

## 4. Linearna regresija II

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

Jan Šnajder, predavanja, v1.10

Prošli smo puta pričali o **linearnoj regresiji**: definirali smo linearni model, funkciju kvadratne pogreške i optimizaciju, koja se svela na izračun pseudoinverza matrice dizajna. Pogledali smo i koja je probabilistička interpretacija regresije i zaključili smo da – ako pretpostavimo da je **šum** u oznakama normalno distribuiran – onda je upravo **kvadratna pogreška** ona koja maksimizira vjerojatnost skupa označenih podataka. To nam je dalo probabilističko opravdanje za uporabu kvadratne pogreške i uspostavilo (našu prvu) vezu prema probabilističkom strojnom učenju.

U ovom predavanju nastavljamo razmatrati regresiju. Konkretno, pričat ćemo o **preslikavanju u prostor značajki** i o **regularizaciji**. Ispostavit će se da su to dva vrlo važna koncepta u strojnom učenju, koja se koriste u mnogim algoritmima strojnog učenja, a ne samo kod regresije.

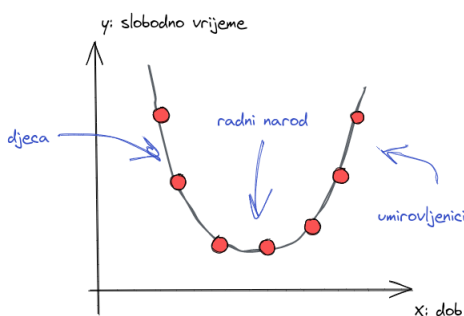
1

### 1 Nelinearna regresija

Za početak, prisjetimo se **modela linearne regresije**. Model je definiran kao linearna kombinacija značajki. Drugim riječima: zavisna varijabla je **linearna funkcija** nezavisnih varijabli:

$$h(\mathbf{x}; \mathbf{w}) = \sum_{j=0}^n w_j x_j = \mathbf{w}^T \mathbf{x}$$

Međutim, često će u praksi veza između nezavisnih varijabli i zavisne varijable biti **nelinearna**. Na primjer, slobodno vrijeme (kao zavisna varijabla) na temelju dobi neki osobe (kao nezavisna varijabla):



#### ► PRIMJER

Još nekoliko primjera za koje nam treba nelinearna regresija:

- (1) Cijena automobila s obzirom na kilometražu – naglo opada pa onda stagnira (ovo smo prošli put uzeli kao primjer linearne ovisnosti, ali u stvarnosti to vjerojatno nije slučaj)
- (2) Box Office Revenue s obzirom na cijenu produkcije – raste pa stagnira
- (3) Prosjek ocjena s obzirom na broj sati provedenih na predavanjima – konkavna funkcija (osim za predmet strojno učenje!)

## 1.1 Model nelinearne regresije

Da bismo modelirali nelinearne ovisnosti, treba nam **nelinearan model**. Npr., recimo da radimo jednostavnu regresiju ( $n = 1$ ). Umjesto linearnog modela

$$h(x; w_0, w_1) = w_0 + w_1 x \quad \in \mathcal{H}_1$$

možemo model definirati kao polinom drugog stupnja:

$$h(x; w_0, w_1, w_2) = w_0 + w_1 x + w_2 x^2 \quad \in \mathcal{H}_2$$

Takvim modelom mogli bismo vrlo točno modelirati ove ranije primjere. Primijetite da je model  $\mathcal{H}_2$  **složeniji** (odnosno “većeg kapaciteta”) od modela  $\mathcal{H}_1$ : model  $\mathcal{H}_2$  sadrži nelinearne hipoteze, a uključuje i sve linearne hipoteze koje sadrži model  $\mathcal{H}_1$ . Konkretno, za  $w_2 = 0$ , isključujemo kvadratnu značajku, pa polinomijalni model degenerira na linearni. Dakle, vrijedi  $\mathcal{H}_1 \subset \mathcal{H}_2$ . Primijetimo također da ta povećana složenost dolazi s cijenom: model  $\mathcal{H}_2$  ima jedan parametar više od modela  $\mathcal{H}_1$  (naime, parametar  $w_2$ ).

Pogledajmo drugi primjer. Radimo višestruku regresiju s  $n = 2$ . Na primjer, želimo predviđati cijene nekretnine na temelju površine ( $x_1$ ) i starosti ( $x_2$ ). Linearan model je:

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

Nelinearan model mogao bi opet biti polinom drugog stupnja:

$$h(\mathbf{x}; \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 + w_5 x_2^2$$

Ovdje, osim kvadratnih značajki, imamo još nešto zanimljivo: značajka  $x_1 x_2$ . Ta se značajka naziva **interakcijska značajka** (ili **cross-term**) i modelira interakciju između dviju ulaznih varijabli. Npr., ako je nekretnina velike površine i vrlo stara, onda će ova značajka imati visoku vrijednost (to, na primjer, može indicirati veliku staru vilu... možda je to poželjna nekretnina s visokom cijenom?).

Korištenje polinoma drugog stupnja samo je jedan način – ne i jedini – kako dobiti nelinearan model. Veću nelinearnost mogli smo ostvariti da smo model definirali polinomom stupnja višeg od dva. Dodavanje interakcijskih značajki drugi je način da se model učini nelinearnim. Možemo, naravno, i kombinirati ova dva načina. Nadalje, nelinearnost se može ostvariti i drugim funkcijama koje nisu polinomi (premda mi u to nećemo ulaziti).

Evo nekoliko različitih regresijskih modela, temeljenih na polinomima, koji se razlikuju po načinu modeliranja nelinearnosti:

- Linearna višestruka regresija:

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

Ovaj model ima više značajki (ukupno  $n$ ), ali je posve linearan, dakle, ne modelira nikakvu nelinearnost.

- Jednostruka (jednostavna) polinomijalna regresija ( $n = 1$ ):

$$h(x; \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \dots + w_d x^d$$

Ovaj model ima samo jednu značajku ( $n = 1$ ), no koristi polinom stupnja  $d$  kako bi modelirao nelinearnost. Primijetite da je  $d$  ovdje **hiperparametar**, budući da određuje složenost modela.

- Višestruka polinomijalna regresija ( $n = 2, d = 2$ ):

$$h(\mathbf{x}; \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 + w_5 x_2^2$$

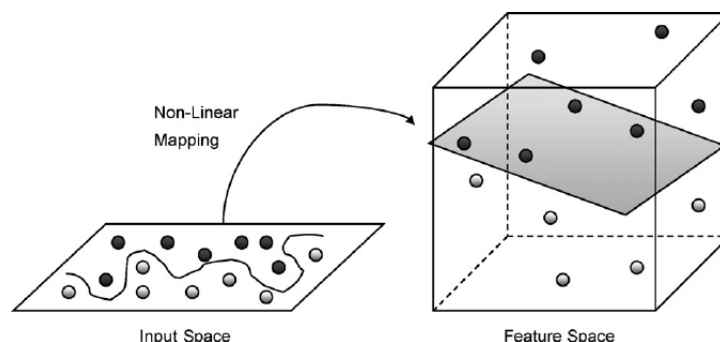
Ovaj model ima dvije značajke ( $n = 2$ ) i koristi polinom drugog stupnja  $d = 2$  kako bi modelirao nelinearnost. Osim kvadratnih značajki  $x_1^2$  i  $x_2^2$ , model sadrži i interakcijsku značajku  $x_1 x_2$ , kojom može modelirati interakciju između značajki  $x_1$  i  $x_2$  (u jeziku statistike: neaditivni efekt nezavisnih varijabli na zavisnu varijablu).

Ovo je samo nekoliko primjera; očito, možemo definirati model proizvodnog stupnja polinoma, možemo uključiti ili ne uključiti interakcijske članove za samo neke parove varijabli, možemo uključiti interakcijske članove za više od dvije varijable, itd. (I opet napomenimo da se nelinearnost može ostvariti i nekim drugim nelinearnim funkcijama, ne nužno polinomijalnim.)

## 1.2 Preslikavanje u prostor značajki

Premda su modeli koje smo upravo razmotrili zapravo različiti, mi bismo ih voljeli nekako **unificirati**, da ih možemo tretirati na isti način kada radimo optimizaciju. To nas dovodi do jedne poprilično moćne ideje u strojnom učenju: uvijek kada želimo s linearnog modela preći na nelinearni, umjesto da mijenjamo model, mi ćemo promijeniti podatke! Način na koji ćemo promijeniti podatke jest da ih preslikamo iz **ulaznog prostora** (engl. *input space*) u neki drugi prostor. Taj drugi prostor zvat ćemo **prostor značajki** (engl. *feature space*). Postupak preslikavanja zvat ćemo **preslikavanje u prostor značajki** (engl. *feature mapping*). Ideja je da, ako odaberemo prikladno preslikavanje, onda će podatci – premda u ulaznom prostoru nisu bili linearni – u prostoru značajki to biti!

Ovakvo preslikavanje je sveprisutno u strojnom učenju, kako u regresiji, tako i u klasifikaciji. Npr., kod klasifikacije:



Prikladnim (nelinearnim) preslikavanjem iz ulaznog prostora u prostor značajki više dimenzije možemo ostvariti da primjeri, koji u ulaznome prostoru nisu bili linearno odvojivi, to budu u prostoru značajki. Na ovom primjeru vidimo da linearna granica (hiperravnina) u prostoru značajki, koja savršeno odjeljuje primjere dviju klasa, u ulaznome prostoru odgovara nelinearnoj granici. No, umjesto da pokušamo u ulaznom prostoru naći nelinearnu hipotezu, mi ćemo primjere nelinearnim preslikavanjem preslikati u prostor značajki veće dimenzije i tamo ih onda odvojiti linearnom hipotezom. U ulaznom prostoru činit će se kao da smo ih odvojili nelinearnom hipotezom.

Ideja će pritom biti uvijek ista: problem koji je ulaznom prostoru nelinearan, nelinearnim preslikavanjem preslikat ćemo u prostor više dimenzije gdje će on (nadamo se) biti linearan, i onda ćemo tamo jednostavno primijeniti linearne modele.

Definirajmo sada formalno kako se radi ovo preslikavanje. Uvest ćemo fiksni skup tzv. **baznih funkcija** (nelinearne funkcije ulaznih varijabli):

$$\{\phi_0, \phi_1, \phi_2, \dots, \phi_m\}$$

gdje  $\phi_j : \mathbb{R}^n \rightarrow \mathbb{R}$ . Dakle, svaka bazna funkcija  $\phi_j$  uzima cijeli vektor primjera  $\mathbf{x}$  i preslikava ga u jedan broj. Taj broj odgovara jednoj od  $m$  značajki u  $m$ -dimenzijskom prostoru značajki. Baznih funkcija ukupno ima  $m + 1$ , po jedna za svaku dimenziju  $m$ -dimenzijskog prostora značajki, plus jedna dodatna bazna funkcija,  $\phi_0$ . Ta dodatna bazna funkcija dogovorno je definirana kao  $\phi_0(\mathbf{x}) = 1$  i ona odgovara “dummy” jedinici kakvu smo imali i kod proširenog vektora  $\mathbf{x}$ . Vrijednost **funkcije preslikavanja** definirana je onda kao vektor dobiven primjenom

pojedinačnih baznih funkcija na primjer  $\mathbf{x}$ :

$$\begin{aligned}\phi &: \mathbb{R}^n \rightarrow \mathbb{R}^{m+1} \\ \phi(\mathbf{x}) &= (\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \dots, \phi_m(\mathbf{x}))\end{aligned}$$

Funkcija preslikavanja značajki preslikava primjere iz  $n$ -dimenzijskog ulaznog prostora u  $m$ -dimenzijski prostor značajki. Tipično je  $m > n$ , odnosno preslikavamo u prostor više dimenzije od ulaznog prostora. Zašto? Zato jer tako imamo veću šansu da će nešto što je u ulaznom prostoru bilo nelinearno u prostoru značajki postati linearno.

Sada to preslikavanje možemo lako ugraditi u naš linearan model, jednostavno tako da vektor  $\mathbf{x}$  zamijenimo vektorom  $\phi(\mathbf{x})$ :

$$h(\mathbf{x}; \mathbf{w}) = \sum_{j=0}^m w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

gdje je, prisjetimo se, vektor  $\phi(\mathbf{x})$  proširen “dummy” značajkom, tj.  $\phi_0(\mathbf{x}) = 1$ .

Ovime smo ostvarili naš cilj: primjeri iz ulaznog prostora preslikavaju se u neki prikladni prostor značajki, ali je sâm model ostao nepromijenjen – i dalje je to linearan model, jedina je razlika što umjesto značajki  $x_j$  koristimo preslikane značajke  $\phi_j(\mathbf{x})$ .

Ovakav model naziva se **model linearne regresije**. To je model koji je **linearan u parametrima** ( $w_j$ ), ali koji – ovisno o funkciji preslikavanja – ne mora biti linearan u izvornim značajkama!

Ovdje treba paziti na suptilne razlike u nazivlju: **model linearne regresije** nije isto što i **linearna regresija**. Linearna regresija (koju smo radili prošli puta) poseban je slučaj modela linearne regresije kod kojega ne radimo baš nikakvo preslikavanje.

Sada kada smo općenito definirali funkciju preslikavanja, pogledajmo i neke konkretne funkcije preslikavanja. Uz odgovarajući odabir funkcije preslikavanja, lako možemo dobiti bilo koji od ranije razmatranih modela linearne regresije.

- Linearna višestruka regresija:

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

Ovdje zapravo nemamo nikakvog preslikavanja, tj. funkcija preslikavanja je zapravo funkcija identiteta (uz proširenje “dummy” jedinicom),  $\phi(\mathbf{x}) = (1, \mathbf{x})$ . S takvom funkcijom preslikavanja naš model linearne regresije je doista linearna regresija.

- Jednostruka (jednostavna) polinomijalna regresija ( $n = 1$ ):

$$\phi(x) = (1, x, x^2, \dots, x^d)$$

Značajku  $x$  preslikavamo u polinom stupnja  $d$ , čime iz 1-dimenzijskog ulaznog prostora ( $n = 1$ ) prelazimo u  $d$ -dimenzijski prostor značajki ( $m = d$ ).

- Višestruka (bivarijatna) polinomijalna regresija drugog stupnja ( $n = 2, d = 2$ ):

$$\phi(\mathbf{x}) = (1, x_1, x_2, x_1 x_2, x_1^2, x_2^2)$$

Ovdje iz dvodimenzijskog ulaznog prostora preslikavamo u 5-dimenzijski prostor značajki primjenom funkcije preslikavanja definirane kao polinom drugog stupnja s interakcijskom značajkom. Prostor značajki sastoji od izvornih značajki  $x_1$  i  $x_2$ , te dodatno od interakcijske značajke  $x_1 x_2$  i dviju kvadratnih značajki,  $x_1^2$  i  $x_2^2$ .

### 1.3 Optimizacijski postupak

Ovime smo riješili pitanje modela. Funkcija gubitka (i, posljedično, funkcija pogreške), ista je kao i kod linearne regresije jer je izlaz identičan za oba modela: dakle, to je kvadratni gubitak odnosno funkcija kvadratne pogreške. Ostaje nam još razmotriti **optimizacijski postupak**. Kako izgleda optimizacijski postupak za model linearne regresije?

Dobra vijest je što se baš ništa suštinski ne mijenja u odnosu na ono što smo već izveli! Naime, model s funkcijom preslikavanja je i dalje linearan model. Jedino što ćemo, umjesto nad izvornom matricom dizajna  $\mathbf{X}$ , optimizaciju provesti nad matricom dizajna u kojoj smo proveli preslikavanje u prostor značajki. Tu matricu označit ćemo sa  $\Phi$ .

Dakle, umjesto izvorne matrice dizajna (koja je definirana s izvornim značajkama):

$$\mathbf{X} = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & x_2^{(N)} & \dots & x_n^{(N)} \end{pmatrix}_{N \times (n+1)}$$

sada ćemo imati matricu dizajna u kojoj smo na svakom primjeru primijenili funkciju preslikavanja  $\phi(\mathbf{x})$ :

$$\Phi = \begin{pmatrix} 1 & \phi_1(\mathbf{x}^{(1)}) & \dots & \phi_m(\mathbf{x}^{(1)}) \\ 1 & \phi_1(\mathbf{x}^{(2)}) & \dots & \phi_m(\mathbf{x}^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(\mathbf{x}^{(N)}) & \dots & \phi_m(\mathbf{x}^{(N)}) \end{pmatrix}_{N \times (m+1)} = \begin{pmatrix} \phi(\mathbf{x}^{(1)})^T \\ \phi(\mathbf{x}^{(2)})^T \\ \vdots \\ \phi(\mathbf{x}^{(N)})^T \end{pmatrix}_{N \times (m+1)}$$

Pritom nemojmo zaboraviti na “dummy” jedinicu (prvi stupac u matrici  $\Phi$ ). Optimizacijskom postupku je, dakako, svejedno s kakvom matricom radi. Prije smo imali:

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \mathbf{X}^+ \mathbf{y}$$

a sada jednostavno podmećemo matricu  $\Phi$  umjesto  $\mathbf{X}$ :

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

gdje je  $\Phi^+ = (\Phi^T \Phi)^{-1} \Phi^T$  pseudoinverz matrice dizajna  $\Phi$ . Za pseudoinverz matrice  $\Phi$ , vrijede, naravno, sve iste napomene kao i one koje smo prošli put dali za pseudoinverz matrice dizajna  $\mathbf{X}$ .

Ovime smo dobili ono što smo i željeli: jedan unificirani algoritam za regresiju, neovisno o tome koju funkciju preslikavanja  $\phi$  odaberemo. Nije li to izvrsno?! No pitanje je sada: koju funkciju preslikavanja odabrati? To nas vodi do iduće teme: prenaučenosť.

## 2 Prenaučenost

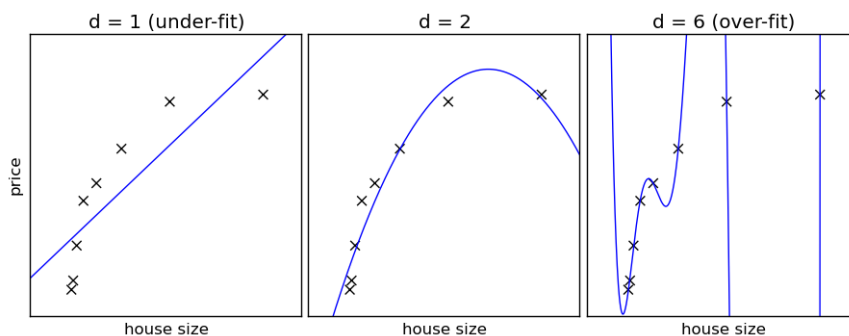
Prošli tjedan rekli smo da, ako na raspolaganju imamo familiju modela, onda trebamo odabrati optimalan model za naš problem. U suprotnom će model biti ili **prenaučen** ili **podnaučen**. Također smo rekli da su različiti modeli indeksirani njihovim **hiperparametrima**.

Kao što smo vidjeli, kod linearnog modela regresije možemo ostvariti različite složenosti modela odabirom odgovarajuće funkcije preslikavanja. Što je onda hiperparametar kod linearnog modela regresije? Pa, hiperparametar je upravo funkcija preslikavanja  $\phi$ . Naime, ovisno o tome kakvu funkciju preslikavanja odaberemo, model će biti više ili manje složen. Npr., model polinoma drugog stupnja je složeniji model od linearnog modela. Dalje, model polinom trećeg stupnja je još složeniji, itd. Istodobno trebamo biti svjesni da, što je model složeniji, to on ima više parametara, i takav je model onda lakše prenaučiti.

### ► PRIMJER

Recimo da želimo naučiti model linearne regresije za predviđanje cijena nekretnina u Bostonu. Jednostavnosti radi, razmotrimo jednostavnu (tj. jednoulaznu) regresiju,  $n = 1$ . Neka je nezavisna varijabla stambena površina nekretnine. Razmatramo tri modela: polinom prvog stupnja (iliti pravac), polinom drugog stupnja (parabola) i polinom šestog stupnja (“sektičan polinom”, u slučaju da vas je zanimalo). Kao što već znamo, sve te modele lako možemo definirati prikladnom funkcijom preslikavanja  $\phi(\mathbf{x}) = (x_1, x_1^2, \dots, x_1^d)$ , gdje je  $d$  željeni stupanj polinoma.

Neka se naš skup označenih primjera sastoji od šest primjera, tj.  $|\mathcal{D}| = 6$ . Nakon što so naučili tri spomenuta modela, dobivamo u ulaznom prostoru ovakve krivulje:



Što se ovdje dogodilo? Polinom prvog stupnja ne čini se baš prikladnim modelom, budući da predikcije tog modela znatno odstupaju od ciljnih vrijednosti. Taj model je podnaučen. Polinom drugog stupnja čini se vrlo dobrim na našem skupu podataka. Za taj model bismo na našem skupu  $\mathcal{D}$  rekli da je optimalne složenosti (premda nam zdrav razum govori da parabola vjerojatno nije najbolji način za modeliranje cijene stana u ovisnosti o površini; odnos je vjerojatno nelinearan, ali ne očekujemo baš da cijene za nekretnine veće površine počnu padati). Polinom šestog stupnja je apsolutni šampion u smislu da je regresijska krivulja doista prošla kroz svaku točku, međutim jasno je da je taj model prenaučeni. Čini se razumnim pretpostaviti da cijena o površini nekretnine ovisi nekako nelinearno, no malo je vjerojatno da je ta ovisnost tako histerična kako ju modelira hipoteza definirana polinomom šestog stupnja. Ovaj se model naučio šumu u podacima, pa je rezultat hipoteza koja žestoko oscilira i koja će očito loše generalizirati na neviđenim podacima.

7

Kako spriječiti prenaučeniost modela? Ovo je općenito pitanje, i koji god odgovor da smislimo on će vrijediti kako za algoritam regresije, tako i za većinu drugih algoritama strojnog učenja. Glavni načini za sprječavanje prenaučeniosti su sljedeći:

- **Koristiti više primjera za učenje** – intuitivno, što imamo više primjera, to će regresijskoj krivulji biti teže da prođe baš kroz svaki od njih. Kada je primjera toliko da krivulja ne može proći kroz svaki od njih, kvadratni gubitak će se pobrinuti da parametri budu takvi da krivulja minimizira zbroj kvadratnog odstupanja, što znači da će krivulja manje oscilirati. Dakle, što više primjera imamo, to stabilnija hipoteza, odnosno to manje prenaučeni model. Premda je ovaj recept za sprječavanje prenaučeniosti jednostavan, u praksi je uglavnom teško izvediv: naime, u praksi je broj označenih primjera redovito ograničen, i ne možemo samo tako pribaviti nove primjere (ili ih nemamo, ili je njihovo označavanje skupo, ili je označavanje dugotrajno);
- **Odabrati model unakrsnom provjerom** – ovo nam je već poznato. Konkretno, kod modela linearne regresije, to bi značilo da na skupu za učenje treniramo modele s različitim funkcijama preslikavanja, koje zatim ispitujemo na ispitnom skupu, te odabiremo onaj model koji ima najmanju pogrešku na ispitnom skupu, jer taj model najbolje generalizira na neviđene podatke. Ako je funkcija preslikavanja definirana kao polinom stupnja  $d$ , onda između takvih funkcija možemo uspostaviti potpuni uređaj po složenosti modela,

gdje složenost određuje hiperparametar  $d$ . Unakrsna provjera svodi se onda na ispitivanje hiperparametra  $d$  iz nekog unaprijed zadanog intervala, npr.,  $d \in \{1, 2, \dots, 6\}$ ;

- **Odabrati podskup značajki** – model je to složeniji što ima više značajki. Ako smanjimo broj značajki, dobit ćemo jednostavniji model. Naravno, ne bi bilo pametno ukloniti značajke koje su korisne; idealno, izbacili bismo iz modela one značajke koje su beskorisne. Postupak pronalaženja optimalnog podskupa značajki naziva se **odabir značajki** (engl. *feature selection*), i o njemu ćemo pričati na samom kraju predmeta. Budući da skup značajki zapravo definira model, na postupak odabira značajki možemo gledati kao na odabir modela;
- **Reducirati dimenzionalnost ulaznog prostora** – umjesto izbacivanja nepotrebnih značajki, možemo cijeli ulazi prostor preslikati u prostor značajki nižih dimenzija. Te nove značajke neće odgovarati izvornim značajkama, no nekad nam to nije ni bitno;
- **Regularizacija** – regularizacija je postupak kojim se implicitno sprječava odabir presloženih modela pri samom postupku optimizacije parametara modela. Regularizacija je univerzalna ideja u strojnom učenju, i o njoj ćemo pričati danas;
- **Bayesovska regresija** (odnosno, općenitije, **bayesovski odabir modela**) – bayesovski postupci u statistici i strojnom učenju oslanjaju se na ideju da se parametri modela, jednako kao i primjeri  $\mathbf{x}$  i oznake  $y$ , tretiraju kao slučajne varijable za koje je potrebno unaprijed definirati odgovarajuće distribucije (apriorne distribucije), a učenje se onda svodi na procjenu tih parametara i primjenu Bayesovog teorema kako bi se dobila aposteriorna distribucija oznake  $y$  za ulazni primjer  $\mathbf{x}$ . Mi na ovom predmetu (nažalost) nećemo pričati ništa o bayesovskim metodama.

8

Stanje s ovim tehnikama je sljedeće. Prva tehnika (ubaciti više primjera) nije uvijek izvediva. Iduće tri tehnike se često koriste. One striktno razdvajaju fazu odabira modela od faze učenja modela. S druge strane, zadnje dvije tehnike (regularizacija i bayesovska regresija) u nekoj mjeri spajaju fazu odabira modela i učenje modela.

Mi ćemo se fokusirati na **regularizaciju**, kao jednu vrlo osnovnu, vrlo učinkovitu i često korištenu tehniku u strojnom učenju.

### 3 Regularizacija

Početak priče o regularizaciji kreće od jednog jednostavnog opažanja: kod linearnih modela, što je model složeniji, to ima veće vrijednosti parametara  $\mathbf{w}$ . Uzmimo kao primjer model polinoma drugog stupnja:

$$h(\mathbf{x}; \mathbf{w}) = w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2 + w_4x_1^2 + w_5x_2^2$$

Ako su težine  $w_3$ ,  $w_4$  ili  $w_5$  velike magnitude (bilo pozitivne ili negativne), hipoteza će biti izrazito nelinearna. S druge strane, što su težine  $w_3$ ,  $w_4$  i  $w_5$  bliže nuli, to je hipoteza manje nelinearna. U krajnosti, ako  $w_3 = w_4 = w_5 = 0$ , model efektivno degenerira u linearan model.

Ovaj primjer ukazuje na to da složenost linearnog modela regresije izravno ovisi o težinama značajki, i da će prenaučeni modeli imati visoke vrijednosti za mnoge težine. Onda, kako bismo to spriječili, mi ćemo već pri učenju modela **ograničiti magnitudu parametara**. To ćemo raditi tako da ćemo **kažnjavati** hipoteze s visokim vrijednostima parametara. Opisani mehanizam nazivamo **regularizacija**.

U praksi, to izgleda tako da krenemo s relativno složenim modelom, kako bismo spriječili podnaučenost, ali onda koristimo regularizaciju kako bismo mu efektivno ograničili složenost. Dakle, krećemo od vrlo složenog, raskošnog modela, ali onda ga sputavamo, da se ne “razuzda” previše. Ako to sputavanje pametno napravimo, spriječit ćemo prenaučенost, odnosno ostvarit ćemo kompromis između jednostavnosti i složenosti modela.

Idealan cilj regularizacije bio bi što više parametara (težina) pritegnuti na nulu. Osim što ćemo time iz složenog modela dobiti jednostavniji model, ako mnoge težine pritegnemo baš na nulu, onda ćemo dobiti tzv. **rijetke modele** (engl. *sparse models*). U strojnom učenju volimo rijetke modele jer su (1) manje skloni prenaučivosti, (2) računalno jednostavniji kod predviđanja i (3) interpretabilniji – znamo koje značajke definitivno ne utječu na izlaz (što može biti važno u statističkoj analizi podataka).

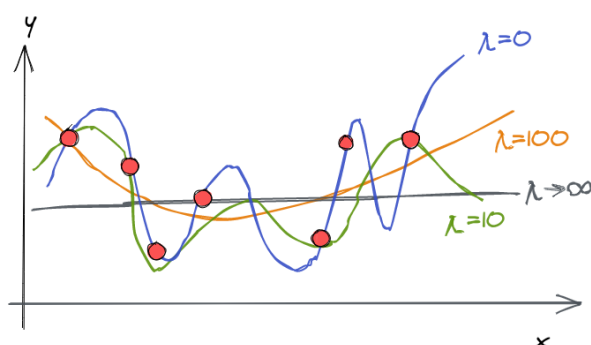
### 3.1 Regularizirana regresija

Pogledajmo sada kako zapravo ostvarujemo regularizaciju. Vrlo jednostavno: u funkciji pogreške (koju minimiziramo), pored empirijske pogreške, dodat ćemo još jedan član, koji karakterizira složenost modela u ovisnosti o njegovim težinama. Tako dobivamo **regulariziranu funkciju pogreške**:

$$E_R(\mathbf{w}|\mathcal{D}) = E(\mathbf{w}|\mathcal{D}) + \underbrace{\lambda\Omega(\mathbf{w})}_{\text{reg. izraz}}$$

U ovom izrazu,  $\Omega(\mathbf{w})$  je **regularizacijski izraz**, a  $\lambda$  je tzv. **regularizacijski faktor**. Ako  $\lambda = 0$ , onda se vraćamo na neregulariziranu funkciju pogreške. Veća vrijednost regularizacijskog faktora  $\lambda$  više će kažnjavati složenost modela i imat će za posljedicu smanjenje efektivne složenosti modela.

#### ► PRIMJER



Treniramo model L2-regularizirane jednostavne regresije ( $n = 1$ ) sa funkcijom preslikavanja definiranom kao polinom 10. stupnja. Razmotrimo četiri modela, koji se međusobno razlikuju po iznosu regularizacijskog faktora  $\lambda$ . Ako ne regulariziramo ( $\lambda = 0$ ), polinom 10. stupnja na našem skupu od 7 primjera ostvaruje savršenu točnost. Kako povećavamo iznos regularizacijskog faktora  $\lambda$ , tako težine  $\mathbf{w}$  sve više pritežemo k nuli, što rezultira sve zaglađenijom regresijskom krivuljom u ulaznome prostoru. Npr., polinom 10. stupnja regulariziran sa  $\lambda = 100$  mogao bi u našem slučaju efektivno odgovarati polinomu drugog stupnja (paraboli). U ekstremnom slučaju, kada  $\lambda \rightarrow \infty$ , sve težine osim težine  $w_0$  (objasniti ćemo ispod zašto je  $w_0$  izuzeta) pritežemo k nuli, pa dobivamo pravac nagiba nula koji os  $y$  siječe na mjestu srednje vrijednosti oznaka  $y$ . Naravno, tako snažna regularizacija nije ono što u praksi želimo.

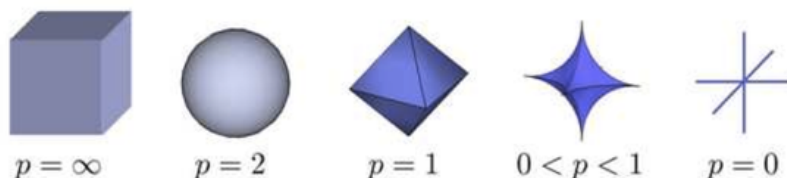
Pogledajmo sada kako bismo konkretno definirali  $\Omega(\mathbf{w})$ . Vrijednost tog izraza ovisi o težinama  $\mathbf{w}$ . Treba nam neka funkcija težina  $\mathbf{w}$ , i to takva da je njezina vrijednost velika, kada su težine velike, a mala, kada su težine male. Pritom, imajmo na umu da je  $\mathbf{w}$  zapravo **vektor** težina. Kakva bi to bila funkcija? Najizravnija opcija je **norma vektora**. Naime, norma je funkcija koja daje duljinu vektora, i ona će biti to veća što su veće magnitude komponentata vektora.



U matematici postoji jedna općenita klasa vektorskih normi koja se zove **p-norma**. Općenito, dakle, regularizacijski izraz  $\Omega(\mathbf{w})$  možemo definirati kao **p-normu vektora težina**:

$$\Omega(\mathbf{w}) = \|\mathbf{w}\|_p = \left( \sum_{j=1}^m |w_j|^p \right)^{\frac{1}{p}}$$

Različite p-norme možemo vizualizirati ovako:



Slike prikazuju gdje, u 3-dimenzijskom prostoru, leže točke (odnosno vrhovi dogovarajućih vektora s hvatištem u ishodištu) koje su jednako udaljene od ishodišta (odnosno vektori koji imaju istu normu). Drugim riječima, slike prikazuju kako izgleda (hiper)površina definirana funkcijom  $\|\mathbf{w}\|_p = \text{konst.}$  Za beskonačnu normu to je kocka, za 2-normu to je kugla, za 1-normu to je oktaedar, za 0-normu to su točke na vrhovima oktaedra (na osima). U višim dimenzijama imali bismo hiperkocku, hiperkuglu, hiperoktaedar odnosno točke u vrhovima hiperoktaedra.

U strojnom učenju tipično koristimo norme sa  $p = 2$  i  $p = 1$ , a spomenut ćemo još i normu sa  $p = 0$ . Njih zovemo L2-norma, L1-norma, odnosno L0-norma. Njihove definicije slijede iz gornje definicije za općenitu p-normu:

- **L2-norma** ( $p = 2$ ), poznata i kao **euklidska norma**:

$$\|\mathbf{w}\|_2 = \sqrt{\sum_{j=1}^m w_j^2} = \sqrt{\mathbf{w}^T \mathbf{w}}$$

- **L1-norma** ( $p = 1$ ), poznata i kao **Manhattan-udaljenost**:

$$\|\mathbf{w}\|_1 = \sum_{j=1}^m |w_j|$$

- **L0-norma** ( $p = 0$ ), koja je u stvari jednaka broju ne-nul elemenata vektora (tj. broju značajki koje nisu uklonjene):

$$\|\mathbf{w}\|_0 = \sum_{j=1}^m \mathbf{1}\{w_j \neq 0\}$$

Primijetite – a to je vrlo važno – da se težina  $w_0$  ne regularizira. Zašto? Zato što  $w_0$  određuje odsječak pravca (odnosno općenito hiperravnine) na  $y$ -osi. Ako predviđamo veličinu stopala s obzirom na dob, onda sigurno ne želimo imati  $w_0 = 0$ , jer bi to značilo da novorođenče ima veličinu stopala jednaku nuli.

9

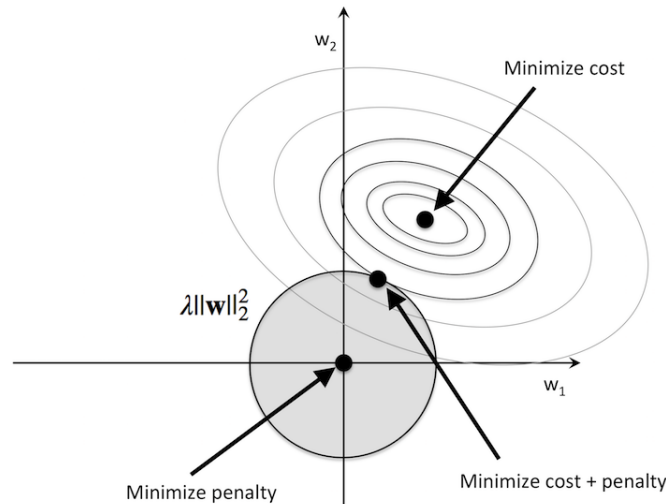
### 3.2 Regularizirani linearni model regresije

Pogledajmo sada kako regularizacija izgleda konkretno kod regresije. **L2-regularizacija** (ili Tikhonovljeva regularizacija) daje nam tzv. **hrbatnu regresiju** (engl. *ridge regression*):

$$E_R(\mathbf{w}|\mathcal{D}) = \frac{1}{2} \sum_{i=1}^N (\mathbf{w}^T \phi(\mathbf{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$$

Primijetite da koristimo L2-normu na kvadrat, i da imamo  $\lambda/2$ . To dvoje, zajedno sa  $1/2$  ispred sume, koristimo zbog kasnije matematičke jednostavnosti. L2-regularizacija kažnjava hipotezu onoliko koliko njezine težine odstupaju od nule u smislu kvadratnog odstupanja. To znači da će veće težine biti kažnjene više, a manje težine će biti kažnjene manje; ova spoznaja će nam biti od koristi nešto kasnije.

L2-regularizacija ima dosta intuitivnu geometrijsku interpretaciju. Kao što znamo, hipoteze su funkcije definirane parametrima  $\mathbf{w}$ . Posljedično, i pogreška  $E(\mathbf{w}|\mathcal{D})$  je funkcija parametara  $\mathbf{w}$ . Možemo pogledati kao izgleda ta funkcija u prostoru parametara:



Slika prikazuje izokonture (skup točaka u kojima funkcija poprima konstantnu vrijednost) funkcije pogreške  $E(\mathbf{w}|\mathcal{D})$  u prostoru parametara  $w_1 \times w_2$  (parametar  $w_0$  smo u prikazu radi preglednosti zanemarili). Eliptične konture u gornjem desnom dijelu slike odgovaraju izokonturama neregularizirane funkcije pogreške  $E(\mathbf{w}|\mathcal{D})$  definirane kao zbroj kvadrata reziduala. Ta je funkcija konveksna, i njezin je minimum u središtu izokontura. Kružnica u donjem lijevom dijelu slike odgovara izokonturi L2-regularizacijskog izraza, čiji je minimum u ishodištu (jer ondje je norma vektora  $\mathbf{w} = (0, 0)$  jednaka nuli). Regularizirana funkcija pogreške  $E_R(\mathbf{w}|\mathcal{D})$  dobiva se zbrojem ovih dviju krivulja. Zbroj dviju konveksnih funkcija daje opet konveksnu funkciju. Izokonture te konveksne funkcije nisu ucrtane u gornjoj slici, ali je ucrtan njezin minimum, kao točka između minimuma neregularizirane funkcije pogreške i minimuma L2-regularizacijskog izraza (ishodišta). Na regularizaciju možemo stoga gledati kao na privlačnu silu koja minimizator funkcije pogreške primiće bliže ishodištu prostora parametara.

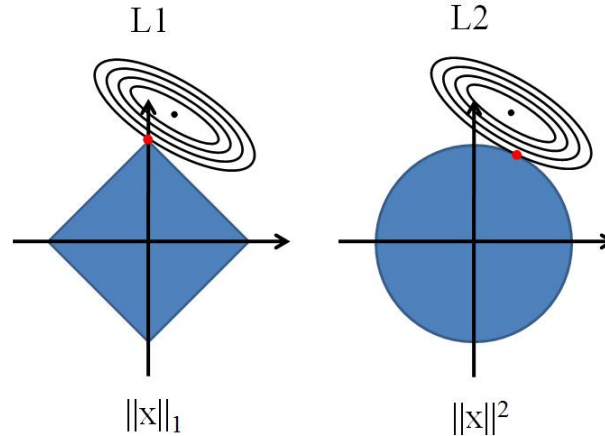
Umjesto L2-regularizacije, možemo odlučiti raditi **L1-regularizaciju** ili **LASSO regularizaciju** (engl. *least absolute shrinkage and selection operator*):

$$E_R(\mathbf{w}|\mathcal{D}) = \frac{1}{2} \sum_{i=1}^N (\mathbf{w}^T \phi(\mathbf{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2} \|\mathbf{w}\|_1$$

L1-regularizacija kažnjava hipotezu onoliko koliko njezine težine odstupaju od nule po apsolutnoj vrijednosti.

Koja je razlika između L2- i L1-regularizacije? Razlika se vidi kod malenih težina, koje su blizu nule. L1-regularizacija nema kvadrat, pa će te malene težine kažnjavati više od L2-regularizacije. S druge strane, L2-regularizacija će male težine kažnjavati vrlo malo, i zapravo će ih vrlo teško dotjerati skroz do nule. Zbog toga će L1-regularizacija općenito rezultirati modelima kod kojih je više težina pritegnuto baš na nulu, tj. L1-regularizacija rezultat će **rjeđim modelima** nego L2-regularizacija.

Razlika između L2- i L1-regularizacije može se protumačiti i grafički, u prostoru parametara:



Slika prikazuje L1-regularizaciju (lijevo) i L2-regularizaciju (desno). Izokonture L1-regularizacijskog izraza su općenito hiperoktaedar, odnosno u dvodimenzijskome prostoru parametara to je kvadrat. S druge strane, izokonture L2-regularizacijskog izraza su hipersfere, odnosno u dvodimenzijskome prostoru parametara to je kružnica. Slika prikazuje da će se, kod L1-regularizacije, zato što izokonture regularizacijskog izraza nisu oble nego imaju vrhove, lakše dogoditi da se minimizator regularizirane funkcije pogreške nađe na nekoj od koordinatnih osi prostora parametara, tj. u jednom od vrhova kvadrata (jer vrhovi su na osima). Nadalje, ako se minimizator nalazi na nekoj od osi prostora parametara, to onda znači da je druga koordinata (tj. težina) jednaka nuli. U trodimenzijskom prostoru parametara izokontura je oktaedar te, ako se minimum nađe u nekom od vrhova oktaedra, onda će druge dvije težine biti jednake nuli. Ako se minimum nađe na bridu oktaedra, onda će jedna težina biti jednaka nula. U višedimenzijskom prostoru, svaki puta kada se minimum nađe na vrhu ili na bridu hiperoktaedra, određeni broj težina će biti jednak nuli. U višedimenzijskom prostoru vjerojatnost da se to dogodi raste s porastom brojem dimenzija, jer hiperoakteader onda ima sve više vrhova i bridova. Iz toga onda zaključujemo da L1-regularizacija lakše dovodi do rijetkih modela (mnoge težine su pritegnute na nulu) nego L2-regularizacija.

Konačno, možemo razmišljati i o **L0-regularizaciji**, kod koje je regularizirana pogreška definirana ovako:

$$E_R(\mathbf{w}|\mathcal{D}) = \frac{1}{2} \sum_{i=1}^N (\mathbf{w}^T \phi(\mathbf{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^m \mathbf{1}\{w_j \neq 0\}$$

L0-regularizacija kažnjava hipotezu onoliko koliko ima ne-nul značajki. To znači da svaku značajku tretira kao da je ili uključena ili posve isključena. Time, efektivno, L0 regularizacija odabire najbolji podskup značajki, što znači da zapravo radi postupak **odabira značajki** (engl. *feature selection*).

Sada kada smo se upoznali sa L2, L1 i L0 regularizacijom, prirodno se nameće pitanje: koja regularizacija je najbolja? Odgovor je: u načelu preferiramo onu regularizaciju koja nam daje što rjeđe modele. To bi bila dakle L0, a onda L1, a onda tek L2. Međutim, problem u praksi predstavlja računalna složenost. Naime, L0-regularizacija je **NP-potpun problem** i nema traktabilno rješenje (treba isprobati sve kombinacije značajki, a njih je  $2^m$ ). S druge strane, L1-regularizacija je traktabilna, ali **nema rješenje u zatvorenoj formi**. Situacija je takva da, kod linearnog modela regresije, jedino L2-regularizacija ima rješenje u zatvorenoj formi.

Mi ćemo u nastavku pogledati L2-regulariziranu regresiju (odnosno hrbatnu regresiju), baš zato što njezina optimizacija ima analitičko rješenje. Hrbatna regresija vrlo često koristi u praksi. U praksi se također koristi i L1-regularizacija (koja nema analitičko rješenje, pa se optimizacija radi iterativno), ili kombinacija L1-regularizacije i L2-regularizacije, koja se zove **elastična mreža** (engl. *elastic net*). Ali o tome nećemo.

### 3.3 Hrbatna regresija: optimizacija

Kao što smo rekli, L2-regularizirana regresija, odnosno hrbatna regresija, ima rješenje u zatvorenoj formi. Lako ga je izvesti. Pogledajmo kako.

Prisjetimo se: ovako smo izveli rješenje za težine  $\mathbf{w}$  za neregularizirani model linearne regresije:

$$\begin{aligned} E(\mathbf{w}|\mathcal{D}) &= \frac{1}{2}(\Phi\mathbf{w} - \mathbf{y})^T(\Phi\mathbf{w} - \mathbf{y}) \\ &= \frac{1}{2}(\mathbf{w}^T\Phi^T\Phi\mathbf{w} - 2\mathbf{y}^T\Phi\mathbf{w} + \mathbf{y}^T\mathbf{y}) \end{aligned}$$

Zatim, da bismo našli minimum, izračunali smo gradijent ove funkcije, izjednačili ga s nulom, i iskazali  $\mathbf{w}$ :

$$\begin{aligned} \nabla_{\mathbf{w}}E &= \Phi^T\Phi\mathbf{w} - \Phi^T\mathbf{y} \\ \mathbf{w} &= (\Phi^T\Phi)^{-1}\Phi^T\mathbf{y} \end{aligned}$$

Sada ćemo raditi isto kao i prije, samo ćemo funkciji pogreške dodati regularizacijski faktor (označeno crvenom):

$$\begin{aligned} E_R(\mathbf{w}|\mathcal{D}) &= \frac{1}{2}(\Phi\mathbf{w} - \mathbf{y})^T(\Phi\mathbf{w} - \mathbf{y}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \\ &= \frac{1}{2}(\mathbf{w}^T\Phi^T\Phi\mathbf{w} - 2\mathbf{y}^T\Phi\mathbf{w} + \mathbf{y}^T\mathbf{y} + \lambda\mathbf{w}^T\mathbf{w}) \end{aligned}$$

Sada računamo gradijent funkcije, izjednačavamo ga s nulom, i iskazujemo  $\mathbf{w}$ :

$$\begin{aligned} \nabla_{\mathbf{w}}E_R &= \Phi^T\Phi\mathbf{w} - \Phi^T\mathbf{y} + \lambda\mathbf{w} \\ &= (\Phi^T\Phi + \lambda\mathbf{I})\mathbf{w} - \Phi^T\mathbf{y} = 0 \\ \mathbf{w} &= (\Phi^T\Phi + \lambda\mathbf{I})^{-1}\Phi^T\mathbf{y} \end{aligned}$$

Ovdje smo iz matricnog diferencijalnog računa iskoristili jednakost  $\frac{d}{dx}\mathbf{x}^T\mathbf{A}\mathbf{x} = \mathbf{x}^T(\mathbf{A} + \mathbf{A}^T)$ , gdje je  $\mathbf{A} = \mathbf{I}$ , a zatim  $\frac{d}{d\mathbf{w}}\mathbf{w}^T\mathbf{w} = 2\mathbf{w}^T$ .

Matrica  $\lambda\mathbf{I}$  je matrica dimenzija  $(m+1) \times (m+1)$  koja na dijagonali ima  $\lambda$ , a izvan dijagonale nule. Dodavanjem te matrice na Gramovu matricu efektivno se ostvaruje regularizacija. Međutim, prisjetimo se da težinu  $w_0$  želimo izuzeti iz regularizacije. To znači da je matrica  $\lambda\mathbf{I}$  zapravo definirana tako da je, pored elemenata izvan dijagonale, i gornji lijevi element jednak nuli, tj.  $\lambda\mathbf{I} = \text{diag}(0, \lambda, \dots, \lambda)$ .

Vidimo da se regularizacija lijepo “ugradila” u rješenje najmanjih kvadrata. Rješenje doduše više nije izravno pseudoinverz Gramove matrice  $\Phi$ , nego treba računati inverz zbroja Gramove matrice i matrice  $\lambda\mathbf{I}$ . Ovo je trivijalno proširenje u računalnom smislu. Međutim, zanimljivo je kako se regularizacija svela na dopunu vrijednosti na dijagonali Gramove matrice. I tu možemo ispričati jednu zanimljivu priču...

## 4 Regularizacija i multikolinearnost

Vratimo se malo na računanje inverza matrice dizajna  $\Phi$ , o kojemu smo kratko pričali prošli put. Rekli smo da inverz ne možemo računati direktno, jer se lako može dogoditi da broj primjera i broj parametara nisu jednaki, što znači da će sustav jednadžbi biti **preodređen ili pododređen**, odnosno da matrica dizajna neće biti kvadratna, pa da nema inverz. Štoviše, rekli smo da čak i ako matrica dizajna kojim čudom bude kvadratna, i dalje može biti da nema inverz, ako odgovarajući sustav jednadžbi nije konzistentan ili ako ima višestruka rješenja. Rješenje

smo onda našli u postupku najmanjih kvadrata, koji nas je vrlo brzo doveo do **pseudoinverza** matrice dizajna:

$$\mathbf{w} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y}$$

Rekli smo i da pseudoinverz uvijek postoji. Stoga, čak i ako pseudoinverz Gramove matrice ne možemo izračunati pomoću inverza matrice, možemo ga izračunati postupkom rastava na singularne vrijednosti (SVD).

Međutim, to ne znači da to uvijek želimo. Objasnimo o čemu se radi.

Prvo, pogledajmo kada jest moguće izračunati pseudoinverz Gramove matrice pomoću inverza matrice. Gramova je matrica dimenzija  $(m + 1) \times (m + 1)$ , tj. sigurno je kvadratna. Ali, da bismo je mogli invertirati, Gramova matrica također mora biti punog ranga. Kada će Gramova matrica biti punog ranga? Kao što smo spomenuli prošli put, rang Gramove matrice jednak je rang matrice dizajna. To znači da će Gramova matrica biti punog ranga ako i samo je rang matrice dizajna  $(m + 1)$ , tj. ako u matrici dizajna imamo  $(m + 1)$  linearno nezavisnih stupaca, odnosno ako su sve značajke **linearno nezavisne**. Nažalost, u praksi je čest problem upravo taj da su stupci linearno zavisni. Zašto dolazi do toga? Zato što skup podataka često ima **redundantne značajke**. Pogledajmo primjer.

#### ► PRIMJER

Radimo predviđanje osobnog prihoda, a kao značajke koristimo dob osobe u danima te kao drugu značajku dob osobe u godinama. No, te dvije značajke linearno su gotovo savršeno korelirane. Matrica dizajna  $\Phi$  u tom slučaju je, npr.:

$$\Phi = \begin{pmatrix} 1 & 9511 & 26 \\ 1 & 11340 & 31 \\ 1 & 8201 & 22 \\ 1 & 18022 & 49 \end{pmatrix}$$

Prvi stupac je “dummy” značajka  $x_0 = 1$ , drugi stupac je značajka  $x_1$  koja odgovara starosti osobe u broju dana, a treći stupac je značajka  $x_2$  koja odgovara starosti osobe u godinama. Drugi i treći stupac su linearno zavisni: treći stupac možemo dobiti iz drugog tako da drugi podijelimo sa 365.25 i zaokružimo na niži cijeli broj. Ovo nije savršena linearna zavisnost (jer zaokružujemo, tj. zato što je broj dana preciznija informacija nego broj godina), ali je dovoljno blizu savršenoj linearnoj zavisnosti da bi u praksi bila problematična. Ako ne bismo zaokruživali, već godine prikazali kao realan broj, imali bismo savršenu linearnu zavisnost drugog i trećeg stupca.

Ovakav fenomen zovemo **multikolinearnost**: dvije ili više ulaznih varijabli su korelirane, pa je neku varijablu moguće vrlo dobro predvidjeti kao linearnu kombinaciju jedne ili više drugih varijabli. Ekstremni slučaj multikolinearnosti je **savršena multikolinearnost**, kao u prethodnom primjeru (ako ne bismo zaokruživali broj godina). Savršena multikolinearnost, međutim, u praksi se rijetko događa. S druge strane, (nesavršena) multikolinearnost događa se razmjerno često. Npr., ako radimo predviđanje osobnog prihoda na temelju dobi i na temelju godina radnog staža, što su razumne značajke za ovaj problem, moglo bi se lako dogoditi da te dvije varijable budu visoko korelirane, što bi dovelo do multikolinearnosti.

Vratimo se sada na našu matricu dizajna. Što se događa ako je neka značajka savršena linearna kombinacija jedne ili više drugih značajki? To onda znači da matrica dizajna  $\Phi$  nema rang  $m + 1$ . Dalje, Gramova matrica (matrica skalarnih produkata)  $\Phi^T \Phi$  ima isti rang kao i matrica dizajna, što znači da ona neće imati puni rang. To onda znači da Gramova matrica nema inverz, tj. **singularna** je. Ako baš želimo, možemo izračunati pseudoinverz od  $\Phi$  SVD-om, no problem je da će naše rješenje tada biti vrlo **nestabilno**. Što mislimo pod time? Mislimo da je rješenje za  $\mathbf{w}$  biti **vrlo osjetljivo na promjene vrijednosti ulaznih varijabli**. Vrlo mala promjena ulaznih varijabli može dati vrlo veliku promjenu u težinama.

Što ako su značajke **multikolinearne**, ali nisu savršeno multikolinearne? Onda je matrica dizajna punog ranga, i Gramova matrica također, pa dakle nije singularna, i možemo izračunati inverz Gramove matrice, dakle možemo izračunati pseudoinverz pomoću inverza matrice. Međutim, trebamo znati da, premda nije singularna, Gramova je matrica tada vrlo blizu tome da bude singularna, pa će naše rješenje opet biti vrlo **nestabilno**.

Takva nestabilnost rješenja je tipično ponašanje koje opažamo kod **prenučenih hipoteza**. Kod regresije, to znači da ćemo za male promjene u označenom skupu podataka imati radikalno različite izlaze. Npr., ako koristimo polinom 10. stupnja, imat ćemo vrlo različite hipoteze ovisno o malim promjenama u ulazu. Pogledajte opet primjer s predviđanjem cijena nekretnina u Bostonu. Tamo smo imali polinom 6. stupnja, čija krivulja će jako divljati za male promjene u ulaznim podacima. To znači da će za male promjene u matrici dizajna hipoteza  $h(x; \mathbf{w})$  imati vrlo različite parametre  $\mathbf{w}$ . To je znak prenaučenosti modela.

Srećom, postoji način da detektiramo ovakvu nestabilnost rješenja. U numeričkoj matematici govorimo o **kondicijskom broju** matrice. Ovdje nemamo vremena ulaziti u detalje, ali je dovoljno da znate da kada je taj broj velik, onda je naše rješenje nestabilno i model je sigurno prenaučen. Za takvu matricu kažemo da je **loše kondicionirana** (engl. *ill-conditioned*).

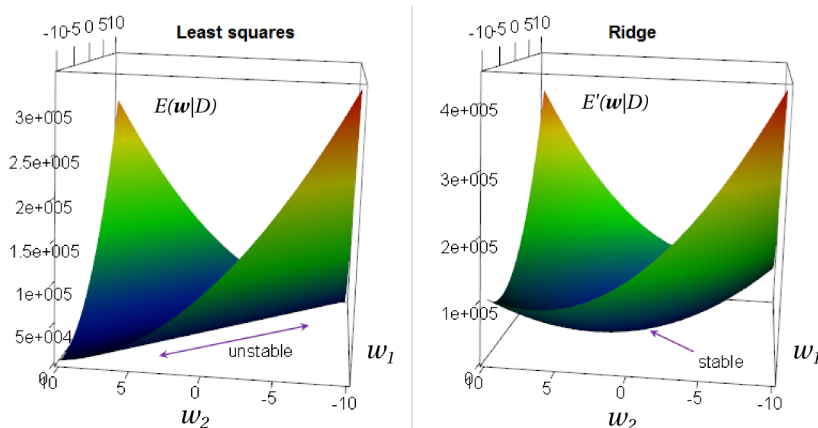
Mi svakako želimo izbjeći situaciju prenaučenosti modela. Primijetimo da, ako je model složen, a podataka je malo, dakle  $m \geq N$ , onda se lako dovodimo u situaciju prenaučenosti. Naime, onda će matrica dizajna biti **“široka i plitka”**, rang matrice će sigurno biti manji od  $(m + 1)$ , tj. bit će najviše  $N$ . To će onda ujedno biti i rang Gramove matrice, koja dakle neće biti punu ranga, pa sigurno imamo multikolinearnost!

Kako izbjeći multikolinearnost? Prvo, prije treniranja modela strojnog učenja uvijek je dobro analizirati korelacije između parova značajki te izbaciti one koje su redundantne. To je jedno rješenje. Drugo rješenje nam je već poznato: **regularizacija**! Pogledajmo, sada kada znamo da je problem pretreniranosti posljedica loše kondicioniranja Gramove matrice, kako regularizacija rješava taj problem. Konkretno, pogledajmo što regularizacija zapravo čini Gramovoj matrici:

$$\mathbf{w} = (\Phi^T \Phi + \lambda \mathbf{I})^{-1} \Phi^T \mathbf{y}$$

Vidimo da svakom stupcu Gramove matrice dodajemo vrijednost  $\lambda$  na drugom mjestu (tj. u drugom retku). Ako su stupci bili linearno zavisni ili skoro linearno zavisni, sada će biti manje linearno zavisni. Što je vrijednost  $\lambda$  veća, to je matrica koju invertiramo “sličnija” dijagonalnoj matrici, to smo dalje od singularne matrice, i to je manja multikolinearnost. U krajnosti, kada je matrica dominantno dijagonalna, sigurno nemamo multikolinearnosti. Dakle, regularizacija zapravo popravljala kondiciju Gramove matrice (smanjuje njezin kondicijski broj). Kažemo da smo matricu **rekondicionirali**.

Konačno, zanimljivo je pogledati koji je efekt rekondicioniranja matrice na oblik **funkcije pogreške** u prostoru parametara:





Slike prikazuju graf funkcije pogreške  $E(\mathbf{w}|\mathcal{D})$  regresije (definirane, sjetimo se, preko kvadratne funkcije gubitka) u prostoru parametara  $w_1 \times w_2$  (opet, radi preglednosti, tj. da bismo uopće mogli nacrtati graf, izostavili smo težinu  $w_0$ , no znamo da po njoj također optimiziramo, premda ju ne regulariziramo). Kao i uvijek, tražimo vektor težina  $\mathbf{w}$  koji minimizira pogrešku. Na lijevoj slici je neregularizirana empirijska pogreška, tj. funkcija koju minimiziramo postupkom običnih najmanjih kvadrata (OLS), dok je na desnoj slici regularizirana empirijska pogreška, tj. funkcija koju minimiziramo kod hrbatne regresije (a to odgovara kriteriju najmanjih kvadrata plus regularizacijski izraz). Kod neregularizirane pogreške vidimo da se minimum nalazi u dugačkoj udolini. Zapravo, događa se to da sve točke u toj udolini, koja ovdje na slici odgovara pravcu, imaju jednaku vrijednost, dakle imamo beskonačno mnogo točaka u kojima funkcija pogreške poprima minimum. Tu udolinu zovemo **“hrbat”**: područje nestabilnog rješenja, gdje je moguće beskonačno mnogo rješenja. Na desnoj slici vidimo regulariziranu funkciju pogreške. Efekt regularizacije, odnosno kondicioniranja matrice dizajna, jest taj da funkcija pogreške postaje “konveksnija”. Time se smanjuje nestabilnost rješenja, i zapravo dovodimo se u situaciju da postoji samo jedan minimum. Također, intuitivno je jasno da, što je konveksnija funkcija pogreške, tj. što jača regularizacija, to će rješenje biti stabilnije. I upravo to je razlog zašto L2-regularizirana regresija nazivamo **hrbatna regresija**: jer može raditi sa takvim “hrptovima” (odnosno udolinama).

13

Što je poanta ove priče? Jedna vrlo konkretna poanta jest da na regularizaciju možemo gledati kao na “popravljanje” Gramove matrice kojim se odmičemo od nestabilnih rješenja (prenaučenih hipoteza). Druga, šira poanta priče jest da u strojnom učenju na jednu te istu stvar često možemo gledati na više različitih, međusobno povezanih načina. Jedna od glavnih svrha ovog predmeta jest da vam to osvijestimo.

## 5 Napomene

Završit ćemo s nekoliko općenitih napomena o linearnom modelu regresije.

- Magnituda parametra  $w_i$  odgovara **važnosti značajke**, a predznak upućuje na njezin utjecaj (pozitivan ili negativan) na izlaznu vrijednost.
- Međutim, ako je model prenaučeni (postoji multikolinearne značajke), onda magnituda parametra ne znači ništa. U tom slučaju, dobro je koristiti regularizaciju.
- Regularizacija **spriječava prenaučenosť** na način da smanjuje složenost modela prigušujući težine pojedinih značajki, odnosno efektivno ih izbacuje (kada  $w_j \rightarrow 0$ ).
- Ako je model nelinearan, regularizacijom smanjujemo nelinearnost.
- Težinu  $w_0$  treba izuzeti iz regularizacijskog izraza (jer ona definira odsječak) ili treba centrirati podatke tako da  $\bar{y} = 0$ , jer onda  $w_0 \rightarrow 0$ .
- L2-regularizacija kažnjava težine proporcionalno njihovom iznosu (velike težine više, a manje težine manje). Zbog toga se teško događa da težine budu pritegnute baš na nulu, i zbog toga L2-regularizacija ne rezultira rijetkim modelima.
- L1-regularizirana regresija rezultira rijetkim modelima, ali nema rješenja u zatvorenoj formi (međutim, mogu se koristiti iterativni optimizacijski postupci).
- Regularizacija je korisna kod modela s puno parametara, jer je takve modele lako prenaučiti.
- Regularizacija smanjuje mogućnost prenaučenosťi i multikolinearnosti, ali ostaje problem odabira hiperparametra  $\lambda$ . Kao što već sigurno pogodate, taj se odabir najčešće radi **unakrsnom provjerom**. Što bismo kao optimalnu vrijednost za  $\lambda$  kada kada bismo optimizaciju radili na skupu za učenje? Odgovor je da bismo sigurno odabrali  $\lambda = 0$ , jer to daje najsloženiji model koji bi imao najmanju pogrešku na označenom skupu primjera za učenje. I to naravno ne bi bilo dobro. Zato ćemo  $\lambda$  odabrati unakrsnom provjerom.

14

## Sažetak

- **Linearan model regresije** linearan je u parametrima
- Nelinearnost regresijske funkcije ostvaruje se preslikavanjem iz ulaznog prostora u **prostor značajki** pomoću nelinearnih **baznih funkcija**
- Parametri linearnog modela uz kvadratnu funkciju gubitka imaju rješenje u zatvorenoj formi u obliku **pseudoinverza matrice dizajna**, neovisno o tome koje preslikavanju u prostor značajki koristimo
- **Regularizacija smanjuje prenaučенost** ugradnjom dodatnog izraza u funkciju pogreške kojim se kažnjava složenost modela
- **L2-regularizirana regresija** ima rješenje u zatvorenoj formi, te efektivno **rekondicionira** matricu dizajna odnosno **uklanja multikolinearnost** značajki

## Bilješke

- [1] Ovo predavanje je lepršavija varijanta poglavlja o regresiji iz (nedovršene) skripte: [http://www.fer.unizg.hr/\\_download/repository/SU-Regresija.pdf](http://www.fer.unizg.hr/_download/repository/SU-Regresija.pdf).
- [2] U jeziku numeričke matematike, uporaba polinomijalne funkcije za modeliranje nelinearnosti funkcije  $h(x; \mathbf{w})$  jest problem **polinomijalne interpolacije**. Jedna alternativa polinomijalnoj interpretaciji, a koja se vrlo često koristi u statistici, jest interpolacija pomoću **spline funkcija**. Spline funkcija je funkcija koja je definirana po dijelovima pomoću funkcije polinoma. Interpolacija pomoću spline funkcije daje slične rezultate kao i polinomijalna interpolacija, ali je rezultat stabilniji, u smislu da, za razliku od polinomijalne funkcije, spline funkcije ne dovode do oscilacija pri rubovima interpolacijskog intervala (tzv. **Rungeov fenomen**), što je u stvari manifestacija **prenaučенosti modela**. U statistici se dominantno koriste **ograničene (prirodne) kubne spline funkcije** (engl. *restricted (natural) cubic splines*), koje su po dijelovima definirane kao polinom trećeg stupnja, uz dodatnu ogradu da je spline funkcija linearna prije prvog segmenta i na kraju zadnjeg segmenta. Više ovdje: <http://fourier.eng.hmc.edu/e176/lectures/ch7/node6.html>.
- [3] Ideja da umjesto modela mijenjamo podatke podsjeća na priču o plavom i ružičastom slonu: Kako upucati plavog slona? Puškom za plave slonove. Kako upucati ružičastog slona? Zadaviti slona dok ne poplavi, onda ga upucati puškom za plavog slona. (Niti jedan slon nije nastradao u ovoj lošoj šali.) Manje nasilan primjer: utrkujete se, imate fiću i loš je u zavojima. Umjesto da pređete na bolid, ostanite u fići, ali izravajte cestu.
- [4] Nameće se pitanje: možemo li preslikavati u prostor niže dimenzije, tj. da je  $m < n$ ? Možemo, naravno. Metode **redukcije dimenzionalnosti** (koje nećemo raditi na ovom predmetu), poput PCA i MSD, zapravo preslikavaju u prostor niže dimenzije, s ciljem da u tom prostoru podatci budu bolje objašnjivi i/ili da model bude manje sklon prenaučенosti. Također, možemo preslikavati u prostor značajki koji je iste dimenzije kao i ulazni prostor (tj.  $m = n$ ); takav primjer imat ćemo u zadacima za vježbu na temu **jezgrenih funkcija** (V10, zadatak 2), koje služe za implicitno preslikavanje u prostor značajki. Međutim, ako je naš cilj ostvariti nelinearnost (tj. nelinearnu granicu između klasa kod klasifikacije, odnosno nelinearnu regresijsku funkciju kod regresije), premda je moguće da nelinearnim preslikavanjem u prostor iste ili manje dimenzije ostvarimo željenu nelinearnost, vjerojatnije je da ćemo to ostvariti ako preslikavamo u prostor veće dimenzije. Ova je intuicija iskazana **Coverovim teoremom**, v. [https://en.wikipedia.org/wiki/Cover%27s\\_theorem](https://en.wikipedia.org/wiki/Cover%27s_theorem). Ako vas zanimaju koji su drugi važni teoremi strojnog učenja, pogledajte ovu raspravu: <https://stats.stackexchange.com/q/321851/93766> (na ovom predmetu spomenut ćemo, bez ulaska u detalje, tri od ovdje navedenih teorema).
- [5] Dakle, pazite: **model linearne regresije**  $\neq$  **linearna regresija**. Vidjet ćemo tijekom semestra da je strojno učenje puno takvih terminoloških mina kojima je lako zbuniti protivnika. Svako znanstveno područje ima svojih terminoloških muka, a pogotovo ona koja u kojima se stapaju utjecaji više drugih



područja i istraživačkih zajednica, kao što je to slučaj sa strojnim učenjem. No, kad god postoji mogućnost terminološke zabune, mi ćemo to posebno istaknuti.

- [6] Primijetite da, kada govorimo o dimenziji ulaznog prostora ili prostora značajki, “dummy” značajku  $x_0$  odnosno  $\phi_0(\mathbf{x})$  ne brojimo kao zasebnu dimenziju. Tako, naprimjer, kažemo da preslikavanje  $\phi(\mathbf{x}) = (1, x_1, x_2, x_1x_2, x_1^2, x_2^2)$  inducira 5-dimenzijski (a ne 6-dimenzijski) prostor značajki. Ovo je pitanje konvencije. Naime, ako je model definiran kao  $h(\mathbf{x}) = w_0 + w_1x_1 + \dots + w_nx_n$ , onda je on zapravo **afina funkcija** u  $\mathbb{R}^n$ . No, svaka se afina funkcija u  $\mathbb{R}^n$  može napisati kao **homogena linearna funkcija** u  $\mathbb{R}^{n+1}$  primijenjena na transformaciju koja dodaje “dummy” jedinicu vektoru značajki  $\mathbf{x}$ , i onda imamo  $h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ . O homogenim funkcijama: [https://en.wikipedia.org/wiki/Homogeneous\\_function](https://en.wikipedia.org/wiki/Homogeneous_function).
- [7] Prema **Lagrangeovom interpolacijskom teoremu**, za skup  $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$  od  $N$  primjera postoji jedinstven polinom  $h(x)$  stupnja ne većeg od  $N - 1$  za koji vrijedi  $h(x^{(i)}) = y^{(i)}$  za  $i = 1, \dots, N$ , tj. taj polinom prolazi kroz svaku točku iz  $\mathcal{D}$  (v. [https://en.wikipedia.org/wiki/Polynomial\\_interpolation#Interpolation\\_theorem](https://en.wikipedia.org/wiki/Polynomial_interpolation#Interpolation_theorem)). U primjeru s nekretninama iz Bostona, za ovaj konkretan skup od 8 točaka dovoljan je već polinom stupnja 6.
- [8] Dobar uvod u **bayesovsku regresiju** možete naći u Bishopovoj knjizi, poglavlju 3.3 (Bishop, 2006). Štovateljima R-a mogao bi se svidjeti i poglavlje o bayesovskoj regresiji iz online knjige (Clyde et al., 2018), dostupno na <https://statswithr.github.io/book/introduction-to-bayesian-regression.html>. Standardna referenca za bayesovske statističke metode općenito je (Gelman et al., 2013). Za kraći uvod u **bayesovski odabir modela**, pogledajte (Bishop, 2006), poglavlje 3.4, ili [https://www.cse.wustl.edu/~garnett/cse515t/fall\\_2019/files/lecture\\_notes/7.pdf](https://www.cse.wustl.edu/~garnett/cse515t/fall_2019/files/lecture_notes/7.pdf). Vrlo dobar pregled postupaka daju (Hooten and Hobbs, 2015).
- [9] Alternativno, možemo centrirati podatke (npr., standardizacijom), tako da ih dovedemo do ishodišta, i onda je  $w_0 = 0$ .
- [10] **Savršena multikolinearnost** događa se kada radimo na sirovim podacima, koji često koriste redundantne značajke. Kada se redundantne značajke uklone, često se događa da su preostale varijable (nesavršeno) multikolinearne (zbog inherentnih korelacija u podacima).
- [11] Matematički gledano, dodajemo lambda na **svojstvene vrijednosti** Gramove matrice, pa će sve svojstvene vrijednosti biti različite od nule, pa onda matrica neće biti singularna. Tehnički gledano, može se dogoditi da matrica nije imala multikolinearnosti, ali da smo upravo dodavanjem lambda na dijagonalu doveli matricu do pogrešne kondicioniranosti. To je moguće, ali u praksi dosta malo vjerojatno.
- [12] Uz manje izmjene, slika sam preuzeo sa <https://stats.stackexchange.com/a/151351/93766>.
- [13] U geologiji, hrbat (greben) je izbočeni i izduženi dio brda. Budući da mi tražimo minimum a ne maksimum, zapravo umjesto hrpta imamo “negativan hrbat” iliti udolinu. Usput, ako vas muči kao što muči mene, ovdje je deklinacija riječi “hrbat”: [http://hjp.znanje.hr/index.php?show=search\\_by\\_id&id=fVxvWhQ%3D](http://hjp.znanje.hr/index.php?show=search_by_id&id=fVxvWhQ%3D).
- [14] Pažljivi čitatelj ovdje će možda primijetiti moguću cirkularnost: regularizaciju smo predstavili kao tehniku za sprječavanje prenaučivosti koja je alternativa unakrsnoj provjeri, i kod koje se složenost modela prilagođava podacima, no sada ispada da ipak treba unakrsna provjera da bismo optimizirali hiperparametar  $\lambda$  (regularizacijski faktor). Točno je da nam unakrsna provjera ipak treba. Međutim, to ne umanjuje korisnost regularizacije. Naime, optimizacijom hiperparametra  $\lambda$  mi modelu dopuštamo da svoju složenost više ili manje prilagodi podacima, ali mu i dalje dajemo slobodu na koji način da se prilagodi (koje težine više a koje manje pritegnuti nuli). U tom smislu, odabir hiperparametra  $\lambda$ , za razliku od izravnog odabira modela, je na neki način “soft” odabir modela, gdje ostavljamo mogućnost da se složenost modela bolje prilagodi podacima. Druga prednost regularizacije nad izravnom optimizacijom hiperparametara vidi se u situacijama kada model ima više hiperparametara: tada nam je lakše optimizirati samo jedan “master” hiperparametar (regularizacijski faktor  $\lambda$ ) nego više hiperparametara odjednom.

## Literatura

- C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.
- M. Clyde, M. Ç. Rundel, C. Rundel, D. Banks, C. Chai, and L. Huang. An introduction to bayesian thinking, 2018.
- A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian data analysis*. CRC press, 2013.
- M. B. Hooten and N. T. Hobbs. A guide to bayesian model selection for ecologists. *Ecological Monographs*, 85(1):3–28, 2015.