Strojno učenje putem regresije i SVM

Bojanić, David

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

David Bojanić

STROJNO UČENJE PUTEM REGRESIJE I SVM

Diplomski rad

Voditelj rada:
doc.dr.sc.Lavoslav
Čaklović

Zagreb, veljača, 2019.

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

Od samih začetaka programiranja, ljudski napori bili su ustrojeni ka stvaranju sustava koji bi u potpunosti mogli automatizirati te zamijeniti iste. Bolje rečeno, htjeli su stvoriti naizgled pravu ljudsku inteligenciju, ironično zvanu umjetna inteligencija. Naspram konvencionalnom eksplicitnom programiranju, takav pothvat zahtijevao je drugačiji pristup. Disciplina umjetne inteligencije, odnosno njezine grane strojnoga učenja latila se upravo takvoga pothvata.

U ovome radu će se predstaviti strojno učenje te dvije metode koje pripadaju toj disciplini; regresija i klasifikacija, dok će se metoda potpornih vektora prezentirati u detalje. Naposljetku, evaluirati će se dani algoritmi na konkretnom problemu.

Poglavlje 1

Strojno učenje

1.1 Motivacija

Prvi pokušaji stvaranja inteligentnih sustava počeli su od ručno pisanih naredbi i uvjeta kojima se dolazi do krajnjih rezultata. Takav pristup je primjenjiv ako se stvara sustav čiji model i proces čovjek može shvatiti i prenijeti u programski kod. Međutim, mnogi današnji problemi previše su zahtjevni da bi se riješili na klasičan način. Zamislite samo pristup učenja autonomnog vozila. Klasičan način učenja bi se sastojao od naredbi oblika "*Ako ___, tada zakoči*". Iz jednostavnog primjera možemo odmah primijetiti kako postoji previše uvjeta koji bi zadovoljavali ovu izjavu, te je očito takav pristup neizvediv. Alternativa ovakvom načinu učenja bi bila da autonomno vozilo jednostavno samo nauči pravila ponašanja ovisno u kojoj se situaciji nalazi uz pomoć ulaznih podataka te statističkih značajki iz njih samih.
Takav pristup je imao i *Arthur Samuel Lee* davnih 1950-tih godina kada je naučio stroj igri *dame* te se smatra jedan od pionira umjetne inteligencije, odnosno strojnoga učenja čiji termin i sam osmišljava godine 1959.

1.2 Kratka povijest strojnog učenja

Iako je strojno učenje usko povezano sa matematikom i statistikom te skupinom ostalih disciplina, ovdje ćemo samo navesti par ključnih trenutaka kako bi održali preglednost i sažetost.

1943 Prvi primjer ”neuronske mreže” kada neuropsiholog i matematičar predstavljaju njihovu percepciju kako ljudski neuroni funkcioniraju.
1950 Alan Turing predstavlja ”Turingov Test” koji određuje da li stroj ima pravu inteligenciju. Kako bi stroj položio test, mora zavarati čovjeka i uvjeriti ga kako je on sam čovjek a ne stroj.
1952 Arthur Samuel piše prvi program koji se uči igrati ”dame” u sklopu kojega formira pojam strojno učenje.
1957 Frank Rosenblatt dizajnira prvu neuronsku mrežu (jedan perceptron) na računalu koja simulira proces razmišljanja ljudskog mozga.
1959 Prva primjena neuronskih mreža. Stanfordov Adaline smanjuje jeku na telefonskim linijama.
1979 Studenti na Stanford fakultetu predstavljaju ”Stanford Cart”; robot koji se može samostalno kretati u prostoriji izbjegavajući prepreke.
1985 Terry Sejnowski predstavlja NetTalk, sustav koji je u tjedan dana naučio pravilno izgovarati 20.000 riječi.
1997 IBMov Deep Blue pobjeđuje tadašnjeg šahovskog prvaka Garrya Kasparova.
2006 Geoffrey Hinton predstavlja termin duboko učenje koji predstavlja novu generaciju algoritama koji dopuštaju računalima da ”vide”, odnosno klasificiraju objekte i tekst na slikama i videima.
2011 IBM-ov Watson pobjeđuje ljudske suparnike u igri ”Jeopardy”
2011 Google-ov X Lab predstavlja algoritam koji samostalno pregledava Youtube videe, te ih kategorizira prema tome da li oni sadrže mačku ili ne.
2014 Facebook razvija DeepFace, algoritam koji prepoznaje lica i ljude na istoj razini kao što to radi i čovjek.
2016 Google-ov AlphaGo algoritam pobjeđuje kineskoga prvaka u igri ”Go”, koja se smatra najkompleksnijom društvenom igrom.

Iz kratkog pregleda povijesti možemo primijetiti kako je zapravo ova disciplina relativno nova. Ako izuzmemo matematičke i statističke fundamente koji su trebali biti razvijeni kako bi uopće došlo do razvitka strojnoga učenja, vidimo da je prošlo samih 50-ak godina od njezinih začetaka.
Ujedno, možemo primijetiti velike kompanije među sudionicima razvijanja ove discipline, što ukazuje na to koliko je ona sama zapravo važna.

Međutim, možda najvažniji zaključak iz same povijesti jest to koliko je strojno učenje interdisciplinarno i primjenjivo. Sama potreba strojnoga učenja proizašla je iz konkretnih problema u raznim disciplinama; među kojima i medicina, fizika, financije, sigurnosne mjere, itd. To znači da za razvijanje modela potrebno je šire znanje od same discipline što ju čini kompleksnijom za savladavanje.

1.3 Definicija i podjela strojnog učenja

Najopćenitije prihvaćenu definiciju strojnoga učenja, predložio je Tom Mitchell:
"Kažemo da program uči na temelju iskustva E obzirom na neki skup zadatka T i evaluacijskom mjerom P, ako se njegov učinak na zadacima T poboljšava sa iskustvom E evaluirano mjerom P."

Pogledajmo jednostavan primjer koji ilustrira danu definiciju:

T klasifikacija neželjene e-pošte i željene e-pošte (*engl. spam/no spam e-mail*)

E program promatra neko vrijeme kakve tipove e-pošte mi klasificiramo kao neželjenu

P broj točno klasificiranih e-pošti

Najjednostavniji prikaz modela strojnoga učenja je sljedeći:

Slika 1.1: Model strojnoga učenja

gdje se modeli razlikuju ovisno o svakom od tih triju dijelova.

Ulazni podaci mogu biti raznovrsni. Postoje modeli koji samo primaju numeričke podatke, modeli koji mogu primati i kategoričke odnosno simboličke varijable, do modela koji primaju audio/vizualne podatke. Ovisno o tome da li su ulazni podaci označeni *(engl. Labeled)*, odnosno da li svaki ulazni podatak ima svoju ciljnu vrijednost, govorimo o nadziranom, odnosno nenadziranom učenju.

1.4 Nadzirano učenje

Kod nadziranog učenja, ulazni podaci su oblika (x, y) gdje je $x = (x_1, x_2, \ldots, x_n)$ vektor značajki a y ciljna vrijednost dane instance. Zadatak nadziranog učenja je pronaći preslikavanje koje nam za ulazne vrijednosti x-a daje ciljnu vrijednost y, odnosno pronalazi funkciju $f(x) = y$.

Ovisno o ciljnoj vrijednosti, modele nadziranog učenja dijelimo na:

- klasifikacijske modele - ciljne vrijednosti su diskretne kategorije odnosno klase
- regresijske modele - ciljne vrijednosti su kontinuirane

1.5 Nenadzirano učenje

Kod nenadziranog učenja, za razliku od nadziranog gdje imamo ciljnu vrijednost y, ulazni podaci su oblika $x = (x_1, x_2, \ldots, x_n)$. Zadatak takvog učenja je pronaći pravilnosti među podacima. Glavne primjene takvoga učenja su:

- grupiranje *(engl. Clustering)* - ciljne vrijednosti su grupe
- otkrivanje iznimki *(engl. Outlier detection)*
- kompresija podataka *(engl. Dimensionality reduction)*

1.6 Uvod u sljedeća poglavlja

Dok je glavna tematika ovoga rada algoritam potpornih vektora, uvodimo općenite pojmove kao što su regresija i klasifikacija te algoritme linearne regresije i logističke regresije kao mjerila usporedbe sa potpornim vektorima. Svaki algoritam ćemo potkrijepiti najjednostavnijim mogućim primjerom koji će služiti dobivanjem boljeg uvida u korake algoritma, dok ćemo u kasnijem poglavlju testirati metode na složenijim problemima.

Poglavlje 2

Regresija

Općenito, kada govorimo o regresiji, mislimo na predviđanje kontinuirane vrijednosti $y \in \mathbb{R}$. Na temelju danih primjera $(x^{(i)}, y^{(i)})$ iz skupa $\mathcal{D}$, gdje je $x^{(i)} = (x_1^{(i)}, \cdots, x_p^{(i)})$, cilj je naučiti nepoznatu funkciju $f : \mathcal{X} \to \mathbb{R}$ tako da je, idealno, $y^{(i)} = f(x^{(i)})$. Međutim, zbog prisustva šuma u podacima, zapravo učimo funkciju $y^{(i)} = f(x^{(i)}) + \epsilon$, gdje je ϵ slučajni šum.

2.1 Motivacija

Motivacija iza regresijskih modela je vrlo intuitivna. Kada bi veza danih primjera iz skupa $\mathcal{D}$ uistinu postojala, tada bi, regresijom, mogli doznati i vrijednost za bilo koji drugi dani primjer.

Promotrimo problem procjene cijene stambenog objekta. Pretpostavimo li na trenutak kako cijena danog objekta ovisi ponajviše o njegovoj kvadraturi, tada bi iz modela naučenog na danom podatkovnom skupu koji predstavlja dio ponude stambenih objekata, mogli procijeniti cijenu bilo kojega drugog danog objekta što bi mogao biti jako koristan model za npr. agencije nekretninama.

2.2 Linearna regresija

Linearnom regresijom nad skupom $\mathcal{D}$ dobivamo funkciju (hipotezu) h kao linearnu aproksimaciju funkcije f. Empirijsku pogrešku hipoteze h nad skupom za učenje $\mathcal{D}$ tada možemo definirati kao:

$$E(h, \mathcal{D}) = \sum_{i=1}^{N} \left(y^{(i)} - h\left(x^{(i)}\right) \right)^2 .$$

Pogrešku mjerimo kao zbroj kvadratnih odstupanja predviđene vrijednosti $h(x)$ i stvarne vrijednosti y. Dakle, želimo izabrati model koji će minimizirati danu empirijsku grešku. Izaberimo model:

$$h(x) = \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \beta_0 = \sum_{i=1}^{N} \beta_i x_i + \beta_0 = \beta^T x + \beta_0$$

gdje su β_i parametri koje treba naučiti na temelju skupa primjera $\mathcal{D}$, te je x oblika $(x_1, \cdots, x_p)$. Budući da izraz $h(x)$ linearno ovisi o težinama β, ovu vrstu regresije nazivamo **linearna regresija**.

Proširimo li vektor x dodatnim stupcem jedinica, te u vektor β ukomponiramo β_0 kao:

$$y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{bmatrix}, \quad X = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_p^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_p^{(2)} \\ \vdots & & & \vdots \\ 1 & x_1^{(N)} & \cdots & x_p^{(N)} \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_0 & \beta_1 & \cdots & \beta_p \end{bmatrix}$$

tada grešku E možemo pisati:

$$E(h, \mathcal{D}) = (y - X\beta)^T (y - X\beta)$$

Kako želimo minimizirati grešku, izraz:

$$\frac{\partial E}{\partial \beta} = -2X^T (y - X\beta)$$

izjednačavamo sa nulom te dobivamo jedinstveno rješenje:

$$\hat{\beta} = \left(X^T X\right)^{-1} X^T y$$

uz pretpostavku da je $X^T X$ pozitivno definitna matrica.
Tada, za bilo koji neviđeni primjer z, u kojem je prvi element postaljven na 1, imamo procjenu:

$$h(z) = \hat{\beta}^T z$$

dok su procjenjene vrijednosti ulaznih podataka:

$$\hat{y} = X\hat{\beta} = X\left(X^T X\right)^{-1} X^T y$$

Postoji mogućnost da stupci matrice X nisu linearno nezavisni. Tada matrica X nije punoga ranga. Takav primjer se može desiti ukoliko su značajke savršeno korelirane (npr. $x_2^{(i)} = 3x_1^{(i)}$ za svaki primjer i). Tada je matrica $X^T X$ singularna pa koeficijenti $\hat{\beta}$ nisu jedinstveno definirani. Općenito se takvi problemi u strojnom učenju rješavaju zanemarivanjem linearno zavisnih stupaca, ukoliko oni ne doprinose novim informacijama danom problemu.

Primjer

U ovom odjeljku ćemo proučiti najjednostavniji primjer kako bi mogli vidjeti korake algoritma, dok ćemo kasnije proučiti konkretne i složenije probleme.

Proučimo problem u kojem je dana progresivnost bolesti dijabetesa godinu od prvog mjerenja naspram nekolicine karakteristika pacijenata, kao što su: godine, spol, indeks tjelesne mase (BMI), itd.

Uzmimo model koji nam ovisno o indeksu tjelesne mase govori o progresivnosti bolesti dijabetesa. Neka su dani sljdeći primjeri:

	indeks tjelesne mase	progresivnost bolesti		indeks tjelesne mase	progresivnost bolesti
1	0.06169621	151	11	-0.08380842	101
2	-0.05147406	75	12	0.01750591	69
3	0.04445121	141	13	-0.02884001	179
4	-0.01159501	206	14	-0.00189471	185
5	-0.03638469	135	15	-0.02560657	118
6	-0.04069594	97	16	-0.01806189	171
7	-0.04716281	138	17	0.04229559	166
8	-0.00189471	63	18	0.01211685	144
9	0.06169621	110	19	-0.0105172	97
10	0.03906215	310	20	-0.01806189	168

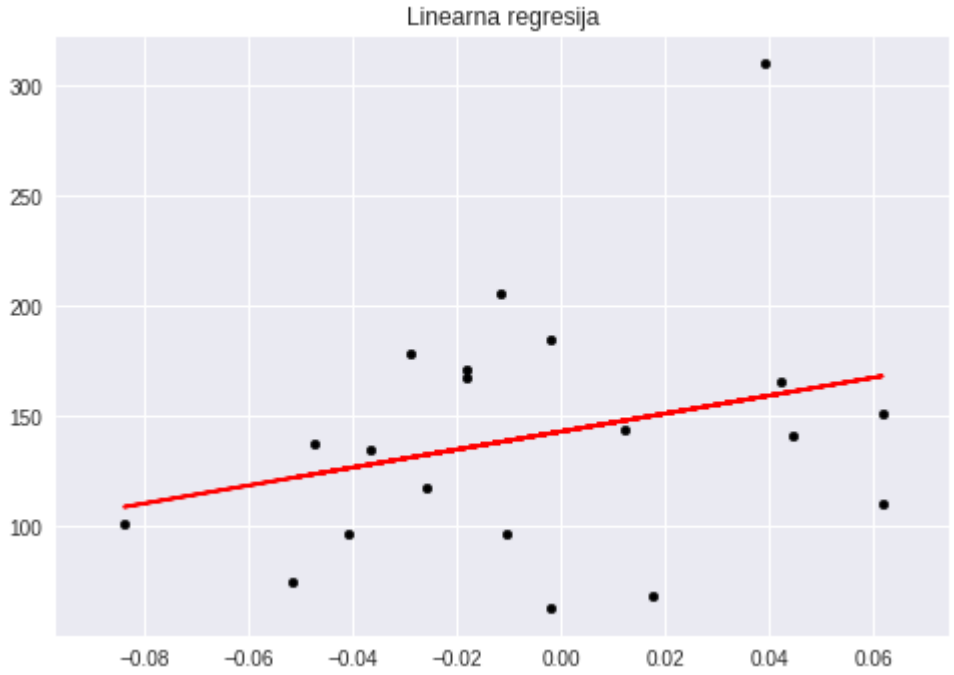
Tražimo $\hat{\beta} = \left(X^T X\right)^{-1} X^T y$ gdje se u matrici X u prvom stupcu nalaze same jedinice, dok u drugom podaci o indeksu tjelesne mase pojedinog pacijenta.

$$X^T X = \begin{bmatrix} 20 & -0.09717378 \\ -0.09717378 & 0.03062671 \end{bmatrix}$$

$$\left(X^T X\right)^{-1} = \begin{bmatrix} 0.05078286 & 0.16112613 \\ 0.16112613 & 33.16246968 \end{bmatrix}$$

$$\left(X^T X\right)^{-1} X^T y = \begin{bmatrix} 143.18115533 & 407.75511977 \end{bmatrix}$$

Dakle dobili smo $\hat{\beta} = \begin{bmatrix} 143.18115533 & 407.75511977 \end{bmatrix}$. Skica primjera izgleda kao:



Slika 2.1: Procjena progresivnosti bolesti dijabetesa

Poglavlje 3

Klasifikacija

Za razliku od regresije koja pokušava predvidjeti stvarnu vrijednost značajke, klasifikacija pokušava predvidjeti klasu primjera. Konkretnije, iz danih primjera $(x^{(i)}, y^{(i)})$ iz skupa $\mathcal{D}$, pokušavamo naučiti kakve točno značajke x spadaju u klasu y.

3.1 Motivacija

Kao i kod regresije, intuicija klasifikacije je vrlo očita, ako ne i očitija. Za dane primjere, želimo znati kojoj oni klasi pripadaju. Promotrimo problem segmentacije tržišta. Pretpostavimo li na trenutak kako korisnike na tržištu možemo preraspodijeliti na konkretne kategorije, tada svakog novog korisnika , ovisno o njegovim karakteristikama, možemo usmjeriti na pravi dio tržišta kako bi oboje mogli maksimizirati svoju korisnost.

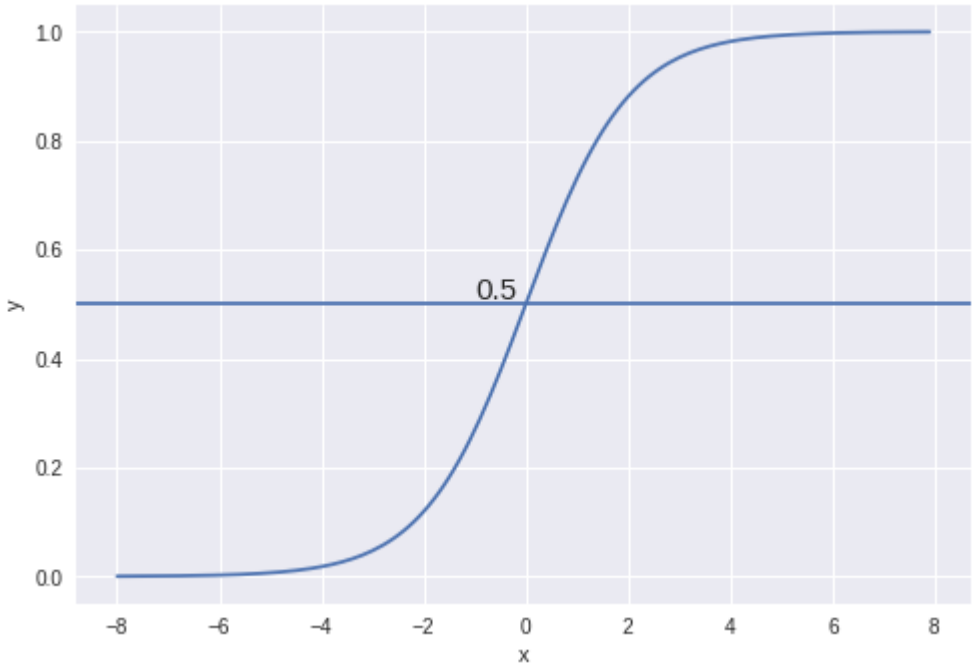
3.2 Logistička regresija

Dok linearna regresija rješava problem predikcije, logistička regresija pokušava riješiti problem klasifikacije. Intuitivno, htjeli prenamijeniti model linearne regresije da rezultira u predikcije klase primjera.
Neka su dani primjeri $(x^{(i)}, y^{(i)})$ gdje je ovoga puta $y \in \{0, 1\}$.

Ideja je pretvoriti rezultat linearnog modela u raspon između 0 i 1 te ga interpretirati kao vjerojatnost pripadnosti nekoj klasi; općenito klasi 1. Ukoliko je ta vjerojatnost veća od nekog pred određenog praga (obično 0.5), tada konkretnom primjeru dodjeljujemo klasu 1 dok je u protivnom on klase 0. Kako bi rezultat linearne regresije bio u rasponu $[0, 1]$, koristimo sigmoidalnu odnosno logit funkciju:

$$g(z) = \frac{1}{1 + e^{-z}}$$

koja ima sljedeći oblik:



Slika 3.1: Sigmoidalna odnosno logit funkcija

Tada naša hipoteza glasi:

$$h(x) = g(\hat{\beta}^T x) = \frac{1}{1 + e^{-\hat{\beta}^T x}}$$

Dakle, ukoliko je $h(x) \geq 0.5$, primjer x je klase 1, dok je u protivnom klase 0. Iz grafa funkcije g, vidimo da je $h(x) \geq 0.5$ ekvivalentno $\hat{\beta}^T x \geq 0$.

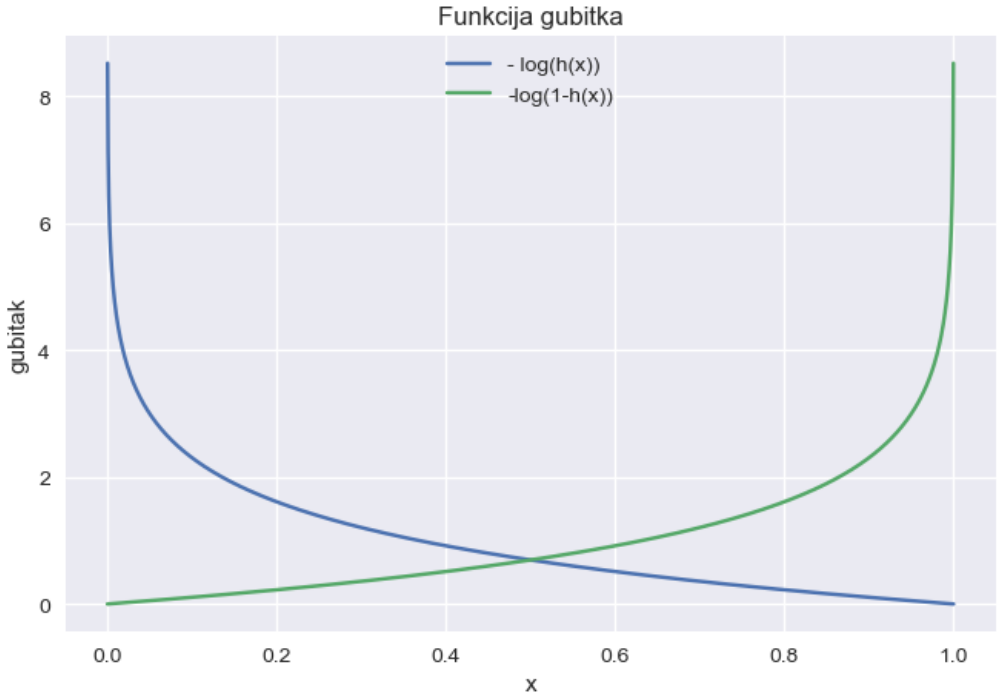
Kao i prije, moramo pronaći optimalne parametre $\hat{\beta}$. Dok smo u linearnoj regresiji, $\hat{\beta}$ mogli pronaći analitički, u ovom slučaju to nije mogućnost te ćemo se zato poslužiti metodom gradijentnog spusta nad funkcijom gubitka.

Funkciju gubitka definiramo na sljedeći način:

$$C(h(x), y) = \begin{cases} -\log(h(x)) & \text{ako } y = 1 \\ -\log(1 - h(x)) & \text{ako } y = 0 \end{cases}$$

Cilj funkcije gubitka je da penalizira krivo klasificirane primjere. Ako je klasa danog primjer $y = 1$, tada u slučaju ispravne klasifikacije imamo $-\log(1) = 0$. Dakle, funkcija gubitka ne penalizira takav slučaj. Obratno, u slučaju pogrešne klasifikacije imali bi

– log(0) što rezultira beskonačnom penalizacijom. Analogon vrijedi za slučaj kada je klasa primjera $y = 0$.



Slika 3.2: Funkcije gubitka

Dakle, kombiniranjem ta dva slučaja za jedan primjer, funkcija gubitka glasi:

$$C(h(x), y) = -y \log\left(h(x)\right) - (1 - y) \log\left(1 - h(x)\right).$$

dok za sve primjere zajedno glasi:

$$J(\beta) = -\frac{1}{N} \sum_{i=1}^{N} \left[y^{(i)} \log\left(h(x^{(i)})\right) + (1 - y^{(i)}) \log\left(1 - h(x^{(i)})\right) \right]$$

gdje je N broj primjera.

Kako želimo da funkcija gubitka bude što manja, koristimo se gradijentnim spustom radi određivanja minimuma. Tada jedan korak iteracija glasi:

$$\beta_j = \beta_j - \frac{\partial}{\partial \beta_j} J(\beta)$$

gdje je

$$\frac{\partial}{\partial \beta_j} J(\beta) = \frac{1}{N} \sum_{i=1}^{N} \left(h(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

Ispada da parametri dobiveni ovom metodom su jednaki kao i oni dobiveni metodom maksimizacije vjerodostojnosti pod pretpostavkom da nam klase y dolaze iz binomne distribucije $B(1, p)$, te da je p vjerojatnost "uspjeha", odnosno klase 1.

Dakle, želimo procjeniti $p(x^{(i)}) = P\left(Y \mid X = x^{(i)}\right)$. Maksimizacijom vjerodostojnosti imamo:

$$\mathcal{L}(\beta) = \prod_{i=1}^{N} P\left(Y = y^{(i)} \mid X = x^{(i)}\right) = \prod_{i=1}^{N} p\left(x^{(i)}, \beta\right)^{y^{(i)}} \left(1 - p(x^{(i)}, \beta)^{1-y^{(i)}}\right)$$

gdje je $p(x, \beta) = \frac{1}{1-e^{-\beta^T x}}$. Uvođenjem oznake $p_i = p(x^{(i)}, \beta)$ te logaritmiranjem dobivamo:

$$\log(\mathcal{L}) = \sum_{y^{(i)}=1} \log(p_i) + \sum_{y^{(i)}=0} \log(1 - p_i)$$

Korištenjem činjenice da je $p_i' = p_i(1 - p_i)$, nužni uvjeti glase:

$$\frac{\partial \log(\mathcal{L})}{\partial \beta_0} = \sum_{y^{(i)}=1} (1 - p_i) - \sum_{y^{(i)}=0} p_i = 0$$
$$\frac{\partial \log(\mathcal{L})}{\partial \beta_i} = \sum_{y^{(i)}=1} (1 - p_i) x^{(i)} - \sum_{y^{(i)}=0} p_i x^{(i)} = 0$$

Rješavanjem sustava dobivamo parametre kao u prošloj metodi.

Primjer

Neka je dan skup pozitivnih primjera

$$\{(3, 1), (3, -1), (6, 1), (6, -1)\},$$

te skup negativnih primjera

$$\{(1, 0), (0, 1), (0, -1), (-1, 0)\}.$$

Kako bi pronašli klasifikacijski pravac, moramo pronaći β. Napišimo najprije funkciju gubitka:

$$\begin{aligned} J(\beta) = -\frac{1}{8}\Bigg[&\log\left(\frac{1}{1+e^{-\beta_0-3\beta_1-\beta_2}}\right) + \log\left(\frac{1}{1+e^{-\beta_0-3\beta_1+\beta_2}}\right) + \\ &\log\left(\frac{1}{1+e^{-\beta_0-6\beta_1-\beta_2}}\right) + \log\left(\frac{1}{1+e^{-\beta_0-6\beta_1+\beta_2}}\right) + \\ &\log\left(1-\frac{1}{1+e^{-\beta_0-\beta_1}}\right) + \log\left(1-\frac{1}{1+e^{-\beta_0-\beta_2}}\right) + \\ &\log\left(1-\frac{1}{1+e^{-\beta_0+\beta_2}}\right) + \log\left(1-\frac{1}{1+e^{-\beta_0+\beta_1}}\right)\Bigg] \end{aligned}$$

Tada su derivacije:

$$\begin{aligned} \frac{\partial}{\partial\beta_0}J(\beta) = 0.5 &- 0.125\frac{e^{-\beta_0-3\beta_1+\beta_2}}{e^{(-\beta_0-3\beta_1+\beta_2}+1)} - 0.125\frac{e^{-\beta_0-3\beta_1-\beta_2}}{e^{-\beta_0-3\beta_1-\beta_2}+1} \\ &- 0.125\frac{e^{-\beta_0-6\beta_1+\beta_2}}{e^{-\beta_0-6\beta_1+\beta_2+1}} - 0.125\frac{e^{-\beta_0-6\beta_1-\beta_2}}{e^{-\beta_0-6\beta_1-\beta_2+1}} \\ &- 0.125\frac{e^{-\beta_0+\beta_2}}{e^{-\beta_0+\beta_2}+1} - 0.125\frac{e^{-\beta_0-\beta_2}}{e^{-\beta_0-\beta_2}+1} \\ &- 0.125\frac{e^{-\beta_0+\beta_1}}{e^{-\beta_0+\beta_1}+1} - 0.125\frac{e^{-\beta_0-\beta_1}}{e^{-\beta_0-\beta_1}+1} \end{aligned}$$

$$\begin{aligned} \frac{\partial}{\partial\beta_1}J(\beta) = &- 0.375\frac{e^{-\beta_0-3\beta_1+\beta_2}}{e^{-\beta_0-3\beta_1+\beta_2}+1} - 0.375\frac{e^{-\beta_0-3\beta_1-\beta_2}}{e^{-\beta_0-3\beta_1-\beta_2}+1} \\ &- 0.75\frac{e^{-\beta_0-6\beta_1+\beta_2}}{e^{-\beta_0-6\beta_1+\beta_2}+1} - 0.75\frac{e^{-\beta_0-6\beta_1-\beta_2}}{e^{-\beta_0-6\beta_1-\beta_2}+1} \\ &+ 0.125\frac{e^{-\beta_0+\beta_1}}{e^{-\beta_0+\beta_1}+1} - 0.125\frac{e^{-\beta_0-\beta_1}}{e^{-\beta_0-\beta_1}+1} \end{aligned}$$

$$\begin{aligned} \frac{\partial}{\partial\beta_2}J(\beta) = &0.125\frac{e^{-\beta_0-3\beta_1+\beta_2}}{e^{-\beta_0-3\beta_1+\beta_2}+1} - 0.125\frac{e^{-\beta_0-3\beta_1-\beta_2}}{e^{-\beta_0-3\beta_1-\beta_2}+1} \\ &+ 0.125\frac{e^{-\beta_0-6\beta_1+\beta_2}}{e^{-\beta_0-6\beta_1+\beta_2}+1} - 0.125\frac{e^{-\beta_0-6\beta_1-\beta_2}}{e^{-\beta_0-6\beta_1-\beta_2}+1} \\ &+ 0.125\frac{e^{-\beta_0+\beta_2}}{e^{-\beta_0+\beta_2}+1} - 0.125\frac{e^{-\beta_0-\beta_2}}{e^{-\beta_0-\beta_2}+1} \end{aligned}$$

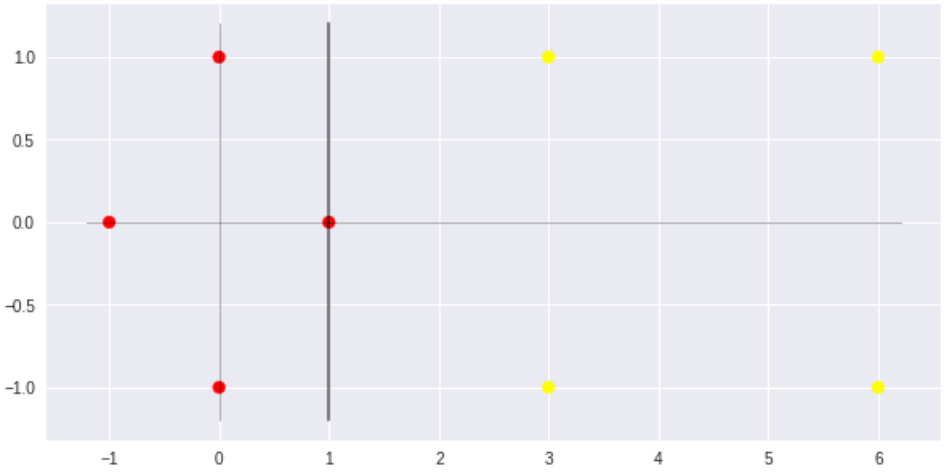
Primjenjujući gradijentni spust došli bi do rezulatata:

$$\beta_0 = -0.8, \quad \beta_1 = 0.8, \quad \beta_2 = 0$$

iz čega bi zaključili kako razdvajajući pravac glasi:

$$x_1 = 1$$

što vidimo na sljedećoj ilustraciji:



Slika 3.3: Primjer

Možemo primijetiti kako bi možda razdvajajući pravac bolje razdvajao primjere kada bi on bio malo ”u desno”. Naime, vidimo da jedna točka klase 0 leži točno na pravcu. U tom smislu govorimo da razdvajajući pravac nije robustan ukoliko postoji velika mogućnost da okolne točke točki $(1, 0)$ budu krivo klasificirane iz razloga što je margina pre blizu klasi 0. U kasnijim poglavljima ćemo moći vidjeti kako SVM pristupa ovom problemu te zašto je upravo takvo rješenje bolje.

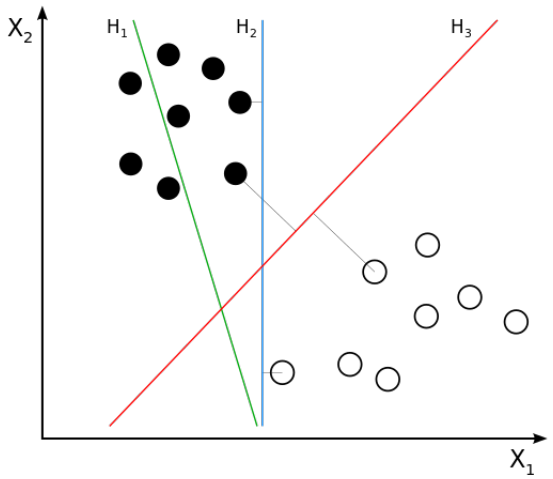
Poglavlje 4

Metoda potpornih vektora

Metoda potpornih vektora *(engl. Support Vector Machine, SVM)* spada pod nadzirane metode učenja koji za dani set podataka predviđa kojoj klasi pojedini podatak pripada. Postoji i varijanta zvana Regresija potpornih vektora *(engl. Support Vector Regression)* koja pokušava predvidjeti kontinuiranu vrijednost a ne klasu.

4.1 Motivacija

Za razliku od drugih klasifikacijskih algoritama, metoda potpornih vektora traži najbolju razdvajajuću hiperravninu. Sljedeća ilustracija prikazuje takvu intuiciju:



Slika 4.1: Model potpornih vektora, preuzeto sa https://en.wikipedia.org/wiki/Support-vector_machine

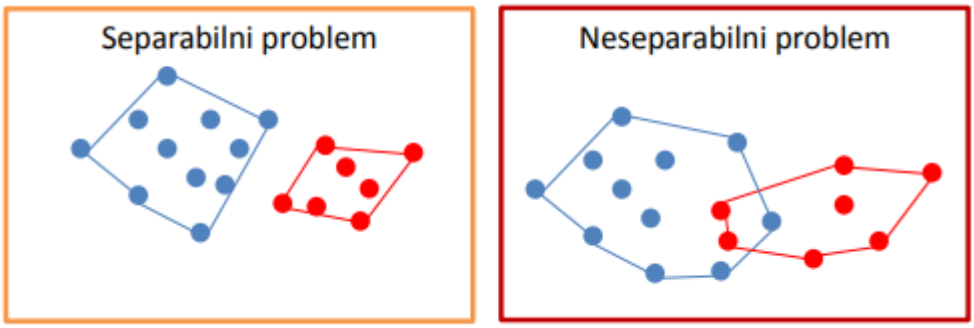
Iz ilustracije vidimo da, u slučaju gdje bi htjeli razdvojiti crne instance od bijelih, SVM bi pronašao hiperravninu koja ima najveće margine, odnosno hiperravninu koja je najudaljenija od bijelih i crnih instanci. Takva hiperravnina bi odgovarala hiperravnini *H3* na ilustraciji.
Intuicija traženja najudaljenije hiperravnine od samih klasa jest pojam generalizacije. Generalizacija je sposobnost modela da točno klasificira (ili predviđa ukoliko se radi o regresivnom problemu) svaku novu instancu neviđenu u skupu na kojem je model učen (odnosno treniran u žargonu strojnoga učenja). Svaki algoritam strojnog učenja teži ka dobroj generalizaciji.

4.2 Linearno (ne)separabilni problemi

U ovom radu ćemo se najprije baviti linearno separabilnim (odvojivim) problemom, dok ćemo kasnije razmotriti slučaj linearno neseparabilnog (neodvojivog) problema, te moć i prednost metode potpornih vektora pristupanja istome naspram drugih metoda.

Kako i sam izraz daje naslutiti, linearno separabilni problemi su problemi za koje postoji hiperravnina koja bi odvojila podatke prema njihovim danim klasama.

Konkretnije to su problemi za koje je presjek konveksnih ljusaka zasebnih klasa, prazan.



Slika 4.2: Linearno separabilni i neseparabilni problem, preuzeto sa https://web.math.pmf.unizg.hr/nastava/su/materijali/

Definicija 4.2.1. *Za dani skup S definiramo konveksnu ljusku skupa S kao:*

$$convS = \left\{ \sum_{i=1}^{k} t_i a_i : a_1, \ldots, a_k \in S, t_1, \ldots, t_k \geq 0, \sum_{i=1}^{k} t_i = 1 \right\}$$

4.3 Formalizacija modela

Neka je dan podatkovni skup $\mathcal{D}$ u kojem se nalaze primjeri za učenje (*en. Training set*) oblika (x, y) gdje je $x = (x_1, x_2, \ldots, x_m)$ vektor značajki, a y ciljna varijabla odnosno klasa toga primjera. Neka je $(x^{(i)}, y^{(i)})$ oznaka za i-ti primjer iz toga skupa.

Linearni model bez aktivacijske funckije ($\phi(x) = x$) definiramo kao:

$$h(x) = w^T \phi(x) + w_0 \tag{4.1}$$

gdje su w i $\phi(x)$ vektori odgovarajućih dimenzija. Radi jednostavnosti pretpostavimo da su klase $y \in \{-1, +1\}$ dok ćemo kasnije promotriti slučaj višeklasne klasifikacije. Predikcija novog primjera x tada odgovara predznaku funkcije $h(x)$, odnosno $y = sgn(h(x))$.

Nadalje, pretpostavimo da su primjeri iz skupa $\mathcal{D}$ linearno odvojivi (ili da su razdvojivi nakon preslikavanja funkcijom ϕ). To znači da postoje w i w_0 takvi da $h(x^{(i)}) \geq 0$ za $y^{(i)} = +1$ te $h(x^{(i)}) < 0$ za $y^{(i)} = -1$. Kraće, možemo pisati:

$$\forall (x^{(i)}, y^{(i)}) \in \mathcal{D} \quad y^{(i)} h(x^{(i)}) \geq 0$$

Ako su primjeri linearno odvojivi, tada postoji beskonačno mnogo rješenja, međutim, nas zanima rješenje maksimalne margine. Time uvodimo pristranost kako bi dobili jedinstveno rješenje.

Udaljenost bilo koje točke x i hiperravnine definirane sa $h(x) = 0$ jest:

$$d = \frac{h(x)}{||w||}, \tag{4.2}$$

no kako tražimo udaljenosti samo ispravno klasificiranih točki, tada gledamo udaljenost kao:

$$\frac{y^{(i)} h(x^{(i)})}{||w||} = \frac{y^{(i)}(w^T \phi(x^{(i)}) + w_0)}{||w||}$$

te vidimo da je ona nenegativna. Marginu će zapravo određivati primjeri koji su najbliži danoj hiperravnini. Stoga minimiziramo gornji izraz:

$$\frac{1}{||w||} \min_i \left\{ y^{(i)}(w^T \phi(x^{(i)}) + w_0) \right\}$$

Kao što smo rekli, želimo maksimizirati marginu:

$$\underset{w, w_0}{\operatorname{argmax}} \left\{ \frac{1}{||w||} \min_i \{ y^{(i)}(w^T \phi(x^{(i)}) + w_0) \} \right\}$$

Ovako formuliran optimizacijski problem ćemo preoblikovati u problem konveksne optimizacije.

Primijetimo da (w, w_0) možemo translatirati bez da utječemo na udaljenost $d = \frac{h(x)}{\|w\|}$, budući da je:

$$\frac{\alpha w^T \phi(x) + \alpha w_0}{\|\alpha w\|} = \frac{\alpha(w^T \phi(x) + w_0)}{\alpha\|w\|} = \frac{w^T \phi(x) + w_0}{\|w\|}$$

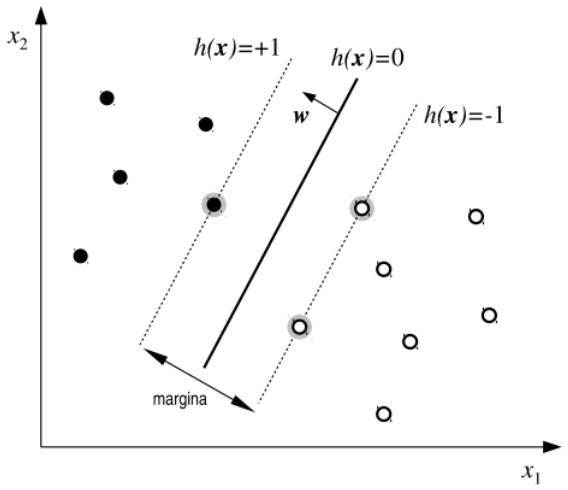
Zbog toga, kako bismo pojednostavili problem, možemo postaviti da za primjere $x^{(i)}$ koji su najbliži margini vrijedi:

$$y^{(i)}(w^T \phi(x^{(i)}) + w_0) = 1$$

Tada za sve preostale primjere mora vrijediti:

$$y^{(i)}(w^T \phi(x^{(i)}) + w_0) \geq 1, \quad i = 1, \ldots, N \tag{4.3}$$

Za one primjere za koje vrijedi jednakost kažemo da su **aktivna** te upravo te točke zovemo **potporni vektori**. Takva maksimizirana margina mora imati barem dva aktivna ograničenja, jedan za svaku klasu. Sljedeća ilustracija prikazuje dosada navedeno.



Slika 4.3: Margina i pripadne hiperravnine, preuzeto iz [2]

Kako za hiperravnine koje određuju margine vrijedi $h(x) = +1$ te $h(x) = -1$, prema definiciji udaljenosti (4.2) slijedi da je svaka od tih ravnina udaljena za $\frac{1}{\|w\|}$ od ravnine $h(x) = 0$. To znači da je maksimalna margina dužine $\frac{2}{\|w\|}$. Maksimiziranje margine, koji je ujedno naš krajnji cilj, tada možemo napisati kao:

$$\underset{w,w_0}{\text{argmax}} \frac{1}{\|w\|}$$

što je ekvivalentno:

$$\underset{w,w_0}{\operatorname{argmin}} \frac{1}{2}\|w\|^2$$

budući da je minimum od $\|w\|$ jednak minimumu od $\|w\|^2$. Faktor $\frac{1}{2}$ smo uključili zbog kasnije matematičke jednostavnosti te on ne mijenja krajnju minimizaciju.

4.4 Optimizacija metodom Lagrangeovih multiplikatora

Nalaženje maksimalne margine sveli smo na problem ograničenog kvadratnog programiranja, budući da minimiziramo kvadratnu funkciju uz ograničenja nejednakosti. Takav problem možemo riješiti Lagrangeovim multiplikatorima.

Metoda Lagrangeovih multiplikatora

Općenito, optimizacijski problem uz ograničenja definiramo na sljedeći način:

$$\begin{array}{ll} \text{minimizirati} & f(x) \\ \text{uz ograničenja} & g_i(x) \leq 0 \quad i = 1, \ldots, m \\ & h_i(x) = 0 \quad i = 1, \ldots, p \end{array}$$

gdje je $f : \mathbb{R}^n \to \mathbb{R}$ funkcija cilja, $h_i : \mathbb{R}^n \to \mathbb{R}$ ograničenja jednakosti te $g_i : \mathbb{R}^n \to \mathbb{R}$ ograničenja nejednakosti. Točke koje zadovoljavaju ograničenja nazivamo **ostvarivim točkama**. Dok smo problem zapisali kao minimizaciju ciljne funkcije, to ne predstavlja nikakvo ograničenje s obzirom da se problem $\min f(x)$ uvijek može prikazati kao $-\max -f(x)$.

Poseban slučaj gornjeg optimizacijskog problema jest onaj kada je funkcija cilja konveksna. Tada govorimo o konveksnome optimizacijskom problemu te ga općenito zapisujemo kao:

$$\begin{array}{ll} \text{minimizirati} & f(x) \\ \text{uz ograničenja} & g_i(x) \leq 0 \qquad i = 1, \ldots, m \\ & a_i^T x - b_i = 0 \quad i = 1, \ldots, p \end{array}$$

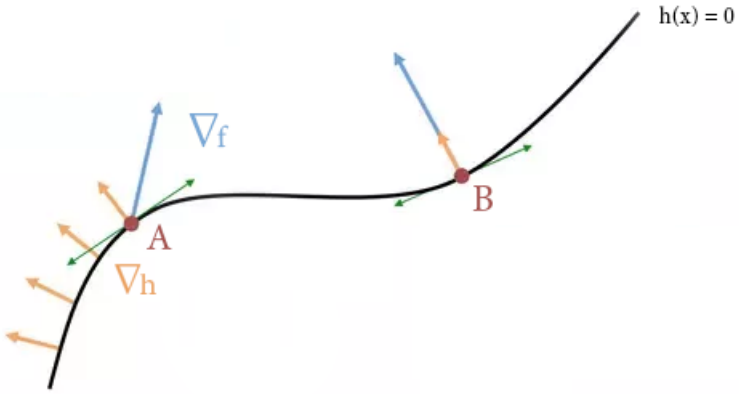
pri čemu su ograničenja nejednakosti g_i također konveksne funkcije, dok ograničenja jednakosti h_i moraju biti afine funkcije. U ovom slučaju, minimizacija ciljne funkcije $f(x)$ zaista predstavlja ograničenje, jer maksimizacija funkcije $-f(x)$ više nije konveksan problem budući da je $-f(x)$ konkavna funkcija. Međutim, u praksi nas maksimizacija konveksnih funkcija i minimizacija konkavih funkcija ne zanima.

Metoda Lagrangeovih multiplikatora nam upravo služi za rješavanje optimizacijskih problema uz ograničenja. Pritom, problem ne mora nužno biti konveksan; jedini uvjet jest da su ciljna funkcija i funkcije ograničenja derivabilne.

Ograničenja jednakosti

Želimo minimizirati funkciju $f(x) : \mathbb{R}^n \to \mathbb{R}$ s obzirom na ograničenje $h(x) = 0$. Ako je funkcija f n-dimenzijska, onda je funkcija h $(n-1)$-dimenzijska površina u $\mathbb{R}^n$. Ujedno, znamo da je za svaku točku na površini ograničenja gradijent $\nabla h(x)$ okomit na nju samu.

Neka je x^* točka na površini ograničenja $h(x) = 0$ koja minimizira $f(x)$. To je točka u kojoj smo se uspjeli najviše približiti globalnome minimumu funkcije $f(x)$, ali smo ipak ostali na površini $h(x) = 0$ koja definira ostvarive točke. To znači da vektor $\nabla f(x^*)$ mora biti okomit na površinu $h(x) = 0$ jer bi se u protivnom uvijek mogli pomaknuti po površini ograničenja tako da se vrijednost $f(x)$ poveća.



Slika 4.4: Lagrangeova optimizacija uz ograničenja jednakosti, preuzeto sa quora.com

Intuitivnije, promotrimo ilustraciju iznad. Dopuštena pomicanja su samo duž krivulje ograničenja h. U točki A primijetimo da gradijent ∇f ima komponentu u smijeru duž kojeg je dopušteno pomicanje. To znači da se možemo pomaknuti duž krivulje h kako bi maksimizirali f. Iz istoga razloga vidimo da, dok su gradijenti kolinearni kao u točki B, tada nemamo komponentu smjera prema kojoj bi se mogli pomaknuti kako bi maksimizirali f. Prema tome, zaključujemo da gradijenti ∇f i ∇h u točki x^* moraju biti jednakih smjerova.

Analogno, u slučaju minimizacije $f(x)$ oni bi trebali biti suprotnih smjerova. Prema tome gradijenti trebaju biti kolinearni, odnosno paralelni ili antiparalelni vektori. Posljedično, u točki x^* mora postojati konstanta $\alpha \neq 0$ za koju vrijedi:

$$\nabla f(x) + \alpha \nabla h(x) = 0 \tag{4.4}$$

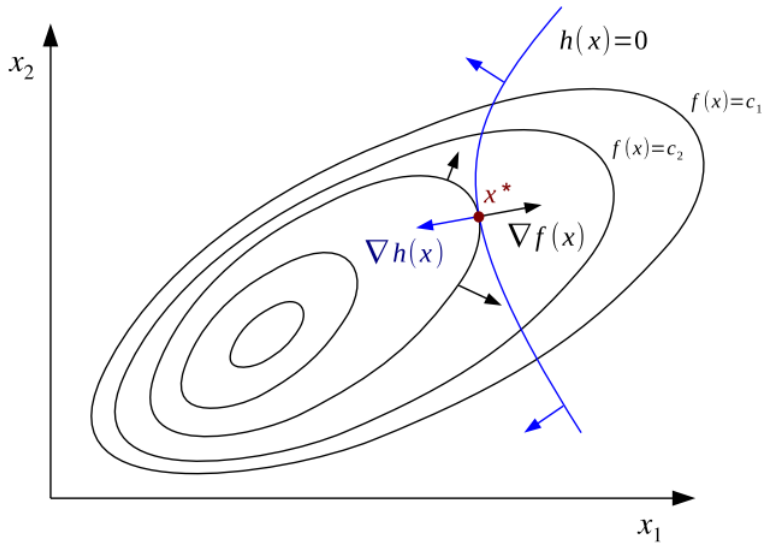
Rješenje ove jednadžbe ujedno je rješenje početnog optimizacijskog problema. Pretpostavimo sada da je jednadžba (4.4) minimum neke funkcije, odnosno odgovara stacionarnoj točci te funkcije u kojoj je njezina derivacija po x jednaka nuli. Ta funkcija tada glasi:

$$L(x, \alpha) \equiv f(x) + \alpha h(x).$$

Ovu funkciju nazivamo **Lagrangeova funkcija**. Kako bismo takvu funkciju minimizirali, njezin gradijent izjednačavamo s nulom i dobivenu jednadžbu rješavamo po x:

$$\nabla_x L(x, \alpha) = 0$$

što odgovara izrazu (4.4). Na sljedećoj ilustraciji je prikazana intuicija dosada objašnjenoga:



Slika 4.5: Lagrangeova optimizacija uz ograničenja jednakosti, preuzeto iz [2]

Pokažimo sada kolinearnost gradijenata u više dimenzija.
Parametriziramo li krivulju $h(x) = 0$ krivuljom $c(t)$ tako da je $c(0) = p$, gdje je p lokalni ekstrem funkcije f te $c'(0) \neq 0$, tada znamo da $f(c(t))$ ima lokalni ekstrem kada je $t = 0$, odnosno:

$$\frac{d}{dt} f(c(t)) \mid_{t=0} = \nabla f(p) \cdot c'(0) = 0$$

Isto tako, vidimo da je $\nabla g(p) \cdot c'(0) = 0$ što znači da postoji parametar α različit od nule tako da:

$$\nabla f(p) = \alpha \nabla g(p).$$

Lagrangeovu funkciju možemo poopćiti na slučaj kada imamo više ograničenja jednakosti. U tom slučaju vektori ∇h_i definiraju hiperravninu okomitu na $f(x)$ u točki x^*, što znači da vrijedi

$$\nabla f(x) + \nabla \sum_{i=1}^{m} \alpha_i h_i(x) = 0 \tag{4.5}$$

te je Lagrangeova funkcija tada:

$$L(x, \alpha) \equiv f(x) + \sum_{i=1}^{m} \alpha_i h_i(x) \tag{4.6}$$

gdje je α vektor Lagrangeovih multiplikatora α_i. $L(x, \alpha)$ je funkcija od $n + m$ varijabli; n varijabli koje čine vektor x i m varijabli koje čine vektor α.

Intuitivnije, uzmimo točku p u zajedničkoj domeni $\Omega \subset \mathbb{R}^n$ funkcije f i m ograničenja h_i. Tada, kao i prije, tangencijalni smjerovi razapinju potprostor U dozvoljenih pomicanja od točke p. Štoviše, smjer X je dopustiv smjer pomicanja ako pripada tangencijalnoj ravnini svih ograničenja h_i. To znači da je smjer X okomit na $\nabla h_i(p)$, odnosno zadovoljava:

$$\nabla h_i(p) \cdot X = 0 \quad i = 1, \cdots, m$$

Ključna pretpostavka Lagrangeove metode je linearna nezavisnost gradijenata $\nabla h_i(p)$ iz čega, prema Kronecker-Capellijevom teoremu slijedi postojanje netrivijalnog rješenja koje tražimo, odnosno regularnost točke p.

Tada $\nabla h_i(p)$ razapinju m-dimenzionalni potprostor V, te slijedi da je $\dim(U) = n - m$. Posljedično je $U \subset V^{\perp}$, te štoviše $U = V^{\perp}$.

Kada je $\nabla f(p) \cdot X \neq 0$ za neki dopušteni smjer pomicanja X, tada funkcija f nije kondicionalno stacionarna u točki p, odnosno, kao i prije, možemo se pomaknuti kako bi povećali svoju vrijednost funkcije f. To znači da želimo:

$$\nabla f(p) \cdot X = 0$$

za sve smjerove $X \in U$. Posljedično je: $\nabla f(p) \in U^{\perp} = V$.
Konkretnije, kada je $\nabla f(p) \in \langle \nabla h_1(p), \cdots, \nabla h_m(p) \rangle$, tada postoje skalari $\alpha_i \quad i = 1, \cdots, m$ tako da:

$$\nabla f(p) = \sum_{i=1}^{m} \alpha_i \nabla h_i(p).$$

Jedna od interpretacija (4.6) jest da je to način "kodiranja" ograničenja. Tražimo x koji minimizira tu funkciju, pritom međutim nemamo nikakvog utjecaja na α. Pretpostavimo da neka točka x zadovoljava sva ograničenja. Tada $\forall i \quad h_i(x) = 0$, pa je $L(x, \alpha) = f(x)$, neovisno o vrijednosti α. S druge strane, ako neko od ograničenja $h_i(x) = 0$ nije ispunjeno, onda dotični α_i može biti postavljen na proizvoljno veliku vrijednost, u najgorem slučaju takvu da $\alpha_i h_i(x) \to \infty$, pa tada $L(x, \alpha) \to \infty$ i točka x nikako ne može biti točka minimuma.

Sustav jednadžbi $\nabla_x L(x, \alpha)$ nam tada daje ostvarivi minimum od $f(x)$. Međutim, moguće je da deriviranjem ne možemo ukloniti multiplikatore α, te tada ne možemo dobiti rješenje u zatvorenoj formi.

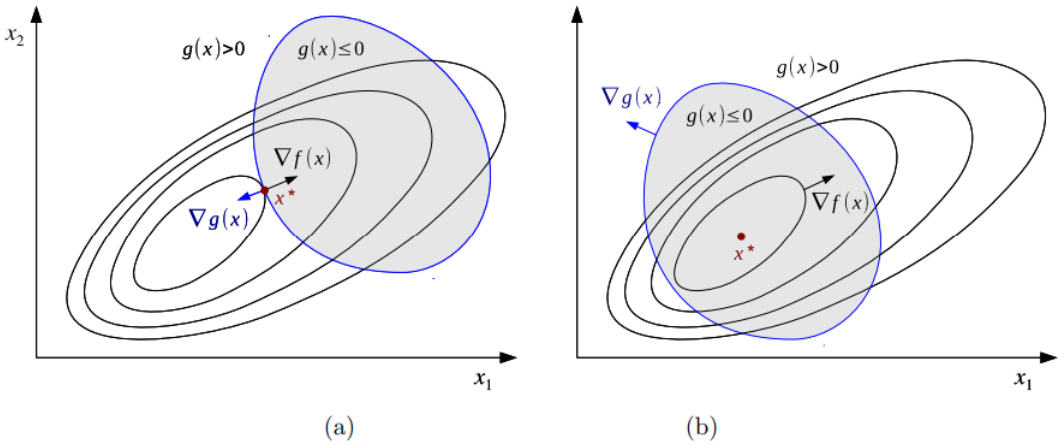
Ograničenja nejednakosti

Želimo minimizirati funkciju $f(x)$ uz ograničenje $g(x) \leq 0$. Kod ovakvog problema postoje dva slučaja:

(a) globalni se minimum x^* nalazi izvan ostvarivog područja $g(x) \leq 0$. Tada govorimo o **aktivnom ograničenju**

(b) globalni se minimum x^* nalazi unutar ostvarivog područja $g(x) \leq 0$. Tada govorimo o **neaktivnom ograničenju** jer ograničenje g ne igra ulogu u pronalasku traženoga minimuma funkcije f

Sljedeće je prikazano na ilustraciji:



Slika 4.6: Lagrangeova optimizacija uz ograničenja nejednakosti, preuzeto iz [2]

Lakši slučaj, slučaj *(b)*, rješavamo "standardno" traženjem stacionarne točke x^* za koju je $\nabla f(x) = 0$, što odgovara Lagrangeovoj funkciji za $\alpha = 0$.

U slučaju *(a)* optimalna točka x^* nalazi se na površini $g(x) = 0$, što je bliže globalnome minimumu. To znači da sve točke x u ostvarivome području imaju vrijednost $f(x)$ veću od minimuma, pa gradijent $\nabla f(x)$ pokazuje prema ostvarivom području, dok $\nabla g(x)$ pokazuje od njega. Posljedično, $\nabla f(x)$ i $\nabla g(x)$ su antiparalelni vektori te vrijedi $\nabla f(x) = -\alpha \nabla g(x)$ za neku konstantu $\alpha > 0$.

Tada, uzevši u obzir ova dva slučaja, minimizacija $f(x)$ uz uvijet $g(x) \leq 0$ odgovara minimizacija sljedeće Lagrangeove funkcije:

$$L(x, \alpha) \equiv f(x) + \alpha g(x)$$

uz $\alpha \geq 0$. Primjetimo, za točku ostvarivog minimuma x^* mora vrijediti da je $\alpha = 0$ (za slučaj *(b)*) ili $g(x) = 0$ (za slučaj *(a)*). Sažetije možemo pisati $\alpha g(x) = 0$.

Općenitije, kada imamo više ograničenja nejednakosti, problem se svodi na minimiziranje Lagrangeove funkcije:

$$L(x, \alpha) \equiv f(x) + \sum_i \alpha_i g_i(x)$$

uz uvjete:

$$\begin{aligned} \alpha_i &\geq 0, \quad i = 1, \ldots, m \\ \alpha_i g_i(x) &= 0 \quad i = 1, \ldots, m \end{aligned}$$

Takvi uvijeti su poznatiji pod imenom **Karush-Kuhn-Tuckerovi (KKT)** uvijeti.

Općeniti slučaj

Razmotrimo sada najopćenitiji slučaj u kojemu istovremeno postoje ograničenja jednakosti i nejednakosti:

$$\begin{array}{lll} \text{minimizirati} & f(x) & \\ \text{uz ograničenja} & g_i(x) \leq 0 & i = 1, \ldots, m \\ & h_i(x) = 0 & i = 1, \ldots, p \end{array}$$

Problem je istovjetan minimizaciji Lagrangeove funkcije:

$$L(x, \alpha, \beta) = f(x) + \sum_{i=1}^{m} \alpha_i g_i(x) + \sum_{i=1}^{p} \beta_i h_i(x)$$

uz KKT uvijete:

$$\begin{aligned} \alpha_i &\geq 0 \quad i = 1, \ldots, m \\ \alpha_i g_i(x) &= 0 \quad i = 1, \ldots, m \end{aligned}$$

Dualni Lagrangeov problem

U teoriji optimizacije, načelo dualnosti odnosi se na činjenicu da se optimizacijski problem može izraziti kao primarni problem ili kao njemu dualni problem. Primarni problem jest minimizacija funkcije $f(x)$, dok je dualni problem nalaženje donje granice primarnog problema. U općenitom slučaju, rješenja primarnog i dualnog problema se ne podudaraju, već postoji tzv. dualni procjep. Međutim, uz određene uvjete je kod konveksne optimizacije, kao što je naš slučaj, dualni procjep jednak nuli, što znači da je rješenje dualnog problema ujedno i rješenje primarnog problema. Tada govorimo o tzv. jakoj dualnosti.

Bez smanjenja općenitosti možemo se ograničiti na slučaj sa samo ograničenjima nejednakosti, pošto svaku jednakost možemo napisati kao dvije nejednakosti. Tada Lagrangeova funkcija glasi:

$$L(x, \alpha) = f(x) + \sum_i \alpha_i g_i(x).$$

U ovom se kontekstu varijable x nazivaju primarne varijable a Lagrangeovi multiplikatori α dualne varijable. Neka je x^* vrijednost koja minimizira Larangeovu funkciju i neka je α^* odgovarajuća vrijednost Lagrangeovog multiplikatora, odnosno:

$$L(x^*, \alpha^*) = \min_{x,\alpha} L(x, \alpha).$$

Vrijednost x^* nalazimo rješavanjem sustava kao što je (4.5). Međutim, kao što smo napomenuli, to ne mora nužno rezultirati uklanjanjem multiplikatora α, budući da sustav rješavamo samo po varijabli x. Umjesto toga, rješenje može rezultirati izrazom za x^* kao funkcije od α. Ta funkcija za svaku vrijednost α daje x koji minimizira Lagrangeovu funkciju uz tako odabrani α, tj:

$$\tilde{L}(\alpha) = \min_x L(x, \alpha) = \min_x \left(f(x) + \sum_i \alpha_i g_i(x) \right).$$

Funkciju $\tilde{L}(\alpha)$ nazivamo dualnom Lagrangeovom funkcijom. Kako bi pronašli optimalnu vrijednost α^*, primijetimo najprije kako vrijedi:

$$\tilde{L}(\alpha) \leq L(x^*, \alpha)$$

budući da $\tilde{L}(\alpha)$ nalazi minimum po svim x, uključivo x^*. Očito je da u točki α^* vrijedi jednakost $\tilde{L}(\alpha^*) = L(x^*, \alpha^*)$. Drugim riječima, funkcija $\tilde{L}(\alpha)$ je donja ograda primarnog optimizacijskog problema. Kako bismo pronašli α^*, moramo maksimizirati donju ogradu, odnosno:

$$\begin{array}{ll} \text{maksimizirati} & \tilde{L}(\alpha) \\ \text{uz ograničenja} & \alpha_i \geq 0 \quad i = 1, \ldots, m \end{array}$$

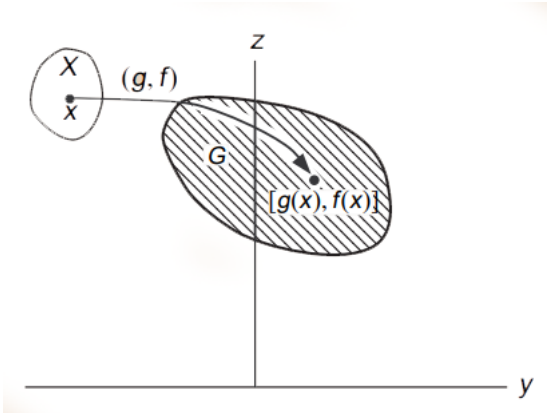
Ovaj se problem naziva dualni Lagrangeov problem. Primjetimo da je ciljna funkcija ovoga problema nužno konkavna, čak i ako izvorni problem nije bio konveksan. Budući da tražimo maksimum konkavne ciljne funkcije uz konveksna ograničenja, još uvijek se bavimo konveksnom optimizacijom.

Geometrijska interpretacija dualnog problema

Promotrimo primarni problem:

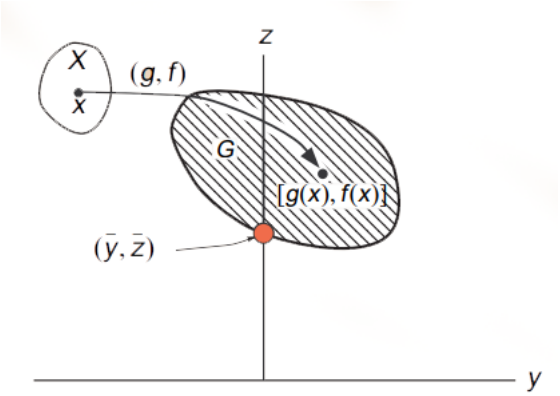
$$\begin{array}{lll} \text{minimizirati} & f(x) & f : \mathbb{R}^n \to \mathbb{R} \\ \text{uz ograničenja} & g(x) \leq 0 & g : \mathbb{R}^n \to \mathbb{R} \\ & x \in X & \end{array}$$

Definirajmo skup: $G = \{(y, z) : y = g(x), z = f(x) \quad \text{za neki x } \in \text{X}\}$



Slika 4.7: Geometrijska interpretacija, preuzeto iz [4]

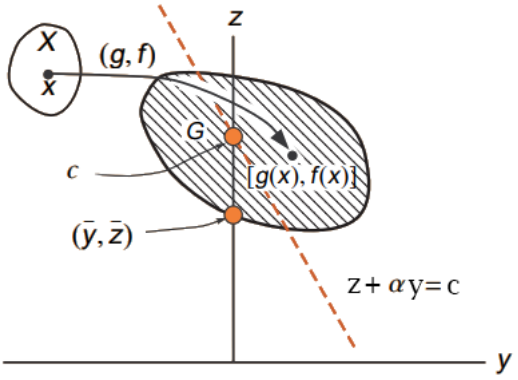
Tada se zapravo primarni problem svodi na pronalazak točke iz skupa G gdje je $y \leq 0$ te minimalnom ordinatom z.



Slika 4.8: Geometrijska interpretacija, preuzeto iz [4]

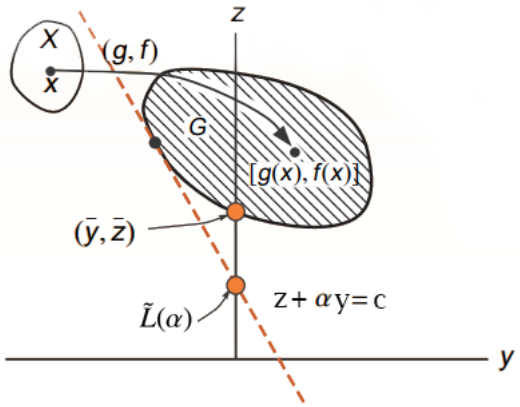
Vidljivo je takva točka $(\bar{y}, \bar{z})$ na ilustraciji.

Kod dualnog problema maksimiziramo funkciju $\tilde{L}(\alpha)$ uz ograničenje $\alpha \geq 0$ gdje je $\tilde{L}(\alpha) = \min_x (f(x) + \alpha g(x))$. Tada, za dani $\alpha \geq 0$ dualni Lagrangeov problem se svodi na minimizaciju $z + \alpha y$ nad točkama (y, z) iz skupa G. Primijetimo da je $z + \alpha y = c$ jednadžba pravca sa koeficijentom smjera $-\alpha$ te odsječkom c na z osi.



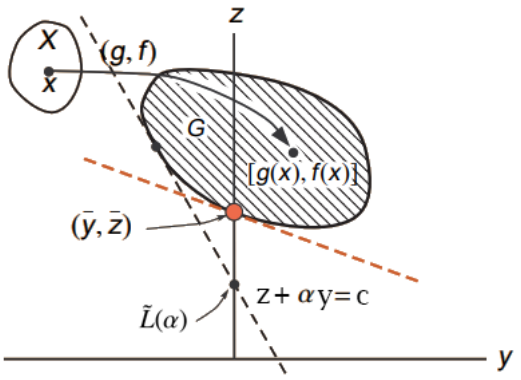
Slika 4.9: Geometrijska interpretacija, preuzeto iz [4]

Kako bi minimizirali izraz $z + \alpha y$ nad skupom G, pomičemo pravac $z + \alpha y = c$ paralelno njemu samom što "niže" moguće dokle god ima neprazan presjek sa skupom G.



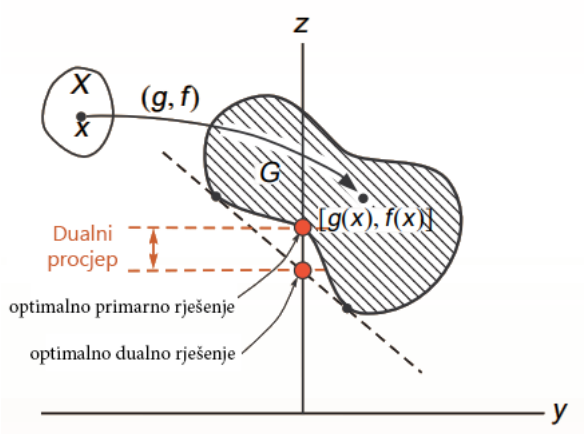
Slika 4.10: Geometrijska interpretacija, preuzeto iz [4]

Odsječak na z osi je tada $\tilde{L}(\alpha)$ sa odgovarajućim $\alpha \geq 0$. Konačno, kako bi riješili dualni problem, moramo pronaći pravac sa nagibom $-\alpha$ takav da je odsječak $\tilde{L}(\alpha)$ na z osi maksimalan. Traženi pravac će imati nagib $-\overline{\alpha}$ te prolaziti kroz točku $(\overline{y}, \overline{z})$ skupa G. Dakle, rješenje dualnog problema je $\overline{\alpha}$ dok je optimalna ciljna vrijednost dualne zadaće $\overline{z}$.



Slika 4.11: Geometrijska interpretacija, preuzeto iz [4]

Primjetimo na trenutak kako ovakvo rješenje ne bi imali da pritom skup G nije bio konveksan.



Slika 4.12: Geometrijska interpretacija, preuzeto iz [4]

Iskažimo teorem jake dualnosti koji nam govori kako se zapravo naš slučaj svodi na gornje geometrijsko razmatranje gdje smo imali podudaranje rješenja primarne i dualne zadaće.

Teorem 4.4.1 (Teorem jake dualnosti)**.** *Neka je X neprazan i konveksan skup u $\mathbb{R}^n$. Neka su $f : \mathbb{R}^n \to \mathbb{R}$ i $g : \mathbb{R}^n \to \mathbb{R}$ konveksne funkcije dok je $h : \mathbb{R}^n \to \mathbb{R}^l$ afina. Pretpostavimo da su ograničenja zadovoljena, odnosno da postoji $\overline{x} \in X$ takav da je $g(\overline{x}) \leq 0$ i $h(\overline{x}) = 0$ (slabi Slaterov uvjet). Tada:*

$$\inf\{f(x) : x \in X, \quad g(x) \leq 0, \quad h(x) = 0\} = \sup\left\{\overline{L}(\alpha), \quad \alpha > 0\right\}$$

gdje je $\overline{L}(\alpha) = \min\left\{f(x) + u^T g(x) + v^T h(x) : x \in X\right\}$

Optimizacija metodom Lagrangeovih multiplikatora

Kombinacijom ciljne funkcije i uvijeta našeg optimizacijskog problema dobivamo sljedeću Lagrangeovu funkciju:

$$L(w, w_0, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i \left\{ y^{(i)} \left(w^T \phi(x^{(i)}) + w_0 \right) - 1 \right\} \tag{4.7}$$

gdje je $\alpha = (\alpha_1, \ldots, \alpha_N)$ vektor Lagrangeovih multiplikatora, po jedan za svako ograničenje. Rješenje ovakvog problema nalazi se u sedlu funkcije L, odnosno vrijednost koja minimizira s obzirom na w i w_0, ali maksimizira u odnosu na α.

Izraz (4.7) predstavlja tzv. primarni problem. Formulirajmo njemu dualni problem koji nam omogućava jednostavnije baratanje ograničenjima. Dualni problem ima i druge prednosti u sklopu SVM-a kao što ćemo vidjeti.

Deriviranjem izraza (4.7), po w, odnosno w_0, te izjednačavanjem sa nulom, dobivamo:

$$w = \sum_{i=1}^{N} \alpha_i y^{(i)} \phi\left(x^{(i)}\right) \tag{4.8}$$

$$0 = \sum_{i=1}^{N} \alpha_i y^{(i)} \tag{4.9}$$

Korištenjem ovih uvjeta možemo iz (4.7) ukloniti varijable w i w_0 te izvesti dualnu Lagrangeovu funkciju:

$$\begin{aligned}
\overline{L}(\alpha) &= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i \left\{ y^{(i)} \left(w^T \phi\left(x^{(i)}\right) + w_0 \right) - 1 \right\} \\
&= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i y^{(i)} w^T \phi\left(x^{(i)}\right) - \underbrace{w_0 \sum_{i=1}^{N} \alpha_i y^{(i)}}_{=0} + \sum_{i=1}^{N} \alpha_i \\
&= \frac{1}{2} \sum_{i=1}^{N} \alpha_i y^{(i)} \phi\left(x^{(i)}\right)^T \sum_{j=1}^{N} \alpha_j y^{(j)} \phi\left(x^{(j)}\right) - \sum_{i=1}^{N} \alpha_i y^{(i)} \phi\left(x^{(i)}\right)^T \sum_{j=1}^{N} \alpha_j y^{(j)} \phi\left(x^{(j)}\right) + \sum_{i=1}^{N} \alpha_i \\
&= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} \phi\left(x^{(i)}\right)^T \phi\left(x^{(j)}\right)
\end{aligned}$$

Sažetije, dualni problem jest maksimizirati izraz:

$$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} \phi\left(x^{(i)}\right)^T \phi\left(x^{(j)}\right) \tag{4.10}$$

uz sljedeća ograničenja na varijablu α:

$$\alpha_i \geq 0, \quad i = 1, \cdots, N$$

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

Prvo ograničenje dolazi od toga što su α_i Lagrangeovi multiplikatori, dok smo drugo ograničenje izveli iznad. Primjetimo da je dualni problem i dalje problem konveksne optimizacije s ograničenjem, odnosno problem kvadratnoga programiranja, no za razliku od primarnog problema gdje smo imali $m + 1$ varijabli (težine w i w_0), ovdje ih imamo N.

Općenito, računalna složenost kvadratnog programiranja je $O(n^3)$, gdje je n broj varijabli. Primijetimo da ako je $m \gg N$, onda se transformacija u dualni problem i računalno isplati.

Treniranje modela se dakle svodi na rješavanje problema kvadratnog programiranja definiranog sa (4.10), a rezultat jest N-dimenzijski vektor parametra α. Nakon što je model naučen, novi primjer x klasificiramo tako da izračunamo $sgn(h(x))$.

Uvrštavanjem (4.8) u (4.1), dobivamo:

$$h(x) = w^T \phi(x) + w_0 = \sum_{i=1}^{N} \alpha_i y^{(i)} \phi(x)^T \phi(x^{(i)}) + w_0 \tag{4.11}$$

Desna strana gornjeg izraza odgovara dualnoj formulaciji problema, u kojoj se, umjesto težina w, pojavljuje vektor α. Kako bismo klasificirali novi primjer, u dualnoj formulaciji ne množimo više njegove značajke s odgovarajućim težinama. Umjesto toga, primjer x uspoređujemo sa svim primjerima $x^{(i)}$ iz skupa za učenje $\mathcal{D}$. Točnije, računamo skalarni produkt $\phi(x)^T \phi(x^{(i)})$, što zapravo znači da uspoređujemo koliko je x sličan primjeru $x^{(i)}$ u prostoru značajki. Pritom svaki primjer $x^{(i)}$ ima pridjeljenu težinu α_i i predznak $y^{(i)}$. Dakle, u dualnoj formulaciji, umjesto da pohranjujemo težine w, moramo pohraniti primjere i njihove oznake. To znači da smo efektivno dobili **neparametarski model**, budući da broj parametara sada ovisi o broju primjera za učenje.

Kao što smo prije napisali, rješenje problema zadovoljava **Karush-Kuhn-Tuckerove (KKT) uvjete**. U našem konkretnom slučaju, ti uvjeti glase:

$$\begin{aligned} \alpha_i &\geq 0 \\ y^{(i)} h(x^{(i)}) - 1 &\geq 0 \\ \alpha_i \left(y^{(i)} h(x^{(i)}) - 1 \right) &= 0 \end{aligned}$$

Iz posljednjeg uvjeta slijedi da za svaki primjer $x^{(i)}$ iz skupa $\mathcal{D}$ vrijedi $\alpha_i = 0$ ili $y^{(i)} h(x^{(i)}) = 1$. To pak znači da će se u zbroju (4.11) pojavljivati samo vektori $x^{(i)}$ za koje $y^{(i)} h(x^{(i)}) = 1$, a to su upravo oni vektori koji leže na ravninama maksimalne margine (s njezine jedne ili druge strane). Kao što smo najavili, te vektore nazivano **potpornim vektorima**. Svi ostali vektori $x^{(i)}$, za koje je $\alpha_i = 0$, uopće ne utječu na izlaz modela i možemo ih posve zanemariti. U praksi to znači da, nakon učenja modela, možemo zadržati samo potporne vektore te je granična hiperravnina definirana linearnom kombinacijom tih vek-

tora. Naravno, to vrijedi samo za predikciju pomoću modela, dok nam za učenje modela trebaju ipak svi primjeri iz skupa $\mathcal{D}$.

Jednom kada je model naučen, uvijek možemo rekonstruirati izvorne varijable primarnog problema. Vektor težina w dan je izrazom (4.8). Primijetimo da samo potporni vektori, dakle oni za koje je $\alpha_i > 0$, definiraju w. Pomak w_0 možemo izračunati temeljem činjenice da za potporne vektore vrijedi $y^{(i)}h(x^{(i)}) = 1$. Neka je S skup indeksa potpornih vektora $x^{(i)}$. Uvrštavanjem (4.11) u taj izraz, dobivamo:

$$y^{(i)}h(x^{(i)}) = y^{(i)}\left(\sum_{j\in S}\alpha_j y^{(j)}\phi(x^{(i)})^T\phi(x^{(j)}) + w_0\right) = 1$$

iz čega slijedi:

$$w_0 = y^{(i)} - \sum_{j\in S}\alpha_j y^{(j)}\phi(x^{(i)})^T\phi(x^{(j)})$$

pri čemu smo iskoristili $\frac{1}{y^{(i)}} = y^{(i)}$ jer je $y^{(i)} \in \{-1, +1\}$. Ovu jednadžbu možemo izračunati na temelju jednog, proizvoljno odabranog označenog primjera $\left(x^{(i)}, y^{(i)}\right)$, no zbog numeričkih odstupanja nećemo za svaki odabir dobiti isto rješenje. Stoga je bolje izračunati prosjek nad svim potpornim vektorima:

$$w_0 = \frac{1}{|\,S\,|}\sum_{i\in S}\left(y^{(i)} - \sum_{j\in S}\alpha_j y^{(j)}\phi(x^{(i)})^T\phi(x^{(j)})\right). \tag{4.12}$$

4.5 Linearno neseparabilni problem odnosno meka margina

Dosada je ključna pretpostavka bila da su primjeri u prostoru značajki linearno odvojivi. Ako ne koristimo preslikavanje, odnosno uzmemo $\phi(x) = x$, prostor značajki jednak je ulaznome prostoru. Prema **Coverovom teoremu** [3] je tada općenito mala vjerojatnost da će problem biti linearno odvojiv. Naime, teorem iskazuje da se vjerojatnost linearne separabilnosti povećava nelinearnom transformacijom vektora u višedimenzionalni prostor, pod uvjetom da prostor nije gusto popunjen.

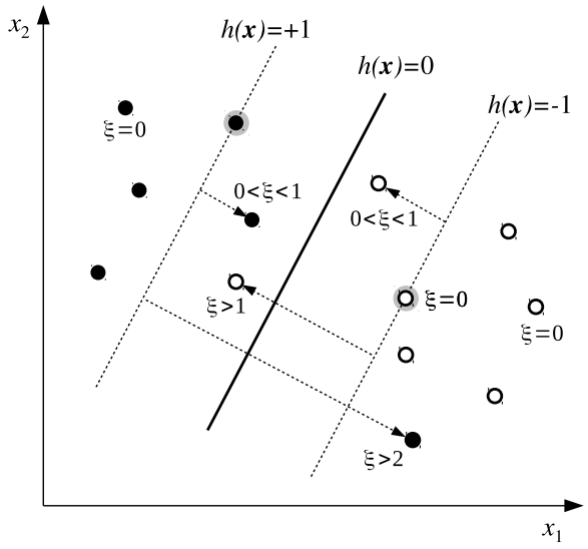
Ako koristimo preslikavanje u prostor značajki tako da je $n \gg m$, povećavamo vjerojatnost da problem bude linearno razdvojiv, međutim nemamo garancije da je to doista tako. S druge strane, ne želimo da model bude suviše nelinearan, jer bi to moglo dovesti do prenaučenosti odnosno priladgodbi šumu u podatcima. Naime, možda je problem baš takav da imamo preklapajuće distribudije klasa, a u takvom slučaju ne želimo da model izgubi svojstvo generalizacije.

Rješenje jest proširenje prethodnog modela tako da on dozvoljava da neki primjeri ipak budu pogrešno klasificirani. Takva formulacija problema se zove **meka margina**.

Kod meke margine, dozvolit ćemo da primjeri budu na pogrešnoj strani granice, ali ćemo ih kažnjavati čim više, što su oni "dublje" na pogrešnoj strani granice. Neka ta kazna raste linearno s udaljenošću primjera od granice. Kažnjavanje ćemo ostvariti uvođenjem **rezervnih varijabli** (*engl. slack variables*), $\xi_i \geq 0, \quad i = 1, \cdots, N$, po jedne za svaki primjer $x^{(i)}$ iz $\mathcal{D}$. Za primjere $x^{(i)}$ koji se nalaze na ispravnoj strani margine vrijedi $\xi_i = 0$, dok za sve druge primjere vrijedi $\xi_i = \mid y^{(i)} - h(x^{(i)}) \mid$. Primjeri koji su unutar margine, ali na ispravnoj strani granice (ili leže upravo na granici), bit će kažnjeni sa $0 < \xi_i \leq 1$.

Primjetimo da, u svrhu dobre generalizacije, ne kažnjavamo samo pogrešno klasificirane primjere, već i primjere koji se nalaze unutar margine.

Sljedeća ilustracija prikazuje dosada navedeno:



Slika 4.13: Meke margine,preuzeto iz [2]

Optimizacijska ograničenja (4.3) sada možemo reformulirati kao:

$$y^{(i)}\left(w^T\phi(x^{(i)}) + w_0\right) \geq 1 - \xi_i, \quad i = 1, \cdots, N. \tag{4.13}$$

Razlika je u tome što sada dozvoljavamo da vrijednost $y^{(i)}h(x)$ za neke primjere bude manja od jedan, odnosno negativna za pogrešno klasificirane primjere. Sada nam je cilj maksimizirati marginu, ali i kazniti primjere koji nisu na pravoj strani margine. Ciljna funkcija koja kombinira ta dva zahtjeva glasi:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \tag{4.14}$$

Parametar $C > 0$ određuje kompromis između veličine margine i ukupne kazne (”mekoće” margine). Veći C dovodi do većeg kažnjavanja pogrešne klasifikacije, što će za posljedicu imati složenije modele. Sada dakle želimo minimizirati:

$$\underset{w,w_0}{\operatorname{argmin}}\left\{\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i\right\}$$

uz ograničenja (4.13) i $\xi_i \geq 0$. To je i dalje problem kvadratnog programiranje te je pripadna Lagrangeova funkcija:

$$L(w, w_0, \xi, \alpha, \beta) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left(y^{(i)}h(x^{(i)}) - 1 + \xi_i\right) - \sum_{i=1}^{N}\beta_i\xi_i$$

gdje su $\alpha_i \geq 0$ i $\beta_i \geq 0$ Lagrangeovi multiplikatori. Primarna formulacija sadrži $m + 3N + 1$ varijabli. Da bismo izveli dualni problem, kao i prije, deriviramo gornji izraz po w, w_0 i ξ_i te ih izjednačimo sa nulom:

$$w = \sum_{i=1}^{N}\alpha_i y^{(i)}\phi(x^{(i)}) \tag{4.15}$$

$$\sum_{i=1}^{N}\alpha_i y^{(i)} = 0 \tag{4.16}$$

$$\alpha_i = C - \beta_i \tag{4.17}$$

Ove jednakosti koristimo kako bismo eliminirali primarne varijable iz Lagrangeove funkcije. Dobivamo dualnu Lagrangeovu funkciju:

$$\tilde{L}(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y^{(i)}y^{(j)}\phi(x^{(i)})^T\phi(x^{(j)}) \tag{4.18}$$

koja je identična funkciji (4.10) za linearno odvojivi slučaj. Međutim, ograničenja su sada nešto drugačija. Naime, vrijedi $\alpha_i \geq 0$ i $\beta_i \geq 0$ jer su to Lagrangeovi multiplikatori, a onda iz (4.17) slijedi $\alpha_i \leq C$. U konačnici, dualni problem jest maksimizirati izraz (4.18) uz ograničenja:

$$0 \leq \alpha_i \leq C$$

$$\sum_{i=1}^{N} \alpha_i y^{(i)} = 0.$$

Prelaskom u dualni oblik, reducirali smo broj varijabli na N. Ponovno je riječ o problemu kvadratnog programiranja čija je općenita računalna složenost $O(N^3)$. Kao i ranije, jednom kada je model naučen i kada imamo izračunat vektor α, za klasifikaciju novog primjera koristima (4.11). Vektori $x^{(i)}$ za koje $\alpha_i = 0$ ne doprinose predikciji. Ostali vektori, za koje $0 < \alpha_i \leq C$, su potporni vektori. Konkretnije, vektori za koje je $\alpha_i = C$ imaju $\xi_i = 0$, tj. leže točno na rubu margine. Vektori za koje je $\alpha_i < C$ leže unutar margine i mogu biti klasificirani ispravno ($\xi_i \leq 1$) ili neispravno ($\xi_i > 1$). Težinu w_0 računamo temeljem izraza (4.12) kao i ranije.

4.6 Nelinearan SVM

SVM je linearan model, no prema Coverovom teoremu, većina problema je nelinearna. Poteškoće nastupaju kada je N, broj parametara u dualnom problemu, mnogo veći od m, dimenzija primjera iz skupa $\mathcal{D}$ jer je tada ulazni prostor gusto popunjen, pa je mala vjerojatnost da je problem linearno odvojiv. Premda postoje problemi kod kojih poteškoća nema jer je $m \gg N$ (npr. problemi genske izražajnosti i klasifikacije teksta), postoje problemi kod kojih niti to nije dovoljno, a da bi oni bili linearno odvojivi (npr. klasifikacija slika temeljem slikovnih elemenata).

Jezgrene funkcije i jezgreni trik

Ideja iza nealinarnog SVM-a leži iza Coverovog teorema. Ako su linearni modeli dovoljno dobri za $m \gg N$, onda to znači da možemo preslikati problem u prostor više dimenzija u kojem je izglednije da će primjeri biti linearno razdvojivi. Dakle, umjesto da transformiramo model i učinimo ga nelinearnim, ideja je transformirati sami problem.

Koristimo preslikavanje $\phi : \mathbb{R}^m \to \mathbb{R}^n$ koja je definirana pomoću baznih funkcija $\phi_j : \mathbb{R}^m \to \mathbb{R}$, tako da $\phi(x) = (\phi_1(x), \cdots, \phi_n(x))$.

Ovakav pristup, međutim, ima niz praktičnih problema:

1. Izračun funkcije preslikavanja može biti računalno zahtjevan

2. Kada bismo unaprijed izračunali preslikavanje za primjere iz skupa $\mathcal{D}$ to bi moglo iziskivati previše memorije, osobito ako je n velik

3. Najveći problem je neznanje o tome da li će preslikavanje stvarno rezultirati u linearno separabilnom problemu

Umjesto izravnog preslikavanja, možemo iskoristiti činjenicu da se u dualnome problemu SVM-a, preslikani vektori $\phi(x)$ uvijek pojavljuju u obliku skalarnoga umnoška. To vrijedi kako za učenje tj. optimizaciju (4.18), tako i za predikciju tj. klasifikaciju novog primjera (4.11). To nam omogućava da iskoristimo tzv. **jezgreni trik** (*engl. kernel trick*) i umnožak dvaju primjera x i x' u prostoru značajki zamijenimo funkcijom:

$$\kappa(x, x') = \phi(x)^T \phi(x')$$

koju nazivamo **jezgrenom funkcijom** (*engl. kernel function*).

Jezgrena funkcija mjeri sličnost dvaju vektora u nekom prostoru značajki. Umjesto da najprije preslikamo vektore u prostor značajki i zatim računamo sličnost vektora kao skalarni produkt, izravno izračunavamo sličnost vektora pomoću jezgrene funkcije. Takav pristup ima dvije važne prednosti. Prva je smanjenje računalne složenosti: izračun jezgrene funkcije često je jednostavniji nego izračun dvaju preslikavanja pa zatim izračun skalarnog umnoška. Druga je ta što prostor značajki koji odgovara jezgrenoj funkciji može biti visoko dimenzionalan (potencijalno beskonačno dimenzionalan).

Dakle, kao što smo vidjeli, za definiranje jezgrene funkcije κ odnosno preslikavanje ϕ imamo tri mogućnosti:

1. *Izravno oblikovanje:* odabrati preslikavanje ϕ, preslikati primjere u prostor značajki te zatim trenirati model
2. *Izravno oblikovanje s jezgrenom funkcijom*: odabrati preslikavanje ϕ, izračunati jezgrenu funkciju te zatim trenirati model s tom funkcijom
3. *Inverzno oblikovanje*: direktno treniranje modela funkcijom κ, gdje funkcija ϕ ostaje nepoznata

Općenito, prednost dajemo inverznom oblikovanju budući da je u načelu lakše definirati jezgrenu funkciju nego preslikavanje ϕ. Pored toga, već postoji niz standardnih jezgrenih funkcija koje su se pokazale učinkovite. Nadalje, prednost inverznog oblikovanja jest što preslikavanje ϕ može biti nepoznato. Jedino što nas zanima jest da, ako definiramo neku jezgrenu funkciju κ, ona doista odgovara skalarnom umnošku u nekom (moguće beskonačno dimenzijonalnom) prostoru značajki.

Primjer 4.6.1. ***(Polinomna jezgrena funkcija i pripadno preslikavanje)*** *Neka je jezgrena funkcija $\kappa(x, z) = (x^T z)^2$. Provjerimo odgovara li ova jezgrena funkcija skalarnome umnošku u nekom prostoru značajki.*

$$\begin{aligned}\kappa(x, z) &= (x^T z)^2 = (x_1 z_1 + x_2 z_2)^2 \\ &= (x_1 z_1)^2 + 2(x_1 z_1)(x_2 z_2) + (x_2 z_2)^2 = x_1^2 z_1^2 + \sqrt{2} x_1 x_2 \sqrt{2} z_1 z_2 + x_2^2 z_2^2 \\ &= (x_1^2, \sqrt{2} x_1 x_2, x_2^2)^T (z_1^2, \sqrt{2} z_1 z_2, z_2^2) = \phi(x)^T \phi(z)\end{aligned}$$

Vidimo da jezgrena funkcija $\kappa(x, z) = (x^T z)^2$ odgovara skalarnom umnošku u prostoru značajki, ako se za preslikavanje upotrijebi funkcija $\phi(x) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2)$. Primijetimo također da je u ovom slučaju jednostavnije izravno izračunati jezgrenu funkciju (4 računske operacije) nego skalarni produkt u prostoru značajki (2×4 operacija za preslikavanje i dodatnih 5 operacija za skalarni produkt).

Mercerove jezgre

Vrijednosti jezgrenih funkcija za sve parove iz skupa za učenje $\mathcal{D}$ možemo izračunati unaprijed i pohraniti u simetričnu matricu dimenzija $N \times N$:

$$K = \begin{pmatrix} \kappa(x^{(1)}, x^{(1)}) & \kappa(x^{(1)}, x^{(2)}) & \dots & \kappa(x^{(1)}, x^{(N)}) \\ \kappa(x^{2)}, x^{(1)}) & \kappa(x^{(2)}, x^{(2)}) & \dots & \kappa(x^{(2)}, x^{(N)}) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(x^{(N)}, x^{(1)}) & \kappa(x^{(N)}, x^{(2)}) & \dots & \kappa(x^{(N)}, x^{(N)}) \end{pmatrix} = \phi^{\mathbf{T}}\phi$$

gdje je ϕ dizajn matrica. Matricu $\mathbf{K}$ nazivamo **Gram-matrica** ili **jezgrena matrica** (*engl. kernel matrix*). Prema **Mercerovom teoremu**, ako je Gram matrica $\mathbf{K}$ pozitivno semidefinitna (tj. $\forall x \neq 0, x^T K x \geq 0$), onda je jezgrenu funkciju κ uvijek moguće rastaviti na skalarni produkt vektora, $\kappa(x, x') = \phi(x)^T \phi(x)$, gdje je prostor značajki Hilbertov prostor H, $\phi(x) \in H$.

Jezgrene funkcije koje to zadovoljavaju nazivamo **Mercerovim jezgrama** ili **pozitivno definitnim jezgrama**. Da bi jezgreni trik funkcionirao, jezgrena funkcija mora biti Mercerova jezgra.

Postoji niz standardnih jezgrenih funkcija koje su Mercerove jezgre. Neke osnovne su sljedeće:

- **Linearna jezgra**: $\kappa(x, x') = x^T x'$, koja efektivno daje linearan model.
- **Polinomijalna jezgra**: $\kappa(x, x') = (x^T x' + 1)^p$, koja uključuje sve kombinacije ulaznih varijabli do uključivo stupnja p. Poseban slučaj polinomijalne jezgre jest $\kappa(x, x') = (x^T x')^p$, koja daje sve kombinacije ulaznih varijabli isključivo stupnja p.
- **Radijalne bazne funkcije** (*engl. radial basis functions, RBF*) ili **homogene jezgre**: $\kappa(x, x') = \kappa(| x - x' |)$, koje ovise samo o udaljenosti između primjera. Poseban slučaj radijalne bazne funkcije jest **Gaussova jezgra:**

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

  gdje parametar $\gamma = \frac{1}{2\sigma^2}$ nazivamo **preciznost**.

Gaussova jezgra mjeri sličnost dvaju primjera temeljem njihove udaljenosti u ulaznom prostoru. Za slične primjere vrijedi $\kappa(x, x') \to 1$, pa su ti primjeri i blizu u prostoru značajki. Za potpuno različite primjere vrijedi $\kappa(x, x') \to 0$, pa su ti primjeri ortogonalni u prostoru značajki (skalarni je umnožak jednak nuli). Parametar γ kontrolira kojom brzinom $\kappa(x, x')$ teže ka nuli u ovisnosti o udaljenosti. Ako je γ malen, $\kappa(x, x') \to 1$ i primjeri su u prostoru značajki grupirani zajedno, što lako dovodi do podnaučenosti. Ako je γ velik, onda je $\kappa(x, x') \to 0$ (izuzev $x = x'$), pa su sve točke u prostoru značajki međusobno ortogonalne, što pak lako dovodi do prenaučenosti. Vidimo da za različite vrijednosti γ-e dobivamo različita preslikavanja, pa je stoga taj parametar vrlo važan.

Općenito, postoji niz operacija nad Mercerovim jezgrama koje zadržavaju to svojstvo i pomoću kojih možemo stvarati nove Mercerove jezgre. Neke od tih su:

$$\begin{aligned}
\kappa(x, x') &= \alpha\kappa_1(x, x') \\
\kappa(x, x') &= \kappa_1(x, x') + \kappa_2(x, x') \\
\kappa(x, x') &= \kappa_1(x, x')\kappa_2(x, x') \\
&\vdots
\end{aligned}$$

Važno je napomenuti da se jezgrene funkcije mogu primijeniti i na slučajeve kada su ulazni primjeri simbolički (dakle ne numerički vektori). Zapravo, prava snaga jezgrenih funkcija dolazi do izražaja upravo onda kad su ulazi strukturirani objekti, npr. grafovi (*engl. graph kernels*), stabla (*engl. tree kernels*), skupovi, nizovi znakova (*engl. string kernels*), tekstualni dokumenti (*engl. text kernels*). U takvim je slučajevima mnogo lakše izravno definirati sličnost dviju struktura nego definirati njihovu vektorizaciju. Npr., jezgrena funkcija za niz znakova može biti definirana kao broj zajedničkih podnizova (i to će biti Mercerova jezgra).

4.7 Optimizacija hiperparametara

Hiperparametar C modela određuje njegovu složenost. Manja vrijednost za C znači da će model dozvoljavati više pogrešaka, tj. bit će jednostavniji. Obrnuto, veća vrijednost za C znači da će model više kažnjavati pogreške, odnosno bit će složeniji.

Ako koristimo (nelinarnu) jezgrenu funkciju, onda trebamo odabrati i hiperparametar jezgrene funkcije (stupanj polinoma p ili preciznost γ). Ti su parametri međusobno povezani. Npr., ako odaberemo visoku vrijednost za γ, dobit ćemo složeniji model, pa će trebati povećati regularizacijski parametar odabirom manje vrijednosti za C. To znači da u fazi "treniranja" moramo optimirati dva parametara istovremeno. To tipično činimo iscrpnim

pretraživanjem u unaprijed definiranom rasponu, tzv. **pretraživanjem po rešetci** (*engl. grid search*). Konkretnije, zamislimo da je skup S_1, konačan skup odabranih mogućih vrijednosti za prvi parametar koji optimiramo, te S_2 analogon za drugi parametar. Tada bi algoritam trenirali nad svakim elementom kartezijevog produkta tih skupova, $S_1 \times S_2$.

4.8 Višeklasna klasifikacija

Do sada je ključna pretpostavka bila da je $y^{(i)} \in \{-1, +1\}$. Iako je algoritam SVM-a stvaran za binarnu klasifikaciju, postoje načini primjene na višeklasne probleme.
Uzmimo da je sada $y^{(i)} \in \{1, \cdots, k\}$.

Metode proširenja SVM-a na višeklasne probleme općenito možemo podijeliti na one direktne i indirektne. Indirektne metode su općenito one koje možemo primijeniti i na bilo koji drugi binarni klasifikator u strojnom učenju, dok su direktne metode specifične za algoritam SVM. Navedimo neke od tih metoda.

Indirektne metode

Jedan-nasuprot-svih (*engl. One-Versus-Rest*)

Jedan-nasuprot-svih metoda ili kraće 1VR, stvara k različitih binarnih klasifikatora za problem klasifikacije sa k klasa. m-ti klasifikator je treniran koristeći primjere iz m-te klase kao pozitivne primjere, te sve preostale primjere iz ostalih $m-1$ klasa kao one negativne.

Konkretnije, m-ti SVM algoritam rješava sljedeći problem:

$$\begin{aligned}
\min_{w^m, b^m, \xi^m} \frac{1}{2} (w^m)^T w^m + C \sum_{i=1}^{l} \xi_i^m \\
(w^m)^T \phi(x^{(i)}) + b^m \geq 1 - \xi_i^m, \text{ ako } y^{(i)} = m, \\
(w^m)^T \phi(x^{(i)}) + b^m \leq 1 - \xi_i^m, \text{ ako } y^{(i)} \neq m, \\
\xi_i^m \geq 0, i = 1, \cdots, l
\end{aligned}$$

Nakon rješavanja ovakvog problema, dobivamo k razdvajajućih hiperravnina.

Neviđeni primjer x klasificiramo tako da:

$$\text{klasa primjera } x \equiv \underset{m=1,\cdots,k}{\operatorname{argmax}} \left((w^m)^T \phi(x) + b^m \right)$$

Problem kod ovakve metode je nebalansiranost klasa. Ukoliko svaka klasa ima jednak broj primjera, tada je omjer pozitivnih i negativnih primjera za svaki model $\frac{1}{k-1}$. Jedan način koji se suprotstavlja tom problemu je povećavanje regularizacijskog parametra C za klasu sa manjim brojem primjera.

Svaki-nasuprot-svakog (*engl. One-Versus-One*)

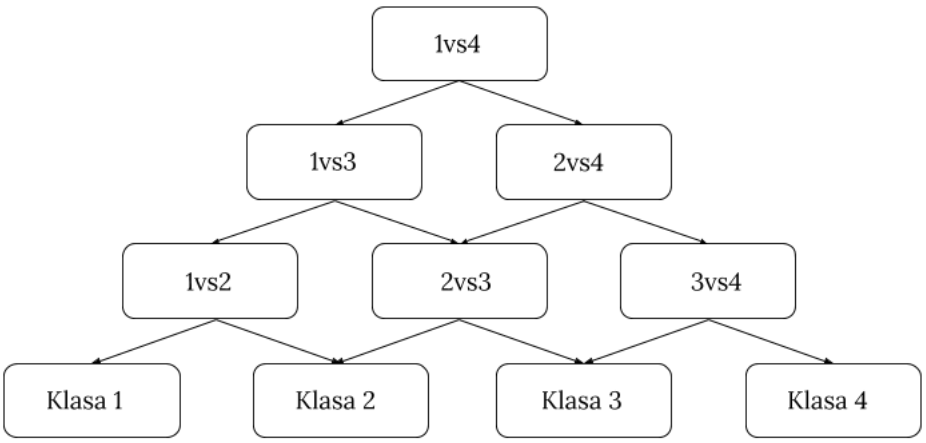
Svaki-nasuprot-svakog metoda, ili kraće 1V1, stvara različitu hiperravninu za svaki par klasa. To znači da na kraju, imamo $\frac{k(k-1)}{2}$ razdvajajućih hiperravnina.

Za primjere iz i-te i j-te klase, rješavamo sljedeći binarni klasifikacijski problem:

$$\begin{aligned}
\min_{w^{ij},b^{ij},\xi^{ij}} & \frac{1}{2}\left(w^{ij}\right)^T w^{ij} + C\sum_t^C \xi_t^{ij} \\
& \left(w^{ij}\right)^T \phi(x^{(t)}) + b^{ij} \geq 1 - \xi_t^{ij}, \text{ ako } y^{(t)} = i, \\
& \left(w^{ij}\right)^T \phi(x^{(t)}) + b^{ij} \leq 1 - \xi_t^{ij}, \text{ ako } y^{(t)} = j, \\
& \xi_t^{ij} \geq 0
\end{aligned}$$

Neviđeni primjer x tada klasificiramo tako da "glasamo" svakim modelom. Konkretnije, svaki model odlučuje kojoj klasi pripada dani primjer, što odgovara jednom glasačkom bodu za tu klasu. Na kraju, primjeru dodjeljujemo klasu koja ima najviše glasova. Ukoliko dvije klase imaju jednak broj glasova, tada postoje razni načini kako se usuglasiti. Najjednostavniji način, koji nije baziran na teoriji, jest biranje klase sa manjim indeksom.

Inačica ovakvog problema jest metoda Direktnog Acikličnog Grafa SVM (*engl. Direct Acyclic Graph SVM*) odnosno kraće DAGSVM. Dok na jednak način treniramo modele, ova metoda se razlikuje po načinu na koji biramo klasu novog neviđenog primjera. Metodu možemo usporediti sa algoritmom stabla odlučivanja. Naime, svaki čvor predstavlja jedan model koji se grana na 2 pod-čvora koji su također modeli. Listovi takvog stabla predstavljaju klase. Intuicija se nalazi na sljedećoj ilustraciji:



Slika 4.14: Dagsvm

Direktne metode

Weston-Watkinsov višeklasni SVM

Umjesto treniranja većeg broja modela, kao u prethodnim slučajevima, ideja je spojiti sve modele u jedan optimizacijski problem. Rezultat je isti kao i u jedan-nasuprot-svih metodi; k dvoklasnih modela sa m-tom hiperravninom razdvajanja $w_m^T\phi(x) + b_m$. Dakle, jedina razlika je način treniranja:

$$\begin{aligned} &\min_{w,b,\xi} \frac{1}{2}\sum_{m=1}^{k} w_m^T w_m + C\sum_{i=1}^{l}\sum_{m\neq y^{(i)}} \xi_i^m \\ &\quad w_{y^{(i)}}^T\phi(x^{(i)}) + b_{y^{(i)}} \geq w_m^T\phi(x^{(i)}) + b_m + 2 - \xi_i^m, \\ &\quad \xi_i^m \geq 0, \quad i = 1,\cdots,l \quad m \in \{1,\cdots,k\} \setminus y^{(i)} \end{aligned} \tag{4.19}$$

Tada, neviđeni primjer x klasificiramo kao:

$$\underset{m=1,\cdots,k}{\operatorname{argmax}} \left(w_m^T\phi(x) + b_m\right).$$

Mana ove metode je njegova složenost. Naime, kada bi htjeli riješiti njegov dualni problem, rješavali bismo optimizacijski problem sa $(k-1)k$ varijabli što može biti itekako veliki broj.

Crammer-Singerov višeklasni SVM

Kao i u prošloj metodi, ideja je riješiti jedan optimizacijski problem u svrhu pronalaska k razdvajajućih hiperravnina.

$$\min_{w_m,\xi_i} \frac{1}{2}\sum_{m=1}^{k} w_m^T w_m + C\sum_{i=1}^{l}\xi_i$$
$$w_{y^{(i)}}^T\phi(x^{(i)}) - w_m^T\phi(x^{(i)}) \geq 1 - \delta_{y^{(i)},m} - \xi_i, \quad i = 1, \cdots, l$$

gdje Kroneckerova delta δ poprima vrijednost 1 ako $y^{(i)} = m$ te 0 ako $y^{(i)} \neq m$.
Tada, neviđeni primjer x klasificiramo kao:

$$\underset{m=1,\cdots,k}{\operatorname{argmax}}\, w_m^T\phi(x).$$

Razlika ove metode i (4.19) jest da ova metoda korsti samo l rezervnih varijabli ξ_i, odnosno, umjesto uzimanja ξ_i^m kao udaljenost dve hiperravnine, ovdje uzimamo maksimum od k takvih udaljenosti kao:

$$\xi_i = \left(\max_m \left(w_m^T\phi(x^{(i)}) + 1 - \delta_{y^{(i)},m}\right) - w_{y^{(i)}}^T\phi(x^{(i)})\right)_+$$

Također ova metoda nema koeficijente b_i te nema potrebu za pisanjem $\xi_i \geq 0$ jer kada je $y^{(i)}$, $1 - \delta_{y^{(i)},m} = 0$ tada imamo:

$$0 \geq 0 - \xi_i$$

što je upravo $\xi_i \geq 0$.

4.9 Primjer

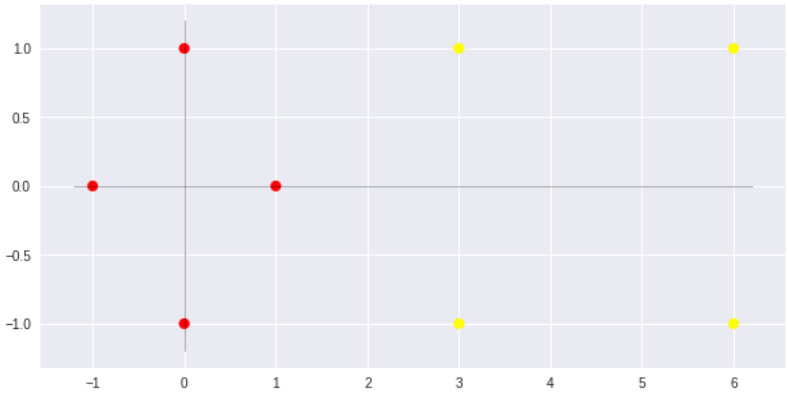
Uzmimo najprije najjednostavniji primjer u kojem, i bez rješavanja problema pomoću SVM-a, je rezultat vidljiv. Cilj takvog jednostavnog primjera je dobivanje što boljeg uvida u korake algoritma, dok ćemo kasnije promotriti i složenije primjere.

Neka su dani sljedeći pozitivni primjeri u $\mathbb{R}^2$:

$$\Big\{(3,1),(3,-1),(6,1),(6,-1)\Big\}$$

i negativni primjeri:

$$\Big\{(1,0),(0,1),(0,-1),(-1,0)\Big\}.$$



Slika 4.15: Primjer

Bez ikakvog računa, odmah vidimo kako bi potporni vektori trebale biti točke $\{(1,0),(3,1),(3,-1)\}$, dok bi razdvajajuća hiperravnina trebala glasiti $x_1 = 2$. Uvjerimo se da je to doista tako.

Kao što smo napomenuli, umjesto rješavanja primarnog problema, možemo odmah preći na njemu dualni problem koji je općenito lakši za rješavanje. Dualni problem glasi:

$$\max \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} \phi\left(x^{(i)}\right)^T \phi\left(x^{(j)}\right)$$

uz ograničenja:

$$\alpha_i \geq 0, \quad i = 1, \cdots, N$$
$$\sum_{i=1}^{N} \alpha_i y^{(i)} = 0$$

Ovaj problem se također može riješiti Lagrangeovom metodom. Napišemo li ciljnu funkciju kao $\frac{1}{2}\alpha^T G\alpha - e^T\alpha$ gdje je $G_{ij} = y^{(i)} y^{(j)} \phi\left(x^{(i)}\right)^T \phi\left(x^{(j)}\right)$ a e vektor jedinica, te ograničenja kao:

$$\alpha_i \geq 0, \quad i = 1, \cdots, N$$
$$y^T\alpha = 0$$

tada Lagrangeova funkcija glasi:

$$L(\alpha, b) = \frac{1}{2}\alpha^T G\alpha - e^T\alpha + by^T\alpha$$

gdje je b Lagrangeov multiplikator.
Deriviranjem izraza po α te b dobivamo sustav:

$$\begin{aligned} G\alpha + by &= e \\ y^T\alpha &= 0 \end{aligned}$$

Tada, naš konkretan problem glasi:

$$\underbrace{\begin{bmatrix} 10 & 8 & 19 & 17 & -3 & -1 & 1 & 3 \\ 8 & 10 & 17 & 19 & -3 & 1 & -1 & 3 \\ 19 & 17 & 37 & 35 & -6 & -1 & 1 & 6 \\ 17 & 19 & 35 & 37 & -6 & 1 & -1 & 6 \\ -3 & -3 & -6 & -6 & 1 & 0 & 0 & -1 \\ -1 & 1 & -1 & 1 & 0 & 1 & -1 & 0 \\ 1 & -1 & 1 & -1 & 0 & -1 & 1 & 0 \\ 3 & 3 & 6 & 6 & -1 & 0 & 0 & 1 \end{bmatrix}}_{G} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_7 \\ \alpha_8 \end{bmatrix}}_{\alpha} + b \underbrace{\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}}_{y} = \underbrace{\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}}_{e}$$

uz

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \end{bmatrix}}_{y^T} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_7 \\ \alpha_8 \end{bmatrix}}_{\alpha} = 0$$

No, znamo da bi $\alpha_3 = \alpha_4 = \alpha_6 = \alpha_7 = \alpha_8 = 0$ iz razloga što oni očito nisu potporni vektori te si stoga možemo olakšati rješavanje sustava na:

$$\underbrace{\begin{bmatrix} 10 & 8 & -3 \\ 8 & 10 & -3 \\ -3 & -3 & 1 \end{bmatrix}}_{G} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}}_{\alpha} + b \underbrace{\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}}_{y} = \underbrace{\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}}_{e}$$

te

$$-\alpha_1 + \alpha_2 + \alpha_3 = 0.$$

Rješavanjem ovog sustava dobivamo: $\alpha_1 = \frac{1}{2}, \quad \alpha_2 = \frac{1}{4}, \quad \alpha_3 = \frac{1}{4}$.
Općenito, razdvajajuća hiperravnina glasi:

$$h(x) = w^T \phi(x) + w_0 = \sum_{i=1}^{N} \alpha_i y^{(i)} \phi(x)^T \phi(x^{(i)}) + w_0 = 0$$

Dakle, potrebno je pronaći w i w_0. Općenito znamo:

$$w = \sum_{i=1}^{N} \alpha_i y^{(i)} \phi(x^{(i)})$$

i

$$w_0 = \frac{1}{\mid S \mid} \sum_{i \in S} \left(y^{(i)} - \sum_j \alpha_j y^{(j)} \phi(x^{(i)})^T \phi(x^{(j)}) \right)$$

što za ovaj primjer znači:

$$w = \alpha_1 y^{(1)} x^{(1)} + \alpha_2 y^{(2)} x^{(2)} + \alpha_3 y^{(3)} x^{(3)} = (1, 0)$$

te

$$w_0 = \frac{1}{3}\left(-1 - \left(-\frac{1}{2} + \frac{3}{4} + \frac{3}{4}\right) + 1 - \left(-\frac{3}{2} + \frac{10}{4} + \frac{8}{4}\right) + 1 - \left(-\frac{3}{2} + \frac{9}{4} + \frac{10}{4}\right)\right) = -2$$

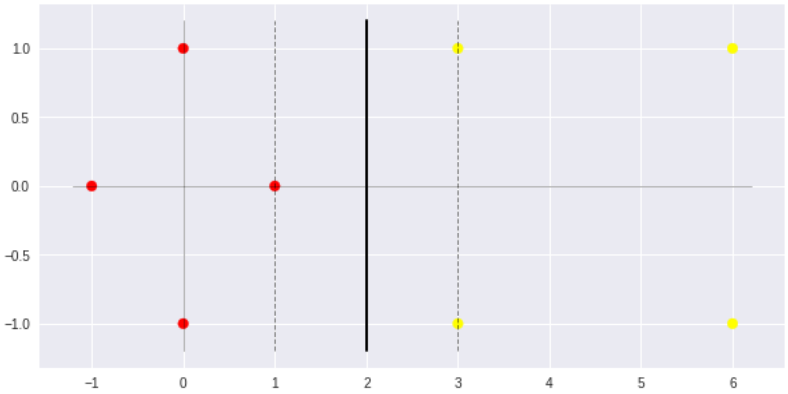
Dakle, razdvajajuća hiperravnina glasi:

$$h(x) = w^T \phi(x) + w_0 = (1, 0)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 2 = x_1 - 2$$

odnosno:

$$x_1 = 2$$

što se lako vidi iz grafa.



Slika 4.16: Primjer

4.10 Primjena

Sada kada imamo bolji uvid u algoritam, možemo intuitivnije zaključiti zašto je SVM primjerenije rješenje za neke probleme, dok za druge nije.

Neke od primjena algoritma su sljedeće:

- Detekcija lica na slikama
- Kategorizacija teksta
- Klasifikacija slika
- Bioinformatika (klasifikacija proteina, klasifikacija gena, itd.)
- Prepoznavanje rukopisa

Općenito postoje par prednosti SVM-a nad drugim metodama:

- **Konveksnost problema**- ne postoje poteškoće u pronalasku globalnog ekstrema. Također, za takve probleme postoje efikasne metode rješavanja
- **Kernelni trik**- odnosno kernelne funkcije koje pružaju mogućnost stvaranja modela sa ekspertnim znanjem u području za koji je model rađen
- **Generalizacija**- SVM je stvaran sa idejom veće generalizacije što mu, osim njegove jednostavne interpretabilnosti daje prednost nad drugim algoritmima strojnoga učenja kao što su logistička regresija ili stabla odlučivanja.

- **Velika dimenzionalnost**- SVM je jako prikladan za visoko dimenzionalne probleme gdje su ulazni podaci x visokodimenzionalni vektori. Rješenje su upravo kernelne funkcije K koje zamjenjuju skalarni produkt raznih vektora $\phi(x)$
- **Memorijska efikasnost**- kako je većina algoritama strojnoga učenja implementirana na računalima, jasno je da postoji problematika memorije. Kao što smo vidjeli, nakon treniranja SVM-a, možemo izbaciti većinu podataka te zadržati samo potporne vektore jer su oni jedini potrebni za određivanje razdvajajuće hiperravnine
- **Regularizacija**- ovisno o linearnoj separabilnosti, SVM sadrži regularizacijski hiperparametar koji navodi korisnika o razmišljanju o prenaučenosti odnosno podnaučenosti

No, kao i svi algoritmi, ima i svoje negativne strane:

- **Odabir kernelne funkcije**- iako su kernelne funkcije prednost SVM-a nad drugim algoritmima, njihov odabir može biti mukotrpan posao

4.11 Regresija potpornih vektora

Stroj potpornih vektora je algoritam koji je prvotno izrađen za rješavanje klasifikacijskih problema. Tek je kasnije adaptiran za regresivne probleme pod imenom regrsija potpornih vektora (*engl. support vector regression*) ili kraće SVR. Kao što je to bilo kod linearne regresije, ideja je pronaći funkciju koja najbolje aproksimira dane primjere. No, ideja ovdje je, pronaći funkciju $f(x)$ koja u najgorem slučaju za primjer $x^{(i)}$ ima ϵ odstupanje od "prave" funkcijske vrijednosti $y^{(i)}$.

Uvedimo najprije oznake. Neka je dani skup podataka za treniranje $\mathcal{D} = \left\{(x^{(1)}, y^{(1)}), \cdots (x^{(l)}, y^{(l)})\right\} \subset \mathcal{X} \times \mathbb{R}$ gdje je $\mathcal{X}$ prostor ulaznih značajki (npr. $\mathcal{X} = \mathbb{R}^m$).

U ϵ-SVR regresiji, dakle, tražimo funkciju:

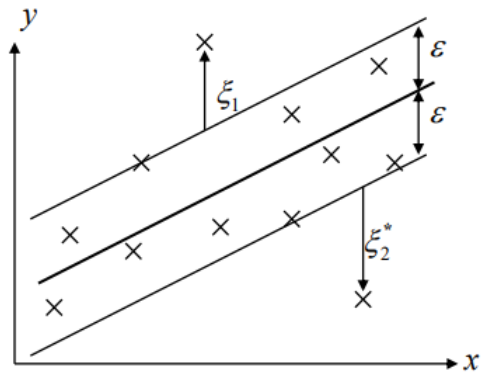
$$f(x) = w^T x + b \quad w \in \mathcal{X},\ b \in \mathbb{R}$$

koja ima najviše ϵ odstupanje od stvarne vrijednosti $y^{(i)}$ za svaki primjer; te je u isto vrijeme što monotonija (*engl. flat*).

Tu monotonost interpretiramo minimiziranjem $\|w\|^2$ kao i kod SVM-a. Više je ciljeva iza minimizacije vektora w. Jedna interpretacija je da su tada predikcije manje osjetljive na perturbacije u značajkama, odnosno outliere. Druga interpretacija je da na neki način biramo nama bitne značajke, stavljajući male koeficijente na one koje ne pridonose modelu.

Dakle, optimizacijski problem možemo sada pisati:

$$\begin{aligned} &\text{minimizirati} && \frac{1}{2}\|w\|^2 \\ &\text{uz ograničenja} && y^{(i)} - wx^{(i)} - b \leq \epsilon \\ & && wx^{(i)} + b - y^{(i)} \leq \epsilon \end{aligned}$$



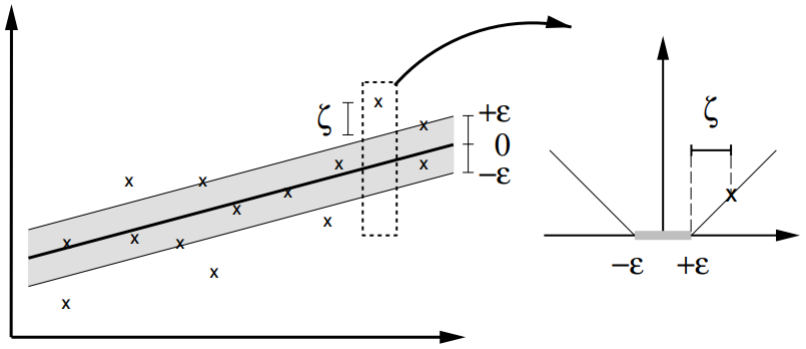
Slika 4.17: Regresija potpornih vektora, preuzeto iz [9]

No, analogno mekoj margini u SVM-u, ponekad i želimo dopustit kakvu veću grešku kako bi održali robustnost modela:

$$\begin{aligned} &\text{minimizirati} && \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ &\text{uz ograničenja} && y^{(i)} - wx^{(i)} - b \leq \epsilon + \xi_i \\ & && wx^{(i)} + b - y^{(i)} \leq \epsilon + \xi_i^* \\ & && \xi_i, \xi_i^* \geq 0 \end{aligned} \tag{4.20}$$

Parametar C je kompromis između monotonost funkcije f, te tolerancije greške veće od ϵ. Tada meku marginu možemo povezati sa funkcijom gubitka:

$$|\,\xi_\epsilon\,| = \begin{cases} 0 & \text{ako } |\,\xi\,| \leq \epsilon \\ |\,\xi\,| - \epsilon & \text{inače} \end{cases}$$



Slika 4.18: Primjer 1

Kao i kod SVM-a zbog lakšeg rješavanja, prelazimo na dualnu formulaciju problema. Slično kao i ranije, Lagrangeova funkcija optimizacijskog problema (4.20) glasi:

$$\begin{aligned} L :=& \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) - \sum_{i=1}^{l}(\eta_i\xi_i + \eta_i^*\xi_i^*) \\ &- \sum_{i=1}^{l}\alpha_i\left(\epsilon + \xi_i - y^{(i)} + w^T x^{(i)} + b\right) \\ &- \sum_{i=1}^{l}\alpha_i^*\left(\epsilon + \xi_i^* + y^{(i)} - w^T x^{(i)} - b\right) \end{aligned} \tag{4.21}$$

gdje su $\eta_i, \eta_i^*, \alpha_i, \alpha_i^*$ Lagrangeovi multiplikatori, odnosno dualne variajble koje dakle trebaju zadovoljavati:

$$\alpha_i^{(*)}, \eta_i^{(*)} \geq 0$$

Parcijalnim deriviranjem po w, b, ξ_i, ξ_i^* dobivamo:

$$\begin{aligned} \nabla_b L &= \sum_{i=1}^{l}(\alpha_i^* - \alpha_i) &&= 0 \\ \nabla_w L &= w - \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)\, x^{(i)} &&= 0 \\ \nabla_{\xi_i^{(*)}} L &= C - \alpha_i^{(*)} - \eta_i^{(*)} &&= 0 \end{aligned} \tag{4.22}$$

Uvrštavanjem (4.22) u (4.21) dobivamo dualni problem:

$$\begin{aligned}
&\text{maksimizirati} \quad -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i-\alpha_i^*)\left(\alpha_j-\alpha_j^*\right)x^{(i)}x^{(j)}-\epsilon\sum_{i=1}^{l}(\alpha_i+\alpha_i^*)+\sum_{i=1}^{l}y^{(i)}(\alpha_i-\alpha_i^*)\\
&\text{uz ograničenja} \quad \sum_{i=1}^{l}(\alpha_i-\alpha_i^*)=0\\
&\qquad\qquad\quad \alpha_i,\alpha_i^*\in[0,C]
\end{aligned} \tag{4.23}$$

Iz (4.22) vidimo da je $w=\sum_{i=1}^{l}\left(\alpha_i-\alpha_i^*\right)x^{(i)}$, iz čega slijedi:

$$f(x)=\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)\,x^{(i)}x+b.$$

Iz KKT uvijeta imamo:

$$\begin{aligned}
\alpha_i\left(\epsilon+\xi_i-y^{(i)}+w^Tx+b\right)&=0\\
\alpha_i^*\left(\epsilon+\xi_i^*+y^{(i)}-w^Tx-b\right)&=0\\
(C-\alpha_i)\,\xi_i&=0\\
(C-\alpha_i^*)\,\xi_i^*&=0
\end{aligned} \tag{4.24}$$

iz čega slijede par zaključaka. Prvo, vidimo da jedino primjeri $(x^{(i)},y^{(i)})$ sa odgovarajućim $\alpha_i^{(*)}=C$ leže izvan ϵ pruge. Drugo, $\alpha_i\alpha_i^*=0$. Tu činjenicu vidimo ako pretpostavimo da za su neki i α_i i α_i^* različiti od nule. Tada iz prve dvije jednadžbe (4.24) slijedi da izrazi u zagradama moraju biti nula. Zbarajnjem tih dvaju izraza dolazimo do kontradikcije. To znači da ne može postojati par dualnih varijabli α_i,α_i^* koje su simultano različite od nule. Prema tome:

$$\begin{aligned}
&\epsilon-y^{(i)}+w^Tx^{(i)}+b\geq 0 \quad \text{te } \xi_i=0 \quad \text{ako } \alpha_i<C\\
&\epsilon-y^{(i)}+w^Tx^{(i)}+b\leq 0 \qquad\qquad\quad \text{ako } \alpha_i>0
\end{aligned} \tag{4.25}$$

Sa tom činjenicom, te analognim zaključcima za α_i^*, slijedi:

$$\max\left\{-\epsilon+y^{(i)}-w^Tx^{(i)}\mid \alpha_i<C \text{ ili } \alpha_i^*>0\right\}\leq b\leq \min\left\{-\epsilon+y^{(i)}-w^Tx^{(i)}\mid \alpha_i>0 \text{ ili } \alpha_i^*<C\right\}$$

Ako vrijedi $\alpha_i^{(*)}\in(0,C)$, tada vrijede jednakosti u jednadžbama iznad. Iz (4.24) vidimo kako Lagrangeovi multiplikatori mogu biti različiti od nule jedino kada vrijedi $\mid f(x^{(i)})-y^{(i)}\mid\geq\epsilon$, odnosno kada se primjer nalazi izvan ϵ pruge. Suprotno tome, kada vrijedi $\mid f(x^{(i)})-y^{(i)}\mid<\epsilon$, tada α_i,α_i^* moraju biti nula kako bi KKT uvijeti bili zadovoljeni.

Upravo te vektore, kojima su α_i,α_i^* različiti od nule zovemo, kao i ranije, **potpornim vektorima**.

Kernelni trik

Kao i ranije, kada smo imali problem linearne nesaparabilnosti, htjeli bi prilagoditi SVR na nelinearne probleme te, kao i ranije, to možemo učiniti preslikavanjem ulaznih primjera funkcijom $\phi : \mathcal{X} \to \mathcal{F}$. No, već smo uočili kompleksnost takvog postupka.

Iz toga razloga koristimo dualnu formu i činjenicu da se u njemu nalazi skalarni produkt koji možemo zamijeniti kernelnom funkcijom. Dakle, izraz (4.23) možemo pisati:

$$\begin{aligned} \text{maksimizirati} \quad & -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)\left(\alpha_j - \alpha_j^*\right)\kappa\left(x^{(i)}, x^{(j)}\right) - \epsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y^{(i)}(\alpha_i - \alpha_i^*) \\ \text{uz ograničenja} \quad & \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \\ & \alpha_i, \alpha_i^* \in [0, C] \end{aligned} \tag{4.26}$$

Primjer

Neka su dani primjeri:

$x^{(i)}$	$y^{(i)}$	$x^{(i)}$	$y^{(i)}$
1	3	11	35
2	4	12	40
3	8	13	45
4	4	14	54
5	6	15	49
6	9	16	59
7	8	17	60
8	12	18	62
9	15	19	63
10	26	20	68

Primijetimo kako u dualnoj formi, prvi član se sastoji od uspoređivanja svaka 2 dana primjera. Ukoliko bi imali 20 danih primjera, prva suma optimizacijskog problema bi se sastojala od 210 članova. Kako bi izbjegli nepreglednost, uzimamo samo 3 primjera i to onih za koji vrijednost od x je u rasponu od 10 do 12. Na prvi pogled možemo vidjeti kako je to malo podataka za ikakav smisleni zadatak, no cilj ovdje nije riješiti konkretan problem, već dati uvid u algoritma.

Tada, jezgrena matrica uz linearnu jezgru izgleda:

$$K = \begin{bmatrix} 100 & 110 & 120 \\ 110 & 121 & 132 \\ 120 & 132 & 144 \end{bmatrix}$$

Iz (4.26) imamo:

$$\begin{aligned} \max_{\alpha,\alpha^*} \frac{1}{2} \Big[& 100\,(\alpha_1 - \alpha_1^*)^2 + 2 \cdot 110 \cdot (\alpha 1 - \alpha_1^*)\,(\alpha_2 - \alpha_2^*) + 2 \cdot 120 \cdot (\alpha 1 - \alpha_1^*)\,(\alpha_3 - \alpha_3^*) \\ & + 2 \cdot 132 \cdot (\alpha 2 - \alpha_2^*)\,(\alpha_3 - \alpha_3^*) + 121\,(\alpha_2 - \alpha_2^*)^2 + 144\left(\alpha_3 - \alpha_3^*\,\right)^2 \Big] \\ & - \epsilon\left[(\alpha_1 + \alpha_1^*) + (\alpha_2 + \alpha_2^*) + (\alpha_3 + \alpha_3^*)\right] \\ & + 26\,(\alpha_1 - \alpha_1^*) + 35\,(\alpha_2 - \alpha_2^*) + 40\,(\alpha_3 - \alpha_3^*) \\ \text{uz ograničenja} \quad & (\alpha_1 - \alpha_1^*) + (\alpha_2 - \alpha_2^*) + (\alpha_3 - \alpha_3^*) = 0 \end{aligned}$$

Kako je naglasak rada na algoritmu SVM, te kako ne bismo produljivali teoriju i dalje, ograničit ćemo se samo na iskaz problema umjesto rješavanja istoga. Naime, potrebne metode za rješavanje ovakvoga optimizacijskog problema bi zahtjevale produbljivanje teorije koje bi bilo izvan područja ovoga rada.

Napomenimo samo kako bi jedna od primjenjivih metoda bila algoritam SMO (*Sequential minimal optimization*) koja dani optimizacijski problem dijeli na manje optimizacijske probleme birajući manji skup varijabli po kojem ga rješava u svakoj iteraciji, te tako dolazi do aproksimacije rješenja uz danu grešku. Detalji o SMO algoritmu se nalaze u [8].

Poglavlje 5

Prilog

U sljedećim poglavljima uspoređujemo SVM i logističku regresiju, te SVR i linearnu regresiju koristeći se jezikom Python. Koristimo gotove pakete kao što su *sklearn* u kojima su implementirani spomenuti algoritmi, te *matplotlib* radi vizualizacije.

5.1 Usporedba SVM-a i logističke regresije

U usporedbi SVM-a i logističke regresije koristimo poznati podatkovni skup "20 newsgroups" u kojem se nalaze novinski članci labelirani jednom od 20 tema. Dvije suprotne teme koje koristimo u usporedbi su religijski članci te ateistički članci.

Učitajmo najprije potrebne pakete:

```
from time import time
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer,
                                            TfidfTransformer
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score,
                            roc_curve,
                            f1_score,
                            classification_report
```

Nadalje učitavamo potrebne podatke:

```
categories = [
    'alt.atheism',
    'talk.religion.misc']

train = fetch_20newsgroups(subset='train', categories=categories,
                           shuffle=True, random_state=1,
                           remove=('headers', 'footers', 'quotes'),
                           data_home='data/',
                           download_if_missing=True)

test = fetch_20newsgroups(subset='test', categories=categories,
                          shuffle=True, random_state=1,
                          remove=('headers', 'footers', 'quotes'),
                          data_home='data/',
                          download_if_missing=True)

print("Ucitavanje '20 newsgroups' skupa podataka za sljedece kategorije:")
print(categories)

print("Ucitano %d dokumenata za treniranje" % len(train.filenames))
print("Ucitano %d dokumenata za testiranje" % len(test.filenames))
```

```
Ucitavanje '20 newsgroups' skupa podataka za kategorije:
['alt.atheism', 'talk.religion.misc']
Ucitano 857 dokumenata za treniranje
Ucitano 570 dokumenata za testiranje
```

Izlistajmo kategoriju i prvih 300 znakova u prvih par članaka:

```
for s,t in zip(train.data[:4],np.take(train.target_names,
                                      train.target[:4])):
    print(t.upper())
    print(s[:300]+'[...]')
    print('-----------------------------------------------------------')
```

```
ALT.ATHEISM
Deletions...
So, you consider the german poster's remark anti-semitic?
Perhaps you imply that anyone in Germany who doesn't
agree with israely policy in a nazi? Pray tell, how does
it even qualify as "casual anti-semitism"?
If the term doesn't apply, why then bring it up?
Your own bigotry is sh[...]
------------------------------------------------------------
ALT.ATHEISM

If the Anne Frank exhibit makes it to your small
little world, take an afternoon to go see it.
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
Bob Beauchaine bobbe@vice.ICO.TEK.COM

They said that Queens could stay, they blew the
Bronx away, and sank Manhattan out at[...]
------------------------------------------------------------
TALK.RELIGION.MISC

(Pleading mode on)
Please! I'm begging you! Quit confusing religious
groups, and stop making generalizations! I'm a Protestant!
I'm an evangelical! I don't believe that my way is the
only way! I'm not a "creation scientist"! I don't think
that homosexuals should be hung by their toenails! [...]
------------------------------------------------------------
ALT.ATHEISM

: }Xenophobia, both *de facto* and *de jure* as implemented
: }in legal systems, is widespread, while the Bible,
: }although not 100% egalitarian, specifically preaches
: }kindness to the stranger, and emphasizes in the Book
: }of Ruth, that a foreigner can join the nation and
: }give rise to one o[...]
------------------------------------------------------------
```

Model logističke regresije:

```
pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('LogReg', LogisticRegression())
])

model = pipeline.fit(train.data, train.target)
y_pred = model.predict_proba(test.data)
classes = model.predict(test.data)
```

Evaluacija modela:

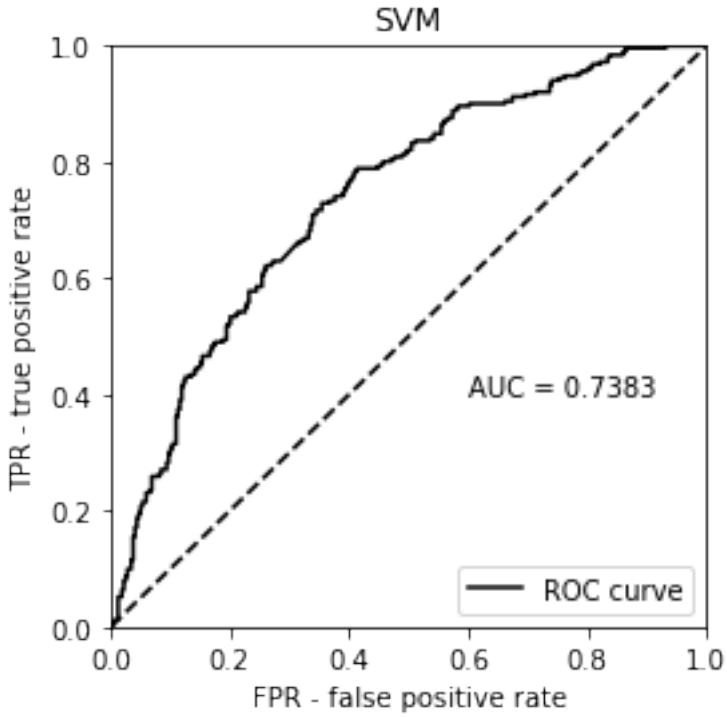
```
fpr, tpr, thresholds = roc_curve(test.target,y_pred[:,1])
auc_score = roc_auc_score(test.target,y_pred[:,1])
print('auc_score: {}'.format(auc_score))
print('')
print(classification_report(test.target, classes))
```

```
auc_score: 0.7383256940888483

             precision    recall  f1-score   support

          0       0.67      0.82      0.74       319
          1       0.69      0.49      0.57       251

avg / total       0.68      0.68      0.66       570
```

ROC krivulja je tada:

```
fig = plt.figure(figsize=(4, 4))
ax = plt.gca()
ax.plot(fpr,tpr,color='black',label='ROC curve')
ax.plot([0,1],[0,1],color='black',linestyle='dashed')
ax.set(xlabel='FPR - false positive rate',
       ylabel='TPR - true positive rate',
       title='SVM',
       xlim=(0,1),ylim=(0,1))
ax.text(0.6,0.4,'AUC = ' + '{0:.4f}'.format(auc_score))
```

```
ax.legend(loc='lower right')

plt.show();
```



Slika 5.1: ROC krivulja

Model SVM-a:

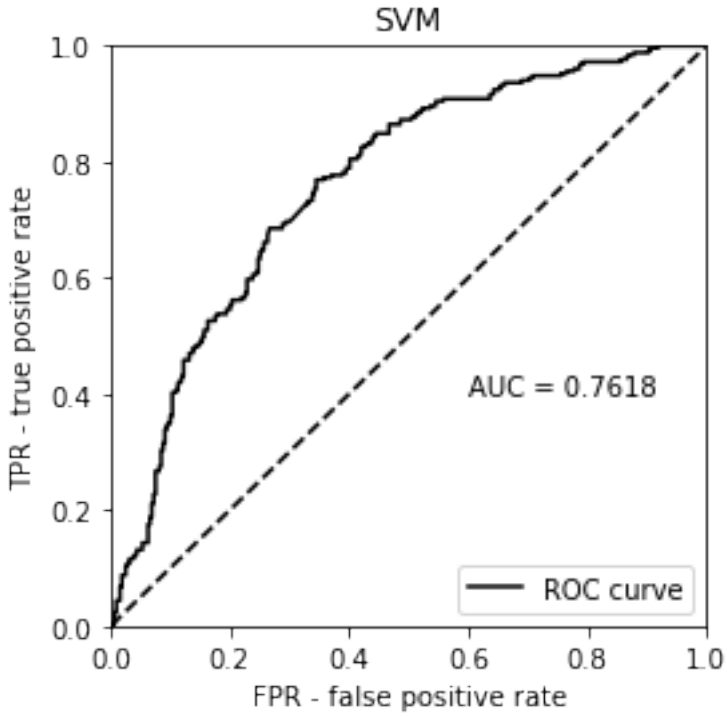
```
pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('SVM', svm.SVC(kernel='linear',probability=True))
])

model = pipeline.fit(train.data, train.target)
y_pred = model.predict_proba(test.data)
classes = model.predict(test.data)

fpr, tpr, thresholds = roc_curve(test.target,y_pred[:,1])
```

```
auc_score = roc_auc_score(test.target,y_pred[:,1])
print('auc_score: {}'.format(auc_score))
print('')
print(classification_report(test.target, classes))
```

```
auc_score: 0.7618366658756823

             precision    recall  f1-score   support

          0       0.70      0.77      0.73       319
          1       0.67      0.58      0.62       251

avg / total       0.68      0.69      0.68       570
```

ROC krivulja:

```
fig = plt.figure(figsize=(4, 4))
ax = plt.gca()
ax.plot(fpr,tpr,color='black',label='ROC curve')
ax.plot([0,1],[0,1],color='black',linestyle='dashed')
ax.set(xlabel='FPR - false positive rate',
       ylabel='TPR - true positive rate',
       title='SVM',
       xlim=(0,1),ylim=(0,1))
ax.text(0.6,0.4,'AUC = ' + '{0:.4f}'.format(auc_score))
ax.legend(loc='lower right')

plt.show();
```



Slika 5.2: ROC krivulja

Pretražimo najbolje hiperparametre C i γ

```
pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('SVM', svm.SVC(kernel='rbf',probability=True))
])

parameters = {
    'SVM__C':(1.0,10.0,100.0,1000.0),
    'SVM__gamma':(0.001,0.01,0.1,1.0)
}

if __name__ == "__main__":
    grid_search = GridSearchCV(pipeline,
                               parameters,
                               n_jobs=-1,
```

```
                                verbose=1,
                                scoring='roc_auc')

    print("Performing grid search...")
    print("pipeline:", [name for name, _ in pipeline.steps])
    print("parameters:")
    print(parameters)
    t0 = time()
    grid_search.fit(train.data, train.target)
    print("done in %0.3fs" % (time() - t0))
    print()

    print("Best score: %0.3f" % grid_search.best_score_)
    print("Best parameters set:")
    best_parameters = grid_search.best_estimator_.get_params()
    for param_name in sorted(parameters.keys()):
        print("\t%s: %r" % (param_name, best_parameters[param_name]))
```

```
Performing grid search...
pipeline: ['vect', 'tfidf', 'SVM']
parameters:
{'SVM__C': (1.0, 10.0, 100.0, 1000.0),
 'SVM__gamma': (0.001, 0.01, 0.1, 1.0)}
Fitting 3 folds for each of 16 candidates, totalling 48 fits

done in 38.123s

Best score: 0.858
Best parameters set:
        SVM__C: 10.0
        SVM__gamma: 0.1
```

Najbolji model je tada:

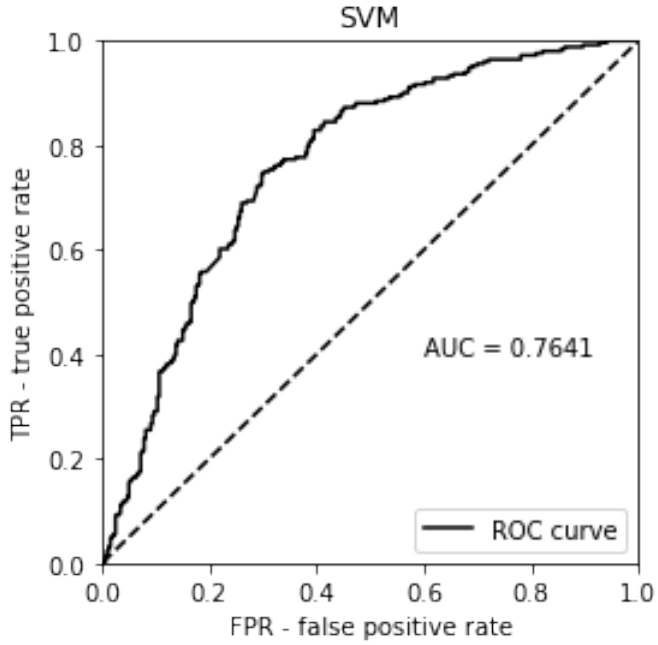
```
model = grid_search.best_estimator_.fit(train.data, train.target)
y_pred = model.predict_proba(test.data)
classes = model.predict(test.data)
```

Evaluacija modela na testnom skupu:

```
fpr, tpr, thresholds = roc_curve(test.target,y_pred[:,1])
auc_score = roc_auc_score(test.target,y_pred[:,1])
print(auc_score)
print('')
print(classification_report(test.target, classes))
```

```
0.7641409284492127

             precision    recall  f1-score   support

          0       0.77      0.70      0.73       319
          1       0.66      0.73      0.69       251

avg / total       0.72      0.71      0.72       570
```

ROC krivulja stvarana istim kodom ko i dosada:



Slika 5.3: ROC krivulja

Možemo primijetiti kako je uz sami linearni kernel, model SVM-a zaista bolji.

5.2 Usporedba SVR-a i linearne regresije

U usporedbi SVR-a i linearne regresije koristimo poznati podatkovni skup ”diabetes” u kojemu se nalaze dani atributi pacijenata oboljelih od dijabetesa kao što su broj godina, spol, bmi,itd. te progresivnost bolesti dijabetesa. Progresivnost bolesti je mjerenje nakon točno jedne godine od prvoga mjerenja statusa bolesti pacijenta čime se može vidjeti razvoj same bolesti.

Učitajmo najprije potrebne pakete:

```
import pandas as pd
from sklearn import datasets
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
```

Nadalje učitavamo potrebne podatke:

```
diabetes = datasets.load_diabetes()
diabetes = pd.DataFrame(data=np.c_[diabetes['data'],
                                   diabetes['target']],
                        columns=diabetes.feature_names+['target'])
diabetes.head()
```

	age	sex	bmi	bp	⋯	s6	target
1	-0.001882	-0.044642	-0.051474	-0.026328	⋯	-0.092204	75
2	0.085299	0.050680	0.044451	-0.005671	⋯	-0.025930	141
3	-0.089063	-0.044642	-0.011595	-0.036656	⋯	-0.009362	206
4	0.005383	-0.044642	-0.036385	0.021872	⋯	-0.046641	135

```
X_train, X_test, y_train, y_test = train_test_split(
    diabetes.drop('target',axis=1),
    diabetes['target'],
    test_size=0.20,
    random_state=42)
```

Model linearne regresije:

```
model = Pipeline([('linear', LinearRegression())])
model = model.fit(X_train,y_train)
y_pred = model.predict(X_test)
```

Evaluacija:

```
MSE = mean_squared_error(y_test, y_pred)
print('MSE: {}'.format(MSE))
R2 = r2_score(y_test, y_pred)
print('R2: {}'.format(R2))
```

```
MSE: 2900.1732878832318
R2: 0.45260660216173787
```

Model SVR-a:

```
X_train, X_test, y_train, y_test = train_test_split(
    diabetes.drop('target',axis=1),
    diabetes['target'],
    test_size=0.20,
    random_state=42)

model = Pipeline([('scale',StandardScaler()),
                  ('SVR', SVR(kernel='linear'))])
model = model.fit(X_train,y_train)
y_pred = model.predict(X_test)

MSE = mean_squared_error(y_test, y_pred)
print('MSE: {}'.format(MSE))
R2 = r2_score(y_test, y_pred)
print('R2: {}'.format(R2))
```

```
MSE: 2939.834541736351
R2: 0.445120736196456
```

```
model = Pipeline([('scale',StandardScaler()),
                  ('SVR', SVR(kernel='linear'))])


parameters = {
    'SVR__C':(0.1,1.0,10.0,100.0,1000.0),
    'SVR__epsilon':(0.001,0.01,0.1,1.0,2.0)
}

if __name__ == "__main__":
    grid_search = GridSearchCV(model,
                               parameters,
                               n_jobs=-1,
                               verbose=1,
                               scoring='mean_squared_error')

    print("Performing grid search...")
    print("pipeline:", [name for name, _ in model.steps])
    print("parameters:")
    print(parameters)
    t0 = time()
    grid_search.fit(X_train, y_train)
    print("done in %0.3fs" % (time() - t0))
    print()

    print("Best score: %0.3f" % grid_search.best_score_)
    print("Best parameters set:")
    best_parameters = grid_search.best_estimator_.get_params()
    for param_name in sorted(parameters.keys()):
        print("\t%s: %r" % (param_name, best_parameters[param_name]))
```

```
Performing grid search...
pipeline: ['scale', 'SVR']
parameters:
{'SVR__C': (1.0, 10.0, 100.0, 1000.0),
'SVR__epsilon': (0.001, 0.01, 0.1, 1.0, 2.0)}

Fitting 3 folds for each of 20 candidates, totalling 60 fits
```

```
[Parallel(n_jobs=-1)]: Done  47 tasks      | elapsed:    7.4s
[Parallel(n_jobs=-1)]: Done  53 out of  60 | elapsed:    8.0s
done in 9.204s

Best score: -3114.069
Best parameters set:
        SVR__C: 1.0
        SVR__epsilon: 2.0
[Parallel(n_jobs=-1)]: Done 60 out of 60 | elapsed: 8.7s
```

Najbolji model je tada:

```
model = grid_search.best_estimator_.fit(X_train, y_train)
y_pred = model.predict(X_test)

MSE = mean_squared_error(y_test, y_pred)
print('MSE: {}'.format(MSE))
R2 = r2_score(y_test, y_pred)
print('R2: {}'.format(R2))
```

```
MSE: 2933.5791374486844
R2: 0.44630141288987424
```

Za razliku od SVM-a, SVR nam ne prikazuje bolje rezultate.

Bibliografija

[1] *Materijali za statistički praktikum 1, PMF, Zagreb*, (2017), https://web.math.pmf.unizg.hr/nastava/statpr/files/linearna.pdf.

[2] B. D. Bašić i J. Šnajder, *Strojno Učenje*, (2014), https://www.fer.unizg.hr/_download/repository/StrojnoUcenje.pdf?forcedefault=true.

[3] T. M. Cover, *Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition*, IEEE Transactions on Electronic Computers **EC-14** (1965), br. 3, 326–334.

[4] Jose De Dona, *Lagrangian Duality*, (2004), http://www.eng.newcastle.edu.au/eecs/cdsc/books/cce/Slides/Duality.pdf.

[5] T. Hastie, R. Tibshirani i J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2009.

[6] C. W. Hsu i C. J. Lin, *A Comparison of Methods for Multi-class Support Vector Machines*, (2010), https://www.csie.ntu.edu.tw/~cjlin/papers/multisvm.pdf.

[7] M. Kuhn i J. Kjell, *Applied predictive modeling*, Springer, 2013.

[8] C. J. Lin, *A Guide to Support Vector Machines*, (2006), https://www.csie.ntu.edu.tw/~b95028/studygroup/optimization/notes/svmguide.pdf.

[9] B Scholkopf i A. J. Smola, *A Tutorial on Support Vector Regression*, (2003), https://alex.smola.org/papers/2003/SmoSch03b.pdf.

Sažetak

Prvo poglavlje ovoga rada uvodi pojam strojnoga učenja, kao i njegovu povijest i primjenu. Sljedeća dva poglavlja uvode regresiju i klasifikaciju te algoritme linearne regresije i logističke regresije koji služe kao mjerilo usporedbe algoritmu potpornih vektora, koji je glavna tematika ovoga rada. Naposljetku slijedi usporedba metode potpornih vektora sa spomenutim metodama logističke regresije i linearne regresije na konkretnom klasifikacijskom i regresijskom problemu.

Summary

The first chapter of this work introduces the concept of machine learning, as well as its history and applications. The next two chapters focus on the concept of regression and classification as well as the appropriate algorithms, linear regression and logistic regression. The aforementioned algorithms are used as a baseline for the main algorithm of this work: the support vector machine algorithm which is talked about in detail in the next chapter. In conclusion, the methods are compared on a concrete regression and classification problem.

Životopis

David Bojanić rođen je 27. veljače 1995. u Rijeci. Nakon završene Prirodoslovno-matematičke taljanske gimnazije "Scuola media superiore italiana Fiume" u Rijeci, upisuje preddiplomski studij matematike na Odjelu za metematiku u Rijeci gdje 2016. godine, pod mentorstvom dr. sc. Andree Švob, piše završni rad na temu "Tri klasična problema u matematici". Iste godine upisuje diplomski studij Financijske i poslovne matematike na Matematičkom odsjeku Prirodoslovno-matematičog fakulteta Sveučilišta u Zagrebu u sklopu kojega piše ovaj rad, završni diplomski rad.