2. Osnovni koncepti

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

Jan Šnajder, predavanja, v2.2

1 Osnovni koncepti

Na uvodnom predavanju motivirali smo strojno učenje i ukratko predstavili osnovne pristupe: **nadzirano i nenadzirano učenje**. Danas, kao i u narednih deset tjedana, fokusirat ćemo se na nadzirano strojno učenje (engl. *supervised machine learning*). [1]

Danas ćemo uvesti niz zanimljivih ideja i novih pojmova, i tako "postaviti scenu" za ono što slijedi idući tjedan. Ovo je predavanje malo apstraktno; taj problem pokušat ćemo ublažiti ubacivanjem ponešto primjera kako bi stvari postale plastičnije. Međutim, kako ćemo se na ove koncepte puno puta vraćati na svakom predavanju tijekom semestra, do kraja semestra dobit ćete vrlo dobar osjećaj za sve ove stvari o kojima ćemo danas pričati, pa ako danas ne sjedne sve na svoje mjesto, nemojte očajavati.

Glavni koncepti koje ćemo danas uvesti su **hipoteza**, **model**, **pogreška i funkcija gubitka**, **prenaučenost**, **unakrsna provjera** i **odabir modela**. Glavni glumac ovdje je **model**, koji će se stalno pojavljivati u svemu što ćemo raditi.

2 Primjena algoritma strojnog učenja

Za početak, razmotrimo širu sliku: koja ja tipična kuharica za primjenu strojnog učenja? Algoritme strojnog učenja tipično primjenjujemo kroz sljedećih osam koraka:

1. **Priprema podataka i preliminarna analiza** – kao i u statistici, prije nego što išta radimo, bitno je dobro se upoznati s podatcima i razumijeti s kakvim podatcima radimo;

2. **Ručno označavanje podataka za učenje i ispitivanje** – ovo nije uvijek potrebno, jer nekada su podatci već označeni, dok nekada radimo s algoritmima koji uopće ne trebaju označene podatke (nenadzirano strojno učenje);

3. **Ekstrakcija značajki** – na ulaz algoritma strojnog učenja dovode se podatci, bilo označeni (nadzirano strojno učenje) ili neoznačeni (nenadzirano strojno učenje), u obliku skupa primjera. Svaki je primjer opisan kao vektor značajki, odnosno ključnih karakteristika koje su indikativne za klasifikaciju/regresiju sličnih, budućih primjera. U ovom koraku potrebno je osmisliti na koji način prikazati primjer kao skup značajki te implementirati postupke ekstrakcije značajki. U većini slučajeva ovo je najkreativniji korak; [2]

4. **Redukcija dimenzionalnosti** – ni ovo nije uvijek potrebno, nego samo ako imamo vrlo mnogo značajki i ako nam to iz nekog razloga predstavlja problem (npr., ako model radi loše jer imamo previše značajki a premalo primjera za učenje, ili ako zbog previše značajki gubimo mogućnost da shvatimo što se zapravo događa i da interpetiramo odluke modela); [3]

5. **Odabir modela** (engl. *model selection*) – ovo je važan korak, koji početnici često ili propuste napraviti ili ga naprave pogrešno, pa ćemo ovome posvetiti mnogo paženje;

6. **Učenje modela** – ovdje algoritam odrađuje svoj posao a mi trebamo samo pričekati da se učenje završi (to nekad traje sekundu-dvije, nekad više sati, a nekad i više tjedana, ovisno o algoritmu i podatcima);
7. **Vrednovanje modela** – kada je model naučen, želimo znati kako dobro radi, jer je model koji dobro radi naš glavni cilj u strojnom učenju. Postoji mnogo načina kako se model može vrednovati, od kojih su mnogi dobri a neki su pogrešni. Naš cilj će biti da vas naučimo kako ispravno vrednovati model;
8. **Dijagnostika i ispravljanje** (engl. *diagnostics and debugging*) – ako model ne radi (a uglavnom ne radi; iluzorno je očekivati da će stvar raditi isprve), treba znati kako debagirati model i natjerati da radi;
9. **Instalacija** (engl. *deployment*) – konačno, kada sve radi dobro, model se može ugraditi u produkciju gdje će odrađivati svoj posao.

Naš fokus na ovom predmetu bit će koraci 5–7. To su koraci koji su zapravo dosta isprepleteni i u praksi se više puta ponavljaju, sve dok ne izgradimo model s kojim smo zadovoljni, odnosno koji je dovoljno dobar za zadani zadatak.

Koracima 1–3 nećemo se baviti jer su ti koraci povezani s ekspertizom iz konkretne domene (ako radite podatkovnu znanost, morat ćete se time baviti, ali i onda je dobro, a nekad i nužno, da tu imate pomoć domenskog stručnjaka). Korak 4 je vrlo zanimljiv i koristan, ali nam ne stane u ovaj predmet. Korak 9 može biti netrivijalan (npr. ako algoritam treba raditi s velikim količinama podataka, javljaju se raznorazni algoritamski izazovi), no taj korak pripada više programskom inženjerstvu, pa se ni njime nećemo baviti.

3 Primjeri, hipoteza, model

3.1 Prostor primjera

Kao i svaki algoritam, tako i algoritam strojnog učenja ima svoj ulaz i svoj izlaz. Na ulazu algoritma strojnog učenja – bilo da tek učimo model ili koristimo već naučeni model za predviđanje – nalazi se **primjer** (engl. *example, instance*). Primjer je jedna podatkovna točka (npr. jedna osoba, jedan putnik, jedna kuća, itd.), dakle jedan redak u našoj bazi podataka za koji želimo da model napravi neko predviđanje (klase, ako radimo klasifikaciju, ili brojčane vrijednosti, ako radimo regresiju).

Svaki ćemo primjer prikazati kao vektor značajki. **Značajka** (engl. *feature*) jest neka karakteristika primjera koja je bitna (ili mi mislimo da bi mogla biti bitna) za problem koji rješavamo, tj. to je neko svojstvo o kojemu ovisi klasifikacija primjera (kod klasifikacije) ili njegova ciljna vrijednost (kod regresije). Npr., ako radimo klasifikaciju putnika Titanica (hoće li putnik preživjeti), onda bi značajke bile dob putnika, socio-ekonomski status, spol, itd. Značajke nekada (pogotovo u dubinskoj analizi podataka) nazivamo **atributi**. Dakle, svaki primjer bit će okarakteriziran s nizom značajki (atributa), i to ćemo formalizirati kao **vektor značajki** (engl. *feature vector*):

$$\mathbf{x} = (x_1, x_2, \ldots, x_n)$$

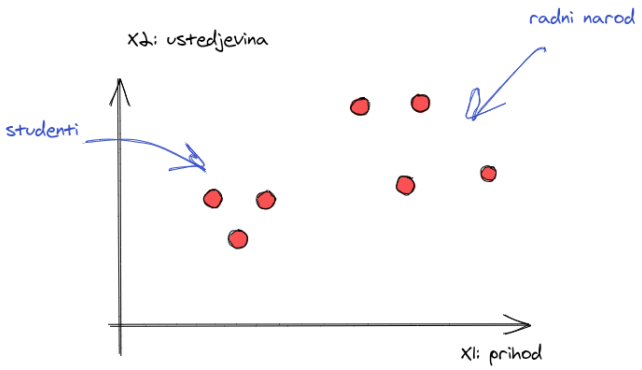
gdje su x_i pojedine značajke (atributi), a n je ukupan broj značajki (atributa). Svaki primjer će imati isti broj značajki. Premda to ne mora uvijek biti tako, mi ćemo si često pojednostaviti život i pretpostaviti da su sve značajke numeričke, tj. $x \in \mathbb{R}$, pa onda $\mathbf{x} \in \mathbb{R}^n$. Vidjet ćemo kasnije da je ta pretpostavka sasvim u redu, jer se numeričkim značajkama mogu kodirati i nenumeričke značajke.

Primjeri žive u **prostoru primjera** ili **ulaznom prostoru** (engl. *instance space, input space*). Taj prostor označavat ćemo s $\mathcal{X}$. Ako su značajke numeričke, tj.$x_i \in \mathbb{R}$, onda je $\mathcal{X} = \mathbb{R}^n$, tj. to je n-dimenzijski vektorski prostor. Njegova dimenzija je n ona odgovara broju značajki.

Dimenzionalnost n će u našim jednostavnim primjerima biti $n = 2$ ili $n = 3$ (samo zato jer nam je takav prostor primjera moguće vizualizirati), međutim u praksi ćemo (željeti) imati puno više značajki, 100 ili 1000, nekad više tisuća ili stotine tisuća, ovisno o problemu. Npr., ako radimo strojno učenje u bininformatici, računalnom vidu ili obradi prirodnoga jezika, lako ćemo imati na desetke tisuća značajki.

Dakle, ponovimo: svaki primjer je jedan vektor u ulaznom prostoru, $\mathbf{x} \in \mathcal{X}$, odnosno jedna točka u tom vektorskom prostoru.

▸ **PRIMJER**



Razmotrimo prostor primjera (ulazni prostor) za problem klasifikacije kreditno sposobnih klijenata banke. Banka treba odlučiti kome će dati kredit. Pojednostavimo problem i pretpostavimo da banke to rade na temelju samo dvije značajke: prihod i ušteđevina. Onda je ulazni prostor dvodimenzijski ($x_1 \rightarrow$ prihod i $x_2 \rightarrow$ ušteđevina), a svaki klijent banke je jedan dvodimenzijski vektor u tom prostoru. Studenti nažalost imaju niske prihode i malo ušteđevine, dok oni koji uživaju plodove dužeg rada imaju naravno veće prihode i veće ušteđevine, pa su bankama draži.

3.2 Označeni primjeri

Kod nadziranog učenja, svaki primjer ima svoju **oznaku** (engl. *label*). To će biti oznaka klase (kod klasifikacije) ili ciljna brojčana vrijednost (kod regresije). Oznaku primjera označit ćemo sa y, a skup svih mogućih oznaka u skupu podataka označit ćemo sa $\mathcal{Y}$.

Za klasifikaciju u K klasa imamo $\mathcal{Y} = \{0, \ldots, K-1\}$. Kada imamo samo dvije klase, $K = 2$, govorimo o **binarnoj klasifikaciji**. Tipično, tada $\mathcal{Y} = \{0, 1\}$ ili $\mathcal{Y} = \{-1, +1\}$, već kako nam je (matematički) jednostavnije. Primjer za koji $y = 0$ ili $(y = -1)$ nazivamo **negativan primjer**, dok primjer za koji $y = 1$ nazivamo **pozitivan primjer**. Za regresiju najopćenitiji (i tipičan) slučaj jest $\mathcal{Y} = \mathbb{R}$, tj. oznake su realni brojevi.

Jedan primjer nam naravno neće biti dovoljan za strojno učenje. Naprotiv, što više primjera, to bolje. Općenito, raspolažemo skupom primjera. Ukupan **broj primjera** označit ćemo sa N. Primijetite da koristimo veliko N za broj primjera, a malo n za broj značajki. Općenito, ta dva broja bit će različita. Tipično, iz razloga koji će postati jasni kasnije, voljeli bismo da primjera imamo više nego značajki, tj. $N > n$, a idealno da ih imamo mnogo više od značajki, tj. $N \gg n$.

Konačno, **skup označenih primjera** (engl. *labeled datataset*) označit ćemo sa $\mathcal{D}$. Formalno, $\mathcal{D}$ je skup parova:

$$\mathcal{D} = \left\{(\mathbf{x}^{(i)}, y^{(i)})\right\}_{i=1}^{N} \subseteq \mathcal{X} \times \mathcal{Y}$$

U strojnom učenju često će nam biti lakše koristiti matrični zapis podataka. Tako skup označenih primjera $\mathcal{D}$ možemo prikazati pomoću dvije komponente: matrica (neoznačenih)

primjera $\mathbf{X}$ i vektor njihovih oznaka $\mathbf{y}$. Ta matrica i taj vektor izgledaju ovako:

$$\mathbf{X} = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ & \vdots & & \\ x_1^{(N)} & x_2^{(N)} & \cdots & x_n^{(N)} \end{pmatrix} \qquad \mathbf{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{pmatrix}$$

Matrica $\mathcal{D}$ sastavljena je od matrice $\mathbf{X}_{N\times n}$ i vektora $\mathbf{y}_{N\times 1}$, tj. $\mathcal{D} = (\mathbf{X}, \mathbf{y})$. Retci matrice $\mathbf{X}$ odgovaraju primjerima, a stupci odgovaraju značajkama. Dakle, svaki redak matrice $\mathbf{X}$ je jedan vektor značajki odnosno primjer, $\mathbf{x}^{(i)}$. S druge strane, svaki stupac matrice $\mathbf{X}$ je jedna značajka kroz sve primjere. Primijetite da indeks primjera pišemo u superskriptu (iznad simbola $\mathbf{x}$), a indeks značajke u supskriptu (ispod simbola $\mathbf{x}$). Npr., x_1^2 je prva značajka (od njih ukupno n) drugog primjera (od njih ukupno N). Analogno, komponente vektora y odgovaraju oznaci za svaki pojedinačan primjer. Pretpostavka je, naravno, da su retci matrice $\mathbf{X}$ i komponente vektora $\mathbf{y}$ poravnate, tj. $y^{(1)}$ je oznaka za primjer $\mathbf{x}^{(1)}$, $y^{(2)}$ je oznaka za primjer $\mathbf{x}^{(2)}$, itd.

Inače, matrica $\mathbf{X}$ naziva se **matrica dizajna** (engl. *design matrix*). Ovaj naziv pogotovo se često koristi u statistici.

3.3 Hipoteza

Svrha nadziranog strojnog učenja jest naučiti funkciju koja primjerima iz $\mathcal{X}$ dodijeljuje oznake iz $\mathcal{Y}$. Drugim riječima, svrha je naučiti preslikavanje iz $\mathcal{X}$ u $\mathcal{Y}$. Ta funkcija naziva se **hipoteza**. Označavat ćemo je sa h. Dakle h je:

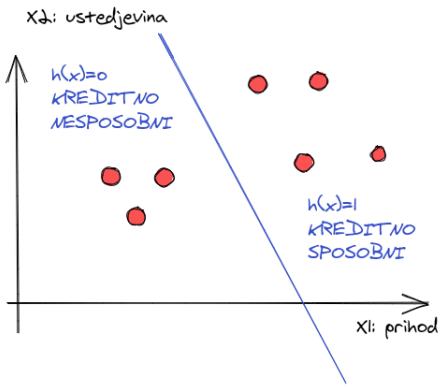
$$h : \mathcal{X} \to \mathcal{Y}$$

Ponovimo ovo: h je funkcija koja svakom primjeru (iz prostora primjera) dodjeljuje oznaku klase (kod klasifikacije) ili brojčanu vrijednost (kod regresije). Konkretno, za binarnu klasifikaciju hipoteza je:

$$h : \mathcal{X} \to \{0, 1\}$$

Ova funkcija zapravo dijeli prostor ulaznih primjera na dva poluprostora: jedan za koji je $h(\mathbf{x}) = 0$ i drugi za koji je $h(\mathbf{x}) = 1$. Pogledajmo primjer.

▸ **PRIMJER**



U dvodimenzijskome ulaznom prostoru hipotezu h mogli bismo definirati tako da odgovara pravcu u ravnini. Taj pravac razdjeljuje dvodimenzijski prostor primjera na dva poluprostora. Poluprostor u kojem $h(\mathbf{x}) = 0$ odgovara kreditno nesposobnim klijentima banke (niski prihodi i niska ušteđevina), a poluprostor u kojem $h(\mathbf{x}) = 1$ odgovara onim kreditno sposobnima (visoki prihodi i visoka ušteđevina). Klasifikacija novih primjera (novih klijenata) bit će određena time s koje strane pravca će primjer sletjeti. Primijetite da ovako definirana hipoteza klasificira sve moguće primjere u $\mathcal{X} = \mathbb{R}^2$, a ne samo ovih sedam koje imamo ucrtane.

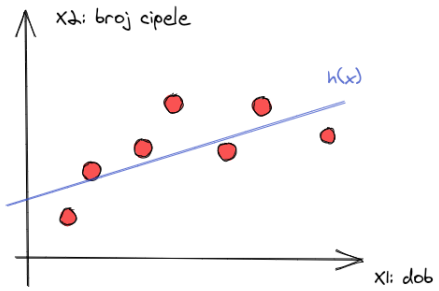
Općenito, mi nećemo unaprijed znati koja je točna funkcija h. Kada bismo to znali, ne bi nam trebalo strojno učenje. Međutim, znat ćemo otprilike – ili, ako ne znamo, nekako ćemo morati odlučiti – što su mogući kandidati za funkciju h. Možda se sjećate da smo u uvodu rekli da je zadatak algoritma strojnog učenja podesiti **parametre** modela, a da je model funkcija **definirana do na parametre**. Evo sad što to točno znači: mi ćemo imati funkciju h koja će biti definirana s nekim parametrima, koje ćemo označavati sa θ. To pišemo ovako:

$$h(\mathbf{x}; \boldsymbol{\theta})$$

(Koristimo ';' kako bismo odvojili argument funkcije koji varira od argumenta funkcije koji je više-manje konstantan. Ova razlika je samo konvencija; formalno, naravno, oba su argumenta varijable dotične funkcije.)

Funkcija h parametrizirana je parametrima θ. Općenito, to neće biti samo jedan parametar, nego niz parametara, tako da zapravo govorimo o **vektoru parametara $\boldsymbol{\theta}$**. U primjerima koji slijede vektor parametara sastojat će se od parametara koji realni brojevi. Međutim, općenito vektor parametara može sadržavati parametre različitog tipa (npr., matrice, vektore, skalare, i razne kombinacije tih tipova). Pogledajmo sada dva primjera.

▸ **PRIMJER**



Želimo predvidjeti broj (tj. veličinu) cipele s obzirom na dob osobe. Kakav je to problem: regresijski ili klasifikacijski? To je regresijski problem, jer želimo predvidjeti brojčanu vrijednost, a ne klasu. Naša hipoteza je funkcija $h : \mathcal{X} \to \mathbb{R}$, a ulazni prostor je jednodimenzijski, $n = 1$, jer imamo samo jednu značajku (dob osobe). Hipotezu bismo mogli definirati ovako:

$$h(x; \theta_0, \theta_1) = \theta_1 x + \theta_0$$

Ovime smo zapravo definirali **pravac**. To je u redu, jer očekujemo (donekle) **linearnu ovisnost** između dobi i veličine cipela.

Funkcija h (a ovdje je to jednadžba pravca) ima dva parametra, θ_1 i θ_0, odnosno naš vektor značajki je $\boldsymbol{\theta} = (\theta_0, \theta_1)$. Jedan parametar ($\theta_1$) određuje nagib pravca, a drugi (θ_0) odsječak na osi y. Trebaju nam oba parametra (odsječak na osi y nam treba jer tek rođena osoba s praktički 0 godina nema broj cipele 0).

▸ **PRIMJER**

Ovo je bio primjer za regresiju. Pogledajmo sada primjer za **klasifikaciju**. Vratimo se opet na klasifikaciju kreditne sposobnosti, na temelju značajki prihoda i uštedevine. Dakle, radimo klasifikaciju u 2-D ulaznom prostoru. Kreditno sposobni su oni koji imaju ili velika primanja ili veliku uštedu.

Recimo da se odlučimo da te dvije klase želimo odvojiti **pravcem**, kao u ranijem primjeru. Taj pravac možemo napisati pomoću **implicitne jednadžbe pravca**:

$$\theta_1 x_1 + \theta_2 x_2 + \theta_0 = 0$$

Općenito, primjeri će se nalaziti s jedne ili s druge strane tog pravca. Dakle, pravac dijeli ulazni prostor na dva poluprostora. Primjeri na "pozitivnoj strani" pravca su oni za koje vrijedi:

$$\theta_1 x_1 + \theta_2 x_2 + \theta_0 \geqslant 0$$

a primjeri na "negativnoj strani pravca" su oni za koje vrijedi:

$$\theta_1 x_1 + \theta_2 x_2 + \theta_0 < 0$$

(Primjeri koji leže točno na pravcu mogu se tretirati kako god; ovdje smo ih uključili u pozitivnu stranu.)

Za primjere koji pozitivnoj strani pravca želimo da hipoteza vrati 1, a za primjere koji su na negativnoj želimo da vrati 0 (ili -1, već kako odlučimo, ali odlučimo se ovdje za 0). To možemo ostvariti ako hipotezu definiramo na sljedeći način:

$$h(x_1, x_2; \theta_0, \theta_1, \theta_2) = \mathbf{1}\{\theta_1 x_1 + \theta_2 x_2 + \theta_0 \geqslant 0\}$$

Ovdje smo upotrijebili funkciju $\mathbf{1} : \{\bot, \top\} \to \{0, 1\}$, koja samo pretvara Booleove vrijednosti u 0 ili 1. Ta je funkcija definirana ovako:

$$\mathbf{1}\{P\} = \begin{cases} 1 & \text{ako } P \equiv \top \\ 0 & \text{inače} \end{cases}$$

Ovime smo definirali hipotezu $h : \mathcal{X} \to \{0, 1\}$, koja interno provjerava s koje se strane pravca primjer našao te ovisno o tome daje 0 ili 1. Granica između ta dva slučaja, tj. između dviju klasa, zapravo odgovara pravcu definiranom implicitnom jednadžbom $\theta_1 x_1 + \theta_2 x_2 + \theta_0 = 0$.

Primijetimo ovdje da imamo tri parametra, $\boldsymbol{\theta} = (\theta_0, \theta_1, \theta_2)$, dok smo u prethodnom primjeru s regresijom imali samo dva. No, prethodni primjer je imao jednodimenzijski ulazni prostor (y-os odgovarala je oznaci), dok ovaj primjer ima dvodimenzijski ulazni prostor (oznaka ne odgovara niti jednoj od osi već je implicitna u poziciji primjera u odnosu na pravac). 4

3.4 Model

U strojnom učenju neprestano pričamo o modelima. Sada možemo demistificirati što je to zapravo. **Model** nije ništa drugo nego **skup hipoteza**. Budući da su hipoteze funkcije, to zapravo znači da je model **skup funkcija**. Označit ćemo ga sa $\mathcal{H}$.

Dakle, model je jedna "vreća" koja sadrži različite hipoteze h. Na primjer, ako je h definiran kao u gornjem primjeru, onda je model skup svih mogućih ravnina u 3-D prostoru. Ili to može biti skup svih hiperravnina u 4-D prostoru. Ili, ako radimo regresiju, skup svih pravaca, ili skup svih krivulja, itd.

Formalno ćemo model definirati ovako:

$$\mathcal{H} = \{h(\mathbf{x}; \boldsymbol{\theta})\}_{\boldsymbol{\theta}}$$

Model $\mathcal{H}$ je, dakle, skup hipoteza h, koje su parametrizirane sa $\boldsymbol{\theta}$, a supskript $\boldsymbol{\theta}$ znači da su elementi tog skupa indeksirani parametrom $\boldsymbol{\theta}$. To pak znači da svaki vektor $\boldsymbol{\theta}$ odgovara jednoj funkciji iz skupa $\mathcal{H}$. Drugim riječima, za zadani vektor parametara $\boldsymbol{\theta}$ možemo dohvatiti njemu odgovarajuću funkciju $h \in \mathcal{H}$. To jest, $\boldsymbol{\theta} \mapsto h$ (θ se preslikava u h, odnosno θ jednoznačno određuje h). Prema tome, model je skup funkcija **parametriziranih (tj. indeksiranih)** s $\boldsymbol{\theta}$. 5

Sada kada sve ovo znamo, evo zanimljivog (i istinitog) pogleda na strojno učenje: **učenje (treniranje modela)** nije ništa drugo nego **pretraživanje** skupa hipoteza $\mathcal{H}$ u nastojanju da se nađe najbolja hipoteza $h \in \mathcal{H}$.

Ujedno se odmah prirodno postavlja sljedeće pitanje: što je nabolja hipoteza? Intuitivno je jasno da bi **najbolja hipoteza** bila ona koja najtočnije klasificira primjere (kod klasifikacije)

odnosno ona daje vrijednosti najbliže ciljnim brojčanim vrijednostima (kod regresije). Budući da se tu onda radi o traženju po nekom kvantitativnom kriteriju dobrote, sada vidimo da je učenje zapravo **optimizacijski problem**. Iz tog je razloga strojno učenje općento vrlo povezano s područjem optimizacije. 6

U stvarnim primjenama, skup $\mathcal{H}$ će biti vrlo velik. Što to znači? To znači da u modelu $\mathcal{H}$ postoji puno mogućih hipoteza h. Zbog toga će nam trebati nekakva **heuristička optimizacija**, koja će pametno pretraživati taj skup, umjesto da ga pretražujemo iscrpno.

Također, primjetite na ovom mjestu da je bilo vrlo mudro što smo parametrizirali funkciju h, odnosno što smo model definirali kao skup parametriziranih funkcija, $\mathcal{H} = \{h(\mathbf{x}; \boldsymbol{\theta})\}_{\boldsymbol{\theta}}$. Naime, da to nismo napravili, i da smo funkcije ostavili sasvim proizvoljnima te da smo rekli da je model skup baš svih mogućih funkcija, onda ne bi postojao način da pretražujemo po tom skupu, pa ne bismo imali po čemu optimizirati. Čak i da smo nekako drugačije ograničili skup mogućih funkcija, opet to ne bi bilo dobro ako funkcije ne bi bile parametrizirane. Ovako, kada su funkcije indeksirane vektorom parametara $\boldsymbol{\theta}$, možemo pretraživati po $\boldsymbol{\theta}$, jer nam svaka promjena u $\boldsymbol{\theta}$ može dati neku drugu funkciju. Zanimljivo je također primijetiti da to što smo parametrizirali funkcije znači da smo ograničili skup mogućih funkcija, ali da to ne znači nužno da je skup $\mathcal{H}$ konačan (npr., možemo se ograničiti na skup hipoteza koje odgovaraju pravcima, no takvih funkcija i dalje može biti beskonačno mnogo).

4 Empirijska pogreška i funkcija gubitka

4.1 Empirijska pogreška

Vratimo se na pitanje određivanja "najbolje hipoteze". Očito nam treba neka **numerička procjena** koliko je hipoteza dobra, a na temelju koje ćemo provoditi optimizaciju. Nas bi zapravo zanimalo koliko bi hipoteza radila dobro na **svim mogućim** primjerima. Međutim, nikada nećemo imati na raspolaganju sve moguće primjere, jer onda ne bismo trebali strojno učenje. Umjesto toga, mjerit ćemo koliko je hipoteza dobra na **skupu označenih primjera** koje imamo na raspolaganju. Ideja je onda da izmjerimo koliko dobro hipoteza radi na tom skupu, tj. u kojoj se mjeri izlaz hipoteze podudara s točnim oznakama. Mjera koliko je hipoteza dobra (odnosno loša) na označenom skupu podataka koji imamo na raspolagajnu naziva se **empirijska pogreška**.

Empirijska pogreška govori nam koliko hipoteza griješi kad klasificira primjere (klasifikacija) ili koliko su vrijednosti daleko od ciljnih vrijednosti (regresija). Pogrešku nazivamo **empirijska** jer je mjerimo na konkretnom skupu podataka (zapravo na statističkom uzorku koji imamo na raspolaganju). Nas zapravo zanima prava pogreška hipoteze na svim mogućim primjerima, ali, naravno, to ne možemo znati jer nemamo sve moguće primjere, pa dakle koristimo empirijsku pogrešku kao procjenu prave pogreške.

Empirijsku pogrešku hipoteze h na skupu označenih primjera $\mathcal{D}$ označit ćemo sa:

$$E(h|\mathcal{D})$$

Ova notacija ima za svrhu naglasiti da je empirijska pogreška E funkcija hipoteze h za neki fiksirani skup označenih primjera $\mathcal{D}$. To jest, za fiksirani skup označenih primjera $\mathcal{D}$, različite hipoteze h mogu dati različite vrijednosti empirijske pogreške. (Oznaku '|' koristit ćemo i inače kako bismo razdvojili argumente funkcije koji su varijable od onih za koje u nekom kontekstu prodrazumijevamo da su konstantni.) Puni naziv za ovu funkciju je **funkcija empirijske pogreške**.

Kako bismo točno definirali empirijsku pogrešku? Tu imamo više mogućnosti. Za klasifikaciju često koristimo **pogrešku klasifikacije** (engl. *misclassification error*):

$$E(h|\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{h(\mathbf{x})^{(i)} \neq y^{(i)}\}$$

Ovo se svodi na to da jednostavno pobrojimo koliko je primjera, od njih ukupno N, za koje hipoteza h daje oznaku koja je drugačija od prave oznake $y^{(i)}$(funkciju $\mathbf{1}\{\cdot\}$ definirali smo u prethodnom primjeru), a onda taj broj podijelimo s ukupnim brojem primjera kako bismo dobili udio netočno klasificiranih primjera.

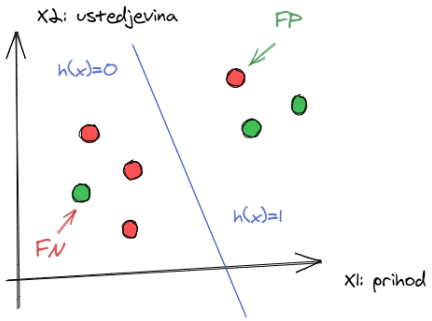
Specifično, za binarnu klasifikaciju s oznakama $\mathcal{Y} = \{0, 1\}$ funkciju pogreške mogli bismo definirati i ovako:

$$E(h|\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} |h(\mathbf{x})^{(i)} - y^{(i)}|$$

Klasifikator će općenito griješiti, tj. empirijska pogreška bit će veća od nule. Kod binarne klasifikacije, klasifikator može griješiti na dva načina: može pozitivan primjer (onaj za koje je $y = 1$) klasificirati kao negativan, ili može negativan primjer (onaj za koji je $y = 0$) klasificirati kao pozitivan. Prvi slučaj, kada klasifikator pozitivan primjer klasificira kao negativan, zove se **lažno negativan** primjer (engl. *false negative, FN*). Drugi slučaj, kada klasifikator negativan primjer klasificira kao pozitivan, zove se **lažno pozitivan** primjer (engl. *false positive, FP*). Zbroj lažno pozitivnih i lažno negativnih primjera u skupu $\mathcal{D}$ daje nam ukupan broj pogrešaka klasifikatora, a kada taj broj podijelimo sa ukupnim brojem primjera N, dobivamo empirijsku pogrešku, odnosno udio pogrešno klasificiranih primjera. Primjere na kojima klasifikator nije pogriješio također možemo stvrstati u dvije skupine: pozitivni primjeri koje je klasifikator ispravno klasificirao kao pozitivni su **istinito pozitivni** (engl. *true positive, TP*), a negativni primjeri koje je klasifikator klasificirao kao negativni su **istinito negativni** (engl. *true negative, TN*).

▸ **PRIMJER**

Razmotrimo opet primjer binarne klasifikacije kreditno sposobnih klijenata banke. Pozitivna klasa (ona za koju $y = 1$) neka odgovara kreditno sposobnim klijentima. Recimo da smo definirali linearan model (kao u ranijem primjeru), naučili ga na skupu $\mathcal{D}$ koji se sastoji od 7 označenih primjera, te dobili ovakvu granicu između dvije klasa u ulaznom prostoru:



Prisjetimo se, hipoteza $h(\mathbf{x})$ ovdje odgovara pravcu te naš dvodimenzijski ulazni prostor dijeli na dva poluprostora, s granicom u točkama za koje $h(\mathbf{x}) = 0$. Zeleni primjeri neka su oni koji su u skupu $\mathcal{D}$ označeni kao pozitivni ($y = 1$), a crveni neka su oni koji su označeni kao negativni ($y = 0$). Na slici vidimo da naš klasifikator na nekim primjerima griješi. Konkretno, klasifikator je tri primjera klasificirao kao pozitivna (gornji desni dio ulaznog prostora), ali je jedan od njih zapravo negativan. Taj primjer je lažno pozitivan (FP). Također, klasifikator je četiri primjera klasificirao kao negativna (doljnji lijevi dio ulaznog prostora), ali je jedan od tih primjera zapravo pozitivan, i to je onda lažno negativan primjer (FN). Ukupno je klasifikator od sedam primjera njih dva pogrešno klasificirao, pa je empirijska pogreška hipoteze h na skupu $\mathcal{D}$ jednaka:

$$E(h|\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{h(\mathbf{x})^{(i)} \neq y^{(i)}\} = \frac{1}{7}(1 + 0 + 0 + 0 + 1 + 0 + 0) = \frac{2}{7} = 0.286$$

odnosno 28.6%. Posljedično, **točnost** klasifikatora je $1 - 0.286 = 0.714$ odnosno 71.4%. Usput,

primijetite da s modelom definiranim tako da odgovara pravcu na ovakvom skupu $\mathcal{D}$ niti nismo mogli ostvariti $E(h|\mathcal{D}) = 0$. Više o tome kasnije.

4.2 Funkcija gubitka

Primijetimo da svaki pojedinačni primjer iz skupa označenih primjera može doprinijeti empirijskoj pogrešci. Npr., kod klasifikacije, svaki će primjer imati doprinos pogrešci koji će biti jednak 0 ili 1, ovisno je li dotični primjer ispravno klasificiran. Iznos pogreške načinjene na pojedinačnom primjeru (funkcija unutar sume) izračunava **funkcija gubitka** (engl. *loss function*). Za dani označeni primjer, funkcija gubitka govori nam koliko je model izgubio na točnosti (tj. dobio na ukupnoj pogrešci) na tom jednom primjeru. Ako je vrijednost funkcije gubitka za neki primjer jednaka nuli, to znači da model ispravno klasificira taj primjer i da taj primjer ne doprinosi ukupnoj pogrešci modela. U gornjoj definiciji empirijske pogreške koristili smo funkciju gubitka $\mathbf{1}\{h(\mathbf{x})^{(i)} \neq y^{(i)}\}$. Ta se funkcija gubitka zove se **gubitak nula-jedan** (engl. *zero-one loss*). Ima i mnogo drugih funkcija gubitaka, kao što ćemo vidjeti u narednim tjednima.

Formalno, funkciju gubitka definirat ćemo kao $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$. To je funkcija koja uzima dvije oznake, jednu točnu i jednu dobivenu kao izlaz hipoteze, te izračunava gubitak (koji je nenegativan realan broj). Konkretno, pisat ćemo $L(y, h(\mathbf{x}))$, gdje je y točna oznaka primjera $\mathbf{x}$, a $h(\mathbf{x})$ je oznaka koju daje za taj primjer daje hipoteza h.

Kako je funkcija gubitka povezana s empirijskom funkcijom pogreške? Veza je ta da ćemo empirijsku funkciju pogreške u većini slučajeva definirati indirektno, preko funkcije gubitka. Jednostavno ćemo izračunati srednju vrijednost funkcije gubitka na skupu označenih primjera. Malo općenitije, reći ćemo da je funkcija pogreške **očekivana vrijednost funkcije gubitka** na svim mogućim primjerima iz $\mathcal{X} \times \mathcal{Y}$, a mi to očekivanje aproksimiramo empirijski kao srednju vrijednost funkcije gubitka na označenom skupu primjera. [7]

Da sažmemo: u modelu $\mathcal{H}$ (koji je skup hipoteza) postoje različite funkcije h. Kada te funkcije primijenimo na označene primjere $\mathcal{D}$, neke će funkcije davati točniju a neke manje točnu klasifikaciju. Koliko je klasifikacija na skupu označenih primjera točna govori nam empirijska pogreška, koja je srednja vrijednost funkcije gubitka na skupu označenih primjera. Kada kažemo da učimo (treniramo) model, mi zapravo pretražujemo model $\mathcal{H}$ kako bismo pronašli hipotezu s najmanjom empirijskom pogreškom na skupu označenh primjera $\mathcal{D}$.

5 Tri komponente algoritma strojnog učenja

Spojimo sada zajedno sve ovo što smo naučili. Bit će nam od velike koristi da svaki, baš svaki algoritam strojnog učenja analiziramo u smislu **tri glavne komponente**. Te komponente su: [8]

1. **Model**

   $$\mathcal{H} = \left\{h(\mathbf{x}; \boldsymbol{\theta})\right\}_{\theta}$$

   Prisjetimo se, model je skup funkcija (hipoteza) parametriziranih vektorom parametara $\boldsymbol{\theta}$.

2. **Funkcija gubitka** odnosno njoj pripadna **funkcija pogreške**

   $$E(h|\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} L(y^{(i)}, h(\mathbf{x}^{(i)}))$$

   S obzirom da parametri $\boldsymbol{\theta}$ jednoznačno određuju funkciju h, tj. $\boldsymbol{\theta} \mapsto h$, to znači da možemo pisati i:

   $$E(\boldsymbol{\theta}|\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} L(y^{(i)}, h(\mathbf{x}^{(i)}; \boldsymbol{\theta}))$$

tj. možemo govoriti o pogrešci hipoteze h ili o pogrešci parametara $\boldsymbol{\theta}$ koji definiraju tu istu hipotezu. U drugom slučaju napisali smo $h(\mathbf{x}^{(i)}; \boldsymbol{\theta})$ kako bismo dodatno naglasili da je funkcija h parametrizirana sa $\boldsymbol{\theta}$ (ona je to uvijek, no nekada to nije potrebno posebno istaknuti).

3. **Optimizacijski postupak**

   To je postupak kojim unutar modela $\mathcal{H}$ nalazimo hipotezu h^* koja minimizira empirijsku pogrešku:

   $$h^* = \underset{h \in \mathcal{H}}{\operatorname{argmin}} E(h|\mathcal{D})$$

   što je, zato što $\boldsymbol{\theta}$ jednoznačno određuju h, istovjetno nalaženju optimalnih parametara $\boldsymbol{\theta}^*$: 9

   $$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} E(\boldsymbol{\theta}|\mathcal{D})$$

   Za hipotezu h^* nekad ćemo koristiti naziv "naučeni/trenirani model". Slično, za parametre $\boldsymbol{\theta}^*$ koristit ćemo naziv "parametri modela" (premda bi točnije bilo "vrijednosti parametara naučenog/treniranog modela").

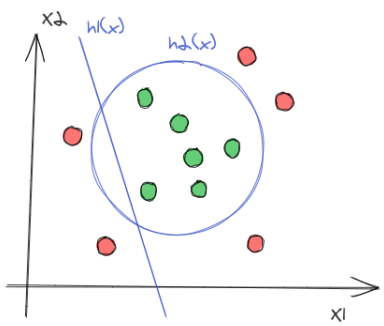
Primijetite da smo ove tri komponente definirali doista vrlo apstraktno. Model smo definirali kao skup hipoteza, ali nismo točno rekli kako su hipoteze parametrizirane. Pogrešku smo definirali kao srednju vrijednost funkcije gubitka, a nismo točno rekli kako je definirana funkcija gubitka. I optimizacijski postupak definirali smo najapstraktnije moguće, kao funkciju koja pronalazi hipotezu (odnosno, ekvivalentno, parametre) koji minimiziraju empirijsku pogrešku. To je namjerno, jer želimo da ove tri komponente budu okvir u koji možemo staviti svaki algoritam strojnog učenja. U tjednima koji slijede, kako budemo uvodili pojedine algoritme strojnog učenja, za svaki ćemo algoritam konkretno specificirati ove tri komponente. Takav unificirani tretman omogućit će nam da bolje uočimo sličnosti i razlike između svih tih algoritama. Zato svakako upamtite ove tri komponente!

6 Složenost modela

U idealnom slučaju, u modelu (skupu hipoteza) $\mathcal{H}$ postoji hipoteza h čija je empirijska pogreška jednaka nula, $E(h|\mathcal{D}) = 0$. No, lako je moguće (i u praksi je to redovito slučaj) da takva h ne postoji, tj. $\forall h \in \mathcal{H}.E(h|\mathcal{D}) > 0$. Tada kažemo da model $\mathcal{H}$ nije dovoljne **složenosti** ili **kapaciteta** za naš problem (definiran našim skupom označenih primjera $\mathcal{D}$).

▸ **PRIMJER**

Želimo naučiti (trenirati) binaran klasifikator za primjere iz skupa $\mathcal{D}$, koji su ulaznome prostoru raspoređeni ovako (zeleni primjeri su pozitivni, crveni su negativni):



(Na primjer, ako je x_1 prihod, a x_2 ušteđevina, ovaj skup označenih primjera mogao bi odgovarati problemu u kojemu banka, možda za neku svoju posebnu ponudu, želi identificirati klijente koji nemaju niti preveliki prihod niti preveliku ušteđevinu.) Razmatramo dva modela: $\mathcal{H}_1$ je skup svih hipoteza koje odgovaraju pravcima (kao u ranijem primjeru), a $\mathcal{H}_2$ je skup svih hipoteza koje odgovaraju kružnicama (s podesivim središtem i radijusom; neka je model definiran tako da su svi primjeri koji se nalaze unutar kružnice pozitivno klasificirani). Na skupu $\mathcal{D}$, niti jedna hipoteza h iz $\mathcal{H}_1$ ne može doseći $E(h|\mathcal{D}) = 0$ (hipoteza $h \in \mathcal{H}_1$ s najmanjom pogreškom ostvaruje $E(h|\mathcal{D}) = \frac{3}{11}$). To znači da model $\mathcal{H}_1$ nije dovoljne složenosti (kapaciteta) za problem definiran skupom $\mathcal{D}$. (Ovdje je to zato jer skup $\mathcal{D}$ nije linearno odvojiv, pa linearan model, definiran pravcem, ne može odvojiti linearno neodvojive primjere). S druge strane, u modelu $\mathcal{H}_2$ postoje neke hipoteze koje mogu savršeno točno klasificirati primjere iz $\mathcal{D}$, tj. $\exists h \in \mathcal{H}_2.E(h|\mathcal{D}) = 0$. Jedna takva hipoteza ucrtana je u gornjoj slici. Stoga zaključujemo da je model $\mathcal{H}_2$ dovoljne složenosti (kapaciteta) za klasifikacijski problem definiran skupom označenih primjera $\mathcal{D}$.

Odmah se postavlja pitanje: je li ovo uopće problem? Zašto jednostavno ne definiramo složeniji model, takav da u $\mathcal{H}$ postoji hipoteza h čija je empirijska pogreška jednaka nuli? Međutim, kao što možete pretpostaviti, u stvarnosti stvari ipak nisu tako jednostavne.

Naime, htjeli mi to ili ne, u našim podatcima (u skupu označenih primjera) uvijek (osim u idealiziranim akademskim primjerima) postoji **šum** (engl. *noise*). Šum je neželjena anomalija u podacima zbog koje dolazi do odstupanja opaženih vrijednosti od pravih vrijednosti, bilo u značajkama pojedinih primjera $\mathbf{x}$ ili njihovim oznakama y. U oba slučaja rezultat će općenito biti taj da će oznake nekih primjera u našem skupu označenih primjera biti netočne. Naravno, problem je u tome što mi ne možemo sa sigurnošću utvrditi koji su to primjeri.

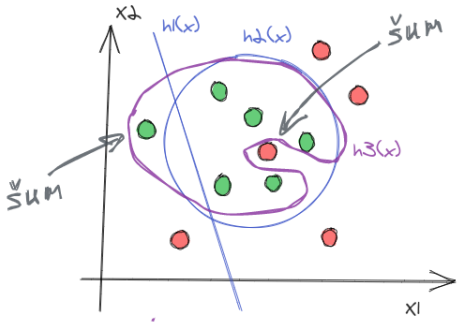
Mogući su razni uzroci šuma u podatcima:

- **Nepreciznost** pri mjerenju značajki (pogotovo ako su značajke neke fizikalne veličine, npr. visina, težina);
- **Pogreške u označavanju** – takozvani **teacher noise** – npr., zamolimo ljude da nam označe podatke, a oni su nešto krivo shvatili, pa krivo označavaju, ili tu i tamo nisu koncentrirani, npr. zato jer rade još nešto paralelno sa označavanjem (npr. prebacuju se između označavanja i Facebooka);
- Postojanje **skrivenih značajki** (latentnih varijabli) – npr., klasificiramo tko će preživjeti potonuće Titanica, ali iz nekog razloga zaboravili smo na značajku starosne dobi. Bez te značajke, izgledat će nam kao da su podatci puni šuma, a zapravo nisu, nego nam nedostaje jedna dimenzija;
- **Nejasne granice** između klasa (subjektivnost) – problem je inherentno subjektivan, i različiti ljudi imaju različito mišljenje koja je točna klasifikacija. (npr., klasifikacija govora mržnje u porukama na Twitteru).

Što god da je uzrok šuma, njegova je posljedica ta da će granica između pozitivnih i negativnih primjera (ili, općenito, granica između primjera iz različitih klasa) u ulaznom prostoru biti složenija nego što bi ona bila da šuma nema, odnosno složenija nego što stvarno jest.

▸ **PRIMJER**

Zamislimo da u skupu $\mathcal{D}$ iz prethodnog primjera postoji šum takav da su neki primjeri pogrešno označeni. Npr., ovako:



Dva primjera su pogrešno označena: jedan pozitivan (zeleni) primjer označen je negativno (crveno), a jedan negativan primjer (crveni) označen je pozitivno (zeleno). U stvarnosti, oznake primjera su onakve kakve su bile u prethodnom primjeru, no sada su, zbog šuma, neke oznake pogrešne. Kao što vidimo, sada više niti model $\mathcal{H}_2$ nije dovoljne složenosti (kapaciteta) da na skupu $\mathcal{D}$ ostvari empirijsku pogrešku jednaku nuli. Želimo li ostvariti savršenu klasifikaciju na $\mathcal{D}$, treba nam neki još složeniji model. Neka je to model $\mathcal{H}_3$, koji odgovara proizvoljno zakrivljenim regijama u ulaznom prostoru. Nazovimo taj model kolokvijalno "model krumpira". Model krumpira je dovoljno složen za ovaj problem i sadrži hipotezu $h_3 \in \mathcal{H}_3$ koja na $\mathcal{D}$ daje savršenu klasifikaciju (odnosno čija je empirijska pogreška jednaka nuli).

Pitanje je, naravno, želimo li koristiti tako složen model. Naime, u stvarnosti su pozitivni primjeri zapravo grupirani zajedno unutar kružnice, ali samo zbog šuma izgleda kao da to nije tako. U tom smislu, ispravna klasifikacija je zapravo ona koju daje hipoteza h_2 iz modela $\mathcal{H}_2$ (kružnica). S druge strane, model $\mathcal{H}_3$ (krumpir) je presložen, pa je zajedno s ispravnom klasifikacijom naučio i onu neispravnu uslijed šuma. Problem je, dakle, što se presložen model može previše prilagoditi šumu. To je problem jer takav model neće dobro generalizirati (v. idući primjer).

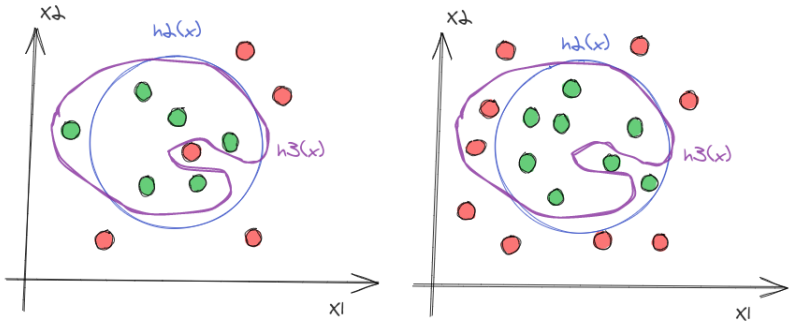
Vidimo da, ako postoji šum, jednostavan model (npr., pravci) ne može doseći $E(h|\mathcal{D}) = 0$. S druge strane, ako je model suviše složen, on će se previše prilagoditi podatcima, pa će naučiti i šum, a ne pravu klasifikaciju bez šuma. Ovdje smo razmatrali primjere u dvodimenzijskome ulaznom prostoru. Situacija je mnogo gadnija u ulaznim prostorima visoke dimenzije, gdje će složeni model imati priliku još se više prilagoditi podatcima odnosno šumu u njima.

Bit problema je dakle ova: prejednostavni modeli imat će previsoku empirijsku pogrešku, dok će presloženi modeli vjerojatno imati nisku empirijsku pogrešku, ali će se previše prilagoditi šumu. Kako god, rezultat je da će takvi modeli **loše generalizirati**. Prisjetimo se, generalizacija je ključno svojstvo modela strojnog učenja: želimo da model radi dobro na primjerima koje nikada nije vidio tijekom treniranja. Prejednostavan model ne može dobro generalizirati jer općenito radi loše (ima visoku empirijsku pogrešku već i na podatcima na kojima je treniran), dok presložen model ne može dobro generalizirati jer se previše prilagodio šumu, a šum će u neviđenim podatcima sigurno biti drugačiji od onoga u podatcima na kojima je model treniran. Situaciju kada za neki klasifikacijski ili regresijski problem koristimo prejednostavan model nazivamo **podnaučenost**. Obratno, situaciju u kojoj koristimo presložen model nazivamo **prenaučenost**.

- **Podnaučenost** (engl. *underfitting*) – model $\mathcal{H}$ je prejednostavan u odnosu na stvarnu klasifikaciju/funkciju, što rezultira lošim predikcijama modela i na viđenim i na neviđenim primjerima;
- **Prenaučenost** (engl. *overfitting*) – model $\mathcal{H}$ je previše složen u odnosu na stvarnu klasifikaciju/funkciju, što rezultira dobrim predikcijama na viđenim primjerima ali lošim predikcijama na neviđenim primjerima.

▸ **PRIMJER**

Vratimo se opet na primjer s pravcima , kružnicama i krumpirima. Za model pravaca ($\mathcal{H}_1$) očigledno je da je taj model prejednostavan, budući da niti na viđenim primjerima ne može ostvariti nisku empirijsku pogrešku (pravac naprosto ne možemo saviti oko kružno raspoređenih primjera). Dakle, model odgovara situaciji podnaučenosti. S druge strane, model krumpira ($\mathcal{H}_3$) je presložen, pa će ga biti lako prenaučiti. To znači da hipoteza iz $\mathcal{H}_3$ može točno klasificirati sve viđene primjere (i tako ostvariti empirijsku pogrešku jednaku nuli), ali neće točno klasificirati mnoge neviđene primjere. To je zato što se hipoteze iz $\mathcal{H}_3$ previše prilagođavaju šumu, a šum na neviđenim primjerima neće biti isti. Razmotrimo sljedeću situaciju:



Na lijevoj strani je slika ulaznog prostora za naš označeni skup primjera $\mathcal{D}$, na kojem se model $\mathcal{H}_3$ prenaučio (prilagodio se dvama neispravno označenim primjerima). Na desnoj strani je slika istog tog ulaznog prostora, ali s nekim još neviđenim primjerima, na kojima modeli $\mathcal{H}_2$ i $\mathcal{H}_3$ nisu učeni. Budući da je šum stohastičke prirode, na tim neviđenim primjerima šum će vrlo vjerojatno biti drugačiji nego na skupu $\mathcal{D}$. To znači da će hipoteza h_3, dobivena modelom $\mathcal{H}_3$ koji je presložen i zato sklon prenaučenosti, pogrešno klasificirati neke (moguće mnoge) neviđene primjere. S druge strane, hipoteza h_2 iz modela kružnice $\mathcal{H}_2$, koji nije dovoljno složen da se previše prilagodio šumu na skupu $\mathcal{D}$, točnije će klasificirati neviđene primjere.

Zaključujemo, dakle, da je u ovom slučaju model $\mathcal{H}_1$ prejednostavan i rezultira podnaučenim hipotezama, model $\mathcal{H}_3$ je presložen i rezultira prenaučenim hipotezama, dok bi model $\mathcal{H}_2$ bio baš taman, ni prejednostavan ni presložen.

Očito, idealan model bio bi onaj koji nije niti prejednostavan (tako da se ne može podnaučiti) niti presložen (tako da se ne može prenaučiti). A ako pored ovoga možemo ili trebamo birati, dajemo prednost što jednostavnijim modelima. Naime, jednostavni modeli imaju niz prednosti nad složenim modelima: 10

- **Bolja generalizacija** – jednostavniji model ima veću šansu da će dati bolje predviđanja na još neviđenim podatcima;
- **Lakše učenje/uporaba** – jednostavniji model ima manje parametara, pa ga je lakše (brže) naučiti, i lakše ga je (brže) koristiti za predikciju;
- **Lakše tumačenje** – jednostavniji je model lakše tumačiti (usporedite model s tri parametra koji opisuje implicitnu jednadžbu pravca i neuronsku mrežu sa 10k parametara). Ovo je bitno ako želimo razmumijeti podatke i moći objasniti klasifikacijsku odluku modela, a ne samo napraviti sustav koji radi vrlo dobru predikciju, ali zapravo funkcionira kao crna kutija. 11

Naravno, očito je pak da model ne smije biti prejednostavan, jer prejednostavan model ne može opisati ni podatke koje imamo, a kamoli još neviđene podatke.

Trebamo, dakle, odabrati model koji točno odgovara **pravoj složenosti** problema koji nastojimo naučiti. To nas dovodi do vrhunca današnjeg predavanja: kako, u neizbježnom prisustvu šuma u podatcima, odabrati model optimalne složenosti, odnosno model s najmanjom pogreškom generalizacije? Ispostavlja se da je to sveprisutan problem u strojnom učenju, poznat

pod nazivom **odabira modela** (engl. *model selection*). Pogledajmo to malo detaljnije. 12

7 Odabir modela

Dakle, moramo odabrati optimalan model $\mathcal{H}$, optimalan u smislu da se neće ni podnaučiti ni prenaučiti. Taj odabir često radimo unutar neke **familije modela**. Budući da je model $\mathcal{H}$ skup, znači familija modela je zapravo skup skupova, npr:

$$\{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_k\}$$

Primijetite da smo ovdje na **jednoj razini više** nego što smo bili ranije, kada smo pričali o modelu: sada smo na razini skupa modela (skup skupova funkcija), dok smo prije bili na razini jednog modela (skup funkcija).

Razmotrimo neke primjere odabira modela. Na primjer, kod klasifikacije, možemo odabrati hoće li granice između klasa biti pravci, kružnice ili nekakve krivulje još veće nelinearnosti. Slično, kod regresije možemo birati stupanj polinoma: počevši od pravca preko parabole pa sve do neke visoko nelinearne krivulje. Zapravo, kada biramo model, često upravo biramo koliki će biti **stupanj nelinearnosti**. Onda modele možemo posložiti u linearni uređaj s obzirom na složenost: od manje složenih do vrlo složenih modela. Npr., kod regresije: pravac, parabola, polinom trećeg stupnja, polinom četvrtog stupnja, itd.

Ako složenost modela možemo nekako kvantificirati i dovesti u vezu s nekim parametrom koji upravlja tom složenošću, onda takav parametar nazivamo **hiperparametar modela**. Npr., stupanj nelinearnosti polinoma je hiperparametar regresijskog modela. Zašto je to hiperparametar a ne parametar? Zato što to nije običan parametar modela. Obični parametri su oni koji čine vektor $\boldsymbol{\theta}$, po kojemu algoritam strojnog učenja radi minimizaciju empirijske pogreške i koji određuju hipotezu h iz modela $\mathcal{H}$. S druge strane, hiperparametar određuje koji skup $\mathcal{H}_i$ ćemo uopće koristiti pri optimizaciji parametara $\boldsymbol{\theta}$. Dakle, **hiperparametri određuju model**, dok **parametri određuju hipotezu**. U tom smislu hiperparametri su jednu razinu iznad običnih parametara, i zato ih tako zovemo ("hiper" u smislu "iznad").

Ovdje primijetimo tri stvari. Prvo, budući da hiperaparametri određuju model, kada kažemo **odabir modela**, to je potpuno isto kao da smo rekli **optimizacija hiperparametara**. Npr., pretpostavimo da želimo odabrati regresijski model optimalne složenosti iz familije modela koja sadrži polinoma prvog, drugog, trećeg, itd. stupnja. Odabrati optimalan model isto je kao i optimizirati (tj. odabrati optimalan) stupanj polinoma, koji je hiperparametar modela.

Drugo, primijetimo da kod odabira i treniranja modela postoji jasan vremenski slijed. Budući da hiperparametri određuju model, očito je da **prvo moramo optimirati hiperparametre**, čime odabiremo model. Tek nakon toga možemo optimizirati parametre, čime odabiremo hipotezu. Dakle, odabir modela uvijek prethodi treniranju modela.

Treća stvar koju treba primijetiti jest da postoji jasna podjela odgovornosti između algoritma strojnog učenje i onoga koji radi odabir modela. Naime, **odabir modela moramo napraviti mi** (čovjek), dok **treniranje modela radi algoritam strojnog učenja**. Ovo je važno, pa ponovimo: algoritam strojnog učenja nije zadužen za optimizaciju hiperaparametara, nego samo za optimizaciju parametara. Odgovornost odabira optimalnog modela je na nama, a ne na algoritmu strojnog učenja. Ako algoritmu strojnog učenja damo suboptimalan model (dakle model koji je prejednostavan ili presložen), ne trebamo se čuditi što će naučen model (tj. hipoteza) raditi loše. Nažalost, ovo se u praksi često zaboravlja, pa se nerijetko događa (pa i u recenziranima znanstvenim radovima) da se prikazuju rezultati dobiveni modelom za koji nije provedena optimizacija hiperparametara.

Spomenimo još da, premda odabir modela moramo napraviti mi a ne algoritam strojnog učenja, to ne znači da ga baš uvijek moramo napraviti ručno: možemo automatski isprobavati više različitih modela (tj. vrijednosti hiperparametara, koristeći, na primjer, neki heuristički postupak optimizacije, npr. genetičke algoritme) te svaki od tih modela trenirati algoritmom

strojnog učenja, te u konačnici onda odabrati najbolji model (onaj koji daje hipotezu koja najbolje generalizira). 13

Možda već možete zaključiti da je odabir modela jedan sumnjiv biznis, budući da moramo naći neku zlatnu sredinu, a ne znamo ni sami gdje bi ta bila. Zbog toga je u strojnom učenju razvijeno mnogo teorije o odabiru modela. U praksi, međutim, najčešće koristimo jedan jednostavan empirijski postupak koji se zove **unakrsna provjera** ili **križna validacija** (engl. *cross-validation*). 14

8 Unakrsna provjera

Unakrsna provjera metoda je za procjenu sposobnosti generalizacije modela. Osnovna ideja je sljedeća. Unutar familije modela koju razmatramo, želimo odabrati onaj model koji najbolje generalizira, odnosno koji dobro radi na **neviđenim primjerima**. Međutim, budući da nemamo neviđene primjere (jer inače očito ne bi bili neviđeni), poslužit ćemo se jednim zgodnim trikom: izdvojit ćemo iz našeg skupa primjera za učenje dio primjera, i oni će glumiti neviđene primjere. Model ih neće vidjeti kod učenja, nego ćemo naučeni model ispitivati na tim primjerima, da vidimo kako dobro generalizira. Primijetite, naravno, da to nisu doista neviđeni primjeri, jer smo ih očito mi vidjeli, međutim ti primjeri nisu korišteni za učenje modela, pa su dakle to neviđeni primjeri iz perspektive algoritma strojnog učenja.

Unakrsnu provjeru provodimo tako da skup primjera dijelimo na **skup za učenje** i **skup za ispitivanje**:

$$\mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}}$$

Skupovi moraju biti međusobno **disjunktni** (dakle, nepreklapajući). Tipično radimo podjelu 70:30 ili 60:40 ili slično, ovisno o tome koliko podataka imamo i kako detaljnu analizu rada modela želimo napraviti. Ovdje očito postoji kompromis: za detaljniju analizu rada modela i bolju procjenu pogreške modela bilo bi dobro imati što više primjera u ispitnom skupu, međutim ako u skupu za učenje nemamo dovoljno primjera, model uopće nećemo moći dobro naučiti. Ovu podjelu obično radimo slučajnim odabirom primjera, kako bismo osigurali da oba skupa ostanu reprezentativna, osim ako postoje dobri razlozi za drugačiji pristup. 15 16

Nakon što smo napravili podjelu skupa primjera na skup za učenje $\mathcal{D}_{\text{train}}$ i skup za ispitivanje $\mathcal{D}_{\text{test}}$, sada računamo dvije pogreške za $h \in \mathcal{H}$ koristeći ta dva skupa:

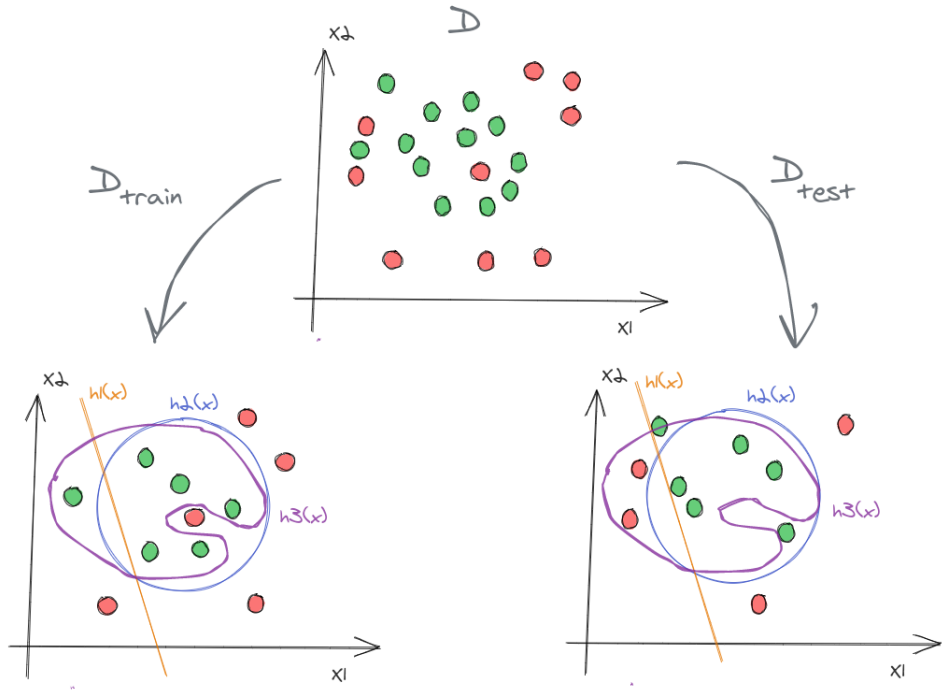
- **Empirijska pogreška učenja** (engl. *empirical training error*) – pogreška hipoteze h izračunata na skupu za učenje $\mathcal{D}_{\text{train}}$, tj. $E(h|\mathcal{D}_{\text{train}})$;
- **Emprijska ispitna pogreška** (engl. *empirical test error*) – pogreška hipoteze h izračunata na skupu za ispitivanje $\mathcal{D}_{\text{test}}$, tj. $E(h|\mathcal{D}_{\text{test}})$.

Ispitna pogreška, budući da je mjerena na skupu primjera koje algoritam nije "vidio" (tj. koji nisu korišteni za treniranje modela), odgovara pogrešci modela na neviđenim podatcima, odnosno kazuje nam koliko će model dobro generalizirati. Zato ispitnu pogrešku zovemo i **pogreška generalizacije**.

Primijetite da kod obje ove pogreške govorimo o **empirijskoj pogrešci**, tj. obje izračunavamo kao očekivanje (iliti prosjek) funkcije gubitka na dotičnim skupovima primjera (za razliku od pravih pogrešaka, koje su nam nepoznate). Međutim, jednostavnosti radi, često se govori samo o **pogrešci učenja** i **pogrešci ispitivanja** (dakle, bez pridjeva "empirijska", koji se podrazumijeva).

▸ **PRIMJER**

Vratimo se opet na prethodni primjer s modelima pravaca ($\mathcal{H}_1$), kružnica ($\mathcal{H}_2$) i "krumpira" ($\mathcal{H}_3$). Pretpostavimo da smo naš skup označenih primjera $\mathcal{D}$ razdijelili, slučajnim uzorkovanjem, na skup za učenje $\mathcal{D}_{\text{train}}$ i skup za ispitivanje $\mathcal{D}_{\text{test}}$, ovako:



U ovom slučaju, skup za učenje ima 11 primjera ($|\mathcal{D}_{\text{train}}| = 11$), dok skup za ispitivanje ima 10 primjera ($|\mathcal{D}_{\text{test}}| = 10$).

Pogledajmo najprije što se događa na skupu za učenje (lijeva slika). Na skupu za učenje treniramo naša tri modela, $\mathcal{H}_1$, $\mathcal{H}_2$ i $\mathcal{H}_3$. Algoritam strojnog učenja (koji god da je) dao nam je za svaki model optimalnu hipotezu, tj. onu koja minimizira empirijsku pogrešku na skupu za učenje. No, vidimo da na skupu za učenje hipoteza h_1 iz modela pravaca $\mathcal{H}_1$ loše klasificira primjere. Preciznije, njezina pogreška učenja je $E(h_1|\mathcal{D}_{\text{train}}) = \frac{5}{11}$. Hipoteza h_2 iz modela kružnica $\mathcal{H}_2$ daje nešto točniju klasifikaciju na skupu skupu za učenje, i njezina je pogreška učenja jednaka $E(h_2|\mathcal{D}_{\text{train}}) = \frac{2}{11}$. S druge strane, model krumpira $\mathcal{H}_3$ je visoke složenosti, i najbolja hipoteza h_3 iz tog modela može bez problema savršeno klasificirati sve primjere iz skupa za učenje, pa je njezina pogreška učenja jednaka $E(h_3|\mathcal{D}_{\text{train}}) = 0$.

Pogledajmo sada kako će se te tri hipoteze, koje su sada naučene na skupu za učenje i fiksirane, provesti na ispitnome skupu (desni grafikon). Tu je situacija dosta drugačija. Zapravo, za hipotezu $h_1 \in \mathcal{H}_1$ situacija i nije toliko drugačija, jer je to podnaučena hipoteza iz prejednostavnog modela, pa ona loše klasificira primjere i na skupu za učenje i na skupu za ispitivanje. Njezina ispitna pogreška je $E(h_1|\mathcal{D}_{\text{test}}) = \frac{2}{10}$. Ispitna pogreška hipoteze iz modela kružnica, $h_2 \in \mathcal{H}_2$, je $E(h_2|\mathcal{D}_{\text{test}}) = \frac{1}{10}$. Konačno, ispitna pogreška hipoteze h_3, koja je iz najsloženijeg modela, modela krumpira, je $E(h_3|\mathcal{D}_{\text{test}}) = \frac{4}{10}$. Općenito, razumno je očekivati da će ispitna pogreška hipoteze biti veća nego njezina pogreška učenja. To je zato jer je hipoteza optimizirana na skupu za učenje, a ne na skupu za ispitivanje, što znači da će se hipoteza bolje prilagoditi skupu za učenje nego ispitnome skupu, koji algoritam nije ni vidio kod učenja. Međutim, to ne mora uvijek biti tako: zbog stohastičnosti uzorkovanja može se dogoditi da ispitna pogreška bude manja od pogreške učenja (ovdje nam se to dogodilo da je za hipotezu h_1).

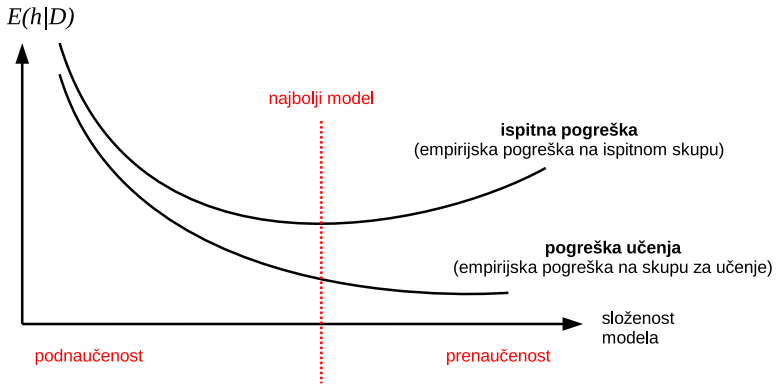
Razmotrimo sada odnose između pogreške učenja i pogreške ispitivanja za svaku od hipoteza. Što se ovdje dogodilo? Dogodilo se to da se hipoteza h_3 previše prilagodila šumu u skupu za učenje, ali takvog šuma nije više bilo u skupu za učenje (tamo je šum drugačiji). Zbog toga h_3 ima vrlo nisku pogrešku učenja, ali vrlo visoku ispitnu pogrešku. Drugim riječima, h_3 loše generalizira, jer loše klasificira neviđene primjere, odnosno model se prenaučio. S druge strane, hipoteza h_1 iz modela pravaca također loše generalizira (ispitna pogreška je visoka), ali ta hipoteza ima i visoku pogrešku učenja, što signalizira da se radi o podnaučenom modelu. Hipoteza h_2 iz modela kružnica najbolje prolazi u ovoj priči: ona doduše griješi i na skupu za učenje i na ispitnom skupu, ali to je hipoteza koja najmanje griješi na ispitnom skupu. Drugim riječima, hipoteza h_2 najbolje generalizira. To znači da je model $\mathcal{H}_2$ model optimalne složenosti za naš klasifikacijski problem definiran skupom

označenih primjera $\mathcal{D}$.

Sažmimo naša opažanja. Model koji je prejednostavan ($\mathcal{H}_1$) bit će podnaučen i dat će hipotezu koja ima visoku i pogrešku učenja i pogrešku ispitivanja. Model koji je presložen ($\mathcal{H}_3$) bit će prenaučen i dat će hipotezu koja doduše ima nisku pogrešku učenja, ali visoku ispitnu pogrešku. Model koji je optimalne složenosti ($\mathcal{H}_2$) dat će hipotezu čija je pogreška učenja između pogrešeke učenja podnaučenog i prenaučenog modela, dok će njezina ispitna pogreška biti manja i od one podnaučenog i prenaučenog modela.

Naša tri modela – $\mathcal{H}_1$ (pravci), $\mathcal{H}_2$ (kružnice) i $\mathcal{H}_3$ (krumpiri) – možemo postaviti u linearan (potpuni) uređaj prema njihovoj složenosti: model $\mathcal{H}_1$ je jednostavniji od modela $\mathcal{H}_2$, koji je pak jednostavniji od modela $\mathcal{H}_3$. Štoviše, mogli bismo definirati da model $\mathcal{H}_2$ uključuje u sebe i sve pravce (recimo da pravce smatramo degeneriranim kružnicama; dakle, to bi onda bio model kružnica i pravaca), dok bismo za model $\mathcal{H}_3$ mogli reći da uključuje kružnice i pravce kao posebne slučajeve. Tada imamo (budući da su modeli skupovi hipoteza): $\mathcal{H}_1 \subset \mathcal{H}_2 \subset \mathcal{H}_3$. U tom slučaju možemo crtati grafikon pogreške učenja i pogreške ispitivanja kao funkcije složenosti modela: x-os tog grafikona odgovara složenosti modela (ili vrijednosti hiperparametra koji određuje složenost modela), a na y-osi je iznos funkcije pogreške učenja odnosno pogreške ispitivanja (v. tekst u nastavku).

Na temelju gornjeg primjera vidimo da pogreška na skupu za učenje, $E(h|\mathcal{D}_{\text{train}})$, pada sa složenošću modela, dok pogreška na ispitnom skupu, $E(h|\mathcal{D}_{\text{test}})$, tipično prvo pada pa zatim raste. Dakle, grafovi pogreške na skupu za učenje (pogreška učenja) i pogreške na skupu za ispitivanje (ispitna pogreška), kao funkcije složenosti modela, očekivano izgledaju ovako:



Optimalan model je upravo onaj koji minimizira ispitnu pogrešku $E(h|\mathcal{D}_{\text{test}})$. Naime, to je pogreška modela na neviđenim primjerima, pa nam ona govori koliko dobro model generalizira. Prema tome, između modela koje razmatramo (familija modela), želimo odabrati onaj model koji ima najmanju pogrešku na neviđenim primjerima jer to znači da taj model najbolje generalizira, tj. da ima najmanju pogrešku generalizacije. Modeli koji su manje složenosti od optimalnog modela (lijeva strana grafikona) nedovoljne su složenosti i oni su skloni podnaučenosti (visoka pogreška učenja i visoka ispitna pogreška). Suprotno, modeli složenosti veće od optimalnog modela (desna strana grafikona) jesu modeli prevelike složenosti i oni su skloni prenaučenosti (mala pogreška učenja ali velika ispitna pogreška).

Gornji graf vjerojatno je najvažniji graf u strojnom učenju. No, trebamo napomenuti da on predstavlja idealizirani slučaj: prikazuje **očekivano** ponašanje pogreške učenja i pogreške ispitivanja u ovisnosti o složenosti modela. U praksi, budući da uvijek radimo sa slučajnim uzorcima podataka, na našem konkretnom skupu primjera može doći do nekih odstupanja (npr., može se dogoditi da ispitna pogreška malo pada pa raste pa opet pada pa raste, ili da je pogreška učenja za model neke složenosti čak veća od ispitne pogreške). Međutim, kada bismo ponavljali uzorkovanje primjera, izračunavali vrijednosti pogrešaka za modele različitih složenosti, i zatim uprosječivali te vrijednosti, možemo očekivati da bismo dobili krivulje kakve smo prikazali na gornjem grafikonu.

Sažetak

- **Hipoteza** je funkcija koje primjere preslikavaju u oznake (klase ili brojeve), definirana do na parametre
- **Model** je skup hipoteza, indeksiran njihovim parametrima
- Učenje (treniranje) modela se svodi na **optimizaciju parametara** modela, što odgovara pretraživanju u skupu hipoteza
- Svaki algoritam strojnog učenja sastoji se od **modela**, **funkcije gubitka/pogreške** i **optimzacijskog postupka**
- Modeli se općenito razlikuju po svojoj **složenosti** (sposobnosti da dosegnu nisku pogrešku učenja)
- Model koji je **podnaučen** ili **prenaučen** loše generalizira
- Odabir modela svodi se na **optimiranje hiperparametara** modela
- **Unakrsnom provjerom** može se procijeniti **pogreška generalizacije** i odabrati optimalan model

Bilješke

[1] Ovo predavanje uglavnom slijedi strukturu drugog poglavlja iz (Alpaydin, 2020), uz neka proširenja. Predavanjem nije pokrivena tema **induktivne pristranosti** (engl. *inductive bias*) – tu temu trebate dodatno proučiti iz skripte (poglavlje 2.3), a proći ćemo ju zajednički na vježbama.

[2] Pristupima temeljenima na dubokom strojnom učenju nastoji se zaobići ekstrakcija značajki, tako da neuronska mreža ne uči samo klasifikaciju/regresiju ulaznog primjera, već implicitno uči i ekstrakciju "značajki" iz sirovih podataka, odnosno **uči reprezentaciju** (engl. *representation learning*). Učenjem reprezentacija posao ekstrakcije značajki prebacuje se s čovjeka na algoritam strojnog učenja. Npr., umjesto da čovjek oblikuje i implementira algoritme za ekstrakciju relevantnih značajki iz slike (npr., detekcija kutova, segmenata, tekstura i sl.), na ulaz neuronske mreže dovodi se slika u izvornome obliku (matrica trokomponentnih slikovnih elemenata) te se tijekom učenja početni slojevi neuronske mreže sami specijaliziraju za ekstrakciju značajki koje su relevantne za zadatak. Učenje reprezentacija pokazalo se posebno korisnim u računalnom vidu i u obradi prirodnog jezika, gdje je ono dovelo do značajnih napredaka. Za uvod u područje učenja reprezentacija, pogledajte (Bengio et al., 2013). Mi se na ovom predmetu nećemo baviti učenjem reprezentacija. Pretpostavit ćemo da je primjer prikazan značajkama, međutim neće nas zanimati na koji smo točno način došli do tih značajki.

[3] Redukcijom dimenzionalnosti nećemo se baviti na ovom predmetu. Međutim, tehnike za redukciju dimenzionalnosti izuzetno su korisne i u praksi je nužno da poznajete barem neke od njih. Dobar pregled tehnika redukcije dimenzionalnosti možete pronaći u (Fodor, 2002) i (Burges, 2010).

[4] Možda izgleda da su tri parametra previše i da bi nam bila dosta dva jer imamo dvodimenzijski prostor primjera, te da bismo mogli koristiti eksplicitni oblik jednadžbe pravca (podijeliti jednadžbu s θ_0) umjesto da koristimo implicitni oblik. No, to nije tako. S eksplicitnim oblikom gubimo mogućnost da definiramo "orijentaciju" pravca (koja strana je pozitivna a koja negativna). Treba nam funkcija koja zapravo definira ravninu ugrađenu u 3D prostoru. Upravo to smo ovdje napravili. Na mjestu gdje ta ravnina siječe ravninu X-Y, tj. u točkama za koje $h(\mathbf{x}) = 0$, tu se nalazi granica između klasa. Općenito, za klasifikaciju u n-dimenzijskome ulaznom prostoru linearan klasifikacijski model treba imati $n + 1$ parametara.

[5] Različite hipoteze imat će različite vektore parametara $\boldsymbol{\theta}$. Međutim, različiti vektori parametara $\boldsymbol{\theta}$ ne moraju nužno davati različite hipoteze. Na primjer, ako je ulazni prostor diskretan (npr. $\mathcal{X} = \mathbb{N}^n$), onda možemo imati pravce koji su malo različiti (dakle parametri $\boldsymbol{\theta}$ im se razlikuju), ali ipak daju identičnu klasifikaciju primjera u dvije klase, tj. funkcija h je jedna te ista

(kako je uobičajeno, jednakost funkcije ovdje definiramo **ekstenzionalno**: dvije funkcije su jednake ako jednako preslikavaju elemente iz domene u kodomenu, tj. $\forall x \in X . h_1(x) = h_2(x)$). To znači da različiti $\boldsymbol{\theta}$ mogu dati identične funkcije $h(\mathbf{x}; \boldsymbol{\theta})$. Dakle, ne vrijedi nužno $h \mapsto \boldsymbol{\theta}$. (Ako ste zainteresirani za ideju funkcijske ekstenzionalnosti, pogledajte ovo: https://ncatlab.org/nlab/show/function+extensionality. Za one s još neiskorijenjenim filozofskim sklonostima, preporučam za početak pročitati o ekstenzionalnosti i odnosu prema intenzionalnosti, ovdje: https://plato.stanford.edu/entries/logic-intensional/).

6 **Optimizacija** i strojno učenje tijesno su povezani. S jedne strane, strojno učenje naveliko koristi algoritme optimizacije i gladno je za sve učinkovitijim postupcima. S druge strane, strojno učenje nerijetko inspirira razvoj novih algoritama optimizacije. Vrlo dobar pregled optimizacije za strojno učenje možete pronaći u (Sra et al., 2012).

7 **Empirijska pogreška** hipoteze $E(h|\mathcal{D})$ na skupu označenih primjera $\mathcal{D}$ je aproksimacija stvarne (ali nama nepoznate) pogreške hipoteze $E(h)$ na svim mogućim primjerima. Formalno:

$$E(h) = \mathbb{E}_{\mathbf{x},y}[L(y, h(\mathbf{x})] \approx \frac{1}{N} \sum_{i=1}^{N} L(y^{(i)}, h(\mathbf{x}^{(i)})) = E(h|\mathcal{D})$$

gdje je $\mathbb{E}_{\mathbf{x},y}[L(y, h(\mathbf{x})]$ **očekivanje funkcije gubitka** po svim označenim primjerima $(\mathbf{x}, y)$ (par $(\mathbf{x}, y)$ tretiramo kao slučajnu varijablu koja se pokorava nekoj nama nepoznatoj distribuciji). Ovo očekivanje aproksimiramo (procjenjujemo) na skupu označenih primjera $\mathcal{D}$ kao srednju vrijednost funkcije gubitka na tom skupu, i to je naša empirijska pogreška $E(h|\mathcal{D})$. U statističkom smislu, empirijska pogreška je procjena prave pogreške hipoteze procijenjena na uzorku $\mathcal{D}$. Kako bi ova procjena bila dobra, skup označenih primjera $\mathcal{D}$ treba biti **reprezentativan** (tj. vjerno predstavljati skup svih mogućih označenih primjera), što se u pravilu postiže ako je skup $\mathcal{D}$ uzorkovan slučajno.

8 Tri komponente mogu se sažeti sljedećom "jednadžbom": **učenje = reprezentacija + evaluacija + optimizacija**. Ovakva "trodioba vlasti" pokazala se kao vrlo koristan način gledanja na algoritme strojnog učenja jer nam omogućava da uočavamo sličnosti i razlike između algoritama. Stoga ćemo na ovom predmetu inzistirati na tome da svaki algoritam strojnog učenja promatramo kroz prizmu ovih triju komponenata. U retrospektivi, definicija strojnog učenja kroz ove tri komponente doimaju se gotovo banalnom, međutim netko se toga ipak trebao prvi dosjetiti. Ne znam tko je to bio. Alpaydin (2020) tri komponente spominje u drugom poglavlju svoje knjige (prvo izdanje je još iz 2004. godine), dok se gornja jednadžba nalazi u Domingos (2012). Zanimljivo je da navedene tri komponente lijepo korespondiraju s različitim vrstama **induktivne pristranosti** algoritma strojnog učenja: tako model definira **pristranost jezikom** (pristranost ograničenja), dok funkcija gubitka i optimizacija definiraju **pristranost preferencijom** (pristranost pretraživanja).

9 **Funkcija argmin** daje minimizator funkcije, tj. $\operatorname{argmin}_x f(x) = x_m$ akko $f(x_m) = \min_x f(x)$. Pazite da ne brkate funkciju min (minimum funkcije) i funkciju argmin (minimizator funkcije). Na primjer, $\min_x (x-1)^2 = 0$, međutim $\operatorname{argmin} (x-1)^2 = 1$. Zašto ovdje koristimo funkciju argmin a ne funkciju min? Zato jer je glavni rezultat optimizacijskoj postupka hipoteza h^* (ili, ekvivalentno, parametri $\boldsymbol{\theta}^*$) koja minimiziraju pogrešku. Naravno, često će nas zanimati i koliko iznosi ta pogreška, međutim to uvijek možemo izračunati jednom kada smo pronašli h^* odnosno $\boldsymbol{\theta}^*$, dok obrat ne vrijedi (ako samo znamo koliko iznosi pogreška, iz toga ne možemo dobiti hipotezu).

10 Ideja da u načelu preferiramo jednostavne modele nad složenijima općenito je prihvaćena u znanosti te je poznata kao **načelo parsmonije** ili **Occamove britve**: ako su dva modela jednake prediktivne točnosti, ispravan model je vjerojatno onaj koji je jednostavniji. No to, naravno, ne znači da jednostavni modeli moraju dati veću predikcijsku točnost. Za raspravu o primjenjivosti načela Occamove britve u strojnom učenju pogledajte (Domingos, 1999).

11 U zadnjih nekoliko godina, a pogotovo s porastom primjene modela dubokih učenja, počelo se tematizirati pitanje **interpretabilnosti i pravednosti** predikcijskih modela. Složeni modeli, poput neuronskih mreža, u tom su smisu posebno problematični jer su netransparentni (*black-box*) te relativno podložni amplificiranju postojećih pristranosti u skupovima podataka. Ovakva razmatranja dovela su do povećanog interesa za tzv. **objašnjivom umjetnom inteligencijom** (engl. *explainable AI, XAI*). Pogledati, na primjer: https://xaitutorial2020.github.io/. Vrlo dobar uvod u problem pravednosti algoritama strojnog učenja daje (još nedovršena) knjiga Barocas et al. (2018), dostupna online

na https://fairmlbook.org/, te pregledni rad (Mehrabi et al., 2019). Poznata knjiga Cathy O'neil (O'neil, 2016) izvrsno tematizira posljedice netransparentnih i nepravednih algoritama na pojedinca i društvo.

12 Problem odabira modela izravno je povezan s tzv. **pretpostavkom induktivnog učenja**, kako ju je definirao Mitchell (1997):

> *"Svaka hipoteza koja koja na dovoljno velikom skupu primjera za učenje dobro aproksimira ciljnu funkciju također će dobro aproksimirati ciljnu funkciju na drugim, neopaženim primjerima."*

Ova hipoteza se u suštini ne može dokazati a da se ne uvedu neke dodatne pretpostavke o ciljnoj funkciji (klasifikaciji). Također je potrebno precizno definirati što znači "dobro aproksimirati". Takvim pitanjima bavi se **teorija računalnog učenja** (engl. *computational learning theory, COLT*), u kojoj su razvijeni teorijski radni okviri za izračun gornje ograde pogreške generalizacije. Npr., radni okvir **vjerojatno približno točno** (engl. *probably approximately correct, PAC*), pomoću kojega možemo izračunati vjerojatnost da će se, za skup primjera određene veličine, pogreška generalizacije biti manja od nekog iznosa. Mitchellova hipoteza induktivnog učenja govori nam da će empirijska pogreška hipoteze, $E(h|\mathcal{D})$, konvergirati stvarnoj pogrešci hipoteze, $E(h)$, s porastom veličine skupa označenih primjera za učenje $\mathcal{D}$. Međutim, u praksi imamo konačan skup primjera za učenje. Nešto praktičniju verziju hipoteze induktivnog učenja dao je Sam Roweis (https://cs.nyu.edu/~roweis/csc2515-2004/notes/lec1x.pdf:

> *"Ako je pogreška hipoteze na dovoljno velikom skupu primjera za učenje mala i ako model nije suviše složen, hipoteza će vjerojatno dobro klasificirati i nove, slične primjere."*

Ovako formulirana pretpostavka induktivnog učenja zapravo tvrdi da je generalizacija moguća, no pod trima uvjetima: (1) empirijska pogreška na skupu za učenje je mala (tj. model nije podnaučen), (2) model nije presložen (tj. model nije prenaučen) i (3) neviđeni primjeri slični su primjerima na kojima je model učen (tj. skup za učenje je reprezentativan uzorak čitave populacije primjera). Zainteresirane za teoriju računalnog strojnog učenja upućujem na izvrsnu knjigu (Shalev-Shwartz and Ben-David, 2014).

13 Automatska optimizacija hiperparametara, odnosno automatski odabir modela, zanimljiv je i netrivijalan problem u strojnom učenju. Potpunom (ili djelomičnom) automatizacijom primjene algoritma strojnog učenja bavi se područje **automatiziranog strojnog učenja** (engl. *automated machine learning, AutoML*). Idealan cilj jest automatizirati sve korake primjene algoritma strojnog učenja (v. dio 2), uključivo i one koje sada još uvijek obavlja čovjek: od pripreme podataka preko ekstrakcije značajki do odabira modela, njegovog vrednovanja i dijagnostike. Više o automatiziranom strojnom učenju možete pronaći u (Hutter et al., 2019).

14 Za pregled različitih pristupa obabira modela, pogledajte (ne baš ažuriranu, ali temeljnu) bibliografiju na http://www.modelselection.org. Dobra referenca na tu temu je (Anderson and Burnham, 2004). Pregled postupaka za unakrsnu provjeru možete naći u (Arlot and Celisse, 2010).

15 Postupak je nešto složeniji kada, uz provjeru pogreške modela na ispitnome skupu, želimo također provesti **odabir modela**. U tom slučaju skup primjera dijelimo na tri međusobno disjunktna skupa: skup za učenje, skup za provjeru i skup za ispitivanje. Više o tome kasnije, u predavanju o vrednovanju modela.

16 Podjelu primjera u skup za učenje i skup za ispitivanje obično radimo slučajnim odabirom, kako bi oba skupa bila reprezentativni uzorci populacije svih primjera. Međutim, u nekim situacijama želimo više kontrole nad time koji će primjeri završiti u skupu za učenje a koji u skupu za ispitivanje. Tipične takve situacije jesu skupovi podataka s vremenskom komponentom, kod kojih u želimo da skup za ispitivanje sadrži novije primjere jer neviđeni primjeri će tipično biti novi primjeri, ili skupovi podataka s mnogo kategorija s neuravnoteženim brojem primjera, gdje želimo da skup za učenje i skup za ispitivanje vjerno zrcale distribuciju klasa, za što onda koristimo **stratificirano uzorkovanje** (engl. *stratified sampling*).

Literatura

E. Alpaydin. *Introduction to machine learning.* MIT press, 2020.

D. Anderson and K. Burnham. Model selection and multi-model inference. *Second. NY: Springer-Verlag*, 63(2020):10, 2004.

S. Arlot and A. Celisse. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

S. Barocas, M. Hardt, and A. Narayanan. Fairness and machine learning. fairmlbook. org, 2018.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

C. J. Burges. *Dimension reduction: A guided tour.* Now Publishers Inc, 2010.

P. Domingos. The role of Occam's razor in knowledge discovery. *Data mining and knowledge discovery*, 3(4):409–425, 1999.

P. Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

I. K. Fodor. A survey of dimension reduction techniques. Technical report, Lawrence Livermore National Lab., CA (US), 2002.

F. Hutter, L. Kotthoff, and J. Vanschoren. *Automated machine learning: methods, systems, challenges.* Springer Nature, 2019.

N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan. A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635*, 2019.

T. Mitchell. *Machine Learning.* McGraw Hill, 1997.

C. O'neil. *Weapons of math destruction: How big data increases inequality and threatens democracy.* Broadway Books, 2016.

S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms.* Cambridge University Press, 2014.

S. Sra, S. Nowozin, and S. J. Wright. *Optimization for machine learning.* MIT Press, 2012.