5. Linearni diskriminativni modeli

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

Jan Šnajder, predavanja, v2.3

Prošli tjedan bavili smo se regresijom te smo razmatrali **linearan model regresije**, koji – uz odgovarajući odabir nelinearne funkcije preslikavanja u prostor značajki – zapravo postaje nelinearan model. Vidjeli smo da, neovisno o tome je li model linearan ili nelinearan, postupak najmanjih kvadrata daje rješenje u zatvorenoj formi, koje se svodi na izračun **pseudoinverza**. Također, pričali smo o **regularizaciji**, kojom se sprječava prenaučenost, i pokazali smo da i za L2-regularizirani linearan model regresije postoji rješenje u zatvorenoj formi.

Danas se više nećemo baviti regresijom, nego **klasifikacijom**. Zapravo, u nastavku predmeta uglavnom ćemo se baviti klasifikacijom. Prisjetimo se: klasifikacija je postupak predviđanja diskretnih oznaka pojedinačnih primjera, npr., klasifikacija elektroničke pošte u spam i ne-spam, ili klasifikacija poruka na Twitteru u pozitivne, negativne i neutralne s obzirom na sentiment.

Naravno, postoji više pristupa klasifikaciji. Mi ćemo se prvih nekoliko tjedana fokusirati na **linearne modele**. Konkretnije, bavit ćemo se modelima koji pripadaju grupi **linearnih diskriminativnih modela**. Kao što ćete vidjeti, to nije jedan model, već čitava familija, od kojih su neki vrlo učinkoviti (npr., logistička regresija i stroj potpornih vektora).

Naš današnji cilj jest objasniti osnovnu ideju linearnih diskriminativnih modela, a onda ćemo u naredna dva tjedna ovu temu detaljnije razraditi.

1 Linearni diskriminativni modeli

Krenimo od toga da naprije razjasnimo naslov današnjeg predavanja: što su **linearni diskriminativni** modeli? Usredotočimo se najprije na "linearan". Da je model linearan znači da modelira linearnu granicu između klasa: npr., pravac (ako je ulazni prostor dvodimenzijski) ili ravnina (ako je ulazni prostor trodimenzijski) ili hiperravnina (za prostore s više od tri dimenzije).

Kako ćemo definirati linearan model? Pa, krenimo od modela regresije:

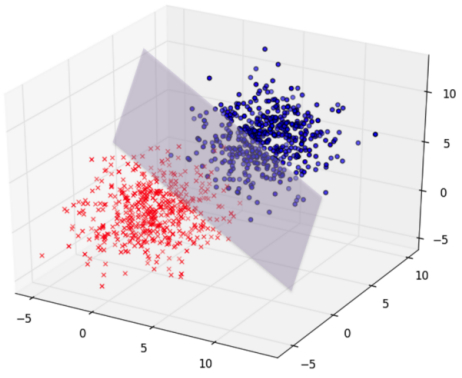
$$h(\mathbf{x}; \mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$$

gdje opet imamo onu "dummy" značajku $x_0 = 1$, kako bismo mogli koristiti ovaj jednostavniji vektorski zapis. Ovako definirana hipoteza (tj. model s fiksiranim težinama $\mathbf{w}$) $h(\mathbf{x}; \mathbf{w})$ daje neku vrijednost iz $\mathbb{R}$ za svaki ulazni vektor $\mathbf{x}$ iz $\mathbb{R}^n$, tj. $h(\mathbf{x}; \mathbf{w}) : \mathbb{R}^n \to \mathbb{R}$.

Kako možemo na ovaj model gledati kao na **binarnan klasifikacijski model**? Tako da razmišljamo kako ovaj model zapravo dijeli ulazni prostor u dva **poluprostora**: jedan čine sve točke u $\mathbb{R}^n$ za koje $h(\mathbf{x}) \geqslant 0$, a drugi sve točke u $\mathbb{R}^n$ za koje $h(\mathbf{x}) < 0$. Binarni klasifikator onda bismo mogli definirati ovako:

$$h(\mathbf{x}; \mathbf{w}) = \mathbf{1}\{\mathbf{w}^{\mathrm{T}}\mathbf{x} \geqslant 0\}$$

gdje je $\mathbf{1}\{\cdot\} : \{\bot, \top\} \to \{0, 1\}$ indikacijska funkcija koju smo već bili definirali. Granica između klasa je točno tamo gdje $h(\mathbf{x}) = 0$ (i te točke koje leže na granici također trebamo svrstati u jedan od dva poluprostora, svejedno je u koji). Za dvodimenzijski ulazni prostor ta granica je pravac, za trodimenzijski to je ravnina, a općenito to je $(n-1)$-dimenzijska hiperravnina ugrađena u n-dimenzijski ulazni prostor ("dummy" značajku x_0 ne računamo u dimenzije ulaznog prostora). Npr., za $n = 3$ to bi izgledalo ovako:

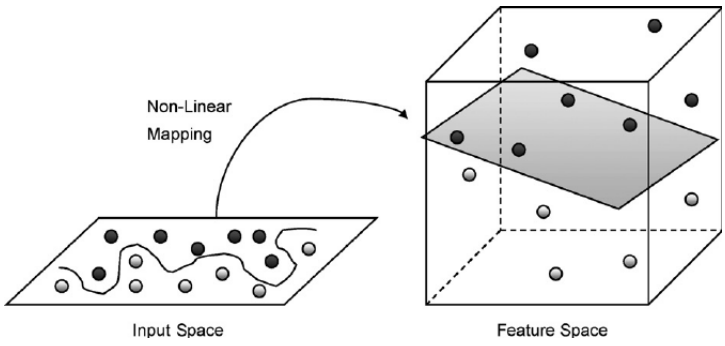


Općenito, dakle, granica je **potprostor** (engl. *subspace*) dimenzije $n-1$ (uvijek za jedan manje od dimenzije ulaznog prostora) koji ulazni prostor dijeli na dva **poluprostora** (engl. *half-spaces*). Tu granicu zovemo **diskriminantna funkcija** (engl. *discriminative function*) ili **granica odluke** ili **decizijska granica** (engl. *decision boundary*), ili jednostavno kažemo **granica između klasa**.

No, što je s nelinearnošću? Znamo da su linearne granice između klasa u stvarnim problemima zapravo rijetkost; većina problema u strojnom učenju je takva da je granica između klasa nelinearna. Kako možemo dobiti nelinearnu granicu? Pa, ako želimo nelinearnu granicu, možemo iskoristiti trik koji smo naučili prošli put: možemo upotrijebiti **nelinarnu funkciju preslikavanja** kako bismo naše podatke iz ulaznog prostora preslikali u neki drugi prostor (prostor značajki), i tamo ih onda napali linearnim modelom. Dakle, uz funkciju preslikavanja $\boldsymbol{\phi} : \mathbb{R}^n \to \mathbb{R}^{m+1}$ dobivamo ovakav model:

$$h(\mathbf{x}; \mathbf{w}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$$

Ovaj model još uvijek ima linearnu granicu u prostoru značajki, no – ako je preslikavanje $\boldsymbol{\phi}$ nelinearno – granica u ulaznom prostor bit će nelinearna. Tehnički, međutim, to je i dalje **linearan model**, jer je linearan u parametrima. Prisjetimo se primjera koji smo pokazali prošli put:



Ovdje preslikavamo iz ulaznog prostora dimenzije $n = 2$ u prostor značajki dimenzije $m = 3$. Ono što nije bilo linearno odvojivo u ulaznom prostoru sada je linearno odvojivo u prostoru značajki. Granica između klasa koja je linearna u prostoru značajki, nelinearna je u ulaznom prostoru.

▸ **PRIMJER**

Jednostavan primjer kako preslikavanje u prostor značajki više dimenzije od ulaznog prostora može problem koji je nelinaran u ulaznom prostoru učiniti linearnim u prostoru značajka jest poznati **XOR-problem** (problem "isključivog ili"). Riječ je o binarnom klasifikacijskom problemu definiranom u

dvodimenzijskome ulaznom prostoru $\mathcal{X} = \{-1, +1\}^2$ sa sljedećim skupom označenih primjera:

$$\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_i = \{((-1,-1),0),((+1,+1),0),((-1,+1),1),((+1,-1),1)\}$$

Ovaj problem ne možemo riješiti linearnim modelom. Drugim riječima, ako je $\mathcal{H}$ linearan model (skup pravaca), onda $\neg\exists h \in H.E(h|\mathcal{D}) = 0$. Međutim, ako primjere iz $\mathcal{D}$ iz dvodimenzijskoga ulaznog prostora preslikamo u trodimenzijski prostor značajki pomoću sljedeće funkcije preslikavanja:

$$\boldsymbol{\phi}(\mathbf{x}) = (1, x_1, x_2, x_1 x_2)$$

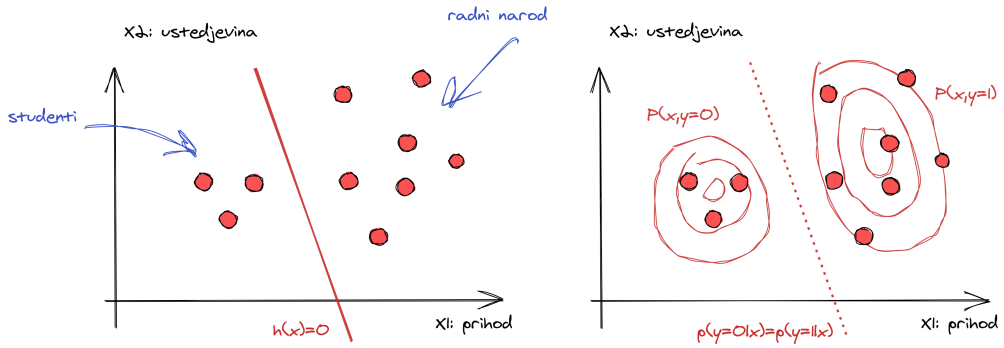
onda su primjeri iz $\mathcal{D}$ u prostoru značajki odvojivi (postoji ravnina takva da su primjeri iz klase $y = 0$ iznad te ravnine, a primjeri iz klase $y = 1$ ispod nje).

Toliko o prvom dijelu naziva: "linearan" model. Pogledajmo sada što znači da je model "diskriminativan". **Diskriminativni modeli** su modeli koji modeliraju granicu koja diskriminira (razlikuje) između klasa, i to tako da parametri modela opisuju tu granicu i ništa više osim te granice. Drugim riječima, diskriminativan model imat će onoliko parametara koliko je potrebno da bi se definirala ta granica.

Ovo se možda čini očitim, no postaje zanimljivije ako razmotrimo što je alternativa. Diskriminativni modeli suprotstavljeni su jednoj drugoj velikoj familiji modela, koji se nazivaju **generativni modeli**. Generativni modeli modeliraju više od granice između klasa, i samu granicu zapravo modeliraju indirektno. Također ipično imaju mnogo više parametara nego diskriminativni modeli. Pomoći će da pogledamo jedan primjer.

▸ **PRIMJER**

Radimo klasifikaciju kreditno sposobnih klijenata banke: banka treba odlučiti kome će dati kredit. Kao što smo to već napravili u uvodnome predavanju, pojednostavimo problem i pretpostavimo da banke to rade na temelju svega dvije značajke: prihod i ušteđevina. Onda je ulazni prostor dvodimenzijski (x1 → prihod, x2 → ušteđevina), a svaki klijent banke je jedan dvodimenzijski vektor u tom prostoru. Neka to izgleda ovako:



Na lijevoj slici prikazana je (linearna) granica između klasa kakvu nalazi diskriminativan model. Granica će biti opisana parametrima tog modela. Na desnoj slici prikazan je generativni model. Generativni model modelirat će vjerojatnosne gustoće svake klase, koje su na slici prikazane kao izokonture funkcije gustoće vjerojatnosti. Granica između klasa sada se može izračunati indirektno na temelju tih distribucija, kao sve one točke za koje je vrijednost funkcije gustoće vjerojatnosti jednaka za objeklase (ta je granica prikazana crtkanom linijom i ona je opet linearna). To je više informacija nego što nudi diskriminativni model, ali te nam informacije možda nisu potrebne.

O generativnim modelima pričat ćemo detaljnije u drugom dijelu predmeta. Danas se fokusiramo na linearne diskriminativne modele. Sada kada smo objasnili pojmove "linearan" i

"diskriminativan", pogledajmo malo detaljnije kako parametri linearnog diskriminativnog modela definiraju granicu u ulaznom prostoru, koji je zapravo euklidski vektorski prostor, tj. pogledajmo "geometriju" linearnog modela.

2 Geometrija linearnog modela

Već smo rekli da su kod linearnih diskriminativnih modela granica između klasa **hiperravnine**. Radi jednostavnosti, fokusirajmo se najprije na slučaj dviju klasa, $K = 2$, pa ćemo poslije proširiti na više klasa. Također, bez smanjenja općenitosti, fokusirajmo se na najjednostavniji model gdje nemamo preslikavanje u prostor značajki, dakle:

$$h(\mathbf{x}; \mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$$

Zapravo, bit će nam lakše ako težinu w_0 izdvojimo iz vektora $\mathbf{w}$ i tretiramo ju zasebno, pa imamo:

$$h(\mathbf{x}; w_0, \mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$$

Granica između klasa dana je jednadžbom $h(\mathbf{x}) = 0$, što definira $(n-1)$- dimenzijsku hiperravninu unutar n-dimenzijskog ulaznog prostora. Opet bez smanjenja općenitosti, u nastavku ćemo si pojednostaviti život i skicirati slučaj dvodimenzijskog ulaznog prostora, $n = 2$, u kojemu je granica između klasa pravac. Dakle, model je onda: 1

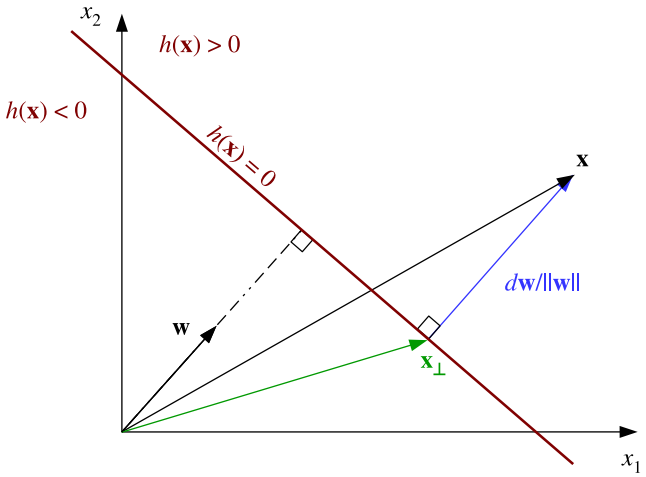
$$h(\mathbf{x}; w_0, \mathbf{w}) = w_1 x_1 + w_2 x_2 + w_0$$

a granica između klasa je:

$$w_1 x_1 + w_2 x_2 + w_0 = 0$$

što prepoznajemo kao implicitnu jednadžbu pravca. Međutim, u nastavku ćemo i dalje govoriti "hiperravnina", jer će naši zaključci vrijediti općenito.

Za daljnja razmatranja poslužit ćemo se sljedećom slikom:



Slika prikazuje pravac (prikazan crveno) kao granicu u dvodimenzijskom ulaznom prostoru $\mathbb{R}^2$ sa značajkama x_1 i x_2. Taj pravac odgovara jednadžbi $h(\mathbf{x}; w_0, \mathbf{w}) = 0$, tj. granicu čine točke (x_1, x_2) za koje hipoteza h daje vrijednost nula. Pravac dvodimenzijski prostor dijeli na dva poluprostora: jedan poluprostor su sve točke (vektori) $\mathbf{x}$ za koje $h(\mathbf{x}; w_0, \mathbf{w}) > 0$, a drugi sve točke (vektori) $\mathbf{x}$ za koje $h(\mathbf{x}; w_0, \mathbf{w}) < 0$. Normala pravca je vektor $\mathbf{w}$, koji je okomit na pravac, i koji pokazuje u smjeru poluprostora za koji $h(\mathbf{x}; w_0, \mathbf{w}) > 0$ (što bi, ako binarni klasifikator definiramo kao $h(\mathbf{x}; w_0, \mathbf{w}) = \mathbf{1}\{\mathbf{w}^{\mathrm{T}}\mathbf{x} \geqslant 0\}$, bio poluprostor pozitivnih primjera). Ovdje je vektor normale $\mathbf{w}$ prikazan kao vektor s početnom točkom u ishodištu koordinatnog sustava (tj. kao

"radijvektor"), no početna točka tog vektora je, naravno, proizvoljna. Na slici je još označen vektor nekog primjera $\mathbf{x}$, koji se nalazi negdje u poluprostoru pozitivnih primjera. Prisjetite se da je svaki primjer $\mathbf{x}$ zapravo vektor, pa ga možemo prikazati kao radijvektor (vektor s početnom točkom u ishodištu koordinatnog sustava). Na slici je nadalje prikazan rastav vektora $\mathbf{x}$ na zbroj dvaju vektora: $d\frac{\mathbf{w}}{\|\mathbf{w}\|}$ (prikazan plavo) i $\mathbf{x}_\perp$ (prikazan zeleno); o tome više u nastavku.

Glavna stvar koja nas ovdje zanima jest s koje se strane hiperravnine nalazi neki primjer te koliko je od nje udaljen. Jasno je da nas zanima s koje se strane nalazi primjer, jer to određuje klasifikaciju primjera. No, zašto nam je zanimljiva udaljenost? Udaljenost nas zanima zato što je ona indikativna za **pouzdanost** klasifikacije primjera: koliko možemo biti sigurni da je primjer pozitivan ili negativan. Npr., kada se primjer nalazi daleko od hiperavnine u pozitivnoj strani, onda smo sigurniji da je primjer pozitivan, nego kada je vrlo blizu toj hiperravnini.

S ciljem, dakle, da utvrdimo stranu i udaljenost na kojoj se nalazi primjer, razmotrit ćemo dvije stvari: (1) što zapravo predstavlja vektor težina $\mathbf{w}$ (bez w_0) i (2) kako izračunati udaljenost primjera od hiperravnine.

Prvo, što je zapravo vektor težina $\mathbf{w}$? Razmotrimo dva primjera, odnosno dvije točke $\mathbf{x}^{(1)}$ i $\mathbf{x}^{(2)}$, koje leže na hiperravnini. Za njih vrijedi:

$$h(\mathbf{x}^{(1)}) = h(\mathbf{x}^{(2)}) = 0$$

Ako umjesto $h(\mathbf{x})$ uvrstimo definiciju modela, onda imamo:

$$\begin{aligned} \mathbf{w}^{\mathrm{T}}\mathbf{x}^{(1)} + w_0 &= \mathbf{w}^{\mathrm{T}}\mathbf{x}^{(2)} + w_0 \\ \mathbf{w}^{\mathrm{T}}(\mathbf{x}^{(1)} - \mathbf{x}^{(2)}) + w_0 - w_0 &= 0 \\ \mathbf{w}^{\mathrm{T}}(\mathbf{x}^{(1)} - \mathbf{x}^{(2)}) &= 0 \end{aligned}$$

Primijetimo da je $(\mathbf{x}^{(1)} - \mathbf{x}^{(2)})$ razlika dvaju vektora, što je opet vektor, i da taj vektor leži upravo na hiperravnini. Budući da smo upravo izveli $\mathbf{w}^{\mathrm{T}}(\mathbf{x}^{(1)} - \mathbf{x}^{(2)}) = 0$, a znajući da skalarni produkt dvaju vektora iščezava kada su ti vektori međusobno **okomiti**, to znači da je vektor $\mathbf{w}$ okomit na sve vektore koji leže na hiperravnini, pa zaključujemo da je vektor $\mathbf{w}$ **normala hiperravnine**. Konkretno, na našoj prethodnoj slici $\mathbf{w}$ je normala pravca, budući da imamo dvodimenzijski ulazni prostor. [2]

Kao što smo rekli, druga stvar koja nas zanima jest koliko je **udaljenost** nekog primjera od hiperravnine. Promotrimo dakle neku točku $\mathbf{x}$ koja je udaljena od hiperravnine. Ta točka ima svoju projekciju na hiperravninu – označimo tu točku sa $\mathbf{x}_\perp$. Sada vektor $\mathbf{x}$ možemo rastaviti na zbroj dvaju vektora:

$$\mathbf{x} = \mathbf{x}_\perp + d\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

Ovaj drugi vektor je jedinični vektor normale, $\mathbf{w}/\|\mathbf{w}\|$, pomnožen sa d. Dakle, d je zapravo udaljenost točke $\mathbf{x}$ od hiperravnine, i to je ono što nas zanima koliko iznosi.

A sad malo algebarske magije. Pomnožimo obje strane jednadžbe sa $\mathbf{w}^T$ i dodajmo s obje strane w_0:

$$\underbrace{\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0}_{h(\mathbf{x})} = \underbrace{\mathbf{w}^{\mathrm{T}}\mathbf{x}_\perp + w_0}_{=h(\mathbf{x}_\perp)=0} + d\frac{\mathbf{w}^{\mathrm{T}}\mathbf{w}}{\|\mathbf{w}\|}$$

$$h(\mathbf{x}) = d\frac{\mathbf{w}^{\mathrm{T}}\mathbf{w}}{\|\mathbf{w}\|} = d\|\mathbf{w}\|$$

gdje smo iskoristili $\mathbf{w}^{\mathrm{T}}\mathbf{w} = \|\mathbf{w}\|^2$. Iz ovoga dobivamo da je udaljenost d primjera $\mathbf{x}$ od hiperravnine s vektorom normale $\mathbf{w}$ jednaka:

$$d = \frac{h(\mathbf{x})}{\|\mathbf{w}\|}$$

Primijetite da ova udaljenost može biti pozitivna ili negativna, pa kažemo da je to **predznačena udaljenost** (engl. *signed distance*). Konkretno, imamo:

- $d > 0 \Rightarrow \mathbf{x}$ je na strani hiperravnine u smjeru normale $\mathbf{w}$
- $d < 0 \Rightarrow \mathbf{x}$ je na suprotnoj strani hiperravnine
- $d = 0 \Rightarrow \mathbf{x}$ je točno na hiperravnini

Sva ova naša razmatranja vrijede i za prostore dimenzije veće od dva, $n > 2$. Dakle, vektor $\mathbf{w}$ (bez w_0) je normala $(n-1)$-dimenzijske ravnine u prostoru $\mathbb{R}^n$, i on pokazuje u smjeru poluprostora za koji $h(\mathbf{x}) > 0$. Predznačena udaljenost primjera $\mathbf{w}$ od hiperravnine $h(\mathbf{x}) = 0$ jednaka je $h(\mathbf{x})/\|\mathbf{w}\|$. Također se može pokazati, no to ćemo preskočiti, da je udaljenost hiperravnine od ishodišta jednaka $-w_0/\|\mathbf{w}\|$. 3

Uočimo na ovom mjestu još i da se jedna te ista hiperravnina može definirati s različitim težinama $\mathbf{w}$: točnije, postoji beskonačno mnogo vektora težina $(w_0, \mathbf{w})$ koji definiraju identičnu hiperravninu. Naime, ako pomnožimo vektor $(w_0, \mathbf{w})$ s nekim faktorom α, onda će vektor normale $\mathbf{w}$ biti α-puta veći, ali se neće promijeniti njegov smjer. Također, budući da će i težina w_0 biti α-puta veća, neće se promijeniti niti udaljenost hiperravnine od ishodišta niti udaljenost primjera od hiperravine, budući da se obje te udaljenosti normiraju sa $\|\mathbf{w}\|$. Ova nam spoznaja sada nije od posebne koristi, ali će to biti za dva tjedna, kada ćemo pričati o modelu SVM. 4

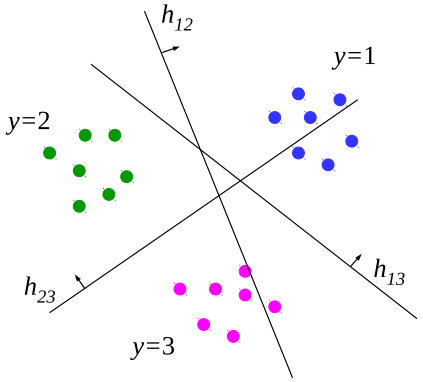
3 Višeklasna klasifikacija

Do sada smo se bavili binarnom klasifikacijom, tj. klasifikacijom u dvije klase. No, nisu svi klasifikacijski problemi s kojima ćemo se susretati binarni. Ponekad nam treba klasifikacija u više od dvije klase. Npr., klasifikacija poruka s Twittera u pozitivne, negativne ili neutralne. Ili klasifikacija novinskih tekstova u rubrike: sport, kultura, crna kronika, politika, itd.

Problem s višeklasnom klasifikacijom jest što su mnogi linearni diskriminativni modeli inherentno **binarni klasifikatori**. Onda se postavlja pitanje kako možemo iskoristiti takve modele kada imamo $K > 2$ klasa? Voljeli bismo ne mijenjati model, nego radije promijeniti problem. Prisjetite se da smo sličan trik već bili primijenili, kada smo primjere preslikavali u prostor značajki, kako bismo mogli primijeniti linearan model ali imati nelinearnu granicu. Sada ćemo napraviti nešto slično: rastavit ćemo višeklasni klasifikacijski problem na skup binarnih klasifikacijskih problema, i onda jednostavno više puta primijeniti binarni klasifikator. Konkretno, imamo dvije mogućnosti kako to napraviti: shema jedan-naspram-jedan i shema jedan-naspram-ostali. Pogledajmo ih redom.

Shema **jedan-naspram-jedan** (engl. *one-vs-one,* **OVO**) višeklasni problem svodi na $\binom{K}{2}$ nezavisnih binarnih klasifikacijskih problema, po jedan za svaki par klasa. Treniramo model h_{ij} tako da nauči razdvojiti primjere iz klase $y = i$ od primjera iz klase $y = j$, zanemarujući pritom primjere iz svih drugih klasa.

Na primjer, za $K = 3$ klase u dvodimenzijskome ulaznom prostoru to bi izgledalo ovako:



Budući da imamo tri klase, u shemi OVO treba nam $\binom{3}{2} = 3$ binarnih klasifikatora. Hipoteze koje odgovaraju tim trima klasifikatorima označene su na slici kao h_{12}, h_{23}, i h_{13}, gdje h_{ij} razdvaja primjere klase s oznakom $y = i$ od primjera klase s oznakom $y = j$. Na slici strelice pokazuju u smjeru pozitivne orijentacije hiperravnina, tj. u smjeru gdje h_{ij} kao pozitivne klasificira primjere iz klase $y = i$. Na slici vidimo da svaka od triju hipoteza uvijek razdvaja samo dvije klase, dok treću klasu potpuno ignorira. Tako, na primjer, ravnina koja odgovara hipotezi $h_{23}(\mathbf{x}) = 0$ razdvaja klase $y = 2$ i $y = 3$, s orijentacijom prema primjerima iz klase $y = 2$, dok međutim prolazi kroz primere iz klase $y = 1$, koje zanemaruje. To ostvarujemo tako da svaki od ovih binarnih klasifikatora h_{ij} treniramo na podskupu označenih primjera $\mathcal{D}$ koji sadrži samo one primjere koji su iz klasa $y = i$ i $y = j$.

Odluku o tome u koju klasu svrstati primjer sada dobivamo većinskim glasanjem. Drugim riječima, **višeklasni model u shemi OVO** definiramo ovako:

$$h(\mathbf{x}) = \operatorname*{argmax}_i \sum_{i \neq j} \operatorname{sgn}\big(h_{ij}(\mathbf{x})\big)$$

pri čemu

$$h_{ij}(\mathbf{x}) = -h_{ji}(\mathbf{x})$$

Uvjerite se da ovakva definicija doista odgovara većinskoj odluci binarnih klasifikatora. Pogledajmo i jedan primjer.

▸ **PRIMJER**

Na prethodnoj slici, razmotrimo klasifikaciju nekog primjera $\mathbf{x}$ iz klase $y = 2$. Za taj primjer imamo:

$$\begin{aligned} h_{12}(\mathbf{x}) = -1 \quad &\Rightarrow \quad h_{21}(\mathbf{x}) = 1 \\ h_{13}(\mathbf{x}) = -1 \quad &\Rightarrow \quad h_{31}(\mathbf{x}) = 1 \\ h_{23}(\mathbf{x}) = 1 \quad &\Rightarrow \quad h_{32}(\mathbf{x}) = -1 \end{aligned}$$

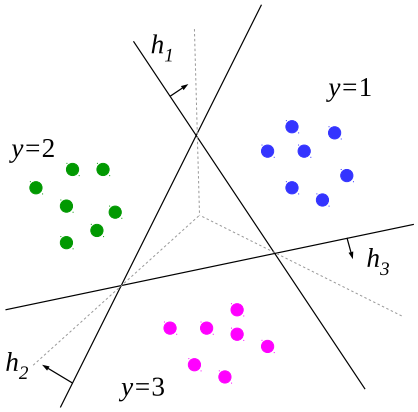
Glasovi hipoteza za pojedinačne klase su:

$$\begin{aligned} (i = 1): \quad & \operatorname{sgn}(h_{12}(\mathbf{x})) + \operatorname{sgn}(h_{13}(\mathbf{x})) = (-1) + (-1) = -2 \\ (i = 2): \quad & \operatorname{sgn}(h_{21}(\mathbf{x})) + \operatorname{sgn}(h_{23}(\mathbf{x})) = 1 + 1 = 2 \\ (i = 3): \quad & \operatorname{sgn}(h_{31}(\mathbf{x})) + \operatorname{sgn}(h_{32}(\mathbf{x})) = 1 + -1 = 0 \end{aligned}$$

pa je većinski izglasana klasa ona s oznakom $y = 2$, što je u skladu sa stvarnom oznakom primjera $\mathbf{x}$.

Shema OVO je jednostavna, no primijetimo da kod nje mogu postojati situacije kada odluka o klasifikaciji nije jednoznačna. To će biti slučaj sa svim onim primjerima koji padnu u dio ulaznog prostora (odnosno prostora značajki, ako koristimo preslikavanje) gdje je broj glasova za pobjedničku klasu izjednačen. Na našoj slici to bi bilo trokutasto područje u sredini ulaznog prostora. Međutim, u praksi se razmjerno rijetko događa da baš imamo popuno izjednačenje. A ako se i dogodi, situaciju možemo razriješiti tako da u obzir uzmemo pouzdanosti klasifikacije (na temelju predznačene udaljenosti primjera od hiperravnine, kako smo objasnili gore).

Druga shema za višeklasnu klasifikaciju pomoću dekompozicije na binarne klasifikacije jest shema **jedan-naspram-ostali** (engl. *one-vs-rest,* **OVR**) (koja se ponekad neispravno naziva *one-vs-all*). Kod te sheme imamo K nezavisnih binarnih klasifikatora, po jedan za svaku klasu. Svaki klasifikator h_i trenira se tako da razdjeluje primjere klase $y = i$ od primjera svih ostalih klasa $y \neq i$, tj. klasifikator je naučen da raspoznaje radi li se o primjeru iz klase $y = i$ ili ne. Prethodni primjer u shemi OVR izgledao bi ovako:



Budući da imamo $K = 3$ klasa, u shemi OVR imat ćemo 3 binarna klasifikatora. Hipoteze koje odgovaraju tim trima klasifikatorima su h_1, h_2 i h_3, gdje hipoteza h_i razdvaja primjere klase s oznakom $y = i$ od primjera svih drugih klasa.

Višeklasni model u shemi OVR odluku donosi na temelju pouzdanosti pojedinačnih klasifikatora tako da primjer svrstava u onu klasu za koju je klasifikacija najpouzdanija. To možemo vrlo jednostavno definirati ovako:

$$h(\mathbf{x}) = \operatorname*{argmax}_j \; h_j(\mathbf{x})$$

dakle primjer se klasificira u klasu onog binarnog klasifikatora koji je najpouzdaniji u svoju pozitivnu klasifikaciju. Ovako definiran model zapravo implicitno definira granicu između dviju susjednih klasa $y = i$ i $y = j$ na mjestu gdje vrijedi $h_i(\mathbf{x}) = h_j(\mathbf{x})$. Na našoj slici takve granice odgovoraju simetralama kuteva (koje su prikazane kao iscrtkane linije).

Primijetite da je kod OVR sheme, za razliku od OVO sheme, mnogo manje područja gdje klasifikacijska odluka nije jednoznačna (može se, doduše, dogoditi da primjer sleti baš na pravac gdje je izlaz dvaju ili više najpouzdanijih klasifikatora izjednačen, ali to onda samo znači da smo urečeni i da bi bilo bolje da se klonimo strojnog učenja i svih koji ga prakticiraju).

Usporedimo sada ove dvije sheme. Očita prednost OVR nad OVO jest da imamo manje modela koje trebamo trenirati: K naspram $\binom{K}{2}$, što je linearna naspram kvadratna ovisnost o broju klasa. U ovom našem primjeru imali smo samo tri klase, pa razlika nije došla do izražaja. No, razlika je značajna ako imamo mnogo klasa i vrlo mnogo puno primjera, jer će onda treniranje mnogo modela vrlo dugo trajati, a i trajanje predikcije bi moglo biti nezanemarivo. Na primjer, ako imamo deset klasa (npr., za problem raspoznavanje znamenki), u shemi OVO trebat će nam 45 klasifikatora, a u shemi OVR samo njih deset.

S druge strane, problem sa shemom OVR jest taj da lako rezultira **neuravnoteženim** brojem primjera između parova klasa za koje treniramo model. Zašto? Zato što ćemo u pravilu za svaki model h_j imati puno više negativnih primjera (svi oni koji ne pripadaju klasi j) od pozitivnih primjera. To je već vidljivo u našem primjeru. Svaka od triju klasa ima po sedam primjera (v. prethodnu sliku). Ako treniramo binarni klasifikator da razdvaja primjere jedne klase od primjera ostalih dviju, onda će svaki takav klasifikator biti treniran sa 7 pozitivnih primjera i 14 negativnih. Situacija postaje to lošija što je veći broj klasa. Za deset klasa, odnos pozitivnih i negativnih primjera bit će 1:9. Općenito, ako su u skupu $\mathcal{D}$ udjeli K klasa uravnoteženi, tj. ako imamo jednak broj primjera za svaku klasu, omjer pozitivnih naspram negativnih primjera za pojedinačne binarne klasifikacijske probleme u shemi OVR bit će $1 : (K - 1)$. (Ako udjeli početno nisu uravnoteženi, onda će omjer za neke binarne probleme biti povoljniji, ali će za neke druge druge biti još lošiji.) Ovo je problematično jer se linearni diskriminativni modeli teško nose sa situacijom kada primjera iz neke klase ima puno više od primjera iz druge klase. Ono što se u takvim situacijama tipično događa jest da optimizacijski algoritam, u legitimnom nastojanju da smanji empirijsku pogrešku, to jednostavno napravi nauštrb primjera iz manje

klase, tako da ih sve klasificira u većinsku klasu. Riječ je o **problemu neuravnoteženosti klasa** (engl. *class imbalance problem*), koji je u strojnom je učenju vrlo dobro poznat i dobro proučen. Problem je vrlo čest u praksi i predložena su neka rješenja, no mi se njima nećemo baviti. 5

Dakle, odabir između višeklasnih shema OVO i OVR svodi se u praksi na kompromis između broja klasifikatora s jedne i neuravnoteženosti klasa s druge strane. Ako klasa nema baš jako puno i ako vremenska složenost nije prevelik problem, OVO je vjerojatno bolja opcija.

4 Klasifikacija regresijom

Do sada još nismo uveli niti jedan klasifikacijski algoritam, pa je vrijeme da to napokon učinimo. Zapravo, nećemo uvesti nov algoritam, nego ćemo pokušati iskoristiti ono što već znamo. Naime, prošli tjedan radili smo **linearan model regresije**. Razumno pitanje je možemo li taj algoritam nekako upotrijebiti za klasifikaciju. Pokazat će se da ne možemo, ali idemo ipak pokušati, jer ćemo iz toga nešto važno naučiti.

4.1 Naivan pristup

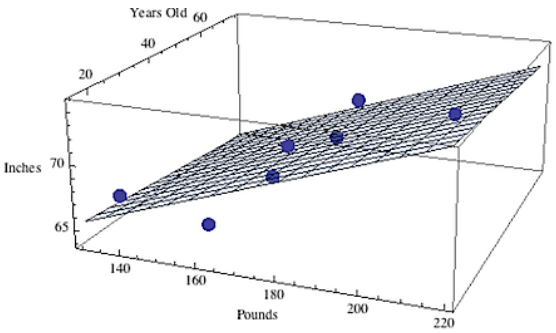
Prisjetimo se, funkcija pogreške (empirijsko očekivanje kvadratnog gubitka) linearnog modela regresije jest:

$$E(\mathbf{w}|\mathcal{D}) = \frac{1}{2}\sum_{i=1}^{N}\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}^{(i)}) - y^{(i)}\right)^2 = \frac{1}{2}(\boldsymbol{\Phi}\mathbf{w} - \mathbf{y})^{\mathrm{T}}(\boldsymbol{\Phi}\mathbf{w} - \mathbf{y})$$

Minimizator pogreške je:

$$\mathbf{w}^* = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{y} = \boldsymbol{\Phi}^{+}\mathbf{y}$$

Kao primjer, razmotrimo regresiju s dvije značajke ($n = 2$). Dakle ulazni prostor je dvodimenzijski i regresijska funkcija je ravnina:



Ova ravnina svakom primjeru $(x_1, x_2) \in \mathbb{R}^2$ pridjeljuje jedan broj, od minus beskonačno do plus beskonačno. Pitanje je: kako bismo to mogli iskoristiti za klasifikaciju? Pa, najjednostavnija stvar koju možemo napraviti jest da klasifikaciju tretiramo kao regresiju: da primjerima iz dviju klasa dodijelimo brojčane oznake iz skupa $\{0, 1\}$ i da na takvom skupu treniramo regresijski model. Drugim riječima, želimo naučiti regresijski model $h(\mathbf{x})$ koji će za primjere iz klase $y = 1$ davati $h(\mathbf{x}) = 1$, a za primjere iz klase $y = 0$ će davati $h(\mathbf{x}) = 0$. Kod predikcije, onda jednostavno gledamo je li $h(\mathbf{x})$ veći ili manji od 0.5. U prvom slučaju primjer klasificiramo u pozitivnu klasu, inače u negativnu. To jest:

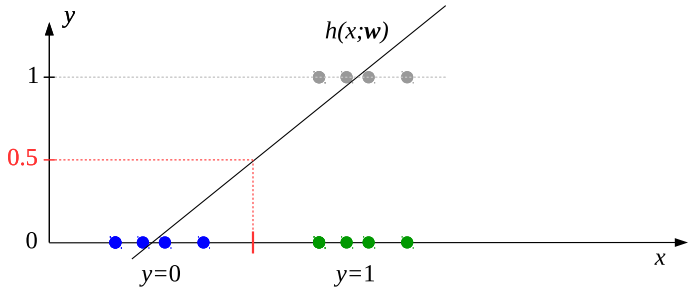
$$h(\mathbf{x}; \mathbf{w}) = \mathbf{1}\{\mathbf{w}^{\mathrm{T}}\mathbf{x} \geqslant 0.5\}$$

Dakle, granicu između klasa $y = 1$ i $y = 0$ čini pravac definiran jednadžbom $h(\mathbf{x}) = 0.5$. Alternativno, model možemo trenirati tako da za primjere iz negativne klase ciljna vrijednost bude $y = -1$. Granica između klasa onda je definirana s $h(\mathbf{x}) = 0$.

Pogledajmo jedan primjer.

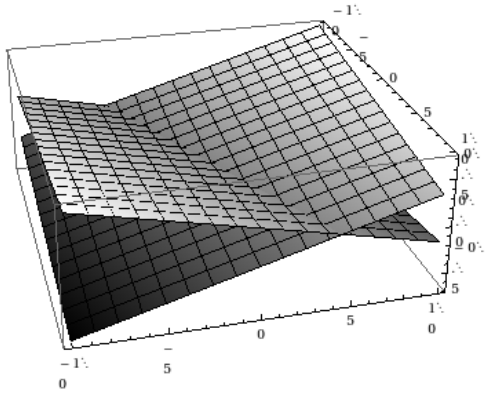
▸ **PRIMJER**

Najjednostavniji primjer je za jednodimenzijski ulazni prostor ($n = 1$). To nije realan primjer, ali poslužit će da objasnimo ideju. Recimo da raspolažemo sa 4 primjera pozitivne klase ($y = 1$) i 4 primjera negativne klase ($y = 0$), i da se oni u jednodimenzijskome ulaznom prostoru (dakle na osi x) mogu razdvojiti u te dvije klase. Treniramo regresijski model tako da za primjere iz klase $y = 1$ postavljamo da je ciljna vrijednost $y = 1$, a primjerima iz klase $y = 0$ da je ciljna vrijednost $y = 0$. To onda izgleda ovako:

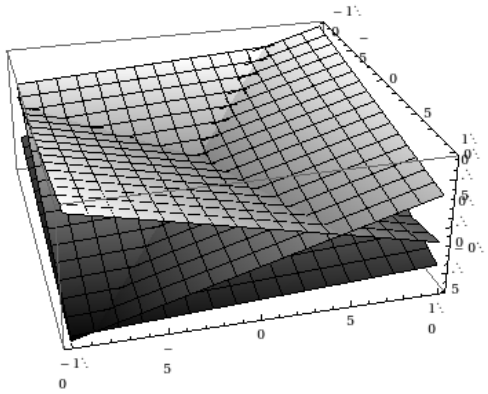


Dobivamo regresijski pravac $h(x; \mathbf{w})$ koji, kao i uvijek, minimizira sumu kvadratnog odstupanja predviđenih vrijednosti od ciljnih vrijednosti. Točka na osi x za koju je $h(x; \mathbf{w}) = 0.5$ predstavlja granicu između klasa u jednodimenzijskome ulaznom prostoru. Za primjere lijevo od te točke vrijedi $h(x; \mathbf{w}) < 0.5$, pa ih klasificiramo kao negativne, a za primjere desno od te točke vrijedi $h(x; \mathbf{w}) \geqslant 0.5$, pa ih klasificiramo kao pozitivne. Ovo zasada izgleda kao da bi moglo raditi, ali vidjet ćemo da ipak postoji problem. (Možda ga već vidite?)

Umjesto jednog modela za dvije klase, mogli smo trenirati dva modela, po jedan za svaku klasu, pa bi granica između klasa bila ondje gdje vrijedi $h_i(\mathbf{x}) = h_j(\mathbf{x})$. Npr., u dvodimenzijskome ulaznom prostoru:



Ako pak želimo klasificirati u više od dvije klase, možemo npr. primijeniti shemu OVR, pa trenirati po jedan model za svaku od K klasa. Granica između susjednih klasa bit će tamo gdje $h_i(\mathbf{x}) = h_j(\mathbf{x})$, gdje su h_i i h_j dvije hipoteze s najvećim vrijednostima za $\mathbf{x}$. Na primjera, za $n = 2$ i $K = 3$:



4.2 Problemi

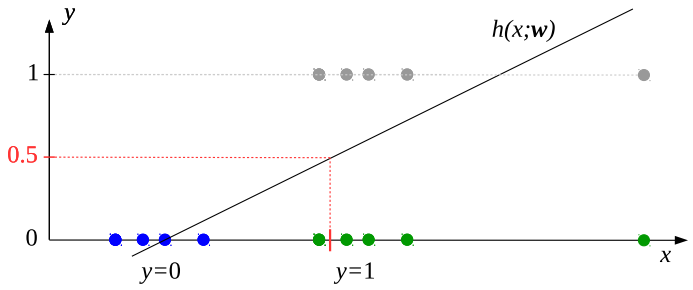
Ovo na prvi pogled izgleda izvrsno! Čini se da smo uspjeli riješiti problem klasifikacije pomoću linearnog modela regresije. Prednosti tog pristupa jesu da je on vrlo jednostavan i postoji rješenje u zatvorenoj formi. Međutim, izgled vara. Postoje nedostatci klasifikacije pomoću linearnog modela regresije, od kojih su neki poprilično problematični. Konkretno, ti nedostatci su:

1. Izlazi modela nemaju vjerojatnosnu intepretaciju, budući da domena hipoteze $h(\mathbf{x})$ nije ograničena na interval $[0, 1]$;
2. Model je nerobusan (neotporan), u smislu da je vrlo osjetljiv na vrijednosti koje odskaču. Konkretno, događa se to da algoritam kažnjava "pretočno" klasificirane primjere te zbog toga, u nekim slučajevima, pogrešno klasificira primjere čak i kada su oni linearno odvojivi.

Pogledajmo detaljnije što mislimo kad kažemo da je model nerobusan.

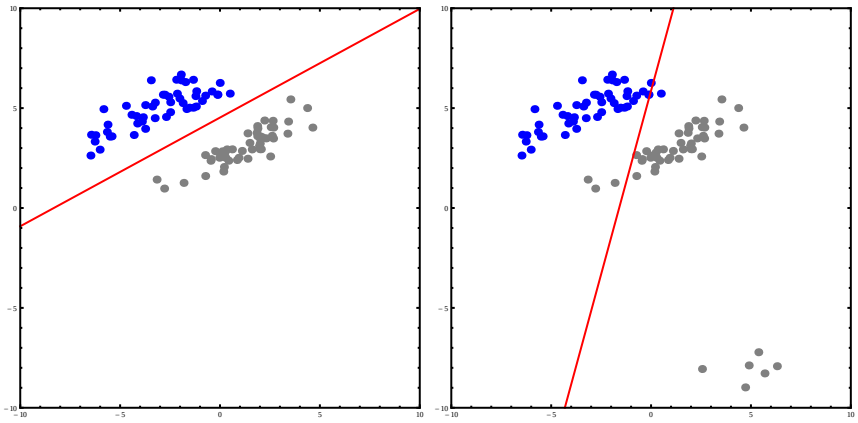
▸ **PRIMJER**

Razmotrimo istu situaciju kao u prethodnom primjeru, dakle klasifikaciju u jednodimenzijskome ulaznom prostoru, ali dodajmo jedan pozitivan primjer ($y = 1$). Neka je taj primjer "duboko" u području pozitivnih primjera, dakle vrlo udesno na osi x. Ovako:



Kakav će to imati efekt na regresijski pravac? Budući da algoritam regresije minimizira kvadratno odstupanje, dodavanje novog primjera imat će efekt da će se cijeli pravac približiti tom novom primjeru, jer bi u protivnom kvadrat reziduala za taj primjer bio vrlo velik. Pravac $h(x; \mathbf{w})$ prikazan na slici bi otprilike bio pravac koji minimizira kvadratno odstupanje (ne baš, bio bi malo većeg nagiba, čini se, ali nema veze, razumijete poantu). S obzirom da je pravac dodavanjem novog primjera promijenio nagib, to se promijenila i vrijednost za koju $h(x; \mathbf{w}) = 0.5$, koja određuje granicu između klasa. Vidimo da se granica između klasa pomakla udesno, i sada je jedan pozitivan primjer pogrešno klasificiran (lažno negativan). Dogodilo se, dakle, da je dodavanje jednog primjera, koji je, primijetite, već bio na ispravnoj strani granice, dovelo do primicanja granice tom primjeru i pogrešne klasifikacije nekog drugog primjera koji je bio blizu granice.

Isti se problem događa i kada je ulazni prostor više dimenzije. Npr., za $n = 2$:



Na lijevoj slici prikazan je dvodimenzijski ulazni prostor s primjerima iz dviju klasa (plave i sive točke) te granica između njih dobivena kao $h(\mathbf{x}; \mathbf{w}) = 0.5$ (crvena linija). Ta je granica dobivena tako da smo učili regresijski model koji primjerima iz jedne klase dodjeljuje vrijednost 1 (npr. plavi primjeri) a primjerima iz druge klase vrijednost 0 (npr. sivi primjeri). Hipoteza odgovara ravnini u trodimenzijskom prostoru $\mathbb{R}^2 \times \mathbb{R}$, a mjesto gdje ta ravnina siječe ravninu $x_1 \times x_2$ jest upravo pravac opisan jednadžbom $h(\mathbf{x}; \mathbf{w}) = 0.5$. Na desnoj slici vidimo što se događa kada skupu za učenje dodamo primjere koji su daleko od granice, a to su sivo obojani primjeri dolje desno. Budući da je model sada treniran tako da i tim primjerima treba dodijeliti vrijednost 0, optimizacijski postupak najmanjih kvadrata nalazi novu ravninu koja minimizira empirijsku pogrešku. Kao što se vidi na slici, granica koju dobivamo kao $h(\mathbf{x}; \mathbf{w} = 0.5)$ sada je značajno primaknuta novododanim primjerima, te model pogrešno klasificira neke primjere iz pozitivne i neke primjere iz negativne klase. Slično kao i u ranijem primjeru, premda je problem i dalje linearno odvojiv, primjeri koji se nalaze daleko od granice imaju velik utjecaj na položaj granice i smanjuju točnost klasifikatora.

Problem koji smo upravo identificirali – da klasifikacija ne radi dobro ako su neki primjeri daleko od granice – zapravo nam ukazuje na to da linearan model regresije nije dobar klasifikator. Ako su primjeri linearno odvojivi u ulaznome prostoru, očekujemo da će nam algoritam strojnog učenja dati hipotezu koja savršeno točno klasificira sve primjere. Međutim, kod linearne regresije to očito nije slučaj. To je velik problem. Ako želimo raditi klasifikaciju, to moramo nekako riješiti.

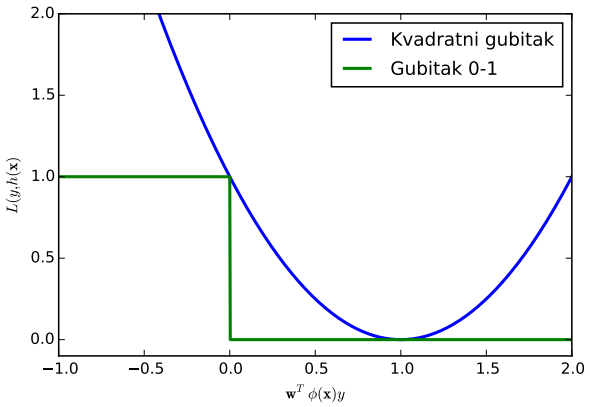
4.3 Tko je kriv?

U čemu je zapravo problem s klasifikacijom pomoću regresije? Prisjetimo se da se svaki algoritam strojnog učenja sastoji od tri komponente: modela, funkcije gubitka (iz koje izvodimo funkciju empirijske pogreške) i optimizacijskog algoritma. Ako postoji problem, on leži u nekoj od ovih triju komponenti. Kojoj? Optimizacijski algoritam definitivno je oslobođen krivnje, jer taj doista samo radi svoj posao na temelju modela i funkcije gubitka koje mu zadamo. Problem je, dakle, u modelu ili funkciji gubitka. Zapravo, možemo reći da je problem *ili* u modelu *ili* u funkciji gubitka. Naime, problem je u modelu, jer za primjere koji su daleko od granice daje vrlo visoke izlazne vrijednosti, koje onda daju veliko kvadratno odstupanje. S druge strane, problem je u funkciji gubitka definiranoj kao kvadratno odstupanje, je ona kažnjava i dobro klasificirane primjere.

Mi ćemo se u nastavku koncentrirati na funkciju gubitka kao krivca za ovaj problem. Cilj nam je da pokažemo zašto funkcija gubitka koju koristimo u algoritmu regresije nije dobra za klasifikaciju. Prisjetimo se, regresija ima **funkciju kvadratnog gubitka**:

$$L(y, h(\mathbf{x})) = \big(y - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})\big)^2$$

Funkcija gubitka L je tipa $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_0^+$, tj. to je funkcija dvije varijable. Bit će nam lakše analizirati ju i grafički prikazati ako je nekako svedemo na funkciju jedne varijable. Srećom, to možemo lako napraviti. Najprije, prebacimo se iz skupa oznaka $y \in \{0, 1\}$ u skup oznaka $y \in \{-1, +1\}$. Sada primijetimo da, ako je klasifikacija nekog primjera $(\mathbf{x}, y)$ ispravna, onda će predznak skalarnog umnoška $\mathbf{w}^{\mathrm{T}}\mathbf{x}$ biti jednak predznaku oznake y, i to bilo da je primjer pozitivan ili negativan. To znači da će umnožak $\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) \cdot y$ uvijek biti pozitivan za točnu klasifikaciju, a negativan za netočnu klasifikaciju (slučaj $\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})y = 0$ možemo uključiti u točnu klasifikaciju, ako je $h(\mathbf{x}) = \mathbf{1}\{\mathbf{w}^{\mathrm{T}}\mathbf{x} \geqslant 0\}$). Ideja je onda da funkciju kvadratnog gubitka L skiciramo kao funkciju jednog argumenta, $\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})y$. Evo kako bi izgledao takav graf za **gubitak 0-1** i za **kvadratni gubitak**:



Ovaj grafikon je važan, pa ćemo ga detaljno objasniti. Na x-osi grafa je vrijednost $\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})y$, koja će, kao što smo već ustanovili, biti pozitivna za točnu klasifikaciju i negativna za netočnu klasifikaciju. Tu vrijednost možemo shvatiti kao "mjeru točnosti klasifikacije". Dakle, pozitivan dio x-osi odgovara slučajevima kada je klasifikacija točna, a negativan dio x-osi slučajevima kada je klasifikacija netočna, a što smo dalje od ishodišta udesno ili ulijevo to je klasifikacija više točna odnosno netočna. Na y-osi je vrijednost funkcije gubitka L. Zelenom je prikazana funkcija gubitka 0-1. Kao što je vidljivo iz grafa, ta funkcija ima vrijednost 1 ako je klasifikacija netočna ($\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})y < 0$), a vrijednost 0 ako je klasifikacija točna ($\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})y \geqslant 0$). Pogledajmo sada kako izgleda graf funkcije kvadratnog gubitka. Gubitak će biti jednak nuli kada je vrijednost predikcije jednaka oznaci y, a to će biti kada $\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})y = 1$. Kada je predikcija jednaka nuli, onda je gubitak jednak 1. Također, kada je predikcija jednaka 2 ili -2, onda će gubitak isto biti 1. Dakle, imamo parabolu, koja je u grafikonu ucrtana plavom bojom.

Razmotrimo što smo dobili. Vidimo da je kvadratni gubitak jednak nuli samo za primjere za koje je $\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x})y = 1$, tj. samo za one primjere za koje je izlaz hipoteze jednak točno 1 (za pozitivne primjere) odnosno točno -1 (za negativne primjere). U svim ostalim slučajima kvadratni gubitak je veći od nule. Pogrešno klasificirani primjeri (negativna strana x-osi) nanose gubitak koji raste kvadratno s udaljenošću primjera od granice. Međutim, vidimo da kvadratni gubitak **kažnjava i točno klasificirane primjere** (pozitivna strana x-osi), i to pogotovo primjere koji su vrlo, vrlo točno klasificirani: primjeri koji su duboko u području ispravne klase području bit će jako kažnjeni. To je i uzrok nerobusnosti rješenja: ako postoji makar jedan primjer koji je duboko u ispravnom području (s koje god strane), gubitak će biti ogroman, i optimizacijski će ga algoritam nastojati smanjiti, pomicanjem hiperravinine tako da se taj gubitak smanji.

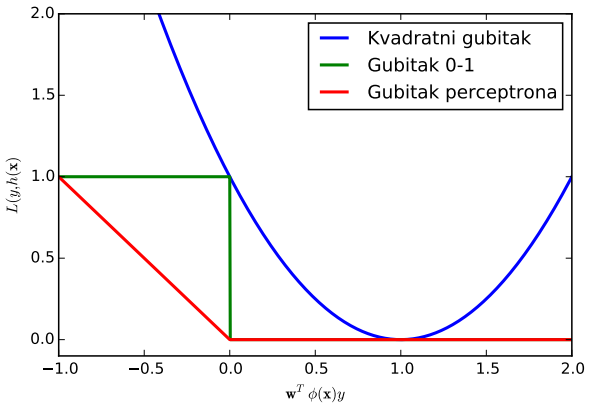
Očito, ako ovo želimo popraviti – a želimo – onda trebamo drugačije definirati funkciju gubitka. Postavlja se pitanje: kakvu funkciju gubitka bismo mi zapravo željeli? Idealno, željeli bismo kažnavati samo netočnu klasifikaciju, i to svaku netočnu klasifikaciju jednako, neovisno o tome koliko je netočna. Drugim riječima, mi želimo **gubitak 0-1** (označen zelenom na prethodnoj skici). 6

Naravno, problem je što ne možemo samo promijeniti funkciju gubitka i opet raditi linearnu

regresiju, jer je to onda više ne bi bila regresija. Naime, funkcija kvadratnog gubitka jedna je od komponenti algoritma linearne regresije. Ako bismo je zamijenili nekom drugom funkcijom gubitka, onda bismo morali promijeniti i optimizacijski postupak (jer postupak najmanjih kvadrata funkcionira samo s kvadratnim gubitkom), i to više ne bi bio algoritam linearne regresije. Dakle, da bismo radili klasifikaciju, trebat će nam ipak neki skroz drugi algoritam. Ne možemo samo tako upotrijebiti linearnu regresiju za klasifikaciju!

Jedna ideja koja odmah pada na pamet jest da za klasifikaciju koristimo algoritam koji kao funkciju gubitka ima gubitak 0-1. To je dobra ideja, ali nažalost nije ostvariva. Naime, funkciju pogreške definiranu kao očekivanje funkcije gubitka 0-1 ne bismo mogli koristiti za optimizaciju. Zašto? Iz dva razloga. Prvo, ta funkcija nije derivabilna, pa ne možemo dobiti rješenje u zatvorenoj formi. Drugo, ako pokušamo optimirati na neki drugi način, npr. gradijentnim spustom, problem s ovom funkcijom jest da je uglavnom (osim za vrijednost 0) konstantna, pa nema nagiba po kojemu bismo se spuštali.

No, evo onda alternativne ideje: umjesto da koristimo gubitak 0-1, zašto ne bismo iskoristili neku funkciju koja joj je što sličnija, ali je derivabilna i ima nagib? Npr., možemo definirati funkciju gubitka tako da je gubitak jednak nuli za točno klasificirane primjere, ali da je gubitak za netočno klasificirane primjere proporcionalan tome koliko je primjer netočan. Na našem grafikonu funkcija gubitaka, takva bi funkcija gubitka izgleda ovako (crvena linija):



Dakle, funkcija gubitka na pozitivnoj strani x-osi je nula, a na negativnoj strani x-osi linearno raste. Taj dio funkcije, kada su primjeri netočno klasificirani, ima derivaciju koja nije nula, što znači da bismo mogli provesti optimizaciju, npr. iterativnom metodom, konkretno npr. **gradijentnim spustom**.

Ispostavlja se već postoji algoritam strojnog učenja koji ima upravo ovakav gubitak. Radi se o algoritmu **perceptrona**, s kojim su mnogi od vas vjerojatno već upoznati. Pogledajmo taj algoritam malo detaljnije, s posebnim naglaskom na funkciju gubitka.

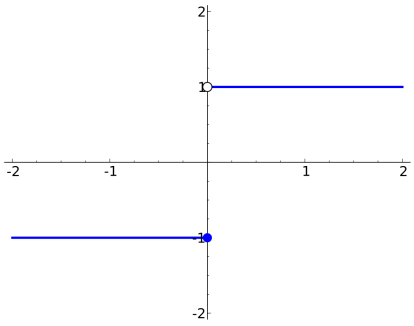
5 Perceptron

Algoritam perceptrona jedan je od najranijih algoritama strojnog učenja i prva uzdanica konekcionističkog (neuronskog) pristupa umjetnoj inteligenciji. U osnovi, riječ o algoritmu za binarnu klasifikaciju. Krenimo od njegovog modela. Osnovna razlika u odnosu na linearan model regresije jest u tome što su kod perceptrona izlazi modela ograničeni na vrijednosti -1 i $+1$. To je ostvareno tako da je skalarni produkt vektora težina i vektora značajki omotan u odgovarajuće definiranu funkciju:

$$h(\mathbf{x}; \mathbf{w}) = f\left(\mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})\right) \tag{8}$$

gdje je f definirana kao **funkcija praga** (step-funkcija): 9

$$f(\alpha) = \begin{cases} +1 & \text{ako } \alpha \geqslant 0 \\ -1 & \text{inače} \end{cases}$$



pa je granica između klasa $y = 1$ i $y = -1$ pravac definiran sa $h(\mathbf{x}; \mathbf{w}) = 0$. Funkciju f u kontekstu perceptrona (i, općenitije, neuronskih mreža) nazivamo **aktivacijska funkcija** (engl. *activation function*) ili **prijenosna funkcija** (engl. *transfer function*). 10

To je, dakle, perceptronov model. Pogledajmo sada njegovu funkciju gubitka. Funkciju gubitka perceptrona već smo bili ucrtali u grafikon funkciju gubitaka (v. gore, crvena linija). Tu funkciju možemo definirati ovako: 11

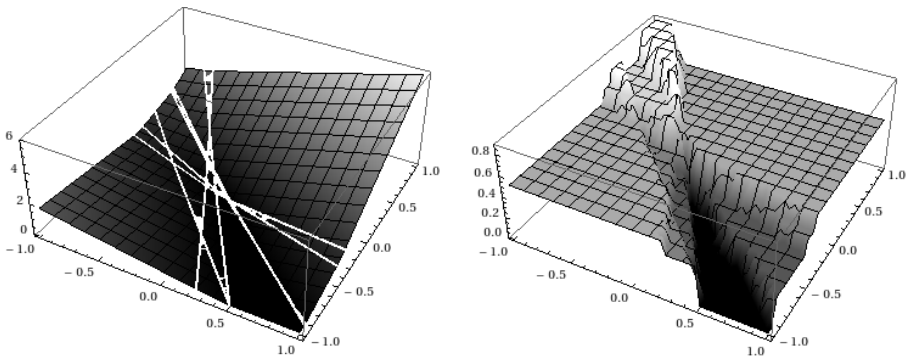
$$\max\left(0, -\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})y\right)$$

Primijetite da koristimo izraz $\max(0, \cdot)$ kako bismo vrijednost funkcije gubitka odozdo ograničili na nulu, budući da ne želimo da gubitak bude negativan (negativan gubitak značio bi nagrada za hipotezu, a mi u nadziranom strojnom učenju nikoga ne želimo nagrađivati). Funkcija pogreške je očekivanje funkcije gubitka, odnosno funkcija empirijske pogreške je prosjek funkcije gubitka na skupu za učenje, no ovdje ćemo zanemariti član $1/N$ (Zašto? Zato jer možemo!), pa je to onda:

$$E(\mathbf{w}|\mathcal{D}) = \sum_{i=1}^{N} \max\left(0, -\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}^{(i)})y^{(i)}\right) = -\sum_{i\,:\,f(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}^{(i)}))\neq y^{(i)}} \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}^{(i)})y^{(i)}$$

Prvi izraz za pogrešku je jednostavno zbroj gubitka po svim primjerima iz $\mathcal{D}$. Drugi izraz za pogrešku je alternativan način da definiramo isto, ali bez funkcije max: suma samo po netočno klasificiranim primjerima. U svakom slučaju, kažnjavamo samo netočno klasificirane primjere, i to proporcionalo tome koliko su oni pogrešno klasificirani.

Sada bi bilo zanimljivo da pogledamo kako izgleda funkcija pogreške $E(\mathbf{w}|\mathcal{D})$ u prostoru parametara $\mathbf{w}$ (tj. u prostoru težina). Za dvodimenzijski prostor parametara $(w_1 \times w_2)$, funkcija pogreške izgleda ovako (lijeva slika):



Vidimo da je funkcija po dijelovima linearna te da je konveksna. Funkcija mora biti konveksna jer je definirana kao zbroj konveksnih funkcija gubitaka. Na ovome grafu, svakoj točki (w_1, w_2) odgovara jedna konkretna hipoteza, tj. jedan pravac definiran parametrima $\mathbf{w}$ kao $h(\mathbf{x}; \mathbf{w}) = 0$. Ta hipoteza akumulira određenu pogrešku, koju izračunavamo tako da pozbrajamo funkciju gubitka za svaki primjer iz $\mathcal{D}$. Promjenom parametara (w_1, w_2) dobivamo različite pravce. Kada pravac pređe preko nekog primjera iz skupa $\mathcal{D}$, klasifikacija tog primjera se mijenja iz $+1$ u -1, ili obrnuto, pa se u toj točki diskretno mijenja vrijednost funkcije pogreške. Zbog toga je površina funkcije pogreške po dijelovima linearna. Na desnoj slici prikazana je funkcija pogreške koju bismo dobili s gubitkom 0-1, što odgovara udjelu pogrešno klasificiranih primjera. Vidimo da je problem s takvom funkcijom pogreške taj što je ona gotovo uvijek konstantna. Nagi pad ima samo u točkama gdje pravac prelazi preko nekog primjera, pa dolazi do promjene udio pogrešno klasificiranih primjera. Takvu funkciju ne možemo minimizirati niti analitički niti numerički. Suprotno tome, pogreška perceptrona (lijeva slika) uglavnom nije konstantna, tj. postoji nagib koji nas vodi prema minimumu. Također možemo vidjeti da je funkcija pogreške perceptrona zapravo aproksimacija udjela pogrešno klasificiranih primjera (tj. funkcija na lijevoj slici je aproksimacija funkcije na desnoj slici).

I, konačno, pogledajmo optimizaciju. Naša definicija funkcije pogreške je vrlo lijepa, no postoji cijena koju moramo platiti što koristimo ovu novu funkciju gubitka, a to je da ne postoji rješenje za minimum funkcije u zatvorenoj formi jer funkcija nije neprekidna. Alternativa je da u primijenimo **gradijentni spust**, i to tako da izračunamo derivaciju funkcije gubitka za svaki primjer koji je netočno klasificiran, i onda za svaki takav primjer promijenimo težine u smjeru suprotnom od porasta gubitka. To će nas postepeno dovesti do minimuma empirijske pogreške. 12

Gradijent gubitka za netočno klasificirane primjere je:

$$\nabla_{\mathbf{w}}\big(-\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})y\big) = -\boldsymbol{\phi}(\mathbf{x})y$$

Primijetite da nas zanima samo gradijent gubitka kada je primjer pogrešno klasificiran. Kada je primjer točno klasificiran, gradijent je nula. Tako smo izbjegli da računamo gradijent po dijelovima linearne funkcije $\max(0, -\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})y)$.

Dobili smo vektor koji pokazuje u kojem smjeru će gubitak rasti (specifično za primjer $\mathbf{x}$). Ako želimo smanjiti gubitak, trebamo ići u suprotnome smjeru. To nam onda daje sljedeće **pravilo ažuriranja težina** (engl. *weight update rule*):

$$\mathbf{w} \leftarrow \mathbf{w} + \eta\boldsymbol{\phi}(\mathbf{x})y$$

gdje vrijednost η određuje tzv. **stopu učenja** (engl. *learning rate*), odnosno veličinu koraka kojim se spuštamo k minimumu. Ovo se pravilo naziva **Widrow-Hoffovo delta-pravilo**.

To sve sada možemo spojiti u vrlo jednostavan algoritam – algoritam perceptrona:

▸ **Algoritam perceptrona**

1: inicijaliziraj $\mathbf{w} \leftarrow (0, \ldots, 0)$
2: **ponavljaj** do konvergencije
3: **za** $i = 1, \ldots, N$
4: **ako** $f\big(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}^{(i)})\big) \neq y^{(i)}$ **onda** $\mathbf{w} \leftarrow \mathbf{w} + \eta\boldsymbol{\phi}(\mathbf{x}^{(i)})y^{(i)}$

Može se dokazati da, ako su primjeri linearno odvojivi, algoritam perceptrona nalazi rješenje (težine koje odgovaraju hipotezi koja savršeno klasificira sve primjere iz $\mathcal{D}$) u konačnom broju koraka. Međutim, ako primjeri nisu linearno odvojivi – a redovito nisu, unatoč funkciji preslikavanja $\boldsymbol{\phi}(\mathbf{x})$ – onda algoritam perceptrona ne konvergira. 13

Sažmimo sada prednosti i nedostatke algoritma perceptrona. Prednosti je da je taj algoritam robustniji od modela linearne regresije, u smislu da točno klasificirani primjeri koji su daleko od

granice između klasa nemaju nikakvog utjecaja na tu granicu (ako je primjer točno klasificiran, gradijent gubitka za taj primjer je nula i težine $\mathbf{w}$ za taj primjer se ne ažuriraju). Druga je prednost da je to očigledno jednostavan algoritam (optimizacijski postupak je jednostavniji od, npr., izračuna pseudoinverza matrice dizajna). No, perceptron ima i niz ozbiljnih nedostataka. Prvo, slično kao i kod regresije, izlazi modela nemaju vjerojatnosnu intepretaciju ($h(\mathbf{x}^{(i)})$ nije ograničena na interval $[0, 1]$). Drugi je problem da rezultat (hipoteza) ovisi o početnim težinama i redoslijedu kojim se provodi ažuriranje težina. Treći, i najveći problem jest što algoritam perceptrona ne konvergira ako primjeri nisu linearno odvojivi.

Prisjetimo se, glavni problem klasifikacije regresijom jest **nerobusnost**: "kažnjavanje" pretočno klasificiranih primjera. Perceptron nema taj problem. Međutim, perceptron ne konvergira ako primjeri nisu linearno odvojivi. Možemo li nekako kombinirati ova dva postupka, i dobiti najbolje od oba? Želimo i robusnost i konvergenciju. Možemo li k tome još dodati probabilistički izlaz? Pokazat će se da možemo. No, to ćemo ostaviti za idući put, kada ćemo govoriti o algoritmu **logističke regresije**.

Sažetak

- **Diskriminativni linearni modeli** daju linearnu granicu između klasa
- Granica između klasa je $(n-1)$-dimenzijska **hiperravnina** u n-dimenzijskom prostoru, definirana preko vektora težina $(w_0, \mathbf{w})$
- Svaki višeklasni klasifikacijski problem može se dekomponirati u skup binarnih klasifikacijskih problema shemom OVO ili OVR, koje imaju svoje prednosti i nedostatke
- **Klasifikacija regresijom** je jednostavna, ali nije robusna, pa se ne koristi
- **Perceptron** je linearan klasifikacijski model koji gradijentnim spustom mimimizira funkciju pogreške koja aproksimira broj pogrešnih klasifikacija
- Perceptron **ne konvergira** za linearno nedvojive probleme, dok za linearno odvojive rješenje ovisi o inicijalizaciji i redoslijedu primjera
- Niti regresija niti perceptron ne daju probabilistički izlaz

Bilješke

1 Ako se prebacimo na zapis s "dummy" jedinicom, $x_0 = 1$, tj. ako težinu w_0 uključimo u vektor $\mathbf{w}$ te model definirao kao $h(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$, onda je granica n-dimenzijska hiperravnina koja prolazi kroz ishodište u $(n+1)$-dimenzijskome proširenom ulaznom prostoru. Kako smo komentirali u bilješci za temu 4 (pročitajte je ako još niste!), ova se razlika svodi na to je li model $h(\mathbf{x})$ definiran kao **afina funkcija** u $\mathbb{R}^n$ (bez "dummy značajke") ili kao **homogena funkcija** u $\mathbb{R}^{n+1}$ (sa "dummy značajkom"). Oba pogleda su, naravno, valjana. Mi ćemo u matričnim izračunima tipično koristiti "dummy" značajku, ali kod skica i interpretacija tipično govorimo o prostoru dimenzije n (odnosno m, ako koristimo preslikavanje), a ne dimenzije $n+1$ (odnosno $m+1$).

2 Prisjetimo se: **vektor normale** na površinu u točki $\mathbf{x}$ je vektor koji je okomit na tangencijalnu ravninu površine u točki $\mathbf{x}$. Kod pravca ili ravnine, normala je okomita na pravac odnosno ravninu. Isto vrijedi i u n-dimenzijskom prostoru: za $(n-1)$-dimenzijsku hiperravninu u $\mathbb{R}^n$ normala je svaki vektor koji je okomit na sve vektore na hiperravnini. Koncept normale može se proširiti i na $(n-1)$-dimenzijske hiperpovršine u $\mathbb{R}^n$: normala hiperpovršine jednaka je vektoru gradijenta (i taj vektor ovisi o točki u kojoj računamo normalu). Normale hiperpovršina nam za sada neće trebati.

3 Pokažimo da je udaljenost $(n-1)$-dimenzijske hiperravnine definirane u $\mathbb{R}^n$ sa $h(\mathbf{x}) = 0$ jednaka $-w_0/\|\mathbf{w}\|$. Neka je $\mathbf{x}$ točka na hiperravnini. Za tu točku onda vrijedi $h(\mathbf{x}) = 0$, tj. vrijedi:

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0 = 0$$

Ako podijelimo obje strane jednadžbe s normom vektora $\mathbf{w}$ i presložimo, dobivamo:

$$\frac{\mathbf{w}^{\mathrm{T}}\mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}\,.$$

Što smo dobili na lijevoj strani? Lijeva strana je **projekcija** vektora $\mathbf{x}$ na vektor normale $\mathbf{w}$. Naime, prisjetimo se definicije skalarnog produkta preko kuta između vektora:

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} = \|\mathbf{w}\|\|\mathbf{x}\|\cos\alpha$$

Ako podijelimo sa $\|\mathbf{w}\|$, dobivamo

$$\frac{\mathbf{w}^{\mathrm{T}}\mathbf{x}}{\|\mathbf{w}\|} = \|\mathbf{x}\|\cos\alpha$$

gdje je $\|\mathbf{x}\|\cos\alpha$ upravo projekcija vektora $\mathbf{x}$ na vektor $\mathbf{w}$. (Općenito, $\mathbf{a}^{\mathrm{T}}\mathbf{b}/\|\mathbf{a}\|$ je projekcija vektora $\mathbf{b}$ na vektor $\mathbf{a}$.) Iz glavne skice možemo vidjeti da je projekcija vektora $\mathbf{x}$ na $\mathbf{w}$ zapravo udaljenost hiperravnine od ishodišta. Dakle, zaključujemo da je udaljenost hiperravnine od ishodišta u prostoru $\mathbb{R}^n$ jednaka $-w_0/\|\mathbf{w}\|$ (u prostoru homogene funkcije $\mathbb{R}^{n+1}$ udaljenost hiperravnine od ishodišta jednaka je nuli; vidi gornju bilješku). Primijetimo da je to **predznačena udaljenost**: može biti pozitivna ili negativna, ovisno o orijentaciji hiperravnine.

4 Da množenje vektora težina $(w_0, \mathbf{w})$ skalarom ne utječe na položaj hiperravnine definirane s $h(\mathbf{x}; \mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0 = 0$ slijedi iz definicije homogenosti linearnog preslikavanja. Naime, ako $h(\mathbf{x})$ definiramo kao homogenu funkciju (uključimo w_0 u vektor $\mathbf{w}$), onda $h(\mathbf{x}; \alpha\mathbf{w}) = h(\alpha\mathbf{x}; \mathbf{w})$, a po definiciji homogenosti $h(\alpha\mathbf{x}) = \alpha h(\mathbf{x})$ te, posljedično, $h(\alpha\mathbf{x}) = 0$ je isti skup točaka kao i $h(\mathbf{x}) = 0$.

5 **Problem neuravnoteženosti klasa** nije specifičan samo za višeklasnu klasifikaciju, nego se može dogoditi i kod binarne klasifikacije. Na primjer, kod klasifikatora za dijagnostiku tumora većina će primjera (srećom) biti negativna. Kod klasifikatora za prepoznavanje sentimenta u pritužbama koje korisnici pišu telekom operaterima većina će primjera (nažalost) također biti negativna. Zapravo, u praksi se vrlo rijetko događa da imamo uravnoteženost klasa. O strategijama rješavanja problema neuravnoteženosti klasa možete pročitati u (Japkowicz, 2000) i (Japkowicz and Stephen, 2002). Tipične strategije su uzorkovanje (naduzorkovanje i poduzorkovanje) ili primjena funkcije gubitka kod koje kazna za pogrešnu klasifikaciju ovisi o tome iz koje je klase primjer (tako da pogrešnu klasifikaciju primjera iz manjinskih klasa više kažnjavamo), što se naziva **klasifikacija osjetljiva na cijenu** (engl. *cost-sensitive classification*).

6 U nekim slučajevima ne želimo svaku netočnu klasifikaciju kažnjavati jednako, već gubitak želimo definirati asimetrično (npr. drugačije kažnjavati lažno pozitivne slučajeve od lažno negativnih). Također, kako je objašnjeno u gornjoj bilješci, nekada gubitak želimo računati drugačije za različite klase. Međutim, i u takvim slučajevima gubitak se može definirati preko gubitka 0-1.

7 **Algoritam perceptrona** osmislio je 1958. godine američki psiholog Frank Rosenblatt (Rosenblatt, 1958), kombiniravši matematički model umjetnog neurona, koji su 1943. godine predožili američki neurofiziolog Warren McCulloch i logičar Walter Pitts (McCulloch and Pitts, 1943), i teoriju o jačanju i slabljenju veza između neurona (sinaptičkoj plastičnosti) kao temelju za učenje, koju je 1949. godine predložio kanadski fiziolog Donald Hebb (Hebb, 1949). Algoritam je postavio temelj za razvoj neuronskih mreža te je bio zaslužan za buđenje velikog interesa i entuzijazma za konekcionistički pristup umjetnoj inteligenciji u šezdesetim godinama prošlog stoljeća, ali i za veliko razočaranje i stagnaciju koji su uslijedili u sedamdesetima, i to ponajviše zahvaljujući kritici u knjizi *Perceptrons* Marvina Minskog i Seymoura Paperta (Minsky and Papert, 1969). Kontroverza koja je u to vrijeme nastala oko perceptona te s time povezane debata o tome kako bi se područje umjetne inteligencije imalo razvijati lijepo su opisane (Olazaran, 1996). Kasnije modificirane varijante perceptrona našle su širu primjenu u strojnom učenju, npr. **perceptron s glasanjem** (engl. *voted perceptron*) (Freund and Schapire, 1999).

8 Ovako definiran model perceptrona čini se istim (do na razliku u pragovima i izlaznim vrijednostima) kao i naš (neuspio) model za klasifikaciju pomoću regresije, gdje smo model definirali kao $h(\mathbf{x}; \mathbf{w}) = \mathbf{1}\{\mathbf{w}^{\mathrm{T}}\mathbf{x} \geqslant 0.5\}$. Doista, modeli su zapravo identični. Međutim, algoritmi će se razlikovati po funkciji

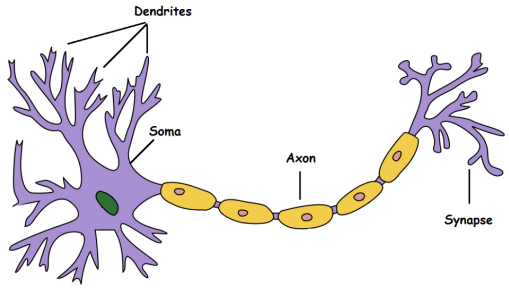
gubitka i optimizacijskome postupku.

9 Alternativno, aktivacijska funkcija perceptrona može se definirati kao:

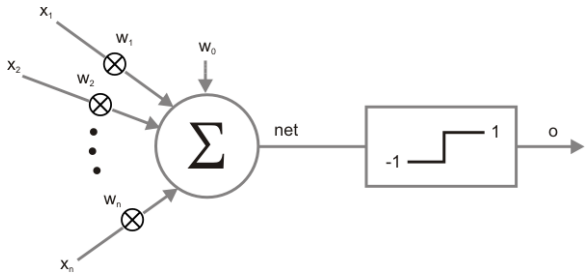
$$f(\alpha) = \begin{cases} 1 & \text{ako } \alpha \geqslant 0 \\ 0 & \text{inače} \end{cases}$$

što je tzv. **Heavisideova funkcija praga**. Mi koristimo aktivacijsku funkciju s kodomenom $\{-1, +1\}$ jer nam to omogućava lakšu usporedbu različitih funkcija gubitaka (naime, uz tako definiranu funkciju, $y \in \{-1, +1\}$, pa umnožak $\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})y$ odgovara "točnosti" klasifikacije, što ne bi bio slučaj s Heavisideovom funkcijom).

10 Naziv **aktivacijska funkcija** (odnosno nešto stariji naziv **prijenosna funkcija**) inspiran je načinom rada biološkog neurona. Pojednostavljeno, biološki neuron preko svojih produžetaka (dendrita) prikuplja u tijelu stanice (somi) elektrokemijsku stimulaciju (preko neuroprijenosnika koji prelaze između sinapsi dendrita) od s njime povezanih ulaznih neurona te se, kada prikupljeni električni potencijal pređe određeni prag, neuron *aktivira* te duž svog živčanog vlakna (aksona) *prenosi* električni impuls prema s njime povaznim izlaznim neuronima:



Analogno, model perceptrona na izlazu daje $+1$ kada vrijednost linearne kombinacija težina i ulaznih značajki ($\mathbf{w}^{\mathrm{T}}\mathbf{x}$) nadmaši vrijednost nula (ili, ekvivalentno, kada iznos $net = \sum_{j=1}^{n} w_i x_i$ nadmaši prag $-w_0$):



Aktivacijsku funkciju nemojte brkati s funkcijom gubitka: aktivacijska funkcija definira preslikavanje skalarnog produkta $\mathbf{w}^{\mathrm{T}}\mathbf{x}$ u izlazne oznake, dok funkcija gubitka definira kaznu uslijed pogrešne predikcije za pojedinačni primjer, gdje je predikcija dobivena kao izlaz aktivacijske funkcije. Aktivacijske funkcije i funkcije gubitka u načelu se mogu proizvoljno kombinirati, no vidjet ćemo da su iz teorijskih ili praktičnih razloga neke kombinacije smislenije od drugih.

11 Poneki pažljivi čitatelj možda će primijetiti da, premda govorimo o funkciji gubitka, nismo eksplicitno napisali da je funkcija gubitka jednaka ovom izrazu, tj. nismo napisali:

$$L\big(y, h(\mathbf{x})\big) = \max\big(0, -\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})y\big)$$

Zašto? Zato što izraz na desnoj strani nije funkcija od $h(\mathbf{x})$, nego od $\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})$. Formalno, dakle, izraz koji smo napisali nije funkcija gubitka. Međutim, kao što smo već pojasnili, izravna usporedba

funkcija gubitaka različitih algoritama mnogo je lakša ako funkciju gubitka promatramo kao funkciju od y i $\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})y$ umjesto kao funkciju od y i $h(\mathbf{x})$.

12 **Gradijentni spust** (engl. *gradient descent*) jednostavna je tehnika optimizacije koja je sveprisutna u strojnom učenju. Ideja gradijentnog spusta jest da u nekoj početnoj točki (vektoru parametara) računamo gradijent funkcije pogreške (smjer porasta) i onda se malo po malo spuštamo u suprotnom smjeru, iterativno, računajući nanovo gradijent u svakoj iteraciji spusta, sve dok gradijent nije jednak nuli, ili vrlo blizu tome. Točka u kojoj je gradijent funkcije jednak nul-vektoru jest ekstrem funkcije (bilo minimumu ili maksimumu). Ako je funkcija konveksna, kao što je to često slučaj s našim funkcijama pogreške, onda je to globalni minimum funkcije. Više o gradijentnom spustu idući put.

13 To je **teorem o konvergencije perceptrona** (Novikoff, 1963). Pristupačniji dokaz možete naći u https://www.cse.iitb.ac.in/~shivaram/teaching/old/cs344+386-s2017/resources/classnote-1.pdf

Literatura

Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.

D. O. Hebb. *The organization of behavior: a neuropsychological theory*. J. Wiley; Chapman & Hall, 1949.

N. Japkowicz. The class imbalance problem: Significance and strategies. In *Proc. of the Int'l Conf. on Artificial Intelligence*, volume 56. Citeseer, 2000.

N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

M. Minsky and S. Papert. *An introduction to computational geometry*. The MIT Press, 1969.

A. B. Novikoff. On convergence proofs for perceptrons. Technical report, Stanford Research Institute, 1963.

M. Olazaran. A sociological study of the official history of the perceptrons controversy. *Social Studies of Science*, 26(3):611–659, 1996.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.