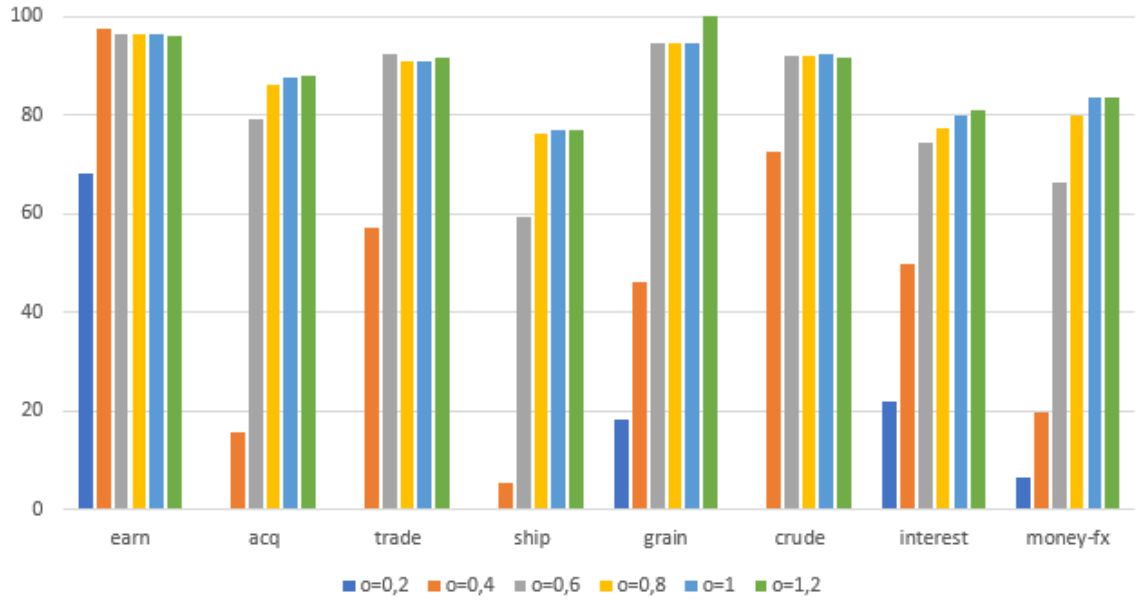
Na kraju, na slici 4.5 vidimo usporedbu najboljih rezultata implementiranih algoritama. Najrobusniji algoritam je SVM s mekom marginom koji koristi *Gaussovu* jezgru, no treba primijetiti kako je izbor parametara itekako bitan. Loš izbor parametara drastično smanjuje uspješnost klasifikatora. Primjerice, za $\sigma = 0.2$ je SVM s mekom marginom lošiji od onoga s tvrdom.



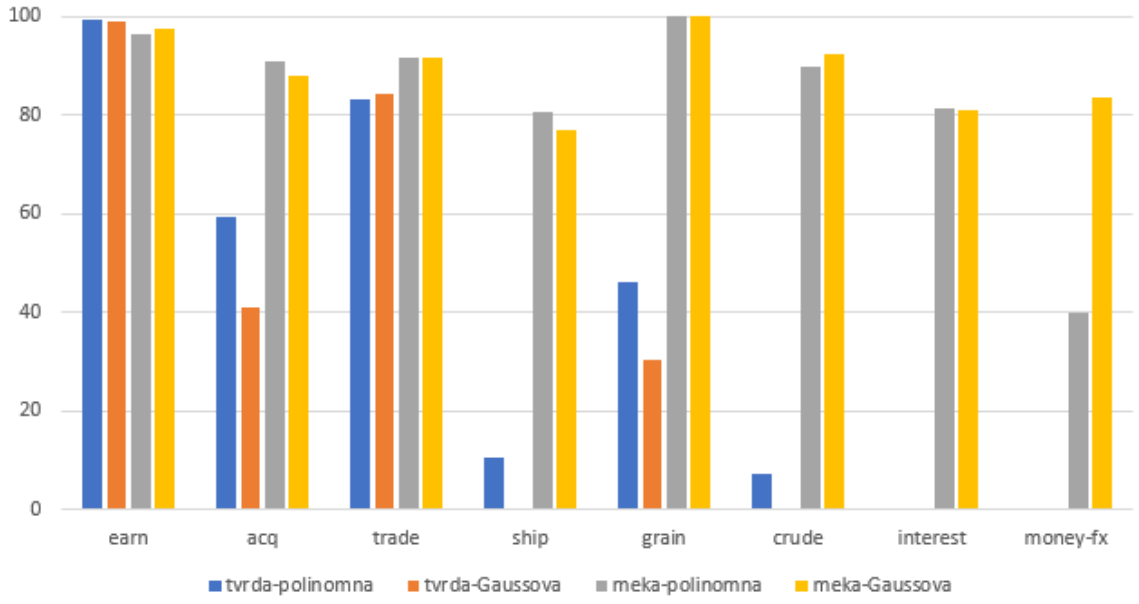
Slika 4.2: Rezultati SVM-a s tvrdom marginom i *Gaussovom* jezgrom



Slika 4.3: Rezultati SVM-a s mekom marginom i polinomnom jezgrom



Slika 4.4: Rezultati SVM-a s mekom marginom i *Gaussovom* jezgrom



Slika 4.5: Usporedba najboljih rezultata SVM-a s tvrdom i mekom marginom

Bibliografija

[1] *ALGLIB user manual*, `http://www.alglib.net/translator/man/manual.csharp.html`, pristupano 28. kolovoza 2017.

[2] Nello Cristianini i John Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*, Cambridge university press, 2000.

[3] Thorsten Joachims, *Text categorization with support vector machines: Learning with many relevant features*, Machine learning: ECML-98 (1998), 137–142.

[4] Alexandros Karatzoglou i Ingo Feinerer, *Kernel-based Machine Learning for Fast Text Mining in R*, Comput. Stat. Data Anal. **54** (2010), br. 2, 290–297, ISSN 0167-9473, `http://dx.doi.org/10.1016/j.csda.2009.09.023`.

[5] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini i Chris Watkins, *Text classification using string kernels*, Journal of Machine Learning Research **2** (2002), br. Feb, 419–444.

[6] Christopher D. Manning, Prabhakar Raghavan i Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press, New York, NY, USA, 2008, ISBN 0521865719, 9780521865715.

[7] John Shawe-Taylor i Nello Cristianini, *Kernel methods for pattern analysis*, Cambridge university press, 2004.

Sažetak

Analiza uzoraka sastavni je dio znanstvenih disciplina kao što su strojno učenje, rudarenje podataka, statistika i bioinformatika. U ovom radu promatrali smo kako primijeniti jezgrine funkcije u različitim algoritmima za učenje te kako takvim pristupom pronaći uzorke među podacima. Vidjeli smo da je takav pristup modularan, odnosno da se svaka jezgra mogla koristiti u svakom algoritmu i obratno. Modularnost te činjenica da je pristup bio u stanju premostiti razlike koje su postojale u različitim disciplinama u kojima se analiza uzoraka koristila, pružili su nam je jedinstveni alat za rad nad podacima svih tipova, bili oni vektori, skupovi, stringovi ili čak neki složeni oblik podatka.

U prvom poglavlju opisali smo problem te smo naveli matematičke definicije koje smo koristili u daljnjem tekstu radi njegovog lakšeg razumijevanja.

U drugom poglavlju smo najprije kroz konkretan primjer uveli pojam jezgrinih funkcija. Vidjeli smo kako smo pomoću njih mogli pronaći linearne relacije u nekom konačnom skupu podataka te smo saznali koje su prednosti njihovog korištenja. Zatim smo napravili klasifikaciju jezgrinih funkcija prema tipu podataka za koji su predviđene.

U trećem poglavlju opisali smo efikasne algoritme učenja za probleme otkrivanja noviteta, klasifikacije i regresije koji su bazirani na potpornim vektorima.

U četvrtom poglavlju implementirali smo dva algoritma za klasifikaciju s dvije različite jezgre, te smo ih primijenili za problem kategorizacije teksta. Iz dobivenih razultata zaključili smo kako robusnost algoritma itekako utječe na uspješnost klasifikacije. Robusniji algoritam je bio daleko uspješniji. Vidjeli smo kako su izbor jezgri i odabir njihovih parametara također utjecali na uspješnost algoritama.

Summary

Pattern analysis is an integral part of many scientific disciplines such as machine learning, data mining, statistics and bioinformatics. In this thesis, we have observed how to apply kernel methods in different learning algorithms and how to find patterns between data. We have seen that such approach is modular, any kernel method can be combined with any algorithm and vice versa. The modularity and the fact that the approach was able to bridge the gaps that existed between the different disciplines in which pattern analysis is used, have provided us with a unique tool for working on data of all types, whether they are vectors, sets, strings, or even a complex form of data.

In the first chapter we have described the problem and we have outlined the mathematical definitions we have used in the following text to make it easier to understand.

In the second chapter we introduced the term kernel method in a concrete example. We've seen how to find linear relations in some finite data set, and we've seen the benefits of using them. Then we've made the classification of the kernel methods according to the type of data they are intended for.

In the third chapter, we have described efficient learning algorithms for novelty detection, classification and regression based on the support vectors.

In the fourth chapter, we implemented two classification algorithms with two different kernels and applied them to the problem of text categorization. From the obtained results, we concluded that the robustness of the algorithm greatly influences the success of the classification. A more robust algorithm was far more successful. We have seen that the choice of the kernels and the selection of their parameters also influenced the success of algorithms.

Životopis

Domagoj Beti rođen je 6. lipnja 1990. godine u Sisku. Pohađao je OŠ Braće Bobetko u Sisku, a potom od 2005. godine Prvu gimnaziju u Zagrebu. Po završetku srednješkolskog obrazovanja upisuje preddiplomski sveučilišni studij Matematika na Matematičkom odsjeku Prirodoslovno-matematičkog fakulteta Sveučilišta u Zagrebu. Godine 2014. stekao je diplomu sveučilišnog prvostupnika matematike. Iste godine, na istoj instituciji, upisuje diplomski sveučilišni studij Računarstvo i matematika.