9. Stroj potpornih vektora II

Strojno učenje 1, UNIZG FER, ak. god. 2022./2023.

Jan Šnajder, predavanja, v2.7

Zadnji put bavili smo se **strojem potpornih vektora** te smo definirali njegov model i optimizacijski postupak. Cilj optimizacije bio je nalaženje hiperravnine koja **maksimizira marginu** između primjera dviju klasa, jer maksimalna margina daje najbolju generalizaciju. Taj se optimizacijski problem sveo na **kvadratno programiranje**, koje smo riješili metodom Lagrangeovih multiplikatora. Također smo pogledali kako se iz primarne formulacije problema možemo prebaciti u **dualnu**. U konačnici smo tako dobili dualni model SVM-a, gdje se predikcija za novi primjer ne radi na temelju težina značajki, nego na temelju sličnosti primjera s nekim prototipnim primjerima, koje smo nazvali **potporni vektori**.

Danas nastavljamo tamo gdje smo stali zadnji put. Glavne dvije teme kojima ćemo se baviti jesu **meka margina** i **gubitak zglobnice**. Prva se tema tiče relaksacije pretpostavke o linearnoj odvojivosti primjera, koje smo se držali prošli put, ali koja je vrlo nerealna. Ova izmjena iziskivat će da malo promijenimo optimizacijski problem, ali je dobra vijest da je ta promjena trivijalna i da ne moramo angažirati nikakvu novu matematičku artiljeriju. Druga se tema tiče alternativnog pogleda na SVM, gdje – umjesto da problem maksimalne margine formaliziramo kao optimizaciju uz ograničenja – optimizaciju provodimo na način koji je sličniji onome što smo radili do sada: gradijentnim spustom po funkciji pogreške definiranoj kao očekivanje funkcije gubitka. Na koncu ćemo pogledati još neke važne detalje u vezi s primjenom SVM-a u praksi.

1 Podsjetnik

Prisjetimo se najprije što smo sve naučili zadnji put. Krenuli smo od pretpostavke da su primjeri iz skupa za učenje linearno odvojivi. Zatim smo definirali problem **maksimalne margine**, a onda ga preformulirali kao problem **optimizacije uz ograničenja**. Ograničenja smo definirali tako da za primjere koji su najbliži hiperravnini (primjeri na margini) vrijedi $yh(\mathbf{x}) = 1$:

▸ **Maksimalna margina**

Početni problem:

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

Reformulirani problem:

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

uz ograničenja:

$$y^{(i)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + w_0) \geqslant 1, \quad i = 1, \ldots, N$$

Zaključili smo da je ovo **kvadratni program**, budući da je ciljna funkcija konveksna a ograničenja nejednakosti su afine funkcije. Taj se problem može riješiti metodom **Lagrangeovih multiplikatora**, kod koje ograničenja ugrađujemo u ciljnu funkciju, tzv. **Lagrangeovu funkciju**, koja sadrži dodatne (dualne) varijable:

▸ **Maksimalna margina – primarni problem**

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

uz ograničenja:

$$y^{(i)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + w_0) \geqslant 1, \quad i = 1, \ldots, N$$

Lagrangeova funkcija:

$$L(\mathbf{w}, w_0, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i \Big\{ y^{(i)}\big(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + w_0\big) - 1 \Big\}$$

gdje $\alpha_i \geqslant 0$. Minimizator $(\mathbf{w}^*, w_0*)$ je stacionarna točka (sedlo) od L. U toj točki vrijede uvjeti KKT:

$$\begin{aligned} y^{(i)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + w_0) &\geqslant 1, & i &= 1, \ldots, N \\ \alpha_i &\geqslant 0, & i &= 1, \ldots, N \\ \alpha_i\big(y^{(i)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + w_0) - 1\big) &= 0, & i &= 1, \ldots, N \end{aligned}$$

Ovo je **primarna** formulacija problema, čije je rješenje minimum po primarnim varijablama $(\mathbf{w}, w_0)$ te maksimum po dualnim varijablama $\boldsymbol{\alpha}$, tj. sedlo Lagrangeove funkcije. Međutim, kod SVM-a iz više nam je razloga bolje preći u **dualnu** formulaciju problema. To radimo tako da najprije definiramo **dualnu Langrangeovu funkciju**:

▸ **Maksimalna margina – prijelaz u dual**

$$L(\mathbf{w}, w_0, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i \Big\{ y^{(i)}\big(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + w_0\big) - 1 \Big\}$$

$$\begin{aligned} \frac{\partial L(\mathbf{w}, w_0, \boldsymbol{\alpha})}{\partial \mathbf{w}} = 0 \quad &\Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \alpha_i y^{(i)} \mathbf{x}^{(i)} \\ \frac{\partial L(\mathbf{w}, w_0, \boldsymbol{\alpha})}{\partial w_0} = 0 \quad &\Rightarrow \quad \sum_{i=1}^{N} \alpha_i y^{(i)} = 0 \end{aligned}$$

Dualna Lagrangeova funkcija:

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

Dualna Lagrangeova funkcija je funkcija isključivo dualnih varijabli $\boldsymbol{\alpha}$ i ona nam za zadane vrijednosti dualne varijable daje minimum Lagrangeove funkcije po primarnim varijablama $(\mathbf{w}, w_0)$. S obzirom da je dualna Lagrangeova funkcija donja ograda primarnog (minimizacijskog) problema, te da vrijedi jaka dualnost (procijep dualnosti jednak je nuli), to je rješenje primarnog problema istovjetno maksimumu dualne Lagrangeove funkcije po dualnim varijablama

$\boldsymbol{\alpha}$, uz određena ograničenja na te dualne varijable. Konkretno, dualna formulacija optimizacijskog problema maksimalne margine je:

▸ **Maksimalna margina – dualni problem**

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \Big(\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} (\mathbf{x}^{(i)})^{\mathrm{T}} \mathbf{x}^{(j)} \Big)$$

uz ograničenja:

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

U točki rješenja vrijede uvjeti KKT:

$$y^{(i)}(\mathbf{w}^{\mathrm{T}} \mathbf{x}^{(i)} + w_0) \geqslant 1, \qquad i = 1, \ldots, N$$
$$\alpha_i \geqslant 0, \qquad i = 1, \ldots, N$$
$$\alpha_i \big(y^{(i)} h(\mathbf{x}^{(i)}) - 1 \big) = 0, \qquad i = 1, \ldots, N$$

Ovaj optimizacijski problem opet je problem kvadratnog programiranja, ovaj put uz ograničenja nejednakosti i ograničenja jednakosti. Prednost ovakve formulacije (između ostaloga) jest da se može učinkovito riješiti algoritmom **slijedne minimalne optimizacije (SMO)**, o kojem, međutim, nismo pričali (niti nećemo). Konačno, utvrdili smo da između modela primarne i dualne formulacije SVM-a možemo uspostaviti sljedeću vezu:

▸ **Primarni i dualni model**

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y^{(i)} \mathbf{x}^{(i)}$$
$$h(\mathbf{x}) = \underbrace{\mathbf{w}^{\mathrm{T}} \mathbf{x} + w_0}_{\text{Primarno}} = \underbrace{\sum_{i=1}^{N} \alpha_i y^{(i)} \mathbf{x}^{\mathrm{T}} \mathbf{x}^{(i)} + w_0}_{\text{Dualno}}$$

Veza se temelji na činjenici da vektor težina $\mathbf{w}$ možemo napisati kao linearnu kombinaciju vektora $\mathbf{x}$, pomnoženih s njihovom oznakom $y^{(i)}$. U linearnoj kombinaciji sudjeluju samo oni vektori za koje $\alpha_i > 0$, i te vektore nazivamo **potporni vektori**.

2 Meka margina

Rješenje za vektor $\boldsymbol{\alpha}$ bit će takvo rješenje koje maksimizira marginu uz pretpostavku **linearno odvojivih** primjera. Prisjetimo, tu smo pretpostavku kodirali ovim ograničenjima:

$$y^{(i)}(\mathbf{w}^{\mathrm{T}} \mathbf{x}^{(i)} + w_0) \geqslant 1, \quad i = 1, \ldots, N$$

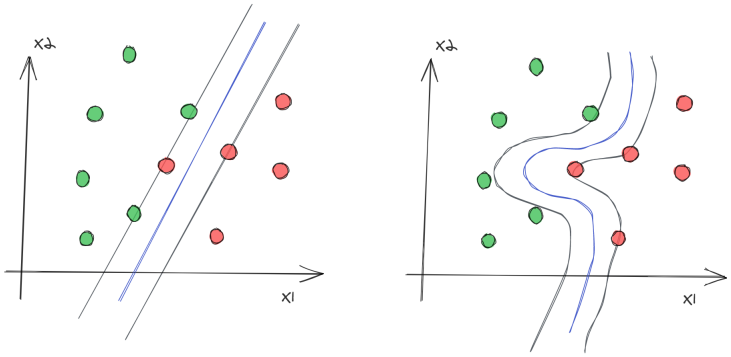
No što ako primjeri nisu linearno odvojivi? Tada jednostavno ne postoji rješenje koje bi zadovoljilo gornji uvjet, pa je ostvarivo područje prazan skup! Drugim riječima, ne postoji hiperravnina koja točno odvaja primjere.

To nije dobro. Ne želimo situaciju u kojoj nemamo nikakvo rješenje. Npr., sjetite se algoritma logističke regresije: ako primjeri nisu linearno odvojivi, ipak ćemo dobiti neko rješenje, i to ono koje minimizira empirijsku pogrešku na skupu za učenje. (Kod perceptrona nećemo dobiti nikakvo rješenje ako primjeri nisu linearno odvojivi, tj. algoritam perceptrona u tom slučaju ne konvergira, ali to smo već ustanovili da je nedostatak koji ne možemo prihvatiti.) Kako bismo to riješili? Ako primjeri nisu linearno odvojivi, želimo da nam algoritam SVM-a, kao i algoritam logističke regresije, da onu hipotezu koja je najbolja moguća, kad već ne postoji savršena hipoteza.

Vjerojatno prva stvar koja nam pada na pamet jest da preslikamo primjere pomoću funkcije ϕ u prostor značajke više dimenzije, kao što smo radili do sada. To je vrlo dobra ideja, i to ćemo svakako raditi, jer ćemo na taj način, kao i do sada, iz linearnog modela efektivno dobiti nelinearan model. Međutim, to nije principijelno rješenje, i to iz dva razloga.

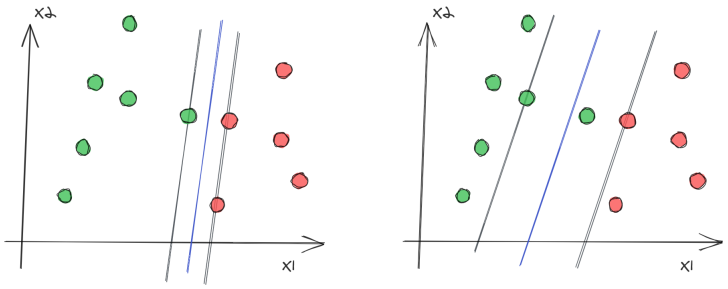
Prvi problem je da se vrlo lako može dogoditi da ne uspijemo primjere preslikati u prostor gdje bi oni bili linearno odvojivi. Općenito nemamo garanciju da ćemo primjere uspjeti preslikati u prostor značajki u kojem su oni linearno odvojivi, odnosno općenito ne znamo koju funkciju ϕ trebamo odabrati, a da bi preslikavanje rezultiralo linearno odvojivim problemom.

Drugi problem je: želimo li stvarno bilo koji ulazni problem preslikati u prostor gdje je on linearno odvojiv? Što ako u ulaznom prostoru postoji **šum**, i dvije klase zbog šuma nisu linearno odvojive? Ako ih preslikamo u toliko visokodimenzijski prostor da one postoru odvojive, sigurno ćemo dobiti **prenaučen** model! Pogledajmo primjer takve situacije:



Ako je najlijeviji crveni primjer na lijevoj slici šum (bilo pogrešno označen ili pogrešno smješten), onda ne bismo željeli da taj primjer utječe na granicu i marginu, i idealno bi bilo da granica i margina izgledaju ovako kako je prikazano na lijevoj slici. Ako, međutim, primjere preslikamo u prostor značajki dovoljno visoke dimenzije i tamo ih odvojimo linearnom granicom, onda će u ulaznom prostoru granica biti visoko nelinarna i previše prilagođena šumu, kao što prikazuje desna slika.

No, čak i ako primjeri jesu linearno odvojivi u ulaznom prostoru, trenutačni postupak inzistira na nalaženju maksimalne margine, čak i onda kada, opet zbog šuma, to možda nije najpametnije:



Na lijevoj slici imamo zeleni primjer koji je šum (bilo pogrešno označen ili pogrešno smješten). Primjeri su, međutim, linearno odvojivi, i granica maksimalne margine je poprilično uska. Na desnoj je slici prikazana margina koja međutim dopušta da dotični zeleni primjer bude unutar margine (ovdje čak i na suprotnoj strani granice!). No, takva margina je znatno šira, što znači da će model bolje generalizirati.

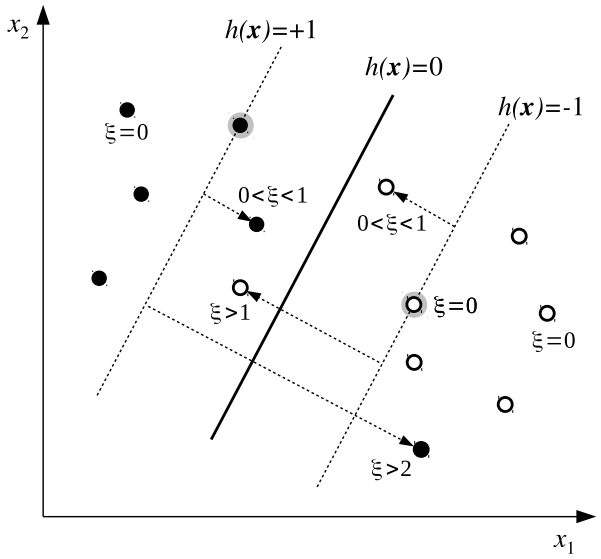
Dakle naš problem je što trenutna formulacija optimizacijskog postupka inzistira na linearnoj odvojivosti primjera. Ako primjeri nisu linearno odvojivi, onda nema rješenja. Ili imamo rješenje maksimalne margine, ili nemamo nikakvo rješenje! Očito je da nam treba malo više fleksibilnosti. Sjetite se kako smo to radili do sada: željeli smo minimizirati empirijsku pogrešku, ali smo dopuštali da se primjeri pogrešno klasificiraju i to je uzrokovalo gubitke. Ideja je da istu stvar napravimo i ovdje: dopustit ćemo da primjeri ulijeću u marginu, pa čak i da budu s krive strane granice, ali ćemo onda malo kažnjavati hipotezu ako se to dogodi. Takva formulacija problema naziva se **meka margina** (engl. *soft margin*). Formulaciju koju smo imali prije, gdje nismo dopuštali nikakve pogreške, onda nazivamo **tvrda margina** (engl. *hard margin*). 1

2.1 Formulacija meke margine

Meku marginu ćemo vrlo jednostavno ostvariti tako da malo promijenimo optimizacijska ograničenja, na sljedeći način:

$$y^{(i)}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + w_0\right) \geqslant 1 - \xi_i, \quad i = 1, \ldots, N$$

Ovdje smo uveli tzv. **rezervnu varijablu** (engl. *slack variable*) ξ_i, po jednu za svaki primjer $\mathbf{x}^{(i)}$. Vrijedi $\xi_i \geqslant 0$. Ta nam varijabla govori koliko je primjer ušao u marginu ili čak prešao granicu. Naime, ako je $\xi_i = 0$, onda znači da je primjer na ispravnoj strani granice i izvan margine, i imamo istu situaciju (odnosno ograničenja) kao što smo imali ranije s tvrdom marginom. Međutim, ako je $\xi_i > 0$, onda znači da više ne vrijedi $y^{(i)}h(\mathbf{x}^{(i)}) \geqslant 1$, nego je sada $y^{(i)}h(\mathbf{x}) < 1$, a to znači da je primjer $\mathbf{x}^{(i)}$ ušao u marginu ili je čak prešao granicu. Konkretno, ako je $\xi_i = 1$, onda to znači da je $y^{(i)}h(\mathbf{x}^{(i)}) = 0$, tj. primjer je točno na granici. Nadalje, ako je $\xi_i > 1$, onda to znači da je $y^{(i)}h(\mathbf{x}^{(i)}) < 0$, što znači da je primjer na drugoj strani hiperravnine i da je pogrešno klasificiran. Navedene slučajeve ilustrira sljedeća skica:



Slika prikazuje dvodimenzijski ulazni prostor s primjerima iz dviju klasa (crni i bijeli kružići). Granica između klasa je hiperravninva (ovdje pravac) za koji $h(\mathbf{x}) = 0$. Primjeri za koje $h(\mathbf{x}) = 1$ i $h(\mathbf{x}) = -1$ su potporni vektori i oni su pozicionirani točno na margini. Na ovoj slici imamo dva potporna vektora, po jedan iz svake klase. Za potporne vektore vrijedi $\xi_i = 0$ i $yh(\mathbf{x}) = 1$. Za primjere koji su izvan margine, a točno klasificirani, vrijedi $\xi_i = 0$ i $yh(\mathbf{x}) > 1$. Za primjere koji su ušli u marginu, ali su još uvijek na ispravnoj strani granice, vrijedi $0 < \xi_i < 1$, tj. ti primjeri će biti kažnjenji sa ξ_i koji je manji od jedinice. Na slici imamo dva takva primjera, po jedan iz svake klase. Primjeri koji pređu na krivu stranu hiperravnine imaju $\xi > 1$, tj. ti su primjeri kažnjeni još više. Na slici imamo jedan takav primjer koji je prešao u poluprostor pozitivne klase. Konačno, primjer koji pređe ne samo na pogrešnu stranu hiperravnine nego još izađe i iz margine s pogrešne strane imat će $\xi_i > 2$. na slici je to jedan primjer koji je prešao u poluprostor negativne klase. Ukupno, za četiri primjera vrijedi $\xi_i > 0$, odnosno četiri primjera ne poštuju ograničenja tvrde margine.

Pa, sažmimo: Ako je primjer $\mathbf{x}^{(i)}$ ispravno klasificiran, onda $\xi_i = 0$. Međutim, ako primjer $\mathbf{x}^{(i)}$ nije ispravno klasificiran, onda $\xi_i > 0$ i ξ_i je jednak odstupanju izlaza modela od točne klasifikacije, tj. $\xi_i = |y^{(i)} - h(\mathbf{x}^{(i)})|$. Primjeri koji su na pogrešnoj strani imat će $\xi_i > 1$. Primjeri koji su na ispravnoj strani granice (ili leže upravo na njoj), ali su ušli unutar margine, imat će $0 < \xi_i \leqslant 1$.

Ovime smo dobili željenu fleksibilnost. Ovako modificirana ograničenja dopuštaju da pojedini primjeri uđu unutar margine ili čak s druge strane granice. No, ne želimo da to bude masovna pojava. Želimo to dopustiti, ali ne previše. Ovdje, dakle, imamo dva kriterija: želimo maksimalnu marginu koja dopušta pogrešne klasifikacije i ulete u marginu, ali istovremeno želimo da je takvih situacija što manje. Ova dva kriterija međusobno su oprečna: možda možemo dobiti savršenu klasifikaciju, ali onda bi margina mogla biti vrlo uska. Obrnuto, možemo dobiti široku marginu, ako dopustimo ulaz mnogih primjera u marginu. Postavlja se pitanje kako ostvariti ovakav kompromis pri optimizaciji?

Prisjetimo se: naša ciljna funkcija koju smo minimizirali bila je:

$$\frac{1}{2}\|\mathbf{w}\|^2$$

i to daje maksimalnu, ali tvrdu marginu. Kako ostvariti kompromis? Na isti način kao što smo to napravili kod regularizacije: linearnom kombinacijom dvaju uvjeta. Ako dozvoljavamo pogreške, a pogreške modeliramo sa ξ_i, onda ćemo ciljnu funkciju redefinirati ovako:

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

Hiperparametar $C > 0$ određuje **kompromis** između širine margine i tvrdoće margine. Veći C dovodi do većeg kažnjavanja pogrešne klasifikacije, što će za posljedicu imati "čišću" odnosno tvrđu marginu. Manji C znači da nas je manje briga za ulete u marginu i za pogrešne klasifikacije, što će nam dati poroznu, ali široku marginu. Kako to utječe na **složenost modela**? Veći C daje tvrđe margine, dakle složenije modele. Manji C daje mekše margine, dakle jednostavnije modele.

Napišimo sad u kompletu naš optimizacijski problem:

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

uz ograničenja:

$$\begin{aligned} y^{(i)}\big(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + w_0\big) &\geqslant 1 - \xi_i, \qquad & i = 1, \ldots, N \\ \xi_i &\geqslant 0, & i = 1, \ldots, N \end{aligned}$$

Primijetite da su varijable ξ_i također varijable po kojima optimiramo, i želimo da su one što manje, kako bi drugi pribrojnik u ciljnoj funkciji bio što manji. Međutim, to nisu parametri modela (za razliku od dualne varijable $\boldsymbol{\alpha}$, koja jest parametar modela u dualnoj formulaciji).

Ovimo smo definirali optimizacijski problem. Iduće pitanje na koje moramo odgovoriti jest: kako ovo optimirati?

2.2 Dualno kvadratno programiranje

Kao i kod tvrde margine, optimizacijski problem koji smo upravo definirali je **kvadratni program**. Kao i kod tvrde margine, prebacit ćemo se u **dualni problem**, zbog prednosti koje smo bili natuknuli prošli put. No, krenimo redom. Najprije definiramo **Lagrangeovu funkciju**:

$$L(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i\big(y^{(i)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + w_0) - 1 + \xi_i\big) - \sum_{i=1}^{N} \beta_i\xi_i$$

Ova Lagrangeova funkcija ima primarne varijable $\mathbf{w}$, w_0 i $\boldsymbol{\xi}$ te dualne varijable $\boldsymbol{\alpha}$ i $\boldsymbol{\beta}$. Rješenje je stacionarna točka (sedlo) ove funkcije, koje je minimum po primarnim varijablama i maksimum po dualnim varijablama. U točki rješenja vrijedit će sljedećih šest uvjeta KKT:

$$\begin{aligned} y^{(i)}\big(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + w_0\big) &\geqslant 1 - \xi_i & i = 1, \ldots, N \\ \xi_i &\geqslant 0 & i = 1, \ldots, N \\ \alpha_i &\geqslant 0 & i = 1, \ldots, N \\ \beta_i &\geqslant 0 & i = 1, \ldots, N \\ \alpha_i\big(y^{(i)}(\mathbf{w}^{\mathrm{T}}\mathbf{x}^{(i)} + w_0) - 1 + \xi_i\big) &= 0 & i = 1, \ldots, N \\ \beta_i\xi_i &= 0 & i = 1, \ldots, N \end{aligned}$$

Sada želimo preći u dualni problem. Za to moramo definirati **dualnu Lagrangeovu funkciju**, koja je funkcija dualnih varijabli i koja minimizira Lagrangeovu funkciju po primarnim varijablama. Dakle, prvo trebamo izračunati minimum Lagrangeove funkcije po primarnim varijablama. U tu svrhu deriviramo Lagrangeovu funkciju po $\mathbf{w}$, w_0 i ξ_i i izjednačavamo s nulom, što nam daje:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{w}} = 0 \quad &\Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \alpha_i y^{(i)} \mathbf{x}^{(i)} \\ \frac{\partial L}{\partial w_0} = 0 \quad &\Rightarrow \quad \sum_{i=1}^{N} \alpha_i y^{(i)} = 0 \\ \frac{\partial L}{\partial \xi_i} = 0 \quad &\Rightarrow \quad \alpha_i = C - \beta_i \end{aligned}$$

Ove jednakosti sada uvrštavamo nazad u Lagrangeovu funkciju L, kako bismo dobili dualnu Lagrangeovu funkciju:

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

Primijetite da je ovo posve ista dualna Lagrangeova funkcija kao i ona za tvrdu marginu! Međutim, ono što je ovdje drugačije su ograničenja. Naime, imamo multiplikatore β_i, koje nismo imali kod tvrde margine. Budući da $\beta_i \geqslant 0$, a da smo gore izveli jednakost $\alpha_i = C - \beta_i$, to znači da:

$$\begin{aligned} \alpha_i &= C - \beta_i \\ \beta_i &= C - \alpha_i \geqslant 0 \\ \alpha_i &\leqslant C \end{aligned}$$

Također, otprije znamo da vrijedi $\alpha_i \geqslant 0$, pa zaključujemo da mora vrijediti ograničenje $0 \leqslant \alpha_i \leqslant C$. Dakle, naš dualni problem je:

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \Big(\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} (\mathbf{x}^{(i)})^{\mathrm{T}} \mathbf{x}^{(j)} \Big)$$

uz ograničenja:

$$\begin{aligned} 0 \leqslant \alpha_i \leqslant C, &\qquad i = 1, \ldots, N \\ \sum_{i=1}^{N} \alpha_i y^{(i)} = 0, &\qquad i = 1, \ldots, N \end{aligned}$$

(Ograničenje jednakosti moramo uvrstiti jer nismo eliminirali varijablu α_i.) Primijetite da je jedina razlika u odnosu na tvrdu marginu ta što, pored donje ograde, postoji i gornja ograda na α, tj. ograničenje $\alpha_i \leqslant C$. I ponovo se ovdje radi o kvadratnom programiranju, koje se može učinkovito (u $\mathcal{O}(N^2)$) riješiti algoritmom **slijedne minimalne optimizacije (SMO)**. 2

2.3 Predikcija

Kao što smo ustanovili zadnji put, u ovoj dualnoj formulaciji, jednom kada je model naučen i kada imamo izračunat vektor $\boldsymbol{\alpha}$, za klasifikaciju novog primjera koristimo izraz:

$$h(\mathbf{x}; \boldsymbol{\alpha}, \mathcal{D}) = \sum_{i=1}^{N} \alpha_i y^{(i)} \mathbf{x}^{\mathrm{T}} \mathbf{x}^{(i)} + w_0$$

Prisjetimo se, predikcija funkcionira tako da primjer $\mathbf{x}$ uspoređujemo po sličnosti s prototipnim primjerima iz $\mathcal{D}$. Ta se uporedba čini pomoću skalarnog produkta $\mathbf{x}^{\mathrm{T}} \mathbf{x}^{(i)}$. Rezultat toga jest da će prototipni primjeri koji su najsličniji ulaznom primjeru imati najviše utjecaja na njegovu klasifikaciju. Prototipni primjeri su oni za koje je $0 < \alpha_i \leqslant C$, i njih zovemo **potporni vektori**. Iz uvjeta KKT slijedi, pomalo protivno intuiciji, da potporni vektori za koje $\alpha_i < C$ leže na margini, dok potporni vektori za koje $\alpha_i = C$ mogu ležati i unutar margine. 3

Primijetite da se, kao i kod tvrde margine, u dualnom modelu SVM-a više ne pojavljuju težine $\mathbf{w}$. Parametar modela više nije n-dimenzijski vektor težina, nego su parametri N-dimenzijski vektor $\boldsymbol{\alpha}$ i pripadni potporni vektori, odnosno označeni primjeri $(\mathbf{x}_i, y^{(i)})$ za koje $\alpha_i > 0$. Pritom, kao i kod tvrde margine, parametar w_0 možemo izračunati pomoću potpornih vektora, dakle w_0 nam ne treba kao zaseban parametar. 4

3 Gubitak zglobnice

Naučili smo već mnogo toga o SVM-u. Pristup koji smo do sada razmatrali, a koji se svodi na formalizaciju problema maksimalne margine kao problema optimizacije uz ograničenja, standardni je pristup i on odgovara povijesnom razvoju SVM-a. Primijetite, međutim, da u tom pristupu uopće nismo pričali o funkciji gubitka niti o pogrešci, kao što smo to radili kod prethodnih algoritama, npr. linearne ili logističke regresije, već smo izravno definirali optimizacijski problem kao kvadratni program. Međutim, sada kada smo objasnili osnovne principe SVM-a, možemo razmotriti i alternativnu formulaciju optimizacijskog problema SVM u smislu gubitka i funkcije pogreške. To će nam omogućiti dvije stvari. Prvo, da SVM ne rješavamo kvadratnim programiranjem, nego npr. nekom varijantom **gradijentnog spusta**. I drugo, da SVM izravno usporedimo s drugim linearnim modelima koje smo već upoznali.

Krenimo od toga da se prisjetimo kako izgleda ciljna funkcija **primarnog** optimizacijskog problema za slučaj meke margine:

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

Zastanite ovdje na trenutak i pogledajte još jednom ciljnu funkciju koju smo upravo napisali. Izgleda li vam ova funkcija poznato? Sjetite se, npr., logističke regresije. Pa, ovo izgleda baš kao **L2-regularizirana empirijska pogreška**. Naime, nju smo definirali ovako:

$$E_R(\mathbf{w}|\mathcal{D}) = \sum_{i=1}^{N} L(y^{(i)}, h(\mathbf{x}^{(i)};\mathbf{w})) + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

Ako ovdje stavimo $\lambda = 1/C$, dobit ćemo (do na faktor C) identičan izraz gornjoj ciljnoj funkciji!

Prema tome, možemo zaključiti da je ciljna funkcija primarnog optimizacijskog problema zapravo L2-regularizirana funkcija empirijske pogreške! Pritom izraz $\frac{1}{2}\|\mathbf{w}\|^2$, kojim osiguravamo da margina bude što šira, zapravo odgovara **regularizacijskom izrazu**, dok izraz $C\sum \xi_i$, kojime kažnjavamo primjere koji ulaze u marginu, odgovara **funkciji gubitka**. Ova prva korespondencija vjerojatno više iznenađuje nego ova druga. Međutim, da tendencija za širom marginom odgovara regularizaciji već smo bili napomenuli, budući da je cijela ideja maksimalne margine upravo ostvariti veću generalizaciju modela, a to znači manju sklonost prenaučenosti, što znači manju složenost, i to upravo ostvarujemo regularizacijom. Dakle, kada malo razmislimo, obje korespondencije imaju smisla. Međutim, ono što je zanimljivo je razlika u načinu kako smo došli do izraza za ciljnu funkciju SVM-a i za L2-regulariziranu pogrešku linearnog modela. Naime, izraz koji odgovara regularizacijskom faktoru imali smo u priči od SVM-a otpočetka, jer je taj izraz bio sastavni dio kriterija maksimalne margine, dok smo izraz s rezervnim varijablama dodali naknadno, kako bismo ostvarili meku marginu. Kod poopćenih linearnih modela išli smo obrnuto: prvo smo definirali empirijsku pogrešku kao očekivanje funkcije gubitka, a tek potom dodali regularizacijski izraz.

Vratimo se ciljoj funkciji SVM-a i pokušajmo je napisati kao L2-regulariziranu pogrešku. Vidimo da regularizacijski faktor već imamo. Pogledajmo možemo li nekako napisati ovaj ξ_i u obliku funkcije gubitka. Prisjetimo se da za primjere koji su na ispravnoj strani granice i izvan margine vrijedi $y^{(i)}h(\mathbf{x}^{(i)}) \geqslant 1$ te da ti primjeri ne nanose gubitak. Primjere koji su unutar margine ili na pogrešnoj strani granice nanose gubitak $\xi_i = |y^{(i)} - h(\mathbf{x}^{(i)})|$, što je jednako $1 - y^{(i)}h(\mathbf{x}^{(i)})$. To onda znači da funkciju gubitka možemo definirati kao:

$$L(y, h(\mathbf{x})) = \max(0, 1 - yh(\mathbf{x}))\,.$$

Ako je $yh(\mathbf{x}) \geqslant 1$, onda će $1 - yh(\mathbf{x}) \leqslant 0$, pa je gubitak 0. Funkcija max pritom osigurava da gubitak ne postane negativan. Ako je $yh(\mathbf{x}) < 1$, onda onda je gubitak veći od nule.

Kao i inače, funkcija gubitka L je funkcija dvije varijable. Kao što smo to radili do sada, radi usporedbe s drugim funkcijama gubitka, preoblikovat ćemo ju u funkciju jedne varijable, $y\mathbf{w}^{\mathrm{T}}\mathbf{x}$. Ovdje je to jednostavno:

$$L(y\mathbf{w}^{\mathrm{T}}\mathbf{x}) = \max(0, 1 - y\mathbf{w}^{\mathrm{T}}\mathbf{x})\,.$$

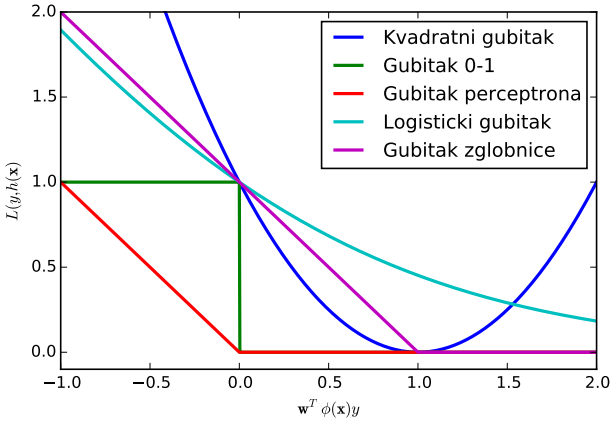
Ovu funkciju gubitka nazivamo **gubitak zglobnice** (engl. *hinge loss*), budući da izgledom podsjeća na zglobnicu. Konačno, uvrštavanjem gubitka zglobnice u ciljnu funkciju dobivamo **funkciju pogreške SVM-a**: 5

$$E(\mathbf{w}|\mathcal{D}) = \sum_{i=1}^{N} \max\left(0, 1 - y^{(i)}\mathbf{w}^{\mathrm{T}}\mathbf{x}\right) + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

gdje $\lambda = 1/C$. Provjerimo još samo vezu između hiperparametara λ i C: veći C daje složeniji model, budući da više kažnjavamo netočne klasifikacije ili ulaske u marginu. Veći C odgovara manjem λ. Manji λ daje manje regularizirani model, tj. složeniji model. Dakle, ovo je u redu. Možemo zaključiti: funkcija pogreške SVM-a je L2-regularizirano očekivanje gubitka zglobnice. (Pritom smo izostavili faktor $\frac{1}{N}$.)

Pogrešku SVM-a možemo minimizirati npr. **stohastičkim gradijentnim spustom**, i to primjenom **podgradijenta**, budući da gubitak zglobnice nije diferencijabilan u svim točkama. Međutim, nećemo dalje o tome.

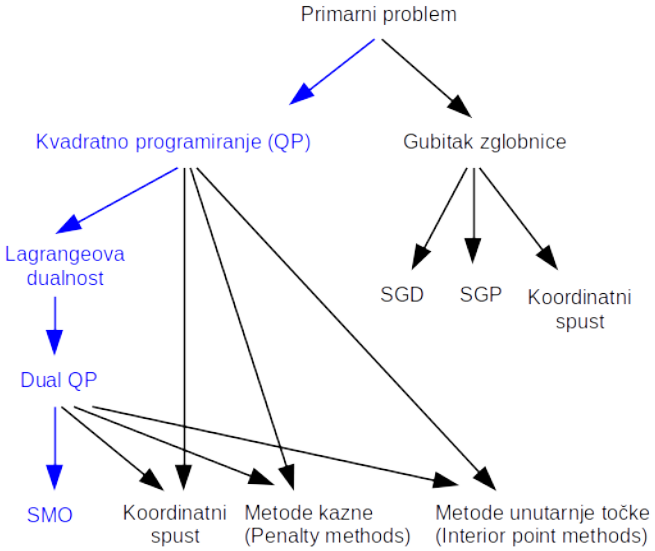
Umjesto toga, usredotočimo se radije na funkciju gubitka zglobnice i usporedimo je s drugim funkcijama gubitka koje smo do sada susreli. Grafikon svih tih funkcija gubitaka izgleda ovako:



Gubitak zglobnice prikazan je ljubičastom bojom. Vidimo da je gubitak zglobnice, isto kao i funkcija logističkog gubitka, aproksimacija funkcije gubika 0-1, koju bismo idealno željeli minimizirati, ali koja nije konveksna i uglavnom je konstantna, pa stoga nije pogodna za optimizaciju. Zapravo, sve funkcije gubitka koje smo dosada promatrali (osim perceptrona, koji je ionako crna ovca u ovoj našoj priči) su tzv. **nadomjesne (surogatne) funkcije gubitka** za gubitak 0-1. Točnije, to su **konveksne gornje međe funkcije gubitka 0-1**. Usporedbimo gubitak zglobnice s gubitkom perceptrona i logističkim gubitkom. Gubitak zglobnice i gubitak perceptrona zapravo su istog oblika, jedina je razlika da je gubitak zglobnice pomaknut udesno za 1. Međutim, efekt tog pomaka je značajan: za razliku od perceptrona, SVM će kažnjavati i one primjere za koje $0 < \mathbf{w}^{\mathrm{T}}\mathbf{x}y < 1$, a to su upravo primjeri koji ulaze u marginu. Od logističkog se gubitka funkcija zglobnice razlikuje po tome što uopće ne kažnjava ispravno klasificirane primjere izvan margine, što rezultira time da su **rješenja rijeđa** (više je težina pritegnuto na nulu nego što je to slučaj kod logističke regresije). Zašto? Zato što primjeri koji su na ispravnoj strani margine ne nanose nikakav gubitak, pa nije potrebno regrutirati težine, makar u malom iznosu, da bi se gubitak dodatno smanjio. 6 7

4 Alternativni algoritmi SVM-a

Optimizacija pogreške SVM-a gradijentnim spustom predstavlja alternativu optimizaciji u dualu pomoću algoritma SMO. Međutim, ova dva postupka ne iscrpljuju sve mogućnosti optimizacije SVM-a. Te mogućnosti grafički možemo prikazati ovako:



Plavom je bojom označen "klasičan" put, kojim smo i mi išli. On uključuje prelazak iz primarnog problema u dualni, primjenu Lagrangeove dualnosti i rješavanje dualnog kvadratnog programa algoritmom SMO. Umjesto algoritma SMO, mogu se koristiti druge optimizacijske metode, poput koordinatnog spusta, metode kazne ili metode unutarnje točke. Alternativno, umjesto prelaska u dualnu formulaciju, istim se metodama može optimirati izravno primarna formulacija problema maksimalne margine. Konačno, umjesto kvadratnog programiranja, može se minimizirati pogreška SVM-a definirana preko gubitka zglobnice, i to opet različitim postupcima, uključivo stohastičkim gradijentnim spustom (SGD), stohastičkom projekcijom (pod)gradijenta (engl. *stochastic (sub)gradient projection*) i koordinatnim spustom.

Sve ove mogućnosti rezultirale su time da postoji niz optimizacijskih algoritama (rješavača) za SVM. Ovdje je dan pregled nekih od njih: [8]

Algorithm	Citation	SVM type	Optimization type	Style	Runtime
SMO	[Platt, 1999]	Kernel	Dual QP	Batch	$\Omega(n^2 d)$
$\text{SVM}^{\text{light}}$	[Joachims, 1999]	Kernel	Dual QP	Batch	$\Omega(n^2 d)$
Core Vector Machine	[Tsang et al., 2005, 2007]	SL Kernel	Dual geometry	Batch	$O(s/\rho^4)$
SVM^{perf}	[Joachims, 2006]	Linear	Dual QP	Batch	$O(ns/\lambda\rho^2)$
NORMA	[Kivinen et al., 2004]	Kernel	Primal SGD	Online(-style)	$\tilde{O}(s/\rho^2)$
SVM-SGD	[Bottou, 2007]	Linear	Primal SGD	Online-style	Unknown
Pegasos	[Shalev-Shwartz et al., 2007]	Kernel	Primal SGD/SGP	Online-style	$\tilde{O}(s/\lambda\rho)$
LibLinear	[Hsieh et al., 2008]	Linear	Dual coordinate descent	Batch	$O(nd \cdot \log(1/\rho))$
SGD-QN	[Bordes and Bottou, 2008]	Linear	Primal 2SGD	Online-style	Unknown
FOLOS	[Duchi and Singer, 2008]	Linear	Primal SGP	Online-style	$\tilde{O}(s/\lambda\rho)$
BMRM	[Smola et al., 2007]	Linear	Dual QP	Batch	$O(d/\lambda\rho)$
OCAS	[Franc and Sonnenburg, 2008]	Linear	Primal QP	Batch	$O(nd)$

Table 1: A comparison of various SVM solvers discussed in this document. "QP" refers to a quadratic programming technique, "SGD" to stochastic (sub)gradient descent, and "SGP" to stochastic (sub)gradient projection. "SL" means the method only works with square-loss. The runtime is for a problem with n training examples and d features, with an average of s non-zero features per example. λ is the SVM regularization parameter, and ρ the optimization tolerance. "Unknown" means there is no known formal bound on the runtime.

Prvonavedeni algoritam je Plattov SMO, koji smo već više puta spomenuli. Taj algoritam može raditi s **jezgrama** (engl. *kernels*) (što znači da granica između klasa može biti nelinearna; više o tome idući put), optimizacija se provodi kvadratnim programiranjem u dualu nad svim primjerima (batch), a složenost treniranja modela je kvadratna u broju primjera. Drugi algoritmi [9]

nude neke druge prednosti, pa su prikladniji u drugim situacijama. Npr., poznati algoritam *LibLinear* prikladan je za slučajeve kada je granica između klasa linearna, jer ima linearnu složenost u broju primjera i broju značajki. Već smo naglasili da je jedna od prednosti dualnih pristupa mogućnost korištenja jezgri, međutim jezgre se zapravo mogu koristiti i u primarnoj formulaciji (npr., algoritam Pegasos).

5 Napomene

SVM dolazi s nizom napomena. Kao i uvijek, vrag je u detaljima. Pogledajmo o čemu se radi.

5.1 SVM za regresiju

Mi smo se bavili SVM-om za klasifikaciju. Međutim, postoji i **stroj potpornih vektora za regresiju** (engl. *support vector regression, SVR*). Taj algoritam koristi istu ideju o margini kao i SVM, samo što ovog puta baš želimo da primjeri budu unutar margine, koja je oko krivulje koju aproksimiramo. Nećemo ići u detalje ovoga. 10

5.2 Hiperparametar C

SVM je vrlo osjetljiv na hiperparametar C, koji definira složenost modela. Treba dobro odabrati vrijednost tog hiperparametra, odnosno treba napraviti **odabir modela**. Kako? Primijetite da, ako optimirate taj hiperparametar na skupu za učenje, dobit ćete prenaučen model (veliki C). Zašto? Zato što, što veći C, to složeniji model, pa to manja empirijska pogreška na skupu za učenje. Zbog toga parametar C, isto kao i parametar λ kod regularizirane logističke regresije, trebamo optimirati na izdvojenom skupu označenih primjera, tj. trebamo raditi **unakrsnu provjeru**.

5.3 Skaliranje

Sjetimo se kako dualni SVM radi predikciju:

$$h(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y^{(i)} \mathbf{x}^{\mathrm{T}} \mathbf{x}^{(i)} + w_0$$

Zamislite što se događa ako radimo klasifikaciju potencijalnih korisnika kredita banke, te imamo značajke, npr. dob osobe i prihod. Što će u skalarnom produktu dominirati? Dominirat će prihod, jer ima veću skalu. To nije dobro. Da bismo to izbjegli, trebamo skalirati značajke. Kako? Tipično radimo **max-min normalizaciju** ili **standardizaciju**. No, budući da ćete to raditi u labosima, ovdje vam ne želim pokvariti zabavu.

5.4 Višeklasje

Kod SVM-a u principu koristimo shemu OVO. Shema OVR se nepreporuča zbog neuravnoteženosti klasa.

5.5 Probabilistički izlaz

SVM nema probabilistički izlaz. Izlaz iz SVM-a je proporcionalan udaljenosti primjera od hiperravnine, i ta vrijednost nije ograničena na interval $(0, 1)$ i ne odgovara vjerojatnosti pripadanja primjera nekoj klasi. Naime, za razliku od poopćenih linearnih modela, SVM nije probabilistički model i nije izveden na temelju pretpostavke da se oznake pokoravaju nekoj teorijskoj distribuciji. Međutim, u praksi često ipak želimo na neki način izračunati vjerojatnost klasifikacije primjera u neku klasu, pa makar i bez teorijske podloge. Rješenje u takvim slučajevima

je **Plattovo skaliranje** (**Plattova kalibracija**). Ideja je da se jednostavno na izlaz SVM-a nalijepi sigmoidna funkcija (odnosno, da treniramo logističku regresiju s izlazom SVM-a $h(\mathbf{x})$ kao jednom jedinom značajkom): [11]

$$P(y = 1|\mathbf{x}) = \sigma(ah(\mathbf{x}) + b)$$

pri čemu se parametri a i b optimiraju na izdvojenom skupu označenih primjera.

Ovo je praktično, ali problematično. Problem je što nemamo nikakvu garanciju da će ovo biti dobro **kalibrirano**, budući da iza ovoga ne stoji nikakav vjerojatnosni model. Pogotovo je to problem ako imamo više klasa, jer neće biti nikakve povezanosti između njih, kao što je to bilo kod multinomijalne logističke regresije, koja koristi funkciju softmax. Stoga je preporuka za Plattovu kalibraciju: koristite, ako baš morate. Ako želite dobiti vjerojatnosti kojima vjerujete, koristite neki probabilistički model (npr., logističku regresiju). [12]

5.6 Nelinearnost

Ovo pitanje smo predugo odgađali, pa je vrijeme da odgovorimo i na njega: kako dobiti nelinearnu granicu? Pa, kao i uvijek: preslikavanjem u prostor značajki, pomoću funkcije $\boldsymbol{\phi}$. Prostor u koji preslikavamo tipično je više dimenzije od ulaznog prostora, čime povećavamo vjerojatnost da će primjeri u tom prostoru značajki biti linearno odvojivi. Ako, dakle, koristimo preslikavanje u prostor značajki, model SVM-a je:

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

Međutim, ovo je tek početak za nešto mnogo moćnije: **jezgrene funkcije**. Jezgrene funkcije sastavni su i ključni dio SVM-a, i o njima ćemo pričati idući put.

Sažetak

- **Meka margina** omogućava kompromis između pogrešne klasifikacije i maksimalne margine, sprječavajući **prenaučenost**
- Problem se opet svodi na **kvadratno programiranje**
- Empirijska pogreška SVM-a je **L2-regularizirano** očekivanje **gubitka zglobnice**
- Postoje razni algoritmi SVM-a, uključivo **SGD** u primarnoj formulaciji
- SVM je osjetljiv na **hiperparametar C** i **skaliranje** značajki
- SVM **nema probabilistički izlaz**, ali može se dobiti, ako baš treba
- Nelinearnost dobivamo preslikavanjem odnosno **jezgrenim funkcijama** (više drugi put)

Bilješke

[1] Formulacija SVM-a s mekom marginom je standardna formulacija SVM-a. Predložili su je 1995. godine Corinna Cortes i Vladimir Vapnik (Cortes and Vapnik, 1995), kao nadogradnju na raniji model SVM-a s jezgrenim funkcijama (Boser et al., 1992).

[2] Ograničenje na dualnu varijablu α je dakle $0 \leqslant \alpha_i \leqslant C$. Ograničenja ovog oblika, gdje postoji gornja i donja ograda, nazivamo **ogradama varijable** (engl. *variable bounds*) ili **ograničenjima kutije** (engl. *box constaints*). Striktno gledano, optimizacijski problem s ogradama nije u standardnoj formi, no lako ga se može pretvoriti u standardnu formu tako da se ograde zamijene dvama odgovarajućim ograničenjima nejednakosti. U našem slučaju, ograde varijable $0 \leqslant \alpha_i \leqslant C$ zamijenili bismo ograničenjima $-\alpha_i \leqslant 0$ i $\alpha_i - C \leqslant 0$. Detaljnije u (Boyd et al., 2004) (poglavlje 4).

[3] Prisjetimo se, kod tvrde margine **potporni vektori** su oni vektori $\mathbf{x}^{(i)}$ iz skupa označenih primjera $\mathcal{D}$ za koje vrijedi $\alpha_i > 0$. Kod meke margine α_i je omeđen odozdo i odozgo, $0 \leqslant \alpha_i \leqslant C$, a potporni

su vektori oni vektori $\mathbf{x}^{(i)}$ iz $\mathcal{D}$ za koje vrijedi $0 < \alpha \leqslant C$. Pokažimo da su potporni vektori za koje vrijedi $\alpha_i < C$ na margini, dok vektori za koje $\alpha_i = C$ mogu ležati i unutar margine. Prisjetimo se da vrijede $\alpha_i = C - \beta_i$ i uvjet komplementarne labavosti $\beta_i \xi_i = 0$. Ako $\alpha_i < C$, onda iz $\alpha_i = C - \beta_i$ slijedi $\beta_i > 0$, te iz toga i $\beta_i \xi_i = 0$ slijedi $\xi_i = 0$, što znači da je primjer na margini. Ako $\alpha_i = C$, onda iz $\alpha_i = C - \beta_i$ sijedi $\beta_i = 0$, te iz toga i $\beta_i \xi_i = 0$ slijedi $\xi_i = 0$ ili $\xi_i > 0$, odnosno primjer je na margini ili unutar nje.

4 Kako na temelju dualnih parametara i potpornih vektora izračunati težinu w_0 objasnili smo u temi 8 (bilješka 21), gdje smo pričali o tvrdoj margini. Međutim, kod meke margine treba paziti da se izračun težine w_0 radi isključivo na temelju potpornih vektora koji su na margini (dakle onih za koje $\alpha_i < C$), a ne i na temelju onih koji su unutar margine (oni za koje $\alpha_i = C$). Naime, izračun se temelji na jednakosti $y^{(i)} h(\mathbf{x}^{(i)}) = 1$, koja vrijedi samo za potporne vektore $\mathbf{x}^{(i)}$ koji su točno na margini.

5 **Zglobnica** (baglama, pant, šarka) je željezni okov pomoću kojega su za okvir pričvršćena vrata, prozorska krila, poklopac na kutiji i sl. http://hjp.znanje.hr/index.php?show=search_by_id&id=d1hnURI%3D

6 Ova sličnost između **gubitka perceptrona** i **gubitka zglobnice** upućuje na to da su i algoritam perceptrona i algoritma SVM-a donekle slični. To je točno, međutim, očigledno postoje i različitosti. Usporedimo ih po trima komponentama. Model perceptrona i model SVM-a su zapravo identični: oba modela su skalarni produkt vektora težina i vektora značajki. Perceptron ima aktivacijsku funkciju praga, međutim ona zapravo ne figurira kod izračuna gubitka, a ista bi se funkcija praga mogla dodati i modelu SVM-a za potrebe krajnje klasifikacijske predikcije. Funkcije gubitka razlikuju se za pomak od 1 duž x-osi, zbog čega algoritam SVM-a kažnjava i one primjere koji su ispravno klasificirani, ali su unutar margine. Osim po funkciji gubitka, funkcije pogreške razlikuju se još i po tome što pogreška SVM-a uključuje i regularizacijski faktor, kojim se promovira širina margine, dok perceptron toga nema. Konačno, optimizacijski se postupci razlikuju zato jer perceptron uči na pojedinačnim primjerima (online), dok SVM može učiti na svim primjerima (kod optimizacije kvadratnim programiranjem) ili, ako optimiramo gradijentnim spustom, na minigrupama primjera ili na pojedinačnim primjerima (stohastički gradijentni spust). Učenje SVM-a stohastičkim gradijentnim spustom najviše nalikuje učenju perceptrona, premda postoji i razlika u pogledu kriterija zaustavljanja (učenje perceptrona zaustavlja se tek kada su svi primjeri zaredom točno klasificirani, dok se stohastički gradijenti spust zaustavlja kada je gradijent jednak nuli ili dovoljno blizu tome). Detaljniju usporedbu algoritama perceptrona i SVM-a možete pronaći u (Collobert and Bengio, 2004).

7 Vrlo je zanimljivo svojstvo SVM-a da rezultira rijetkim modelima, premda zapravo koristi L2-regularizaciju a ne L1-regularizaciju. Rijetkost je ovdje posljedica toga što funkcija gubitka ne kažnjava ispravno klasificirane primjere izvan margine, a ne toga što provodimo L1-regularizaciju. Ovo je netipično u usporedbi s poopćenim linearnim modelima, i mnogim drugim probabilističkim modelima strojnog učenja. Zbog toga Murphy (2012) konstatira da je SVM iz probabilističke perspektive "vrlo neprirodan", budući da (1) rijetkost ostvaruje pomoću funkcije gubitka umjesto pomoću regularizacijskog izraza (odnosno apriorne gustoće vjerojatnosti nad težinama), (2) uvodi jezgre pomoću jezgrenog trika, umjesto da su one eksplicitno ugrađene u model i (3) ne rezultira vjerojatnosnim izlazom, što dovodi do toga da vjerojatnosti nisu dobro kalibrirane, pogotovo kod višeklasne klasifikacije (Murphy, 2012) (poglavlje 14.5).

8 Preuzeto iz (Menon, 2009).

9 Algoritam SMO implementiran je u enormno popularnoj bilbioteci **LibSVM** (Chang and Lin, 2011), koja se pak koristi u mnogim drugim alatima, uključivo alatu **sklearn**, koji koristimo za labose.

10 Zainteresirane upućujem na (Drucker et al., 1997) i (Smola and Schölkopf, 2004).

11 Plattovo skaliranje osmislio je John Platt 1999. godine. Detalje možete naći u (Platt et al., 1999). Platt je godinu dana prije toga osmislio algoritam **slijedne minimalne optimizacije (SMO)**.

12 Intuitivno, probabilistički model je dobro **kalibriran** ako vjerojatnosti koje daje doista odgovaraju udjelima u skupu primjera. Npr., ako za neki podskup primjera vrijedi $h(\mathbf{x}) \sim 0.7$, onda za dobro

kalibrirani model očekujemo da će u tom podskupu primjera doista njih $\sim 70\%$ biti pozitivni.

Literatura

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

R. Collobert and S. Bengio. Links between perceptrons, mlps and svms. In *Proceedings of the twenty-first international conference on Machine learning*, page 23, 2004.

C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik. Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161, 1997.

A. K. Menon. Large-scale support vector machines: algorithms and theory. *Research Exam, University of California, San Diego*, 117, 2009.

K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

J. Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.