Primjene teorije informacija na probleme dubokog učenja

Tot, Jakob

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Jakob Tot

PRIMJENE TEORIJE INFORMACIJA NA PROBLEME DUBOKOG UČENJA

Diplomski rad

Voditelj rada:
Matej Mihelčić, doc. dr. sc.
Suvoditelj rada:
Hrvoje Planinić, doc. dr. sc.

Zagreb, veljača, 2024.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Rasmusu za monitor

Sadržaj

Uvod

Većina današnje globalne populacije ne može zamisliti svakodnevicu bez nekog oblika digitalne komunikacije. Bilo da se radi o privatnoj korespondenciji fizičkih osoba ili o poslovnim procesima najvećih svjetskih kompanija, digitalna komunikacija je neizbježna. Nije teško zamisliti situaciju u kojoj vrlo sitna greška u prijenosu neke digitalne poruke rezultira katastrofalnim posljedicama. Istovremeno, rijetko tko je upoznat s pozadinskim mehanizmima procesa o kojem ovisimo u toliko velikoj mjeri. Među prvima koji je postavio fundamentalna pitanja o procesu komunikacije bio je Claude Shannon koji je tada radio u Bell Laboratories. No, njegova veličina leži u tome što je na mnogo njih ustvari i dao odgovor. Za vrijeme drugog svjetskog rata bavio se kriptografijom zbog čega je razvio posebno viđenje komunikacije općenito. Jedno od njegovih bitnih opažanja bilo je da ljudska komunikacija sadrži redundantnosti. Štoviše, procijenio je da je redundantnost engleskog jezika oko 50%. Iako ne postoje nikakve procjene, ona definitivno postoji u nekoj mjeri i u hrvatskom jeziku. Tome u prilog ide čnjnca da ov rčncu mžte prčtati bz vćh prblma. Pokazat će se da se vještom manipulacijom redundatnosti mogu postići različiti ciljevi efikasne i pouzdane komunikacije. Davši do tada neviđen pogled na informaciju, Shannon je razvio područje koje ima dalekosežan utjecaj na mnoge druge grane.

Jedna od tih grana je i duboko učenje koje je isto tako danas sveprisutno. Iako grana ima bogatu istraživačku prošlost, sjeme tog istraživanja je tek nedavno palo na plodno tlo. Tako se šira javnost u potencijal modela dubokog učenja uvjerila tek pojavom velikih jezičnih modela (eng. *large language model*) poput ChatGPT-a[1] te raznih generativnih modela poput Midjourney-a[2]. Osim toga, razni modeli dubokog učenja danas postižu nadljudske rezultate kod problema koji su lagani za ljude, ali su tradicionalno bili nemogući za računala: prepoznavanje objekata, prepoznavanje govora, prevođenje itd. Preostaje vidjeti kakvu će ulogu duboko učenje imati na našem putu prema *superinteligenciji* za koju mnogi znanstvenici vjeruju da će se razviti u ovom stoljeću.

[1] openai.com/chatgpt
[2] midjourney.com

Poglavlje 1

Teorija informacija

Već od svojih začetaka, teorija informacija bavi se proučavanjem fenomena komunikacije. Štoviše, Shannonov članak u kojem su po prvi put formalizirani koncepti teorije informacija zove se "Matematička teorija komunikacije" ([20]). U istom radu predložen je Shannon-Weaverov model komunikacije u kojem se navode tri različite vrste problema u komunikaciji: tehnički (koliko se precizno simboli komunikacije mogu odaslati), semantički (koliko se precizno može prenijeti značenje poruke), efektivni (koliko efikasno primljena poruka utječe na željeno ponašanje). Prema tome, ne čudi što je informacija, kao i neke fizikalne veličine (npr. energija), pojam kojem ljudi pripisuju određena kvalitativna svojstva. Međutim, teorija informacija bavi se isključivo tehničkim problemom komunikacije. U svrhu maksimalnog apsolviranja ove materije, bitno je informaciju razmatrati kao strogo kvantitativnu, izmjerivu veličinu. To prvenstveno znači da je značenje odašiljanih poruka irelevantno. Shannon se toga pronicljivo dotaknuo rekavši da "poruke često imaju značenje".

1.1 Osnovne definicije i teoremi

U ovom potpoglavlju promatramo diskretne slučajne varijable definirane na vjerojatnosnom prostoru $(\Omega, \mathcal{F}, \mathbb{P})$, s time da je Ω konačan. Ako nije drugačije navedeno, kod logaritama se podrazumijeva baza 2.

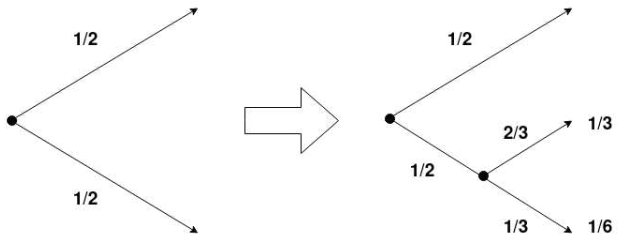
Entropija

Za početak uvodimo pojam entropije, veličinu koja će nam služiti kao mjera za prosječnu "informaciju" slučajne varijable. Temeljna ideja je da slučajna varijabla "sadrži" tim više informacije što je ona više neprevidljiva, odnosno što nas više njena vrijednost može iznenaditi. Stoga je pomalo neočekivano što ćemo entropiju definirati na način da će ona biti

potpuno određena vjerojatnosnom distribucijom slučajne varijable, dok će same vrijednosti koje ona poprima biti nebitne. Označimo s H našu traženu mjeru. Neka naša slučajna varijabla poprima vrijednosti iz n-članog skupa $\mathcal{A}$ te neka su $p_1, p_2, \ldots, p_n$ vjerojatnosti da varijabla poprimi te vrijednosti. Kako bi ikoja takva mjera bila matematički korisna i u skladu s našom intuicijom, smisleno je zahtijevati da ona zadovoljava sljedeće uvjete:

1. H je neprekidna po p_i.
2. H je simetrična, tj. poredak varijabli od H nije bitan.
3. H poprima maksimalnu vrijednost kada je $p_i = \frac{1}{n}, i = 1, \ldots, n$.
4. $H(p_1, \ldots, p_n) = H(p_1 + p_2, p_3, \ldots, p_n) + (p_1 + p_2)H\left(\frac{p_1}{p_1+p_2}, \frac{p_2}{p_1+p_2}\right)$ (aksiom grupiranja).

Ako bi neki sustav promijenili tako da jedan ishod rascijepamo na podishode, tada bi željeli da neizvjesnost sustava bude uvećana za neizvjesnost uzrokovanu tim cijepanjem pomnoženu s vjerojatnosti originalnog ishoda. Zbog toga je prirodno zahtijevati da vrijedi aksiom grupiranja.



Slika 1.1: Primjer podjele jednog ishoda na dva podishoda. Po aksiomu grupiranja vrijedi da je $H(\frac{1}{2}, \frac{1}{3}, \frac{1}{6}) = H(\frac{1}{2}, \frac{1}{2}) + \frac{1}{2}H(\frac{2}{3}, \frac{1}{3})$.

Može se pokazati ([20]) da je jedina funkcija s gornjim svojstvima oblika $-C\sum_{i=1}^{n} p_i \log p_i$ za $C > 0$.

Definicija 1.1.1. *Neka je X slučajna varijabla koja poprima vrijednosti iz skupa $\mathcal{X}$ i $p(x) = \mathbb{P}(X = x)$ njena diskretna funkcija gustoće. Tada je* entropija *slučajne varijable X definirana s*

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x),$$

uz konvenciju da ako je $p(x) = 0$, onda stavljamo $p(x) \log p(x) = 0$ zbog $\lim_{x \to 0^+} x \log x = 0$. Pripadajuće mjerne jedinice za baze logaritama 2, e i 10 su bitovi, natovi i ditovi, redom.

Vidimo da je entropija slučajne varijable potpuno određena njenom distribucijom. Ako je skup vrijednosti koje slučajna varijabla poprima n-člani, onda se umjesto $H(X)$ može pisati i $H(p_1, \ldots, p_n)$, gdje su p_i vjerojatnosti poprimanja pripadnih vrijednosti. Jedna od mnogobrojnih interpretacija entropije je prosječan broj bitova (sada govorimo o binarnim znamenkama) potrebnih da bi se opisala dana slučajna varijabla.

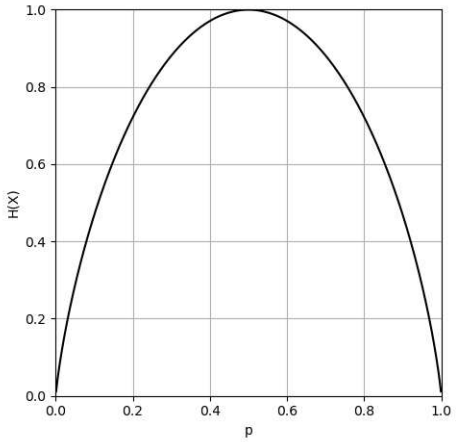
Teorem 1.1.2. *Neka je X slučajna varijabla koja poprima vrijednosti iz skupa $\mathcal{X}$ i H(X) njena entropija. Tada vrijedi:*

- *$H(X) \geq 0$, dok se jednakost postiže kada je $p(x) = 1$, za neki $x \in \mathcal{X}$ (siguran događaj),*
- *$H(X) \leq \log(|\mathcal{X}|)$, dok se jednakost postiže kada je $p(x) = \frac{1}{|\mathcal{X}|}$, za sve $x \in \mathcal{X}$ (jednako vjerojatni događaji).*

Primjer 1.1.3. *Neka je X Bernoullijeva slučajna varijabla s vjerojatnostima p i $q = 1 - p$, $p \in [0, 1]$, odnosno*

$$X \sim \begin{pmatrix} 0 & 1 \\ q & p \end{pmatrix}.$$

Tada je entropija ove slučajne varijable jednaka $H(X) = -p \log p - (1-p)\log(1-p)$. Graf ove funkcije prikazan je na slici ispod. On zorno ilustrira svojstva entropije iz prethodnog teorema. Naime, možemo vidjeti da je entropija jednaka nuli za $p = 0$ ili $p = 1$, što odgovara sigurnim događajima, a entropija je jednaka $1 = \log(|\mathcal{X}|) = \log(2)$ kada je $p = \frac{1}{2}$.



Slika 1.2: Graf entropije Bernoullijeve slučajne varijable.

Napomena 1.1.4. *Entropija slučajne varijable u definiciji 1.1.1 je specijalan slučaj entropije slučajnog vektora. Neka su $X_1, \ldots, X_n$ slučajne varijable definirane na $\Omega_1, \ldots, \Omega_n$ koje poprimaju vrijednosti iz skupova $\mathcal{X}_1, \ldots, \mathcal{X}_n$ te neka je $p(x_1, \ldots, x_n) = \mathbb{P}(X_1 = x_1, \ldots, X_n = x_n)$ njihova zajednička diskretna funkcija gustoće. Tada je*

$$H(X_1, \ldots, X_n) = -\sum_{(x_1,\ldots,x_n)\in\mathcal{X}_1\times\cdots\times\mathcal{X}_n} p(x_1, \ldots, x_n) \log p(x_1, \ldots, x_n).$$

Definicija 1.1.5. *Neka su X i Y slučajne varijable, $p(x, y)$ njihova zajednička diskretna funkcija gustoće, te $p_X(x)$ i $p_Y(y)$ marginalne funkcije gustoće varijabli X i Y. Tada je* uvjetna entropija *od X uz dano Y definirana s*

$$H(X|Y) = -\sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x, y) \log p(x|y),$$

gdje je $p(x|y) = \frac{p(x,y)}{p_Y(y)}$. Podrazumijeva se da sumiramo samo po onim $y \in \mathcal{Y}$ za koje je gornji izraz dobro definiran, dakle po y za koje vrijedi $p_Y(y) > 0$.

Slično kao i definicija 1.1.1, definicija 1.1.5 je specijalan slučaj uvjetne entropije m-dimenzionalnog slučajnog vektora $\mathbf{X} = (X_1, \ldots, X_m)$ uz dani n-dimenzionalni slučajni vektor $\mathbf{Y} = (Y_1, \ldots, Y_n)$.

Teorem 1.1.6. *Za slučajne varijable X i Y vrijedi $H(X, Y) = H(X) + H(Y|X)$.*

Dokaz.

$$\begin{aligned}
H(X, Y) &= -\sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x, y) \log p(x, y) \\
&= -\sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x, y) \log \left[p(x)p(y|x)\right] \qquad (1.1) \\
&= -\sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x, y) \log p(x) - \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x, y) \log p(y|x) \\
&= -\sum_{x\in\mathcal{X}} p(x) \log p(x) - \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x, y) \log p(y|x) \\
&= H(X) + H(Y|X)
\end{aligned}$$

gdje smo u (1.1) koristili definiciju uvjetne vjerojatnosti.

□

Korolar 1.1.7. *$H(X, Y) \leq H(X) + H(Y)$ dok se jednakost postiže ako i samo ako su X i Y nezavisne slučajne varijable.*

Dokaz. Obje tvrdnje slijede iz prethodnog teorema. Za prvu, potrebno je pokazati da $H(Y|X) \leq H(Y)$. Računamo:

$$\begin{aligned} H(Y) - H(Y|X) &= -\sum_{y\in\mathcal{Y}} p(y)\log p(y) + \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(y|x) \\ &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(y) + \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log\frac{p(x,y)}{p(x)} \\ &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log\frac{p(x)p(y)}{p(x,y)} \\ &\geq -\log\left(\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\,\frac{p(x)p(y)}{p(x,y)}\right) = 0. \end{aligned} \tag{1.2}$$

Nejednakost u (1.2) slijedi primjenom Jensenove nejednakosti za slučajne varijable jer je negativni logaritam konveksna funkcija. Time je dokazana prva tvrdnja. Za nezavisne X i Y vrijedi $H(Y|X) = H(Y)$ pa po prethodnom teoremu vrijedi i druga tvrdnja. □

Napomena 1.1.8. *Iako vrijedi nejednakost $H(Y|X) \leq H(Y)$, to nije slučaj kada promatramo uvjetnu entropiju za danu konkretnu vrijednost $H(Y|X = x)$. To znači da u nekim slučajevima specifična vrijednost od X neće nužno prenijeti nikakvu informaciju o Y, ali u prosjeku, promatrajući sve moguće vrijednosti, hoće (vidi Primjer 1.1.9).*

Primjer 1.1.9. *Funkcija gustoće slučajnog vektora (X,Y) dana je s*

Y \ X	**1**	**2**	**3**	**4**
1	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{32}$
2	$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{32}$	$\frac{1}{32}$
3	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$
4	$\frac{1}{4}$	0	0	0

Marginalna gustoća od X je tada $\left(\frac{1}{2},\frac{1}{4},\frac{1}{8},\frac{1}{8}\right)$, dok je marginalna gustoća od Y jednaka $\left(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}\right)$. Slijedi da je $H(X) = \frac{7}{4}$ i $H(Y) = 2$. Nadalje, za kraći račun koristimo alterna-

tivni izraz za uvjetnu entropiju

$$H(X|Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x|y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(y)p(x|y)\log p(x|y)$$
$$= -\sum_{y\in\mathcal{Y}} p(y)\sum_{x\in\mathcal{X}} p(x|y)\log p(x|y) = -\sum_{y\in\mathcal{Y}} p(y)H(X|Y=y)$$

te uvjetne distribucije od X uz dano Y=i, za i=1,2,3,4. Sada računamo:

$$H(X|Y) = -\sum_{i=1}^{4}\mathbb{P}(Y=i)\,H(X|Y=i)$$
$$= \frac{1}{4}\left[H\left(\frac{1}{2},\frac{1}{4},\frac{1}{8},\frac{1}{8}\right) + H\left(\frac{1}{4},\frac{1}{2},\frac{1}{8},\frac{1}{8}\right) + H\left(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}\right) + H(1,0,0,0)\right] = \frac{11}{8}.$$

Analognim računom dobivamo da je $H(Y|X) = \frac{13}{8}$. *Po definiciji možemo izračunati i zajedničku entropiju H(X,Y). Alternativno, nakon računanja jedne od uvjetnih entropija, H(X,Y) možemo izračunati pomoću Teorema 1.1.6. pa je tako* $H(X,Y) = H(Y) + H(X|Y) = 2 + \frac{11}{8} = \frac{27}{8}$. *Uočimo kako je* $H(X) = \frac{7}{4} < H(X|Y=3) = 2$.

Napomena 1.1.10. *Općenito ne vrijedi* $H(X|Y) = H(Y|X)$. *Međutim, vrijedi* $H(X) - H(X|Y) = H(Y) - H(Y|X)$.

Uvedimo sada pojam relativne entropije, mjeru koja opisuje koliko se neke dvije vjerojatnosne distribucije razlikuju. Iako ona na neki način predstavlja udaljenost između distribucija, ona nije udaljenost u strogom smislu riječi jer nije simetrična i ne zadovoljava nejednakost trokuta.

Definicija 1.1.11. *Neka su P i Q vjerojatnosne distribucije definirane na istom prostoru elementarnih događaja.* Relativna entropija *ili* Kullback-Leiblerova divergencija *distribucija P i Q dana je s*

$$D_{KL}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\log\frac{P(x)}{Q(x)}$$

uz konvenciju da je $0\log\frac{0}{0} = 0$, $0\log\frac{0}{q} = 0$ *te* $p\log\frac{p}{0} = \infty$.

Relativna entropija $D_{KL}(P\|Q)$ je mjera koja nam govori koliko smo neučinkoviti kada pretpostavimo da je Q distribucija neke slučajne varijable, dok je njena stvarna distribucija ustvari P. Točnije, kada bismo znali P, mogli bismo opisati tu slučajnu varijablu s u prosjeku $H(P)$ bitova. Ako bismo istu varijablu pokušali opisati tako što za nju pretpostavimo distribuciju Q, trebalo bi nam u prosjeku $D_{KL}(P\|Q)$ bitova više.

Ponovno se vraćamo slučajnim varijablama. Želimo kvantificirati koliko poznavanje neke slučajne varijable smanjuje količinu informacije druge.

Definicija 1.1.12. *Neka su X i Y slučajne varijable, $p(x, y)$ njihova zajednička diskretna funkcija gustoće te $p_X(x)$ i $p_Y(y)$ njihove marginalne funkcije gustoće.* Uzajamna informacija *varijabli X i Y dana je s*

$$I(X;\ Y) = \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p_X(x)p_Y(y)} = D_{KL}(p\|p_X p_Y).$$

Primijetimo da je $I(X;\ Y) = I(Y;\ X)$. Količina informacije koju nam poznavanje varijable X pruža o varijabli Y jednaka je količini informacije koju nam poznavanje varijable Y pruža o varijabli X. Slično kao i korelacija, uzajamna informacija je na neki način mjera zavisnosti dviju varijabli. No, radi se o drugačijem konceptu zavisnosti. Dok korelacija prati kretanje vrijednosti slučajnih varijabli, uzajamna informacija je indikator općenitije veze dviju varijabli koja kvantificira koliko se njihova zajednička distribucija razlikuje od produkta njihovih marginalnih distribucija.

Teorem 1.1.13. *Vrijedi $D_{KL}(P\|Q) \geq 0$.*

Dokaz. Neka je $A = \{x : P(X) > 0\}$ nosač distribucije P, odnosno njene pripadne funkcije gustoće. Tada imamo:

$$\begin{aligned} D_{KL}(P\|Q) &= \sum_{x\in\mathcal{X}} P(x) \log \frac{P(x)}{Q(x)} = -\sum_{x\in A} P(x) \log \frac{Q(x)}{P(x)} \\ &\geq -\log\left(\sum_{x\in A} P(x)\frac{Q(x)}{P(x)}\right) = -\log\left(\sum_{x\in A} Q(x)\right) \qquad (1.3) \\ &\geq -\log\left(\sum_{x\in\mathcal{X}} Q(x)\right) = 0. \end{aligned}$$

gdje smo za nejednakost u (1.3) opet iskoristili Jensenovu nejednakost. □

Korolar 1.1.14. *Vrijedi $I(X;\ Y) \geq 0$.*

Dokaz. Kako je $I(X;\ Y) = D_{KL}(p(x,y)\|p(x)p(y))$, tvrdnja slijedi iz prethodnog teorema ako uzmemo $P = p(x, y)$ te $Q = p(x)p(y)$. Alternativno, već smo u korolaru 1.1.7 pokazali da je

$$H(Y) - H(Y|X) = -\sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x,y) \log \frac{p(x)p(y)}{p(x,y)} = I(X;\ Y) \geq 0.$$

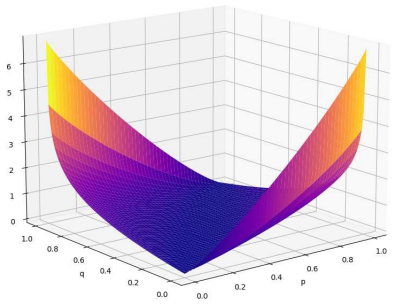
□

Primjer 1.1.15. *Neka je* $\Omega = \{0, 1\}$ *te neka su* P *i* Q *dvije Bernoullijeve distribucije na* Ω *takve da je* $P(0) = 1 - p, P(1) = p$ *te* $Q(0) = 1 - q, Q(1) = q$ *za* $p, q \in [0, 1]$. *Tada imamo*

$$D_{KL}(P\|Q) = (1-p)\log\frac{1-p}{1-q} + p\log\frac{p}{q}$$

$$D_{KL}(Q\|P) = (1-q)\log\frac{1-q}{1-p} + q\log\frac{q}{p}$$

Očito je $D_{KL}(P\|Q) = D_{KL}(Q\|P) = 0$ *za* $p = q$, *što je u skladu s našom prethodnom diskusijom. Za* $p = \frac{1}{2}$ *i* $q = \frac{1}{4}$ *imamo* $D_{KL}(P\|Q) = 0.2075$ *bitova, dok je* $D_{KL}(Q\|P) = 0.1887$ *bitova. Treba nam više dodatne informacije (pa je time naša neučinkovitost veća) da bi opisali nepristrani novčić ako za njega pretpostavimo da je pristran nego obratno.*



Slika 1.3: Graf KL divergencije $D_{KL}(P\|Q)$ za P i Q Bernoullijeve.

Sada dokazujemo tzv. lančana pravila za dosad definirane mjere.

Teorem 1.1.16 (Lančano pravilo za entropiju)**.** *Neka je* $X_1, X_2, \ldots, X_n$ *niz od* n *slučajnih varijabli. Tada je*

$$H(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} H(X_i|X_{i-1}, \ldots, X_1).$$

Dokaz. Jer je $p(x, y|z) = p(y|x, z)p(x|z)$, analognim postupkom kao u dokazu teorema 1.1.6 pokaže se da vrijedi $H(X, Y|Z) = H(X|Z) + H(Y|X, Z)$. Uzastopnom primjenom tog rezultata

slijedi da je

$$\begin{aligned} H(X_1, X_2) &= H(X_1) + H(X_2|X_1) \\ H(X_1, X_2, X_3) &= H(X_1) + H(X_2, X_3|X_1) \\ &= H(X_1) + H(X_2|X_1) + H(X_3|X_2, X_1) \\ &\vdots \\ H(X_1, X_2, \ldots, X_n) &= H(X_1) + H(X_2|X_1) + \cdots + H(X_n|X_{n-1}, \ldots, X_1) \\ &= \sum_{i=1}^{n} H(X_i|X_{i-1}, \ldots, X_1). \end{aligned}$$

□

Korolar 1.1.17. *Vrijedi*

$$H(X_1, X_2, \ldots, X_n) \leq \sum_{i=1}^{n} H(X_i).$$

Dokaz. Po teoremu 1.1.16 vrijedi

$$H(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} H(X_i|X_{i-1}, \ldots, X_1).$$

Prije smo pokazali da vrijedi $H(X|Y) \leq H(X)$. Kako je uvjetna entropija za slučajne varijable specijalan slučaj uvjetne entropije za slučajne vektore, vrijedi i $H(X_i|X_{i-1}, \ldots, X_1) < H(X_i)$, za $i = 1, 2, \ldots, n$. Time je tvrdnja dokazana. □

Definicija 1.1.18. Uvjetna uzajamna informacija *slučajnih varijabli X i Y uz danu slučajnu varijablu Z definirana je kao*

$$I(X;\ Y|Z) = H(X|Z) - H(X|Y,\ Z).$$

Teorem 1.1.19 (Lančano pravilo za uzajamnu informaciju)**.** *Neka su $X_1, X_2, \ldots, X_n, Y$ slučajne varijable. Tada je*

$$I(X_1, X_2, \ldots, X_n;\ Y) = \sum_{i=1}^{n} I(X_i;\ Y|X_{i-1}, \ldots, X_1).$$

Dokaz.

$$\begin{aligned} I(X_1, X_2, \ldots, X_n;\ Y) &= H(X_1, X_2, \ldots, X_n) - H(X_1, X_2, \ldots, X_n|Y) \\ &= \sum_{i=1}^{n} H(X_i|X_{i-1}, \ldots, X_1) - \sum_{i=1}^{n} H(X_i|X_{i-1}, \ldots, X_1, Y) \\ &= \sum_{i=1}^{n} I(X_i;\ Y|X_1, \ldots, X_{i-1}). \end{aligned}$$

□

Definicija 1.1.20. Relativna entropija za uvjetne distribucije $P(Y|X)$ *i* $Q(Y|X)$ *(tzv. uvjetna relativna entropija) definirana je kao*

$$D_{KL}(P(Y|X)||Q(Y|X)) = \sum_{x\in\mathcal{X}} P(x) \sum_{y\in\mathcal{Y}} P(y|x) \log \frac{P(y|x)}{Q(y|x)}.$$

Teorem 1.1.21 (Lančano pravilo za relativnu entropiju)**.** *Vrijedi*

$$D_{KL}(P(X,Y)||Q(X,Y)) = D_{KL}(P(X)||Q(X)) + D_{KL}(P(Y|X)||Q(Y|X))$$

Dokaz.

$$\begin{aligned} D_{KL}(P(X,Y)||Q(X,Y)) &= \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} P(x,y) \log \frac{P(x,y)}{Q(x,y)} \\ &= \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} P(x,y) \log \frac{P(x)P(y|x)}{Q(x)Q(y|x)} \\ &= \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} P(x,y) \log \frac{P(x)}{Q(x)} + \sum_{x\in\mathcal{Y}} \sum_{y\in\mathcal{Y}} P(x,y) \log \frac{P(y|x)}{Q(y|x)} \\ &= D_{KL}(P(X)||Q(X)) + D_{KL}(P(Y|X)||Q(Y|X)). \end{aligned}$$

□

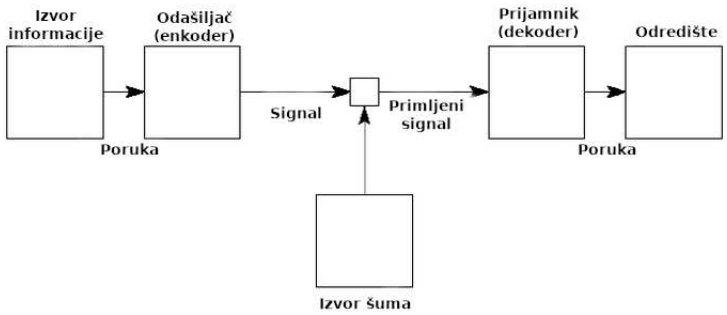
1.2 Kodiranje izvora

Već smo spomenuli kako je teorija informacija duboko povezana s procesom komunikacije. U slučaju digitalne komunikacije, bilo da se radi o raznim kablima ili o elektromagnetskom spektru, kod svih komunikacijskih kanala prisutna su različita fizikalna ograničenja. Jedno od tih ograničenja je kapacitet kanala. U suštini, to je najveća stopa po kojoj se informacije mogu slati kroz kanal uz proizvoljno malu vjerojatnost pogreške. Iz tog razloga vrlo je bitno da kanal koristimo što je efikasnije moguće. Što to točno znači?

Primjer 1.2.1. *U konjskoj utrci sudjeluje osam konja čije su vjerojatnosti pobjede redom* $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}$*. Entropija ove konjske utrke je*

$$H(X) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{4}\log\frac{1}{4} - \frac{1}{8}\log\frac{1}{8} - \frac{1}{16}\log\frac{1}{16} - \frac{4}{64}\log\frac{1}{64} = 2 \text{ bita}.$$

Recimo da želimo poslati poruku iz koje će se nekako moći zaključiti koji je konj pobjedio. Možda su konjima dodijeljena imena pa bi jedno rješenje bilo poslati ime pobjednika.



Slika 1.4: Shematski prikaz općenitog komunikacijskog sustava

Drugo, mnogo jednostavnije rješenje bilo bi poslati indeks konja koji je pobjedio. Na ovaj način, svaki konj može se jedinstveno identificirati s tri binarne znamenke (bita). Prema tome, prosječna duljina poruke kojom bi se obznanio pobjednik je 3 bita. Možemo li bolje? Neka su redom sljedeći nizovi bitova oznake konja: 0, 10, 110, 1110, 111100, 111101, 111110, 111111. Jer distribucija pobjednika nije uniformna, sada je prosječna duljina poruke jednaka $\frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{16} \cdot 4 + \frac{4}{64} \cdot 6 = 2$ *bita.*

Dakle, način na koji reprezentiramo informaciju utječe na prosječnu duljinu poruke kojom istu želimo prenijeti. Taj proces drugačijeg obilježavanja informacije (tzv. kodiranje) nije ništa drugo nego kompresija podataka. Postoje dvije vrste: s gubitcima (eng. *lossy*) ili bez gubitaka (eng. *lossless*). U praksi je kompresija s gubicima vrlo raširena zbog ogromne uštede memorije. Tako na primjer danas najpopularniji formati digitalnih audio i slikovnih zapisa (mp3 i JPEG) koriste kompresiju s gubicima. Međutim, mi ćemo promatrati samo kompresiju bez gubitaka u kojoj se kodiranjem uklanja redundantnost. Primijetimo da smo u prethodnom primjeru osmislili kodiranje takvo da je prosječna duljina poruke bila jednaka entropiji. U nastavku ćemo vidjeti da bolje od toga ne možemo.

Uvedimo za početak par osnovnih definicija vezanih za kodiranje.

Definicija 1.2.2. *Neka su S i T konačni skupovi simbola. Preslikavanje $C : S \to T^*$, gdje je T^* skup svih konačnih nizova simbola iz T, zovemo* kodiranje. *Elemente skupa T^* zovemo* kodne riječi.

Nama će u ulozi skupa S uglavnom biti skup $\mathcal{X}$, a u ulozi skupa T skup $\{0, 1\}$.

Definicija 1.2.3. *Za kodiranje $C : S \to T^*$ kažemo da je* nesingularno *ako je C injektivno preslikavanje, tj.*

$$(\forall x \in S)\, x \neq x' \implies C(x) \neq C(x').$$

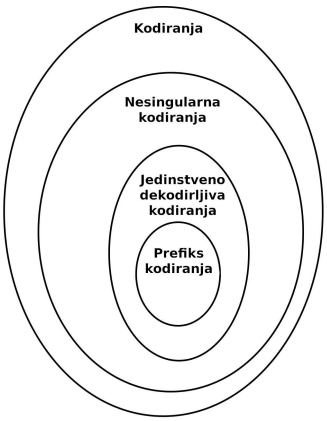
Definicija 1.2.4. *Preslikavanje $C^* : S^* \to T^*$ definirano s*

$$C(x_1 x_2 \cdots x_n) = C(x_1)C(x_2)\cdots C(x_n),$$

gdje je s $C(x_1)C(x_2)\cdots C(x_n)$ označena konkatenacija pripadnih kodnih riječi, zovemo ekstenzija kodiranja *C.*

Definicija 1.2.5. *Za kodiranje C kažemo da se može dekodirati na jedinstven način ako je pripadna ekstenzija C^* nesingularna.*

Definicija 1.2.6. *Za kodiranje C kažemo da je* prefiksno *ako nijedna kodna riječ nije prefiks nijedne druge kodne riječi.*



Slika 1.5: Skupovni odnos različitih klasa kodiranja

Da bi kodiranje bilo od ikakve praktične koristi, očigledno se mora moći dekodirati na jedinstven način. Zbog toga od sada nadalje podrazumijevamo da se kodiranja koja spominjemo mogu dekodirati na jedinstven način, iako to nije eksplicitno naglašeno.

Teorem 1.2.7 (Svojstvo asimptotske ekviparticije)**.** *Neka je $X_1, X_2, \ldots$ niz nezavisnih jednako distribuiranih slučajnih varijabli s diskretnom funkcijom gustoće p_X. Tada vrijedi*

$$-\frac{1}{n}\log p(X_1, X_2, \ldots, X_n) \xrightarrow{\mathbb{P}} H(X) \ \text{ kada } n \to \infty.$$

Ovdje je p diskretna funkcija gustoće slučajnog vektora $(X_1, \ldots, X_n)$ pri čemu, ovdje i u nastavku, radi jednostavnosti izostavljamo pripadni indeks iz oznake.

Dokaz. Jer su funkcije nezavisnih slučajnih varijabli i same nezavisne slučajne varijable, slijedi da je $\log p_X(X_1), \log p_X(X_2), \ldots, \log p_X(X_n)$ niz nezavisnih jednako distribuiranih slučajnih varijabli. Po slabom zakonu velikih brojeva sada imamo:

$$\begin{aligned} -\frac{1}{n}\log p(X_1, X_2, \ldots, X_n) &= -\frac{1}{n}\sum_{i=1}^{n}\log p_X(X_i) \xrightarrow{\mathbb{P}} -\mathbb{E}\left[\log p_X(X)\right] \\ &= -\sum_{x\in\mathcal{X}} p_X(x)\log p_X(x) = H(X). \end{aligned}$$

□

Svojstvo asimptotske ekviparticije je svojevrsni analogon zakona velikih brojeva za entropiju.

Definicija 1.2.8. *Neka je $X_1, X_2, \ldots$ niz nezavisnih jednako distribuiranih slučajnih varijabli te neka su $\epsilon > 0$ i $n \in \mathbb{N}$. Skup nizova $(x_1, \ldots, x_n) \in \mathcal{X}^n$ koji zadovoljavaju svojstvo*

$$2^{-n(H(X)+\epsilon)} \leq p(x_1, x_2, \ldots, x_n) \leq 2^{-n(H(X)-\epsilon)}$$

zovemo tipičnim skupom *s obzirom na danu distribuciju i označavamo ga s $A_\epsilon^{(n)}$.*

Teorem 1.2.9. *Za tipični skup $A_\epsilon^{(n)}$ vrijedi:*

1. *Ako $(x_1, x_2, \ldots, x_n) \in A_\epsilon^{(n)}$, tada $H(X) - \epsilon \leq -\frac{1}{n}\log p(x_1, x_2, \ldots, x_n) \leq H(X) + \epsilon$.*
2. *$\mathbb{P}\left((X_1, \ldots, X_n) \in A_\epsilon^{(n)}\right) > 1 - \epsilon$, za n dovoljno velik.*
3. *$|A_\epsilon^{(n)}| \leq 2^{n(H(X)+\epsilon)}$, za svaki n.*
4. *$|A_\epsilon^{(n)}| \geq (1-\epsilon)2^{n(H(X)-\epsilon)}$, za n dovoljno velik.*

Dokaz. Prvo svojstvo slijedi direktno iz definicije tipičnog skupa tako što niz nejednakosti prvo logaritmiramo, a zatim podijelimo s $-n$. Drugo svojstvo slijedi iz prvog svojstva te iz svojstva asimptotske ekviparticije (Teorem 1.2.7), odnosno definicije konvergencije po vjerojatnosti. Za dokaz trećeg svojstva, računamo:

$$\begin{aligned} 1 &= \sum_{(x_1,x_2,\ldots,x_n)\in\mathcal{X}^n} p(x_1, x_2, \ldots, x_n) \geq \sum_{(x_1,x_2,\ldots,x_n)\in A_\epsilon^{(n)}} p(x_1, x_2, \ldots, x_n) \\ &\geq \sum_{(x_1,x_2,\ldots,x_n)\in A_\epsilon^{(n)}} 2^{-n(H(X)+\epsilon)} = 2^{-n(H(X)+\epsilon)}|A_\epsilon^{(n)}|. \end{aligned}$$

Konačno, zbog drugog svojstva imamo

$$1 - \epsilon < \mathbb{P}\left((X_1, \ldots, X_n) \in A_\epsilon^{(n)}\right) \leq \sum_{(x_1,x_2,\ldots,x_n)\in A_\epsilon^{(n)}} 2^{-n(H(X)-\epsilon)} = 2^{-n(H(X)-\epsilon)}|A_\epsilon^{(n)}|$$

čime smo dokazali četvrto svojstvo. □

U kontekstu komunikacijskih sustava, izvor informacije predstavlja "generator" simbola iz nekog skupa $\mathcal{X}$ gdje se svaki simbol tog skupa generira s određenom vjerojatnošću. Dakle, izvor možemo reprezentirati slučajnom varijablom X s distribucijom p koja poprima vrijednosti u $\mathcal{X}$. Tada je entropija izvora jednaka $H(X)$. Naivno kodiranje nezavisnog niza slučajnih varijabli $X_1, \ldots, X_n$, $X_i \sim X$, podrazumijeva (bijektivno) preslikavanje iz skupa $\mathcal{A} \subseteq \mathcal{X}^n$ u indeksni skup $\mathcal{I} = \{1, 2, \ldots, M\}$, gdje je $M = |\mathcal{A}|$. Kao veličinu koja mjeri koliko je ovo kodiranje efikasno možemo uzeti tzv. *stopu kodiranja* $\frac{\log M}{n}$, prosječan broj bitova potrebnih da se kodira izvorni simbol. Na izlazu, dekoder pretvara dobiveni indeks natrag u originalni niz simbola. Primijetimo da ukoliko je $|\mathcal{A}| < |\mathcal{X}|^n$, nećemo moći točno dekodirati sve nizove $x^n = (x_1, \ldots, x_n) \in \mathcal{X}^n$. Pošto za nizove iz skupa $\mathcal{A}$ znamo da ih možemo točno dekodirati, vjerojatnost pogreške pri dekodiranju jednaka je $P_e = \mathbb{P}((X_1, \ldots, X_n) \notin \mathcal{A}) = \mathbb{P}(X^n \notin \mathcal{A})$. Sljedeći teorem govori da čak i ovakvo naivno kodiranje ima optimalnu stopu kodiranja uz proizvoljnu malu vjerojatnost pogreške kada kodiramo vrlo dugačke nizove.

Teorem 1.2.10 (Teorem o kodiranju izvora)**.** *Neka je X slučajna varijabla s entropijom $H(X)$, te neka je $X_1, \ldots, X_n$ niz nezavisnih jednako distribuiranih slučajnih varijabli, $X_i \sim X$, $i = 1, 2, \ldots, n$. Tada za svaki $\epsilon > 0$ postoji kodiranje niza čija stopa kodiranja zadovoljava $\left|\frac{\log M}{n} - H(X)\right| \leq \epsilon$ i čija je vjerojatnost pogreške $P_e < \epsilon$, kada $n \to \infty$. Obratno, neka je $\zeta > 0$. Tada za svako kodiranje niza sa stopom kodiranja manjom od $H(X) - \zeta$ vrijedi $P_e \to 1$, kada $n \to \infty$.*

Dokaz. Neka je $\epsilon > 0$ proizvoljan. Ideja je kodirati samo nizove iz tipičnog skupa $A_\epsilon^{(n)} \subseteq \mathcal{X}^n$, dok ostale možemo preslikati u bilo koji fiksni kod. Po teoremu 1.2.9 postoji $n \in \mathbb{N}$ takav da je $\mathbb{P}\left(X^n \in A_\epsilon^{(n)}\right) > 1 - \epsilon$. Kako je $M = |A_\epsilon^{(n)}|$, opet po teoremu 1.2.9 slijedi da je

$$(1 - \epsilon)2^{n(H(X)-\epsilon)} \leq M = |A_\epsilon^{(n)}| \leq 2^{n(H(X)+\epsilon)}.$$

Sada za stopu kodiranja $\frac{\log M}{n}$ vrijedi

$$\frac{1}{n}\log(1 - \epsilon) + H(X) - \epsilon \leq \frac{\log M}{n} \leq H(X) + \epsilon.$$

Puštanjem $n \to \infty$ dobivamo traženu tvrdnju. Nadalje, greška ovog kodiranja dana je s

$$P_e = \mathbb{P}\left(X^n \notin A_\epsilon^{(n)}\right) = 1 - \mathbb{P}\left(X^n \in A_\epsilon^{(n)}\right) < \epsilon.$$

Time je dokazana prva tvrdnja. Obratno, neka je $\zeta > 0$ proizvoljan i neka je C kodiranje niza čija je stopa kodiranja manja od $H(X) - \zeta$. Tada vrijedi $M \leq 2^{n(H(X)-\zeta)}$. Dakle, tada možemo odabrati najviše $2^{n(H(X)-\zeta)}$ nizova iz skupa $\mathcal{X}^n$ koje ćemo točno dekodirati. Označimo s $\mathcal{A}$ skup koji sadrži te nizove. Primijetimo da $\mathcal{A}$ može sadržavati i nizove iz

tipičnog skupa i one koji nisu u tipičnom skupu $A_\epsilon^{(n)}$. Tada je za svaki $\epsilon > 0$ i dovoljno velik $n \in \mathbb{N}$:

$$\begin{aligned}\mathbb{P}(X^n \in \mathcal{A}) &= \mathbb{P}\left(X^n \in \mathcal{A}, X^n \in A_\epsilon^{(n)}\right) + \mathbb{P}\left(X^n \in \mathcal{A}, X^n \notin A_\epsilon^{(n)}\right) \\ &\leq 2^{n(H(X)-\zeta)} 2^{-n(H(X)-\epsilon)} + \mathbb{P}\left(X^n \notin A_\epsilon^{(n)}\right) \qquad (1.4) \\ &< 2^{-n(\zeta-\epsilon)} + \epsilon, \qquad (1.5)\end{aligned}$$

gdje nejednakosti (1.4) i (1.5) slijede iz teorema 1.2.9. Greška ovog kodiranja je

$$P_e = \mathbb{P}(X^n \notin \mathcal{A}) = 1 - \mathbb{P}(X^n \in \mathcal{A}) > 1 - (2^{-n(\zeta-\epsilon)} + \epsilon).$$

Uzmimo $\epsilon < \zeta$. Tada je $2^{-n(\zeta-\epsilon)} < \epsilon$ za dovoljno velik n. Stoga je $P_e > 1 - 2\epsilon$, kada $n \to \infty$. Kako ovo vrijedi za svaki $\epsilon \in \langle 0, \zeta \rangle$, to dokazuje drugu tvrdnju. □

Ovaj teorem pruža nam teoretski ideal kompresije podataka bez gubitaka. Nadalje, pokazuje kako je suštinska količina informacije nekog izvora upravo njegova entropija. No, koliko god rezultati ovog teorema bili fascinantni, on ima jednu veliku manu. Naime, svi rezultati vrijede samo asimptotski, a u praksi nemamo luksuz slati poruke proizvoljne duljine. Dakle, teorem nam garantira postojanje efikasnog kodiranja, no ne sadrži način konstrukcije takvog kodiranja.

Primjer 1.2.11. *Neka je $X_1, X_2, \ldots, X_n$ niz nezavisnih Bernoullijevih slučajnih varijabli t.d. $p(0) = 0.2, p(1) = 0.8$ i neka je $\epsilon = 0.05$. Entropija je tada jednaka $H(X) \approx 0.722$. Pogledajmo kako izgledaju tipični skupovi za različite duljine nizova takvih varijabli. Za $n = 25$, računanjem vjerojatnosti svih nizova dobijemo da jedino nizovi s 20 jedinica upadaju u tipični skup. Primijetimo kako je očekivani broj jedinica ovdje upravo $np = 20$. Vjerojatnost svakog takvog niza je $0.8^{20} \cdot 0.2^5 = 3.689 \cdot 10^{-6}$, a ima ih $\binom{25}{20} = 53130$. To znači da je $\mathbb{P}\left(A_{0.05}^{(25)}\right) = 53130 \cdot 3.689 \cdot 10^{-6} \approx 0.196$. Vidimo da duljina niza nije dovoljno velika da bi vrijedilo $\mathbb{P}\left(A_\epsilon^{(n)}\right) > 1 - \epsilon = 0.95$. Štoviše, prvi n za koji to vrijedi je $n = 969$. Sada u tipični skup upadaju svi nizovi s brojem jedinica iz skupa $\{751, 752, \ldots, 799\}$. Tehnički, oni nisu jednako vjerojatni, ali gornja ograda za razliku dviju vjerojatnosti je $2^{-nH(X)} \cdot (2^\epsilon - 2^{-\epsilon})$ što je reda veličine 10^{-213}. Slijedi da je $|A_{0.05}^{(969)}| = \sum_{i=751}^{799} \binom{969}{i}$ što je otprilike reda veličine 10^{223}, odnosno 2^{740}, dok je $\mathbb{P}\left(A_{0.05}^{(969)}\right) \approx 0.951$. Iako se radi o ogromnim brojevima, nemojmo previdjeti činjenicu da je $A_{0.05}^{(969)}$ otprilike 2^{229} puta manji skup od skupa svih binarnih nizova duljine 969 čiji je kardinalitet 2^{969}. Za kraj, spomenimo još samo kako niz s najvećom vjerojatnosti $x_{max} = (1, 1, \ldots, 1)$ nije u tipičnom skupu. Naime, tipični skup možemo interpretirati kao skup svih nizova koji nam daju informaciju otprilike jednaku prosječnoj informaciji izvora, tj. entropiji. Zbog toga tipični skupovi nizova kakve smo promatrali u ovom primjeru uvijek sadrže nizove s np jedinica (kada je to cijeli broj) jer je to upravo očekivani broj jedinica.*

Primjer 1.2.12 (Huffmanovo kodiranje)**.** *Za kraj, pogledajmo jedno praktično kodiranje, tzv. Huffmanovo kodiranje. Radi se o kodiranju koje se temelji na "bottom-up" konstrukciji binarnog stabla pomoću prioritetnog reda gdje veći prioritet imaju simboli s manjom vjerojatnošću. Algoritam izgleda otprilike ovako:*

Algorithm 1 Huffmanovo kodiranje

1: kreiraj list stabla za svaki simbol i dodaj ga u prioritetni red
2: **while** postoji više od jednog čvora u redu **do**
3: ukloni dva čvora s najmanjim vjerojatnostima iz reda
4: kreiraj novi čvor stabla kojem su ta dva čvora djeca i koji ima vjerojatnost jednaku zbroju vjerojatnosti djece
5: dodaj novi čvor u red
6: **end while**
7: preostali čvor je korijen stabla i stablo je gotovo

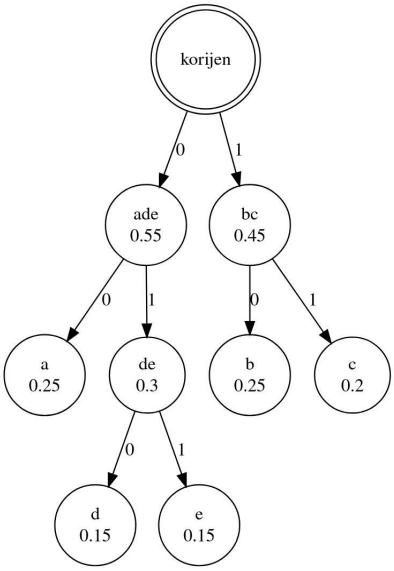
Preostaje još samo dodijeliti pripadnu kodnu riječ svakom simbolu. To radimo tako da, počevši od korijena stabla, djecu označimo s 0 i 1 te nastavljamo tako što za svaki sljedeći par djece jednom djetetu na kraj kodne riječi roditelja dodamo 0, a drugom 1 (konvencija je lijevom djetetu uvijek dodati 0). Vidimo da ovaj proces nije jedinstven. Sprovedimo ovaj algoritam za varijablu

$$X \sim \begin{pmatrix} a & b & c & d & e \\ 0.25 & 0.25 & 0.2 & 0.15 & 0.15 \end{pmatrix}.$$

Na kraju je naše preslikavanje dano u sljedećoj tablici:

x_i	p_i	$c(x_i)$
a	0.25	00
b	0.25	10
c	0.2	11
d	0.15	010
e	0.15	011

Entropija ove slučajne varijable je $H(X) = 2.2855$ bitova, dok je prosječna duljina kodnih riječi jednaka 2.3 bita. Primijetimo kako je ovako dobiveno kodiranje ujedno i prefiksno. Ova klasa kodiranja ima veliku prednost u odnosu na kodiranja koja se mogu dekodirati na jedinstven način. Naime, čitajući niz dobivenih kodiranih simbola s lijeva na desno, čim dođemo do niza simbola koji odgovara kodnoj riječi nekog simbola, možemo sa sigurnošću reći koji simbol je u pitanju, dok kod kodiranja koja se mogu dekodirati na jedinstven način moramo uzeti u obzir i iduće simbole u nizu.



Slika 1.6: Prikaz dobivenog binarnog stabla

1.3 Kodiranje kanala

U prethodnom poglavlju vidjeli smo kako uklanjanjem redundantnosti možemo komprimirati podatke i time biti efikasniji u procesu digitalne komunikacije. No, u cijeloj toj priči potpuno smo izostavili jednu od najvećih prepreki u komunikaciji uopće: šum. U realnim okolnostima nema garancije da će naš signal (kodirani simboli) stići do odredišta "netaknut", tj. nepromijenjen. Iako to često ne možemo direktno vidjeti, proces slanja signala je svejedno fizički proces koji nije savršen kao naš matematički model pa tako dijelovi signala mogu postati korumpirani ili se čak u potpunosti izgubiti. Kako u takvim uvjetima osigurati pouzdanu komunikaciju? Rješenje je, pomalo ironično, dodati redundantnost.

Definicija 1.3.1. Diskretni kanal *definiramo kao sustav koji se sastoji od ulaznog skupa $\mathcal{X}$, izlaznog skupa $\mathcal{Y}$ te skupa uvjetnih vjerojatnosti $p(y|x)$ koje označavaju vjerojatnosti izlaznog simbola y za poslani ulazni simbol x, $x \in \mathcal{X}, y \in \mathcal{Y}$. Za kanal kažemo da je* bez memorije *ako izlaz ovisi samo o trenutnom ulazu, tj.*

$$\mathbb{P}(Y_i = y_i | X_i = x_i, X_{i-1} = x_{i-1}, \dots) = \mathbb{P}(Y_i = y_i | X_i = x_i).$$

Definicija 1.3.2. Kapacitet *diskretnog kanala bez memorije definiran je kao*

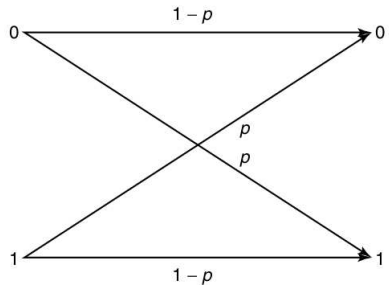
$$C = \max_{p(x)} I(X;\ Y),$$

gdje maksimum uzimamo po svim ulaznim distribucijama $p(x)$.

Primjer 1.3.3 (Binarni simetrični kanal)**.** *Neka su $\mathcal{X} = \mathcal{Y} = \{0, 1\}$, odnosno šaljemo i primamo binarne znamenke, te neka se ulazni bitovi invertiraju s vjerojatnošću p, a ostanu ispravni s vjerojatnošću $1 - p$. Konkretno, kada pošaljemo 0, s vjerojatnošću p će na odredište stići 1 i obratno. Na odredištu ne možemo znati kada je došlo do pogreške pa su na neki način svi bitovi sumnjivi. Računamo kapacitet ovakvog kanala:*

$$\begin{aligned} I(X;Y) &= H(Y) - H(Y|X) = H(Y) - \sum p(x)H(Y|X = x) \\ &= H(Y) - \sum p(x)H(p, 1-p) = H(Y) - H(p, 1-p) \leq 1 - H(p, 1-p), \end{aligned}$$

gdje posljednja nejednakost slijedi zbog činjenice da je Y binarna slučajna varijabla. Jednakost se postiže za uniformnu distribuciju ulaza pa je kapacitet ovakvog kanala $C = 1 - H(p, 1-p)$. Uočimo kako je kapacitet kanala jednak 0 za $p = \frac{1}{2}$ jer je tada svaki primljeni bit ispravan ili pogrešan s jednakom vjerojatnošću pa ustvari primamo nasumičan niz bitova. Nadalje, primijetimo kako kapacitet poprima maksimalnu vrijednost 1 za $p = 0$, ali i za $p = 1$. Naime, ako su svi bitovi invertirani, opet su primljeni bitovi deterministički određeni kao i da su svi ispravni.

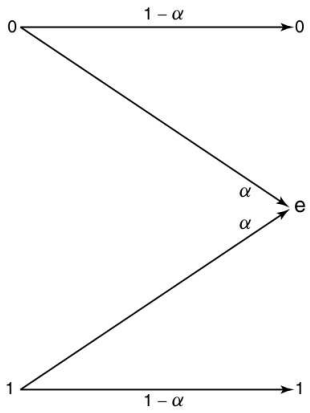


Slika 1.7: Prikaz binarnog simetričnog kanala

Primjer 1.3.4 (Binarni brišući kanal)**.** *Promotrimo sada analogon binarnog simetričnog kanala. Neka je sada $\mathcal{X} = \{0, 1\}$ te $\mathcal{Y} = \{0, 1, e\}$. Ovdje se bitovi ne invertiraju s određenom vjerojatnošću, već dio njih jednostavno nikad ne stigne na odredište, no sada znamo koji su to (njihova oznaka na izlazu je e). Za takve bitove kažemo da su izgubljeni ili obrisani. Označimo s α udio izgubljenih bitova te s π vjerojatnost $p(1)$. Tada je kapacitet ovog kanala jednak*

$$\begin{aligned} I(X;\ Y) &= H(Y) - H(Y|X) = H((1-\pi)(1-\alpha), \pi(1-\alpha), \alpha) - \sum_x p(x)H(Y|X = x) \\ &= \cdots = H(\alpha, 1-\alpha) + (1-\alpha)H(\pi, 1-\pi) - H(\alpha, 1-\alpha) = (1-\alpha)H(\pi, 1-\pi). \end{aligned}$$

Maksimum se očito postiže za $\pi = \frac{1}{2}$, te slijedi da je $C = 1 - \alpha$. U većini današnjih komunikacijskih kanala prisutan je mehanizam pomoću kojeg prijamnik može slati povratnu informaciju odašiljaču. Time se mogu popraviti greške u ovakvim kanalima tako da se izgubljeni bit jednostavno opet zatraži i tako sve dok ne stigne. Zanimljivo je da kapacitet kanala u kojem izgubljeni bitovi nisu posebno naznačeni nije poznat, već postoje samo gornje i donje ograde (vidi [5], [18] i [22]).



Slika 1.8: Prikaz binarnog brišućeg kanala

Definicija 1.3.5. (M, n) kôd *diskretnog kanala* $(\mathcal{X}, \mathcal{Y}, p(x|y))$ *sastoji se od:*

1. *indeksnog skupa* $\{1, 2, \ldots, M\}$.
2. *kodiranja* $X^n : \{1, 2, \ldots, M\} \to \mathcal{X}^n$.
3. *dekodiranja* $g : \mathcal{Y}^n \to \{1, 2, \ldots, M\}$*, koje deterministički dodijeljuje vrijednost svakom primljenom signalu.*

Definicija 1.3.6. Stopa *R (M,n) kôda definirana je s*

$$R = \frac{\log M}{n}.$$

Sada smo spremni iskazati drugi glavni rezultat teorije informacija, tzv. teorem o kodiranju kanala. Iako idejno vrlo sličan dokazu teorema o kodiranju izvora, dokaz ovog teorema je mnogo kompliciraniji pa ćemo ga izostaviti. Temelji se na korištenju analogona tipičnog skupa iz prošlog poglavlja, nasumičnog odabira (M, n) kôda te ograničavanju vjerojatnosti pogreške. Znatiželjni čitatelj može čitavi dokaz pronaći u [6].

Teorem 1.3.7 (Teorem o kodiranju kanala)**.** *Za diskretni kanal bez memorije, dostižne su sve stope $R \leq C$. Točnije, za svaku stopu $R < C$, postoji niz $(2^{nR}, n)$ kodova s maksimalnom vjerojatnošću pogreške $\lambda^{(n)} \to 0$.*
Obratno, za svaki niz $(2^{nR}, n)$ kodova takvih da $\lambda^{(n)} \to 0$ vrijedi $R \leq C$.

Iako smo u prethodnim primjerima vidjeli da pogreške pri odašiljanju signala smanjuju kapacitet kanala, teorem garantira postojanje kodiranja kojim ćemo se moći proizvoljno približiti stopi odašiljanja jednakoj tom smanjenom kapacitetu uz proizvoljno malu vjerojatnost greške. Nažalost, kako je dokaz vrlo sličan teoremu o kodiranju izvora, tako i ovdje imamo samo egzistenciju traženog kodiranja te rezultat vrijedi asimptotski, što znači da je potrebno slati nizove simbola vrlo velike duljine. Opet, u praksi nemoguće. No, to nije spriječilo mnoge istraživače da nedugo nakon objave ovog teorema počnu razvijati sve bolja i bolja kodiranja za ispravljanje greški.

Primjer 1.3.8 (Hammingovo kodiranje)**.** *Jedna od najstarijih kodiranja za ispravljanje greški su kodiranja iz familije Hammingovih kodiranja. Ona funkcioniraju tako što bloku od $2^r - r - 1$ podatkovnih bitova dodaju r tzv. paritetnih bitova na pomno odabrana mjesta, za $r \geq 2$. Neka je $r = 3$. Tada govorimo o Hammingovom $(7, 4)$ kodiranju, tj. bloku od 4 podatkovna bita dodajemo 3 paritetna. To radimo na način da paritetni bitovi stoje na mjestima čiji su indeksi potencije broja 2, dakle 1,2 i 4. Ostala mjesta popune se podatkovnim bitovima.*

001	010	011	100	101	110	111
p_1	p_2	d_1	p_3	d_2	d_3	d_4

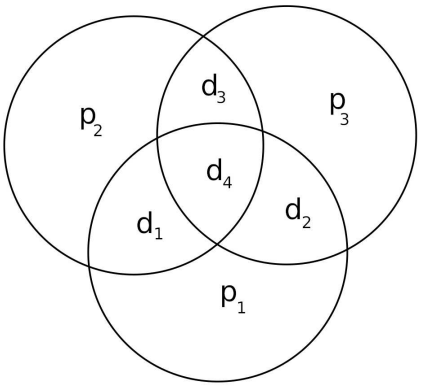
Vrijednosti i-tog paritetnog bita p_i mora biti takva da paritet svih bitova na pozicijama gdje je i-ti najmanje značajan bit jednak 1 bude 0, tj. da ima paran broj jedinica na tim mjestima. To možemo kompaktnije zapisati preko logičkog operatora XOR:

$$p_1 = d_1 \oplus d_2 \oplus d_4.$$
$$p_2 = d_1 \oplus d_3 \oplus d_4.$$
$$p_3 = d_2 \oplus d_3 \oplus d_4.$$

Sada smo dobili 3 kontrolne skupine i znamo da je paritet svake od njih jednak 0. Uvedimo sada kontrolne bitove za te 3 skupine:

$$c_1 = p_1 \oplus d_1 \oplus d_2 \oplus d_4.$$
$$c_2 = p_2 \oplus d_1 \oplus d_3 \oplus d_4.$$
$$c_3 = p_3 \oplus d_2 \oplus d_3 \oplus d_4.$$

Ukoliko bi došlo do greške, barem jedan od ovih bitova bi bio jednak 1. Tada ujedno znamo i koji bit je pogrešan. Naime, to je upravo bit na mjestu s binarnim indeksom $c_3c_2c_1$. *Ekvivalentno, taj indeks možemo dobiti tako što zbrojimo modulo 2 sve indekse gdje se nalaze jedinice. Ovo kodiranje, kao i sva iz Hammingove familije, je ograničeno time što može ispraviti samo pogreške u jednom bitu. Nadalje, može i detektirati pogreške u dva bita, ali takvu pogrešku ne može ispraviti jer ne razlikuje 1-bitne i 2-bitne pogreške. Neka je* $d = 1101$ *blok bitova koji želimo odaslati. Pripadna kodna riječ je tada* 1010101. *Pretpostavimo da je došlo do greške na predzadnjem mjestu, odnosno* d_3 *je invertiran, pa je primljen blok bitova* 1010111. *Sada je* $c_1 = 0, c_2 = 1, c_3 = 1$, *pa je indeks pogreške jednak* $c_3c_2c_1 = 110$. *Primijetimo da bi iste vrijednosti kontrolnih bitova dobili i za greške u bitovima* d_1 *i* d_2, *odnosno za primljeni blok* 1000001. *Za 3 ili više greške može se dogoditi da svi kontrolni bitovi ostanu 0. Zbog toga je ovo kodiranje pogodno u situacijama u kojima rijetko dolazi do pogreške, npr. u RAM memoriji računala (ECC memorija).*



Slika 1.9: Vizualizacija Hammingovog (7,4) kodiranja: paritetni bitovi su postavljeni tako da paritet u svakom krugu bude jednak 0.

1.4 Diferencijalna entropija

U ovom potpoglavlju uvodimo pojam entropije i ostalih izvedenica, ali za neprekidne slučajne varijable. Iako postoje mnoge sličnosti i poveznice s entropijom diskretnih varijabli, postoje i značajne razlike. Tako recimo entropija neprekidne varijable može biti negativna te nije invarijantna na linearne transformacije. Zbog česte pojave Euleorovog broja e u gustoćama poznatih slučajnih varijabli, u ovom potpoglavlju s log označavamo prirodni logaritam zbog jedostavnijeg računa.

Definicija 1.4.1. *Neka je X neprekidna slučajna varijabla i f njena pripadna funkcija gustoće.* Diferencijalna entropija *slučajne varijable X definirana je kao*

$$h(X) = -\int_S f(x)\log f(x)\,dx,$$

gdje je S nosač gustoće f, kada integral postoji.

Kao i u diskretnom slučaju, diferencijalna entropija ovisi samo o gustoći od X pa se ponekad označava i s $h(f)$.

Primjer 1.4.2 (Uniformna distribucija)**.** *Neka je $X \sim U(0,a)$. Gustoća ove slučajne varijable je*

$$f(x) = \begin{cases} \frac{1}{a} & za\ 0 \le x \le a, \\ 0 & inače. \end{cases}$$

Računamo diferencijalnu entropiju:

$$h(X) = -\int_0^a \frac{1}{a}\log\frac{1}{a}dx = \log a,$$

Vidimo da je $h(X) < 0$ za $a < 1$.

Primjer 1.4.3 (Normalna distribucija)**.** *Neka je $X \sim N(\mu,\sigma^2)$. Gustoća ove varijable je dana s*

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

Stoga je diferencijalna entropija jednaka

$$\begin{aligned} h(X) &= -\frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{\infty}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\log\left(\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\right)\mathrm{d}x \\ &= \frac{\sqrt{2}\sigma}{\sigma\sqrt{2\pi}}\int_{-\infty}^{\infty}\exp\left(-t^2\right)\log\left(\sigma\sqrt{2\pi}\exp\left(t^2\right)\right)\mathrm{d}t \\ &= \frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty}\left(\log(\sigma\sqrt{2\pi}) + \log\left(\exp\left(t^2\right)\right)\right)\exp\left(-t^2\right)\mathrm{d}t \\ &\vdots \\ &= \log(\sigma\sqrt{2\pi}) + \frac{1}{2}. \end{aligned}$$

Uočimo kako entropija ne ovisi o očekivanju μ, već samo o varijanci σ^2. To ima smisla jer promjenom parametra μ samo horizontalno translatiramo graf gustoće, dok ga promjenom parametra σ^2 činimo širim ili užim.

Definicija 1.4.4. *Neka je $X_1, X_2, \ldots, X_n$ niz slučajnih varijabli takvih da je $(X_1, X_2 \ldots, X_n)$ neprekidan slučajan vektor sa zajedničkom gustoćom $f(x_1, x_2, \ldots, x_n)$. Njihova* zajednička diferencijalna entropija *dana je s*

$$h(X_1, X_2, \ldots, X_n) = -\int_{x^n \in \mathbb{R}^n} f(x^n) \log f(x^n)\, dx^n.$$

Teorem 1.4.5. *Neka je X n-dimenzionalna normalna slučajna varijabla s vektorom očekivanja $\boldsymbol{\mu}$ i kovarijacijskom matricom $\boldsymbol{\Sigma}$, tj. $X \sim N_n(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Tada je*

$$h(X) = \frac{1}{2} \log \left[(2\pi e)^n |\boldsymbol{\Sigma}|\right],$$

gdje je $|\boldsymbol{\Sigma}|$ determinanta matrice $\boldsymbol{\Sigma}$.

Dokaz. Vidi [6], str. 249. □

Definicija 1.4.6. *Neka su X i Y slučajne varijable te $f(x, y)$ njihova zajednička funkcija gustoće.* Uvjetna diferencijalna entropija *dana je s*

$$h(X|Y) = -\int f(x, y) \log f(x|y)\, dx\, dy.$$

Kako je $f(x|y) = \frac{f(x,y)}{f_Y(y)}$, vrijedi $h(X|Y) = h(X, Y) - h(Y)$, isto kao u diskretnom slučaju.

Definicija 1.4.7. Relativna entropija *ili* Kullback-Leiblerova divergencija *dviju funkcija gustoća f i g definirana je s*

$$D_{KL}(f \parallel g) = \int f(x) \log \frac{f(x)}{g(x)} dx.$$

Primijetimo da je $D_{KL}(f \parallel g)$ konačan samo kada je nosač od f sadržan u nosaču od g. Ovdje je isto konvencija staviti $0 \log \frac{0}{0} = 0$.

Definicija 1.4.8. *Neka su X i Y slučajne varijable te $f(x, y)$ njihova zajednička gustoća. Njihova* uzajamna informacija *definirana je s*

$$I(X;\ Y) = \int f(x, y) \log \frac{f(x, y)}{f(x) f(y)} dx\, dy.$$

Iz definicije slijedi da je $I(X;\ Y) = h(X) - h(X|Y) = h(Y) - h(Y|X) = h(X) + h(Y) - h(X, Y)$.

Primjer 1.4.9. *Neka su X i Y dvije normalne slučajne varijable s proizvoljnim očekivanjima, zajedničkom varijancom σ^2 te koeficijentom korelacije ρ. Tada je kovarijacijska matrica jednaka*

$$\Sigma = \begin{bmatrix} \sigma^2 & \rho\sigma^2 \\ \rho\sigma^2 & \sigma^2 \end{bmatrix}.$$

Već od prije znamo da je tada $h(X) = h(Y) = \log(\sigma\sqrt{2\pi}) + \frac{1}{2}$, te $h(X,Y) = \frac{1}{2}\log\left[(2\pi e)^2|\Sigma|\right]$ $= \frac{1}{2}\log\left[(2\pi e)^2\sigma^4(1-\rho)\right]$. Dakle, imamo da je

$$I(X;\ Y) = h(X) + h(Y) - h(X,Y) = -\frac{1}{2}\log(1-\rho^2).$$

Za $\rho = 0$, varijable su nezavisne pa je uzajamna informacija jednaka 0. Za $\rho = \pm 1$, varijable su u potpunosti korelirane pa je uzajamna informacija beskonačna.

Teorem 1.4.10 (Svojstvo asimptotske ekviparticije za neprekidne varijable)**.** *Neka je $X_1, X_2, \ldots, X_n$ niz nezavisnih jednako distribuiranih slučajnih varijabli s funkcijom gustoće $f_X(x)$. Tada vrijedi*

$$-\frac{1}{n}\log f(X_1, X_2, \ldots, X_n) \xrightarrow{\mathbb{P}} h(X).$$

Dokaz. Analogno kao u diskretnom slučaju, dokaz slijedi direktno iz slabog zakona velikih brojeva te činjenice da su funkcije nezavisnih varijabli i same nezavisne:

$$\begin{aligned} -\frac{1}{n}\log f(X_1, X_2, \ldots, X_n) &= -\frac{1}{n}\log\left(\prod_{i=1}^{n} f_X(X_i)\right) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\log f_X(X_i) \xrightarrow{\mathbb{P}} -\mathbb{E}\left[\log f_X(X)\right] = h(X). \end{aligned}$$

□

Definicija 1.4.11. *Neka je $X_1, X_2, \ldots, X_n$ niz nezavisnih jednako distribuiranih slučajnih varijabli s gustoćom $f(x)$ te neka su $\epsilon > 0$ i $n \in \mathbb{N}$.* Tipični skup *za danu distribuciju definiran je s*

$$A_\epsilon^{(n)} = \left\{(x_1, x_2, \ldots, x_n) \in S^n \ :\ \left|-\frac{1}{n}\log f(x_1, x_2, \ldots, x_n) - h(X)\right| \leq \epsilon\right\}.$$

Primijetimo da je ovo identična definicija kao i u diskretnom slučaju kada bismo koristili logaritam s bazom 2. Sada ćemo vidjeti da tipični skup zadovoljava analogna svojstva onima iz diskretnog slučaja. Ovdje ulogu kardinaliteta preuzima Lebesgueova mjera skupa.

Teorem 1.4.12. *Tipični skup $A_\epsilon^{(n)}$ zadovoljava sljedeća svojstva:*

1. $\mathbb{P}\left((X_1, \ldots, X_n) \in A_\epsilon^{(n)}\right) > 1 - \epsilon$ *za n dovoljno velik.*
2. $\lambda\left(A_\epsilon^{(n)}\right) \leq e^{n(h(X)+\epsilon)}$ *za svaki n.*
3. $\lambda\left(A_\epsilon^{(n)}\right) \geq (1-\epsilon)e^{n(h(X)-\epsilon)}$ *za n dovoljno velik.*

Dokaz. Po teoremu 1.4.10 imamo $-\frac{1}{n}\log f(X_1, X_2, \ldots, X_n) \xrightarrow{\mathbb{P}} h(X)$. Prvo svojstvo sada slijedi iz definicije konvergencije po vjerojatnosti i definicije tipičnog skupa. Nadalje, imamo

$$\begin{aligned}
1 &= \int_{S^n} f(x_1, x_2, \ldots, x_n)\, dx_1 dx_2 \cdots dx_n \\
&\geq \int_{A_\epsilon^{(n)}} f(x_1, x_2, \ldots, x_n)\, dx_1 dx_2 \cdots dx_n \\
&\geq \int_{A_\epsilon^{(n)}} e^{-n(h(X)+\epsilon)}\, dx_1 dx_2 \cdots dx_n \\
&= e^{-n(h(X)+\epsilon)} \int_{A_\epsilon^{(n)}} dx_1 dx_2 \cdots dx_n \\
&= e^{-n(h(X)+\epsilon)} \lambda\left(A_\epsilon^{(n)}\right),
\end{aligned}$$

čime je dokazano drugo svojstvo. Za dokaz trećeg svojstva, neka je n dovoljno velik tako da vrijedi prvo svojstvo. Tada je

$$\begin{aligned}
1 - \epsilon < \mathbb{P}\left((X_1, \ldots, X_n) \in A_\epsilon^{(n)}\right) &= \int_{A_\epsilon^{(n)}} f(x_1, x_2, \ldots, x_n)\, dx_1 dx_2 \cdots dx_n \\
&\leq \int_{A_\epsilon^{(n)}} e^{-n(h(X)-\epsilon)}\, dx_1 dx_2 \cdots dx_n \\
&= e^{-n(h(X)-\epsilon)} \int_{A_\epsilon^{(n)}} dx_1 dx_2 \cdots dx_n \\
&= e^{-n(h(X)-\epsilon)} \lambda\left(A_\epsilon^{(n)}\right).
\end{aligned}$$

□

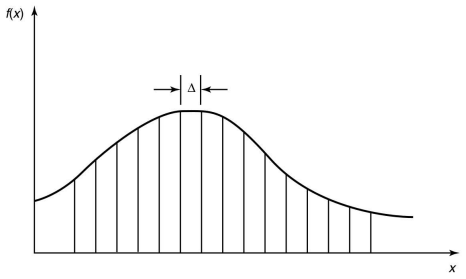
Dakle, Lebesgueova mjera tipičnog skupa je otprilike $e^{nh(X)}$. Također, može se pokazati da je tipični skup s najmanjim volumenom (tj. Lebesgueovom mjerom) koji sadrži većinu vjerojatnosti. Kako se radi o n-dimenzionalnom volumenu, prosječna "duljina stranice" je tada $e^{h(X)}$. To nam daje interpretaciju diferencijalne entropije kao logaritam prosječne duljine stranice najmanjeg skupa koji sadrži većinu informacije.

Za kraj poglavlja, ispitajmo odnos diskretne i diferencijalne entropije. Neka je X neprekidna slučajna varijabla s gustoćom $f(x)$. Podijelimo njen raspon (sliku) na podintervale duljine Δ. Pretpostavimo da je f neprekidna na svakom podintervalu. Tada po Lagrangeovom teoremu srednje vrijednosti postoji x_i u svakom podintervalu takav da je

$$f(x_i)\Delta = \int_{i\Delta}^{(i+1)\Delta} f(x)\,dx.$$

Sada definiramo diskretizaciju naše slučajne varijable X na način da ona poprima upravo vrijednost x_i na svakom podintervalu, tj.

$$X^\Delta = x_i,\ i\Delta \leq X < (i+1)\Delta.$$



Slika 1.10: Podjela raspona od X na podintervale jednake duljine Δ

Označimo s p_i vjerojatnost $\mathbb{P}\left(X^\Delta = x_i\right)$. Tada je

$$p_i = \int_{i\Delta}^{(i+1)\Delta} f(x)\,dx = f(x_i)\Delta.$$

Entropija naše diskretizirane slučajne varijable je sada jednaka:

$$\begin{aligned} H(X^\Delta) &= -\sum_{-\infty}^{\infty} p_i \log p_i \\ &= -\sum_{-\infty}^{\infty} f(x_i)\Delta \log(f(x_i)\Delta) \\ &= -\sum_{-\infty}^{\infty} \Delta f(x_i) \log f(x_i) - \sum_{-\infty}^{\infty} f(x_i)\Delta \log \Delta \\ &= -\sum_{-\infty}^{\infty} \Delta f(x_i) \log f(x_i) - \log \Delta, \end{aligned} \tag{1.6}$$

gdje posljednja jednakost slijedi zbog $\sum f(x_i)\Delta = \int f(x) = 1$. Ako je $f(x)\log f(x)$ Riemman-integrabilna, prvi član u (1.6) teži prema integralu od $-f(x)\log f(x)$ kada $\Delta \to 0$. Upravo smo dokazali sljedeći teorem.

Teorem 1.4.13. *Ako je gustoća $f(x)$ neprekidne slučajne varijable X Riemman-integrabilna, tada vrijedi*

$$H(X^{\Delta}) + \log\Delta \to h(f) = h(X), \ \textit{kada}\ \Delta \to 0.$$

Poglavlje 2

Duboko učenje

Inteligencija je jedan od onih pojmova koje je teško precizno definirati, ali svatko od nas ima vrlo dobro intuitivno shvaćanje tog pojma. Potencijalno stvaranje inteligentnih strojeva fasciniralo je ljude još u doba stare Grčke. Akumulacijom znanja i boljim razumijevanjem procesa učenja u biološkim organizmima, sredinom 20. stoljeća došlo se do ideje da strojevi "uče" oponašanjem mozga. Temeljna građevna jedinica živčanog sustava je živčana stanica ili *neuron*, a velika većina ih se nalazi upravo u mozgu koji je glavni organ središnjeg živčanog sustava. Neuroni međusobno komuniciraju slanjem električnih impulsa i na taj način tvore veze koje pak čine mreže. Izlaganjem podražajima te veze postaju jače ili slabije i to je zapravo ključni mehanizam učenja. Pojednostavljeni model ovog sustava temelj je grane strojnog učenja zvane duboko učenje. Iako se izraz odnosi na općenit pristup učenju putem više[1] razina apstrakcije, velika većina modela bazirana je upravo na neuronskim mrežama. Kao što je već spomenuto, začeci ove grane sežu u sredinu 20. stoljeća, točnije u četrdesete godine, dok su se mnoge metode koje su i danas u upotrebi razvile kroz nadolazeća desetljeća. No, arhitekture dubokog učenja su u masovnoj upotrebi tek zadnjih petnaestak godina. Primarni razlozi toga su nedavni pristup ogromnim količinama podataka u sve više digitaliziranom dobu te razvoj boljeg hardvera s većom računskom moći. Obje ove stvari su ključne u procesu učenja (treniranju) modela. Današnji modeli dubokog učenja postižu najbolje rezultate kod mnogih problema strojnog učenja, posebice u područjima računalnog vida i obrade prirodnog jezika.

Pretpostavlja se da je čitatelj upoznat s osnovama strojnog učenja zbog čega su izostavljene formalne definicije mnogih pojmova. Dobar osnovni pregled može se naći u [12].

[1] Postoji nekoliko definicija dubine modela te ne postoji konsenzus oko toga kolika bi ona trebala biti da se model smatra "dubokim" (vidi [9],str. 7-8).

2.1 Umjetni neuron

Najranije i najjednostavnije arhitekture neuronskih mreža sastavljene su od jednog jedinog neurona. Osim što je neuron okosnica kompliciranijih modela, sam po sebi ima značajnu predikcijsku moć. Štoviše, neke poznate tehnike klasičnog strojnog učenja, kao npr. logistička i linearna regresija, mogu se reprezentirati kao mreže s jednim neuronom.

McCulloch-Pitts neuron

Prvi model umjetnog neurona uveli su McCulloch i Pitts 1943. godine. U [15] pokazali su da se pomoću njihovog modela neurona (MCP) mogu vršiti razne logičke operacije. Iako je vrlo brzo postao irelevantan u praktičnom smislu, MCP neuron ima veliku povijesnu važnost kao rani pokušaj formalizacije rada bioloških neurona. Model se sastoji od:

- m binarnih ulaznih signala $x_1, x_2, \ldots, x_m \in \{0, 1\}$.
- m pripadnih težina $\omega_1, \omega_2, \ldots, \omega_m \in \{-1, 0, 1\}$.
- aktivacijske funkcije $f : \{0, 1\}^m \to \{0, 1\}$.
- granične vrijednosti $t \in \mathbb{Z}$.
- izlaznog signala $y \in \{0, 1\}$, takvog da je $y = f(x_1, \ldots, x_m)$.

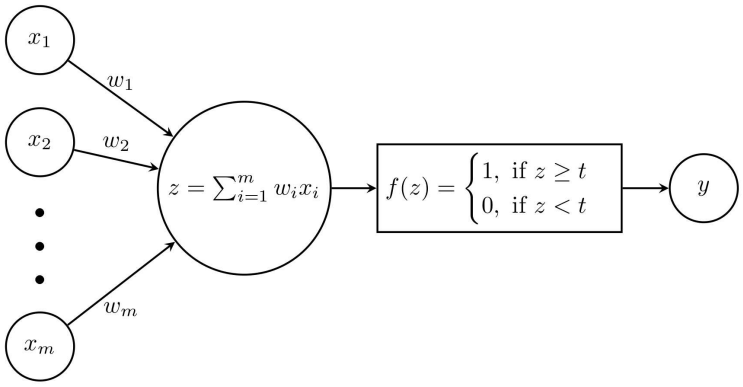
Vrijednost izlaznog signala dana je s

$$y = \begin{cases} 1, & \text{kada je } \sum_{i=1}^{m} \omega_i x_i \geq t \\ 0, & \text{inače.} \end{cases}$$

U donjoj tablici dane su potrebne vrijednosti parametara ω_i i t za izvedbu logičkih operacija konjunkcije ($\wedge$) i disjunkcije ($\vee$) u slučaju m binarnih ulaznih signala, odnosno za izvedbu logičke negacije ($\neg$) u slučaju jednog signala.

	$\wedge$	$\vee$	$\neg$
ω_i	1	1	-1
t	m	1	0

Uočimo kako je do ovih vrijednosti ”ručno” došao čovjek koji ih je potom samo unio u model. Drugim riječima, u MCP neuronu ne postoji *mehanizam učenja*.



Slika 2.1: Skica MCP neurona.

Perceptron

Najraniji binarni klasifikator u kontekstu neuronskih mreža je Rosenblattov perceptron iz 1957. godine ([17]). Unatoč tome što je perceptron uvelike baziran na MCP neuronu, sada se nalazimo u kontekstu nadziranog učenja pa moramo uvesti neke promjene. Za početak, sada postoji skup za treniranje čiji je svaki element oblika $(\mathbf{x}, y)$, gdje je $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ vektor ulaznih signala (kovarijata), a $y \in \{-1, 1\}$ oznaka pripadne klase. Osim toga, vrijednosti ulaznih signala x_i kao i pripadnih težina ω_i više nisu ograničene. Granična vrijednost t iz MCP modela neurona ovdje se može uvesti dodavanjem dodatnog konstantnog ulaza $x_0 = -1$, što odgovara tzv. članu pristranosti (eng. *bias term*) kojeg nakon ovog dijela rada više nećemo eksplicitno označavati. Aktivacijska funkcija u ovom slučaju je funkcija signum, definirana s:

$$\operatorname{sign}(x) = \begin{cases} 1 & \text{za } x \geq 0, \\ -1 & \text{za } x < 0. \end{cases}$$

Stoga je izlazna vrijednost jednaka $\hat{y} = \operatorname{sign}(\sum_{i=0}^{n} \omega_i x_i) = \operatorname{sign}(\sum_{i=1}^{n} \omega_i x_i - \omega_0)$, odnosno

$$\hat{y} = \begin{cases} 1 & \text{za } \sum_{i=1}^{n} \omega_i x_i \geq \omega_0, \\ -1 & \text{za } \sum_{i=1}^{n} \omega_i x_i < \omega_0. \end{cases}$$

Oznaka je sugestivna jer izlazna vrijednost predstavlja predviđenu vrijednost stvarne klase y. Ono po čemu se ovaj model najviše razlikuje od svog prethodnika je sposobnost učenja, što ovdje konkretno podrazumijeva korištenje algoritma za ažuriranje težina ω_i koje se na početku inicijaliziraju na nasumične vrijednosti. Zatim se iterativno prolazi po svim elementima skupa za treniranje u nasumičnom poretku i za svaki element vektor težina $\mathbf{w}$

se ažurira na sljedeći način:

$$\mathbf{w} = \mathbf{w} + \alpha(y - \hat{y})\mathbf{x},$$

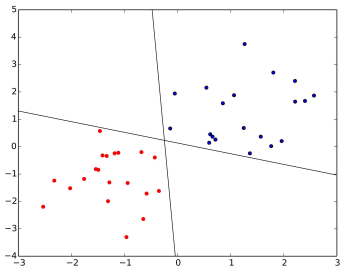
gdje je $\alpha > 0$ stopa učenja, parametar kojim kontroliramo magnitudu promjena težina. Jedan prolaz po čitavom skupu nazivamo epoha. Primijetimo da se težine ažuriraju samo u slučaju da je predviđena klasa pogrešna, tj. $y \neq \hat{y}$. Algoritam staje kada u epohi više nema pogrešnih klasifikacija. Pitamo se za kakve skupove za treniranje će ovaj algoritam konvergirati. U odgovoru na to pitanje može nam pomoći činjenica da je granica odluke modela dana jednadžbom $\sum_{i=1}^{n} \omega_i x_i = \omega_0$, što odgovara jednadžbi hiperravnine u n-dimenzionalnom euklidskom prostoru.

Definicija 2.1.1. *Za skupove točaka A i $B \subseteq \mathbb{R}^n$ kažemo da su linearno separabilni ako postoji $n + 1$ realnih brojeva $\omega_1, \omega_2, \ldots, \omega_n, k$ takvih da za svaku točka skupa A vrijedi $\sum_{i=1}^{n} \omega_i a_i > k$, a za svaku točku skupa B vrijedi $\sum_{i=1}^{n} \omega_i b_i < k$.*

U skladu s time, vrijedi sljedeći rezultat čiji se dokaz može naći u [2].

Teorem 2.1.2 (Teorem o konvergenciji perceptrona)**.** *Neka je X skup za treniranje koji je linearno separabilan s obzirom na pripadne klase elemenata. Tada će perceptronova procedura učenja konvergirati u konačno mnogo koraka.*

Dakle, gore opisanom procedurom, perceptron može postići savršenu klasifikaciju u slučaju linearno separabilnih klasa. No, originalni algoritam će se izvršavati vječno kada klase nisu linearno separabilne. Mogli bismo izmijeniti kriterij zaustavljanja tako da algoritam stane kada dođe do određenog broja pogrešnih klasifikacija. Bez obzira na to, nemamo veličinu kojom bismo usporedili dvije granice odluke, izuzev broj grešaka. Kako je krajnji cilj ovakvih modela predikcija klase dosad neviđenih ulaza, nepostojanje takve veličine je velik nedostatak jer model neće imati jednaku sposobnost generalizacije za različite granice odluke, iako one možda rezultiraju istim brojem grešaka na skupu za treniranje.



Slika 2.2: Dva (od beskonačno mnogo) pravca koji razdvajaju dva skupa 2D točaka.

Najpoznatija kritika perceptrona je činjenica da ne može "naučiti" logičku operaciju *XOR* ([16]).

ADALINE

Vrlo blizak perceptronu je ADALINE (Adaptive Linear Neuron) model. Arhitektura modela je identična te je jedina razlika u metodi učenja. Osim toga, ovaj model može se koristiti i za predviđanje neprekidne varijable, odnosno za regresijske probleme. Stoga više nemamo ograničenja ni na vrijednosti zavisnih varijabli y iz skupa za treniranje. U općenitim uvjetima, izlazna vrijednost je sada jednaka $\hat{y} = \sum_{i=0}^{n} \omega_i x_i$, tj. mogli bismo reći da je aktivacijska funkcija ustvari identiteta. Pravilo ažuriranja vektora težina ostaje isto kao u perceptronu:

$$\mathbf{w} = \mathbf{w} + \alpha(y - \hat{y})\mathbf{x}.$$

U slučaju klasifikacije, na izlaznu vrijednost se primijenjuje funkcija signum ili neka druga step funkcija kao posljednji korak. No, to se napravi tek nakon što se ažuriraju težine za dani ulazni primjer i pripadnu predviđenu izlaznu vrijednost. Dakle, što se klasifikacije tiče, ADALINE model se razlikuje od perceptrona po tome što se aktivacijska funkcija primijenjuje na $\sum_{i=0}^{n} \omega_i x_i$ nakon ažuriranja težina, dok se kod perceptrona primijenjuje prije. Posljedica toga je što se težine ažuriraju i za one ulaze koji su točno klasificirani (osim ako je baš $\sum_{i=0}^{n} \omega_i x_i = \hat{y} = y$). Druga posljedica je to što više ne minimiziramo direktno broj pogrešnih klasifikacija, već kvadratnu pogrešku. Više o tome u nastavku.

Poveznica s klasičnim modelima strojnog učenja

Većina (parametarskih) metoda strojnog učenja u suštini se sastoji od određivanja parametara koji optimiraju neku funkciju koja služi za određivanje koliko je metoda uspješno obavila zadatak za dani skup podataka. Tu funkciju nazivamo funkcijom gubitka. Ona se najčešće formulira tako da njen optimum bude njen minimum. Kao prvi primjer promotrimo *linearnu regresiju* kao predstavnika klasičnih modela strojnog učenja. Pretpostavka linearne regresije je da je veza između nezavisnih i zavisne varijable linearna, odnosno predviđena vrijednost modela dana je s:

$$\hat{y} = \sum_{i=0}^{n} \omega_i x_i.$$

Za procjenu parametara ω_i uglavnom se koristi metoda najmanjih kvadrata u kojoj ulogu funkcije gubitka ima srednja kvadratna greška (MSE):

$$\text{MSE}(\mathbf{w}) = \frac{1}{m} \sum_{j=1}^{m} (y_j - \hat{y}_j)^2,$$

gdje je m broj elemenata u skupu za treniranje, a y_j i $\hat{y}_j$ su stvarna i predviđena vrijednost za j-ti ulazni primjer. U klasičnoj metodi gradijentnog spusta sada bi bilo potrebno izračunati ovaj gradijent na temelju cijelog skupa za treniranje i tek tada ažurirati parametre u smjeru negativnog gradijenta. No, postoji inačica ove metode, tzv. stohastički gradijentni spust, u kojem se "stvarni gradijent" (onaj na cijelom skupu za treniranje) procjenuje na temelju nasumičnog podskupa skupa za treniranje. Jedna mogućnost je procijeniti gradijent na temelju samo jednog ulaznog primjera. Doprinos greški jednog fiksnog primjera jednak je

$$L(\mathbf{w}) = (y - \hat{y})^2 = \left(y - \sum_{i=0}^{n} \omega_i x_i \right)^2 .$$

Parcijalne derivacije su tada jednake

$$\frac{\partial L}{\partial \omega_i} = -2 \left(y - \sum_{i=0}^{n} \omega_i x_i \right) x_i = -2(y - \hat{y}) x_i .$$

Dakle, svaku komponentu vektora težine ω_i ažuriramo tako da nakon svakog primjera iz skupa za treniranje postavimo njenu vrijednost na $\omega_i + 2\beta(y - \hat{y})x_i$. U vektorskom zapisu, uz supstituciju $\alpha = 2\beta$ imamo:

$$\mathbf{w} = \mathbf{w} + \alpha(y - \hat{y})\mathbf{x},$$

što je upravo pravilo ažuriranja u ADALINE modelu! Vidimo da proces učenja ADALINE modela u potpunosti odgovara onome u linearnoj regresiji kada koristimo stohastički gradijentni spust.

Bitno je uočiti da u slučaju perceptrona ovim načinom ne možemo doći do istog zaključka, iako je pravilo ažuriranja težina identično definirano. Naime, kod perceptrona govorimo o klasifikaciji gdje stvarne i predviđene vrijednosti više nisu neprekidne, već diskretne pa ne možemo koristiti istu funkciju gubitka za računanje gradijenta. Problem stvara funkcija signum koja nije derivabilna. Promotrimo sljedeću funkciju gubitka:

$$L(\mathbf{w}) = \sum_{j=1}^{m} \max\{-y_j(\mathbf{w}^T \mathbf{x}_j), 0\},$$

gdje je $\mathbf{w}$ vektor težina te $\mathbf{x}_j$ vektor ulaznih signala j-tog elementa iz skupa za treniranje. Pokažimo da minimizacija ove funkcije gubitka odgovara procesu učenja perceptrona. Uočimo da je maksimum unutar sume veći od nule kada su y_j i $\mathbf{w}^T \mathbf{x}_j$ različitih predznaka, što odgovara slučaju pogrešne klasifikacije. Zbog toga se funkcija može zapisati u obliku:

$$L(\mathbf{w}) = - \sum_{j \in \mathcal{M}(\omega)} y_j(\mathbf{w}^T x_j),$$

gdje je $\mathcal{M}(\omega) \subseteq \{1, 2, \ldots, m\}$ skup indeksa primjera koje smo pogrešno klasificirali. Fokusirajući se ponovno na jedan fiksni element skupa za treniranje (kojeg smo pogrešno klasificirali), odnosno koristeći stohastički gradijentni spust, vidimo da se tada vektor težina ažurira na sljedeći način:

$$\mathbf{w} = \mathbf{w} + y\mathbf{x}.$$

Kako je $y \in \{-1, 1\}$, u slučaju pogrešne klasifikacije vektor $(y - \hat{y})\mathbf{x}$ je istog smjera kao i vektor $y\mathbf{x}$, uz drugačiju amplitudu. Znači, perceptronova metoda učenja nije ništa drugo doli minimizacija gore definirane funkcije gubitka putem stohastičkog gradijentnog spusta.

Napomena 2.1.3. *Optimalne težine u linearnoj regresiji pomoću metode najmanjih kvadrata mogu se dobiti i analitički. Uvedimo sada klasičnu notaciju u općenitom slučaju s m elemenata skupa za treniranje* $(\mathbf{x}_i, y_i)$ *gdje je* $\mathbf{x}_i = [x_{i1}, \ldots, x_{in}]^T$ *n-dimenzionalni vektor-stupac. Pomoću tih vektora tvorimo tzv. matricu dizajna X:*

$$X = \begin{bmatrix} -\mathbf{x}_1^T - \\ \vdots \\ -\mathbf{x}_m^T - \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ & \ddots & \\ x_{m1} & \cdots & x_{mn} \end{bmatrix} \in \mathbb{R}^{m \times n}.$$

Zavisne varijable kao i težine isto su dane u vektorskom obliku: $\mathbf{y} = [y_1, \ldots, y_m]^T \in \mathbb{R}^{m \times 1}$, $\mathbf{w} = [\omega_1, \ldots, \omega_n]^T \in \mathbb{R}^{n \times 1}$. *Funkcija gubitka tada je jednaka*

$$L(\mathbf{w}) = \|\mathbf{y} - X\mathbf{w}\|^2 = (\mathbf{y} - X\mathbf{w})^T (\mathbf{y} - X\mathbf{w}).$$

Deriviranjem po $\mathbf{w}$ *te izjednačavanjem gradijenta s nulom, dobijemo da je vektor težina koji minimizira funkciju gubitka dan s:*

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

Međutim, u praksi se često preferira korištenje iterativnih metoda poput gradijentnog spusta jer je računanje inverza matrice računski zahtjevna operacija.

Zadržimo se na slučaju klasifikacije. Bilo bi poželjno znati koliko je model "siguran" u svoju donesenu odluku, tj. željeli bismo znati koliku vjerojatnost model pripisuje nekom događaju. Kod klasifikacije, to odgovara pitanju kolika je vjerojatnost da neki ulazni primjer pripada određenoj klasi. Predstavnik takvih modela je *logistička regresija*. U ovom modelu pretpostavljamo da je zavisna varijabla y uz dane nezavisne prediktore $\mathbf{x}$ Bernoullijeva slučajna varijabla. Dakle, sada je $y \in \{0, 1\}$. Vjerojatnost da y dolazi iz klase označene jedinicom dana je s:

$$p = \mathbb{P}(y = 1 \mid \mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^T \mathbf{x}),$$

gdje je σ tzv. logistička funkcija definirana s:

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

Ovdje se parametri procijenjuju metodom maksimalne vjerodostojnosti (MLE). Kako je pretpostavka da se radi o Bernoullijevoj distribuciji, funkcija vjerodostojnosti za dani m-člani skup za treniranje dana je s:

$$\mathcal{L}(p_i|\mathbf{y}) = \prod_{i=1}^{m} p_i^{y_i}(1 - p_i)^{1-y_i}.$$

Maksimiziranje ove funkcije ekvivalentno je minimiziranju negativnog logaritma ove funkcije što se često preferira jer je lakše baratati sumom nego produktom te je proces računski stabilniji:

$$NLL = -\log(\mathcal{L}(p_i|\mathbf{y})) = -\sum_{i=1}^{m} y_i \log p_i + (1 - y_i)\log(1 - p_i).$$

Ova funkcija ima ulogu funkcije gubitka kod logističke regresije. Parcijalne derivacije za jedan fiksni element su dane s:

$$\frac{\partial NLL}{\partial \omega_i} = x_i(p - y),$$

pa bi stohastičkim gradijentnim spustom u svakom koraku vektor težina ažurirali na sljedeći način:

$$\mathbf{w} = \mathbf{w} + \alpha(y - p)\mathbf{x}.$$

Ako bi u perceptron modelu kao aktivacijsku funkciju koristili logističku funkciju, tada bi izlazna vrijednost modela bila upravo jednaka p, tj. $\hat{y} = p$. Sada vidimo da smo opet došli do istog postupka kao i u metodi učenja perceptrona. Prokomentirajmo sad još malo našu funkciju gubitka. Definiramo dvije distribucije P i Q koje sadrže stvarne i predviđene vjerojatnosti da dani ulazni primjer pripada klasi 1, tj. $P \in \{y, 1 - y\}$, $Q \in \{p, 1 - p\}$. Tada je

$$\begin{aligned} H(P) + D_{KL}(P\|Q) &= -y\log y - (1 - y)\log(1 - y) + y\log\frac{y}{p} + (1 - y)\log\frac{1 - y}{1 - p} \\ &= -y\log p - (1 - y)\log(1 - p), \end{aligned}$$

a to je upravo negativna log-vjerodostojnost za jedan ulazni primjer. Ova veličina se u literaturi naziva unakrsna entropija (eng. *cross-entropy*) distribucije Q s obzirom na distribuciju P. Sad vidimo da minimizacijom funkcije gubitka smanjujemo KL divergenciju $D_{KL}(P\|Q)$, odnosno tražimo distribuciju Q kojom ćemo se maksimalno približiti stvarnoj distribuciji P. Općenito, kada god pretpostavimo nekakvu uvjetnu distribuciju $p(\mathbf{y} \mid \mathbf{x}; \mathbf{w})$, metoda maksimalne vjerodostojnosti nalaže da koristimo $-\log p(\mathbf{y} \mid \mathbf{x}; \mathbf{w})$ kao našu funkciju gubitka. Stoga je unakrsna entropija sveprisutna u dubokom učenju.

2.2 Neuronske mreže

U prethodnom dijelu smo se bavili modelima s jednim umjetnim neuronom, a sada se okrećemo neuronskim mrežama - modelima s više međusobno povezanih neurona najčešće podijeljenih u slojeve. Podrazumijevamo da se neuronska mreža sastoji od 2 ili više slojeva, s time da se ulazni sloj ne računa jer on služi samo za unos podataka u model te nema nikakvih računskih operacija. To implicira postojanje tzv. skrivenih slojeva (eng. *hidden layer*) koji se tako zovu jer njihovi izlazi nisu vidljivi krajnjem korisniku. U suštini, neuronska mreža je aproksimacija neke (često vrlo kompleksne) funkcije pomoću kompozicije više jednostavnijih funkcija. Korištenjem više slojeva ili više neurona u slojevima možemo reprezentirati funkcije veće kompleksnosti. Zbog već spomenutog negativnog pogleda na područje uzrokovano ograničenjima ranih modela, pojava arhitekture neuronskih mreža kakvu znamo danas bila je značajno prolongirana. Najveću ulogu u preporodu odigrao je tzv. algoritam unazadne propagacije (eng. *back-propagation*) za računanje gradijenta funkcije gubitka. Iako je bilo i ranijih prijedloga za njegovu upotrebu([23]), oni su bili ignorirani. Korištenje algoritma počelo se ozbiljno razmatrati tek nakon objave rada Davida Rumelharta 1986. godine ([19]) koji implementira rezultate iz [13].

Unaprijedne neuronske mreže

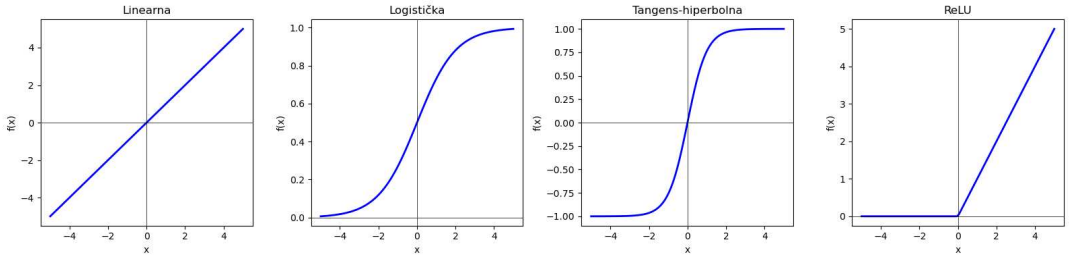
Unaprijedne (eng. *feedforward*) neuronske mreže ili višeslojni perceptron su fundamentalni modeli dubokog učenja. Naziv su dobile zbog toka informacije unutar modela. Naime, podaci s ulaza prolaze kroz skrivene slojeve prema izlazu jednosmjerno, "prema naprijed". Na neuronske mreže možemo gledati i kao na grafove gdje čvorove predstavljaju neuroni, a neuroni su povezani putem bridova. Konkretno, unaprijedne neuronske mreže su težinski aciklički usmjereni grafovi u kojima se težine modela pridružuju bridovima grafa. U standardnoj arhitekturti, svi čvorovi j-tog sloja povezani su sa svim čvorovima $(j+1)$-vog. Neka je $\mathbf{x} \in \mathbb{R}^{n\times 1}$ ulazni vektor-stupac, te neka je k broj slojeva mreže. Neka su $p_1, \ldots, p_k$ brojevi neurona u svakom od k slojeva. Težine veza između ulaznog i prvog skrivenog sloja tada su sadržane u matrici $W_1 \in \mathbb{R}^{n\times p_1}$, dok su težine između r-tog i $(r+1)$-vog skrivenog sloja dane u matrici $W_r \in \mathbb{R}^{p_r \times p_{r+1}}$. Ako se u izlaznom sloju nalazi o čvorova, tada je finalna matrica W_{k+1} dimenzije $p_k \times o$. Proces prolaska ulaznog vektora kroz model sve do izlaznog vektora opisan je sljedećim rekurzivnim jednadžbama:

$$
\begin{aligned}
&H^{(1)} = \phi_1(W_1^T \mathbf{x}) && \text{[Iz ulaznog sloja u skriveni]}\\
&H^{(p+1)} = \phi_{p+1}(W_{p+1}^T H^{(p)})\ \forall p \in \{1, 2, \ldots, k-1\} && \text{[Iz skrivenog u skriveni sloj]}\\
&\hat{\mathbf{y}} = \phi_k(W_{k+1}^T H^{(k)}) && \text{[Iz skrivenog sloja u izlazni]}
\end{aligned}
$$

Ovdje je s $H^{(p)}$ označen vektor vrijednosti p-tog sloja, a s ϕ_p aktivacijska funkcija p-tog sloja koja se na svoj vektorski ulaz primijenjuje element po element. Najčešće korištene

aktivacijske funkcije su:

- linearna (identiteta): $\phi(x) = x$
- logistička: $\phi(x) = \frac{1}{1+e^{-x}}$
- tangens-hiperbolna: $\phi(x) = \frac{e^{2x}-1}{e^{2x}+1}$
- rektifikacijska (ReLU): $\phi(x) = \max\{0, x\}$.



Slika 2.3: Prikaz najpopularnijih aktivacijskih funkcija

Postoji mnogo varijanti ReLU funkcije. Jedna od njih je tzv. propusna (eng. *leaky*) ReLU gdje se negativne vrijednosti ne postavljaju na nulu, već se pomnože nekom malom konstantom:

$$\phi(x) = \begin{cases} 0.01x & \text{za } x < 0, \\ x & \text{inače.} \end{cases}$$

Ovo je poboljšanje u odnosu na običnu ReLU funkciju jer sada ni gradijent funkcije više nije nula pa će tijekom računanja gradijenta funkcije gubitka manje neurona biti ”trajno neaktivno”.

Napomena 2.2.1. *Vrlo je bitno imati nelinearnu aktivacijsku funkciju u barem jednom sloju. Naime, kada ga nebi bilo, onda bi cijeli model bio linearan jer je kompozicija linearnih funkcija linearna pa bi njegova ekspresivnost bila vrlo ograničena.*

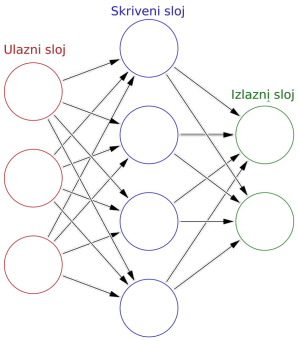
U slučaju klasifikacije s K različitih klasa, izlazni sloj se sastoji od K čvorova, dok je aktivacijska funkcija tzv. *softmax* funkcija. Izlaz funkcije predstavlja procjenu vjerojatnosti da ulazni primjer pripada određenoj klasi. Poznato je da je ta vjerojatnost nekalibrirana, tj. ne odgovara stvarnoj vjerojatnosti. Koristeći oznake od prije, $\forall i \in \{1, \ldots, K\}$ imamo:

$$\phi(\mathbf{z})_i = \frac{e^{\mathbf{z}_i}}{\sum_{j=1}^{K} e^{\mathbf{z}_j}},$$

gdje je $\mathbf{z} = W_{k+1}^T H^{(k)}$. Predviđena klasa je tada:

$$\hat{y} = \text{argmax}_{i \in \{1,\ldots,K\}} \phi(\mathbf{z})_i.$$

Ako je $\mathbf{z}_k \gg \mathbf{z}_j$ za sve $j \neq k$, tada je $\phi(\mathbf{z})_k \approx 1$, a $\phi(\mathbf{z})_j \approx 0$ (vidi [3], str. 196-198). Za više detalja o problemu kalibracije vjerojatnosti, vidi [10]. Primijetimo kako je $\sum_{i=1}^{K} \phi(\mathbf{z})_i = 1$. Dakle, primjenom softmax funkcije pretvorili smo ulazni vektor u vjerojatnosnu distribuciju.



Slika 2.4: Arhitektura unaprijedne neuronske mreže s jednim skrivenim i jednim izlaznim slojem.

Danas se preporuča koristiti ReLU funkciju. Iako nije derivabilna u nuli, u praksi se pokazalo da to nije veliki problem te se bez većih utjecaja na performanse derivacija može nadopuniti u toj jednoj točki. Funkcija gubitka modela bira se ovisno o pozadinskom problemu, a najčešće su to već spomenute srednja kvadratna pogreška i unakrsna entropija. Na kraju se funkcija gubitka minimizira nekom inačicom gradijentnog spusta. Ovime smo opisali kompletnu arhitekturu unaprijednih neuronskih mreža. Međutim, fali nam još jedna stvar: kako ćemo zapravo izračunati gradijent funkcije gubitka? Iako teoretski moguće, naivno računanje gradijenta kompozicija funkcija s kakvima se susrećemo u dubokom učenju nije praktično.

Unazadna propagacija

Kod umjetnog neurona, funkcija gubitka se može direktno izračunati kao funkcija težina te je tada računanje gradijenta relativno jednostavno. U neuronskim mrežama to nije tako jer je izlazna vrijednost, a time posljedično i funkcija gubitka, kompliciranа kompozicija funkcija. Algoritam unazadne propagacije ([19]) koristi lančano pravilo diferencijalnog računa. Sastoji se od sljedeće dvije faze:

1. Unaprijedna faza: u ovoj fazi ulazni signal iz skupa za treniranje prolazi kroz neuronsku mrežu i računa se izlazna vrijednost koristeći trenutne vrijednosti težina (koje se na početku inicijaliziraju na nasumične vrijednosti). U ovom trenutku može se izračunati derivacija funkcije gubitka s obzirom na izlaz.

2. Unazadna faza: sada računamo gradijente s obzirom na težine modela koje ćemo kasnije koristiti u procesu treniranja. Gradijenti se računaju unazad, od izlaznog sloja prema ranijim slojevima, pa je po tome i algoritam dobio ime. Neka je sada $h_1, h_2, \ldots, h_k$ niz čvorova u skrivenim slojevima poslije kojih dolazi izlaz modela o. Nadalje, neka je težina veze između čvorova h_r i h_{r+1} označena s $\omega_{(h_r,h_{r+1})}$, $0 \leq r \leq k$, gdje h_0 predstavlja ulazni signal. Ako pretpostavimo da postoji samo jedan put od čvora h_1 do izlaza o u mreži, tada se gradijent funkcije gubitka L u odnosu na bilo koju težinu može dobiti pomoću lančanog pravila:

$$\frac{\partial L}{\partial \omega_{(h_{r-1},h_r)}} = \frac{\partial L}{\partial o} \cdot \left[\frac{\partial o}{\partial h_k} \prod_{i=r}^{k-1} \frac{\partial h_{i+1}}{\partial h_i} \right] \cdot \frac{\partial h_r}{\partial \omega_{(h_{r-1},h_r)}}, \forall r \in \{1, \ldots, k\}.$$

Generalizirani izraz koji uzima u obzir sve puteve od h_r do izlaza o tada je:

$$\frac{\partial L}{\partial w_{(h_{r-1},h_r)}} = \underbrace{\frac{\partial L}{\partial o} \cdot \left[\sum_{[h_r,h_{r+1},\ldots h_k,o] \in \mathcal{P}} \frac{\partial o}{\partial h_k} \prod_{i=r}^{k-1} \frac{\partial h_{i+1}}{\partial h_i} \right]}_{\Delta(h_r,o)=\frac{\partial L}{\partial h_r}} \frac{\partial h_r}{\partial w_{(h_{r-1},h_r)}}, \tag{2.1}$$

gdje je $\mathcal{P}$ skup svih puteva od h_r do o. Rekurzija za $\Delta(h_r, o)$ može se zapisati i na sljedeći način:

$$\Delta(h_r, o) = \frac{\partial L}{\partial h_r} = \sum_{h:h_r \Longrightarrow h} \frac{\partial L}{\partial h} \frac{\partial h}{\partial h_r} = \sum_{h:h_r \Longrightarrow h} \frac{\partial h}{\partial h_r} \Delta(h, o), \tag{2.2}$$

gdje je h svaki čvor na koji se h_r neposredno spaja. Vrijednost za čvor u izlaznom sloju se postavlja na $\Delta(o, o) = \frac{\partial L}{\partial o}$. Kako je svaki h u sloju koji dolazi nakon sloja u kojem je h_r, vrijednosti $\Delta(h, o)$ su nam već dostupne iz prijašnjih koraka. Evaluirajmo sada izraz $\frac{\partial h}{\partial h_r}$. Neka je težina veze između h_r i h dana s $\omega_{(h_r.h)}$, te neka je a_h linearna kombinacija ulaza čvora h iz sloja neposredno prije. Tada je $h = \phi(a_h)$, gdje je ϕ aktivacijska funkcija čvora h. Tada ponovno primjenom lančanog pravila slijedi:

$$\frac{\partial h}{\partial h_r} = \frac{\partial h}{\partial a_h} \cdot \frac{\partial a_h}{\partial h_r} = \frac{\partial \phi(a_h)}{\partial a_h} \cdot \omega_{(h_r,h)} = \phi'(a_h) \cdot \omega_{(h_r,h)}. \tag{2.3}$$

Dakle, gradijenti se akumuliraju unatrag, s desna na lijevo. Pri tome se svaki čvor procesira samo jednom u svakom prolazu. Preostaje izračunati izraz $\frac{\partial h_r}{\partial w_{(h_{r-1},h_r)}}$, no

koristeći istu logiku kao u prethodnom računu slijedi da je:

$$\frac{\partial h_r}{\partial w_{(h_{r-1},h_r)}} = h_{r-1} \cdot \phi'(a_{h_r}). \tag{2.4}$$

Finalno, uvrštavajući (2.2),(2.3) i (2.4) u (2.1) dobivamo:

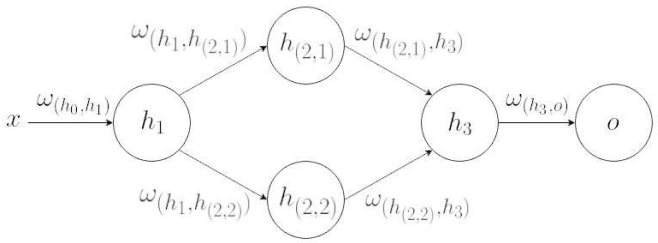
$$\frac{\partial L}{\partial \omega_{(h_{r-1},h_r)}} = h_{r-1} \cdot \phi'(a_{h_r}) \sum_{h:h_r \Longrightarrow h} \omega_{(h_r,h)} \cdot \phi'(a_h) \cdot \Delta(h,o), \forall r \in \{1,\ldots,k\}. \tag{2.5}$$

Primijetimo kako pomoću formule (2.5) ne možemo izračunati parcijalne derivacije u odnosu na težine iz zadnjeg skrivenog sloja u izlazni sloj $\omega_{(h_k,o)}$. No, uz $o = \phi(a_o) = \phi(h_k\omega_{(h_k,o)})$, po lančanom pravilu opet slijedi da je:

$$\frac{\partial L}{\partial \omega_{(h_k,o)}} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial \omega_{(h_k,o)}} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial a_o} \cdot \frac{\partial a_o}{\partial \omega_{(h_k,o)}} = \frac{\partial L}{\partial o} \cdot \phi'(a_o) \cdot h_k.$$

Valja napomenuti da se pojam unazadne propagacije odnosi isključivo na računanje gradijenta, dok se funkcija gubitka naknadno minimizira gradijentnim metodama.

Primjer 2.2.2. *Ilustrirajmo sada algoritam unazadne propagacije na vrlo jednostavnoj neuronskoj mreži:*



Dakle, u ovom primjeru je $k = 3$. Zbog jednostavnijeg zapisa i računa, neka su sve aktivacijske funkcije jednake logističkoj, tj. $\phi(x) = \frac{1}{1+e^{-x}}$. Tada vrijedi $\phi'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{1+e^{-x}}\left(1 - \frac{1}{1+e^{-x}}\right) = \phi(x)\,(1-\phi(x))$. Nadalje, neka je funkcija gubitka dana s $L = (y-o)^2$, gdje je y pripadna zavisna varijabla za dani ulazni signal x. Slijedi da je $\frac{\partial L}{\partial o} = \Delta(o,o) = -2(y-o)$. Uvedimo oznake $h_4 := o$, $h_0 := x$. Nadalje, označimo s a_m ulaz neurona h_m. Tada je za sve neurone u mreži $h_m = \phi(a_m)$. Kao što smo već rekli, prvo računamo parcijalne derivacije s obzirom na težine najbliže izlazu:

$$\begin{aligned}\frac{\partial L}{\partial \omega_{(h_3,o)}} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial a_4} \cdot \frac{\partial a_4}{\partial \omega_{(h_3,o)}} &= -2(y-o)\phi'(a_4)h_3 \\ &= -2(y-o)\phi(a_4)\,(1-\phi(a_4))\,h_3 \\ &= -2(y-o)o(1-o)h_3.\end{aligned}$$

Riješili smo slučaj $r = k + 1 = 4$. Od sada nadalje možemo koristiti formulu (2.5).

$$\begin{aligned}\frac{\partial L}{\partial \omega_{(h_{(2,1)},h_3)}} &= \Delta(h_3, o)\frac{\partial h_3}{\partial \omega_{(h_{(2,1)},h_3)}} \\ &= [\omega_{(h_3,o)} \cdot \phi'(a_o) \cdot \Delta(o, o)] \cdot [h_{(2,1)} \cdot \phi'(a_3)] \\ &= -2(y - o) \cdot h_{(2,1)} \cdot \phi(a_3)\,(1 - \phi(a_3)) \cdot \omega_{(h_3,o)} \cdot \phi(a_o)\,(1 - \phi(a_o)) \\ &= -2(y - o) \cdot h_{(2,1)} \cdot h_3(1 - h_3) \cdot \omega_{(h_3,o)} \cdot o(1 - o).\end{aligned}$$

$$\begin{aligned}\frac{\partial L}{\partial \omega_{(h_{(2,2)},h_3)}} &= \Delta(h_3, o)\frac{\partial h_3}{\partial \omega_{(h_{(2,2)},h_3)}} \\ &= [\omega_{(h_3,o)} \cdot \phi'(a_o) \cdot \Delta(o, o)] \cdot [h_{(2,2)} \cdot \phi'(a_3)] \\ &= -2(y - o) \cdot h_{(2,2)} \cdot \phi(a_3)\,(1 - \phi(a_3)) \cdot \omega_{(h_3,o)} \cdot \phi(a_o)\,(1 - \phi(a_o)) \\ &= -2(y - o) \cdot h_{(2,2)} \cdot h_3(1 - h_3) \cdot \omega_{(h_3,o)} \cdot o(1 - o).\end{aligned}$$

Uočimo kako u svakom sljedećem koraku koristimo vrijednosti iz prethodnog:

$$\begin{aligned}\frac{\partial L}{\partial \omega_{(h_1,h_{(2,1)})}} &= \Delta(h_{(2,1)}, o)\frac{\partial h_{(2,1)}}{\partial \omega_{(h_1,h_{(2,1)})}} \\ &= h_1 \cdot \phi'(a_{(2,1)}) \cdot \omega_{(h_{(2,1)},h_3)} \cdot \phi'(a_{h_3}) \cdot \Delta(h_3, o) \\ &= h_1 \cdot h_{(2,1)}(1 - h_{(2,1)}) \cdot \omega_{(h_{(2,1)},h_3)} \cdot h_3(1 - h_3) \cdot \omega_{(h_3,o)} \cdot \phi'(a_o) \cdot \Delta(o, o) \\ &= -2(y - o)h_1 \cdot h_{(2,1)}(1 - h_{(2,1)}) \cdot \omega_{(h_{(2,1)},h_3)} \cdot h_3(1 - h_3) \cdot \omega_{(h_3,o)} \cdot o(1 - o).\end{aligned}$$

$$\begin{aligned}\frac{\partial L}{\partial \omega_{(h_1,h_{(2,2)})}} &= \Delta(h_{(2,2)}, o)\frac{\partial h_{(2,2)}}{\partial \omega_{(h_1,h_{(2,2)})}} \\ &= h_1 \cdot \phi'(a_{(2,2)}) \cdot \omega_{(h_{(2,2)},h_3)} \cdot \phi'(a_3) \cdot \Delta(h_3, o) \\ &= h_1 \cdot h_{(2,2)}(1 - h_{(2,2)}) \cdot \omega_{(h_{(2,2)},h_3)} \cdot h_3(1 - h_3) \cdot \omega_{(h_3,o)} \cdot \phi'(a_o) \cdot \Delta(o, o) \\ &= -2(y - o)h_1 \cdot h_{(2,2)}(1 - h_{(2,2)}) \cdot \omega_{(h_{(2,2)},h_3)} \cdot h_3(1 - h_3) \cdot \omega_{(h_3,o)} \cdot o(1 - o).\end{aligned}$$

Finalno:

$$\begin{aligned}\frac{\partial L}{\partial \omega_{(h_0,h_1)}} &= \Delta(h_1, o)\frac{\partial h_1}{\partial \omega_{(h_0,h_1)}} \\ &= \Big[\phi'(a_{(2,1)}) \cdot \omega_{(h_1,h_{(2,1)})} \cdot \Delta(h_{(2,1)}, o) + \phi'(a_{(2,2)}) \cdot \omega_{(h_1,h_{(2,2)})} \cdot \Delta(h_{(2,2)}, o)\Big]\, h_0 \cdot \phi'(a_1) \\ &\vdots \\ &= -2x(y - o)h_1(1 - h_1)\omega_{(h_1,h_{(2,1)})}h_{(2,1)}(1 - h_{(2,1)})\omega_{(h_{(2,1)},h_3)}h_3(1 - h_3)\omega_{(h_3,o)}o(1 - o) \\ &\quad - 2x(y - o)h_1(1 - h_1)\omega_{(h_1,h_{(2,2)})}h_{(2,2)}(1 - h_{(2,2)})\omega_{(h_{(2,2)},h_3)}h_3(1 - h_3)\omega_{(h_3,o)}o(1 - o).\end{aligned}$$

Sada imamo sve parcijalne derivacije te bi mogli ažurirati težine pomicanjem u smjeru negativnog gradijenta.

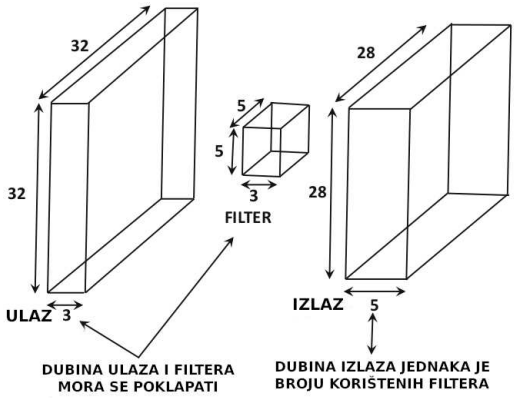
Konvolucijske neuronske mreže

Specijalni slučaj unaprijednih neuronskih mreža su tzv. konvolucijske (eng. *convolutional*) neuronske mreže čija je namjena rad s ulazima rešetkaste strukture, npr. 2D slike. Već smo rekli kako su neuronske mreže općenito inspirirane biologijom. Ovdje je to isto slučaj i to u još većoj mjeri. Naime, vršenjem eksperimenata na vidnom korteksu mačaka ([11]) ustanovilo se da su ćelije u određenim dijelovima korteksa osjetljive samo na određene dijelove vidnog polja. Nadalje, različiti oblici i orijentacije objekata aktiviraju različite ćelije koje su pak spojene u slojevite strukture. To otkriće dalo je naslutiti da se kod sisavaca vidno polje rekonstruira pomoću više razina apstrakcije. Ovi modeli daju najbolje rezultate u problemima koji se tiču slikovnih podataka te pokazuju koliko je bitno prilagoditi arhitekturu modela pozadinskom problemu.

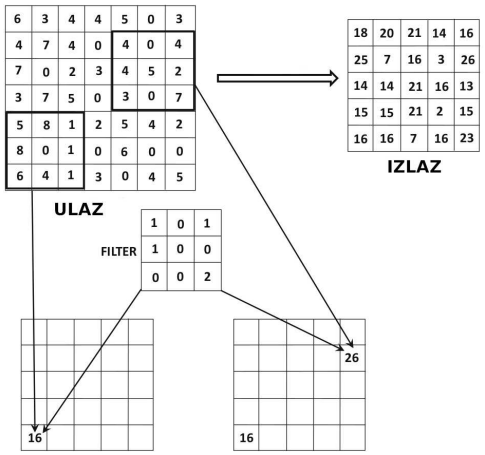
Glavna odlika konvolucijskih neuronskih mreža je prisutnost konvolucijskih slojeva koji su dobili naziv po matematičkoj operaciji konvolucije, iako je operacija koja se koristi zapravo tzv. unakrsna korelacija - slična, ali ne identična operacija. Konvolucijski sloj uglavnom dolazi neposredno nakon ulaznog. Ulazni primjer sada više nije dan u obliku vektora, već on ima visinu, širinu i dubinu, gdje je dubina broj *kanala*. Na primjer, u slučaju RGB slike, dubina je 3. Dakle, općenito, to je 3D matrica. Na taj način čuvamo prostornu informaciju slike. Parametri konvolucijskog sloja također su spremljeni u 3D strukturu, tzv. filter ili jezgru (eng. *kernel*). Visina i širina filtra su najčešće jednaki te puno manji od visine i širine ulaznog primjera, dok dubina filtera mora odgovarati dubini ulaza. Broj različitih filtera u sloju određuje dubinu izlaza sloja. Za fiksni indeks dubine, pripadnu 2D matricu zovemo mapom značajki (eng. *feature map*). Neka je sada ulaz q-tog sloja dimenzije $L_q \times B_q \times d_q$ (visina $\times$ širina $\times$ dubina), a filtri iz q-tog u $(q+1)$-vi sloj dimenzije $F_q \times F_q \times d_q$. Neka su parametri p-tog filtra u q-tom sloju dani u 3D matrici $W^{(p,q)} = \left[\omega_{ijk}^{(p,q)}\right]$ te mape značajki u q-tom sloju u 3D matrici $H^{(q)} = \left[h_{ijk}^{(q)}\right]$. Tada je rezultat operacije konvolucije u q-tom sloju 3D matrica dimenzije $(L_q - F_q + 1) \times (B_q - F_q + 1) \times d_{q+1}$, gdje je d_{q+1} broj različitih filtera u q-tom sloju. Vrijednosti elemenata dane su s:

$$h_{ijp}^{(q+1)} = \sum_{r=1}^{F_q} \sum_{s=1}^{F_q} \sum_{k=1}^{d_q} \omega_{rsk}^{(p,q)} h_{i+r-1,j+s-1,k}^{(q)} \quad \begin{array}{l} \forall i \in \left\{1 \ldots, L_q - F_q + 1\right\}, \\ \forall j \in \left\{1 \ldots B_q - F_q + 1\right\}, \\ \forall p \in \left\{1 \ldots d_{q+1}\right\}. \end{array}$$

Sada vidimo zašto se ova operacija još naziva i *klizeći skalarni produkt*. Naime, vrijednosti se dobiju tako da se svaki filter postavlja unutar mape značajki i računa se skalarni produkt vrijednosti. Filter se tada pomiče tako da se prođu sve pozicije u kojima je filter u potpunosti sadržan u mapama značajki. Primijetimo kako se ovakvim postupkom dimenzija ulaznog primjera potencijalno može značajno smanjiti prolaskom kroz model (pa time i više konvolucijskih slojeva). Ovdje ćemo samo spomenuti da postoje tehnike kojima se to



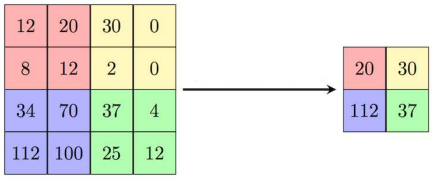
Slika 2.5: Konvolucija između ulaza dimenzija $32 \times 32 \times 3$ i pet različitih filtera dimenzija $5 \times 5 \times 3$. Izlaz za svaki filter je prostorne dimenzije 28×28, dok je dubina izlaza određena brojem različitih filtera, u ovom slučaju 5 (preuzeto iz [1], str. 320).



Slika 2.6: Primjer konvolucije između ulaza dimenzije $7 \times 7 \times 1$ i filtera dimenzije $3 \times 3 \times 1$. Zbog mogućnosti prikaza u 2D, odabrana je dubina 1, iako primjena jednog filtera rezultira jednom mapom značajki, dakle 2D objektom. Za dubine veće od 1, vrijednosti u svakoj mapi značajki bi se samo zbrojile (preuzeto iz [1], str. 321).

može izbjeći proširenjem ulaza "praznim" pikselima (eng. *padding*). Kao zadnji korak, na izlaz se primjenjuje neka aktivacijska funkcija (najčešće ReLU) element po element. Moguće je ručno namjestiti filtre tako da oni detektiraju neke pojave na slici (npr. obrube objekata), ali ideja je da model sam nauči filtre koji će najbolje uhvatiti relevantne značajke slika.

Još jedna vrsta sloja koja se pojavljuje u ovim modelima su tzv. slojevi sažimanja (eng. *pooling layer*). Operacija sažimanja u q-tom sloju prolazi po svakoj mapi značajki ulaza i za svaki kvadrat dimenzije $P_q \times P_q$ vraća maksimum vrijednosti iz tog kvadrata (eng. *max-pooling*). Izlaz je objekt dimenzije $(L_q - P_q + 1) \times (B_q - P_q + 1) \times d_q$. Za razliku od konvolucije, sažimanje se vrši na svakoj mapi značajki zasebno te su dubine ulaza i izlaza iste. Umjesto da prolazimo po svim kvadratnim područjima dimenzije $P_q \times P_q$, možemo definirati korak S_q tako da se kvadrat pomiče za S_q mjesta u horizontalnom i vertikalnom smjeru. Štoviše, najčešći pristup u praksi je uzeti $S_q = P_q$ kako nebi imali presjek između regija koje se sažimaju. Tada je visina izlaznog objekta jednaka $(L_q - P_q)/S_q + 1$, a širina $(B_q - P_q)/S_q + 1$. Postoji i inačica sažimanja koja vraća prosječnu vrijednost u svakom kvadratu (eng. *average-pooling*). Sažimanjem postižemo neku dozu invarijantnosti na translacije jer translacijom ulaznih vrijednosti izlaz operacije sažimanja neće se značajno promijeniti.



Slika 2.7: Primjer sažimanja mape značajki dimenzije 4×4 pomoću kvadrata dimenzije 2×2 s korakom $S_q = 2$.

Nedavno se počelo sumnjati u potrebu za slojevima sažimanja te se pokazalo ([21]) da se njihovim izostavljanjem mogu postići jednako dobri rezultati.

Nakon svih konvolucijskih i slojeva sažimanja u mreži, izlaz se ”spljošti” u vektor koji služi kao ulaz za klasični skriveni sloj s neuronima kakve nalazimo u običnim unaprijednim mrežama. Izlazni sloj se konstruira shodno prirodi problema, s nekom od aktivacijskih funkcija koje smo već prije spomenuli.

Poglavlje 3

Eksperimentalni rezultati

U ovom poglavlju implementiramo modele dubokog učenja i testiramo njihovu točnost na nekoliko poznatih skupova podataka za klasifikaciju slika. Uz to, za usporedbu su dani i rezultati klasičnih modela strojnog učenja na istim skupovima podataka. Konkretno, implementiramo jednu regularnu unaprijednu te jednu konvolucijsku neuronsku mrežu. Od klasičnih modela koristimo multinomijalnu logističku regresiju te stablo odluke. U svrhu implementiranja koristimo programski jezik *Python* te softverske biblioteke *Tensorflow/Keras* i *scikit-learn*. Što se tiče arhitekture mreža, kao aktivacijska funkcija u svim skrivenim slojevima koristi se ReLU funkcija. Regularna mreža sastoji se od 2 skrivena sloja s 64 neurona u svakom. Konvolucijska mreža sastoji se od 3 konvolucijska sloja s 32, 64 i 64 različitih filtera dimenzije 3×3, redom. Između prvog i drugog te drugog i trećeg konvolucijskog sloja nalazi se jedan sloj za sažimanje u kojem se koristi kvadrat dimenzije 2×2, te je korak postavljen na 2. Nakon toga dolazi standardni skriveni sloj s 64 neurona. Izlazni sloj obiju mreža sastoji se od 10 neurona i korištena je softmax aktivacijska funkcija. Pred-procesiranje podataka je svedeno na minimum te se sastoji od normalizacije vrijednosti piksela na vrijednosti u intervalu $[0, 1]$ dijeljenjem s 255. Kao malu pomoć, odlučili smo klasičnim modelima osim samih vrijednosti piksela slika dodati i značajke koje predstavljaju horizontalne i vertikalne rubove slike. One su dobivene prolaskom ručno određenih filtera po vrijednostima piksela slike, dakle konvolucijom.

-1	0	+1
-1	0	+1
-1	0	+1

+1	+1	+1
0	0	0
-1	-1	-1

Slika 3.1: Prikaz tzv. Prewitt filtera za detekciju vertikalnih i horizontalnih rubova.

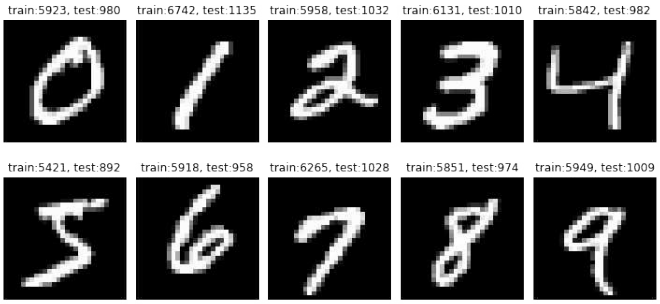
Osim toga, dodat ćemo i značajku dobivenu tzv. HOG (eng *Histogram of Oriented Gradients*) metodom. Tu ćemo osim samih promjena u vrijednostima susjednih piksela (slično kao kod Prewitt filtera) dobiti i "orijentaciju" te promjene pa korištenjem ove značajke očekujemo bolje rezultate. Detaljno objašnjenje metode može se naći u originalnom radu ([7]).
Funkcija gubitka neuronskih mreža je unakrsna entropija i to je zapravo direktna poveznica s teorijom informacija. Mreže su trenirane stohastičkim gradijentnim spustom opisanim u prethodnom poglavlju, a stopa učenja je postavljena na $\alpha = 0.01$. Jedina razlika je što ne ažuriramo težine nakon svakog ulaznog primjera, već to radimo nakon grupe (tzv. *batch*) od 64 ulaza. Nadalje, koristi se inačica s momentumom koja uzima u obzir i smjer promjene iz prethodnog koraka s ciljem brže konvergencije.

Što se tiče modela stabla odluke, nismo ograničavali maksimalnu dubinu stabla. Tada se čvorovi granaju sve dok svaki čvor ne sadrži primjere koji pripadaju jednoj klasi ili sve dok se u svakom čvoru nalazi više od k primjera. Kao i za sve ostale parametre, za parametar k koristit ćemo standardnu vrijednost, a to je 2. Dakle, dubina stabla je određena podacima na kojima se model trenira.

3.1 MNIST skup podataka

Često zvan "Hello World!"-om dubokog učenja, MNIST (*Modified National Institute of Standards and Technology*) skup podataka sastoji se od crno-bijelih (*grayscale*) slika decimalnih znamenki koje želimo klasificirati. Slike su dimenzije 28×28 piksela. Kako je svaki piksel određen samo jednom vrijednosti, ulaz ima dubinu 1. Skup za treniranje sastoji se od 60000 slika. Klase su gotovo balansirane, tj. ima otprilike 6000 slika svake znamenke. Ista situacija je i kod testnog skupa gdje se nalazi 10000 slika, s otprilike 1000 slika svake znamenke. Kao što je već rečeno, mjerimo točnost modela, tj. postotak testnih primjera koje model točno klasificira. Nakon 20 epoha treniranja neuronskih mreža, dobiveni su sljedeći rezultati:

Stablo odluke (dubina 50)	87.54%
Stablo odluke+Prewitt (dubina 57)	88.37%
Stablo odluke+HOG (dubina 49)	88.22%
Multinomijalna logistička regresija	92.55%
Multinomijalna logistička regresija (+Prewitt)	92.64%
Multinomijalna logistička regresija (+HOG)	97.83%
Unaprijedna neuronska mreža	97.61%
Konvolucijska neuronska mreža	99.12%



Slika 3.2: Primjer jedne slike iz svake klase MNIST skupa podataka. Iznad svake slike dan je broj slika pripadne znamenke u skupovima za treniranje i testiranje.

Vidimo da je dodavanje značajki u model stabla odluke rezultiralo sitnim poboljšanjima. Kod multinomijalne logističke regresije vidimo značajan porast točnosti nakon dodavanja HOG značajke te je ona čak i veća od točnosti unaprijedne neuronske mreže. Iako su ovo dobri rezultati, napomenimo da koristimo relativno jednostavne mreže te je poznato da ovdje modeli mogu postići točnost od oko 99.9% (vidi [4]).

3.2 CIFAR-10 skup podataka

Slično kao i MNIST, CIFAR-10 (*Canadian Institute For Advanced Research*) je jedan od najpoznatijih skupova podataka u dubokom učenju. Ovdje su podaci nešto kompleksniji što problem čini težim u odnosu na klasifikaciju znamenaka. Tako su ovdje slike podijeljene u sljedećih 10 klasa: zrakoplovi, automobili, ptice, mačke, jeleni, psi, žabe, konji, brodovi i kamioni. Slike su sada dimenzije 32×32 piksela te su u boji pa je sada dubina ulaza 3. Skup za treniranje sastoji se od 50000 slika, a skup za testiranje od 10000. Klase su ovog puta u potpunosti balansirane.

Slika 3.3: Primjer jedne slike iz svake klase CIFAR-10 skupa podataka. Vidimo da su i skup za treniranje i skup za testiranje savršeno balansirani.

Korištenjem potpuno identičnih modela kao i prije i opet nakon 20 epoha treniranja neuronskih mreža, dobiveni su sljedeći rezultati:

Stablo odluke (dubina 44)	26.63%
Stablo odluke+Prewitt (dubina 52)	25.80%
Stablo odluke+HOG (dubina 45)	31.01%
Multinomijalna logistička regresija	38.16%
Multinomijalna logistička regresija (+Prewitt)	46.74%
Multinomijalna logistička regresija (+HOG)	57.01%
Unaprijedna neuronska mreža	46.57%
Konvolucijska neuronska mreža	68.21%

Općenito, dobiveni rezultati su puno lošiji što je i bilo za očekivati jer je zadatak sada puno teži. Što se tiče dodavanja dodatnih značajki, vidimo da je kod stabla odluke dodavanjem Prewitt značajke točnost čak sitno pala, dok je dodavanjem HOG značajke značajno porasla. Na ovom skupu podataka točnost modela stabala odluke može se povećati povećanjem parametra k. Tako za $k = 200$ dobijemo sljedeće rezultate:

Stablo odluke (dubina 43)	30.54%
Stablo odluke+Prewitt (dubina 51)	30.37%
Stablo odluke+HOG (dubina 45)	34.56%

Kod multinomijalne logističke regresije, obje dodatne značajke značajno su podigle točnost. Štoviše, ona je veća od točnosti unaprijedne neuronske mreže u oba slučaja. Jer je mreža relativno plitka, ona sama nije mogla naučiti reprezentacije slike koje daju bolje značajke od

značajki koje smo mi ručno dodali modelu multinomijalne logističke regresije. Očekivano, konvolucijska mreža ima najbolji rezultat. Nadalje, duljim treniranjem ovih modela ne bi mogli postići mnogo bolji rezultat jer se validacijska greška svake epohe nije smanjivala već od desete epohe nadalje, dok se trening greška naravno konstantno smanjuje. Dakle, daljnje treniranje dovelo bi do *pretreniranja*. No, kako smo već spomenuli, koristimo relativno jednostavne mreže. Čisto u ilustrativne svrhe, istrenirat ćemo još jedan dublji model koji koristi više slojeva. Dodatno, sada koristimo i *padding*, tj. proširenje slike praznim pikselima kako bi prostorne dimenzije ulaza i izlaza konvolucijskog sloja ostale iste. Kao mjera sprječavanja pretreniranja uvode se *dropout* slojevi koji postavljaju vrijednost svakog neurona iz prethodnog sloja na nulu s određenom vjerojatnosti.

```
cnn_model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu',padding='same', input_shape=(32, 32, 3)),
    layers.Conv2D(32, (3, 3), activation='relu', padding='same'),
    layers.MaxPooling2D((2, 2)),
    layers.Dropout(0.25),
    layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
    layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
    layers.MaxPooling2D((2, 2)),
    layers.Dropout(0.25),
    layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
    layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
    layers.MaxPooling2D((2, 2)),
    layers.Dropout(0.25),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

Slika 3.4: Kôd kojim smo definirali dublji model konvolucijske neuronske mreže.

Nakon 50 epoha treniranja, ova mreža postigla je točnost od 80.6%. Koristeći još više slojeva, točnost bi vjerojatno mogli dodatno podići. No, s dodavanjem novih slojeva potrebno je sve više i više vremena za treniranje pa smo donekle ograničeni računskom moći hardvera. Inače, najbolji rezultati modela na ovom skupu podataka kreću se oko 99.5% (vidi [8]).

Bibliografija

[1] C. C. Aggarwal, *Neural Network and Deep Learning: A Textbook*, Springer, Cham, 2018.

[2] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[3] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, NY, 2006.

[4] A. Byerly, T. Kalganova i I. Dear, *No Routing Needed Between Capsules*, 2021, https://arxiv.org/abs/2001.09136v6.

[5] M. Cheraghchi, *Capacity upper bounds for deletion-type channels*, Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC '18, ACM, 2018, http://dx.doi.org/10.1145/3188745.3188768.

[6] T. M. Cover i J. A. Thomas, *Elements of Information Theory*, Wiley, Hoboken, NJ, 2006.

[7] N. Dalal i B. Triggs, *Histograms of oriented gradients for human detection*, 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), sv. 1, 2005, str. 886–893.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit i N. Houlsby, *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, 2021, https://arxiv.org/abs/2010.11929v2.

[9] I. Goodfellow, Y. Bengio i A. Courville, *Deep Learning*, MIT Press, Cambridge, MA, 2016.

[10] C. Guo, G. Pleiss, Y. Sun i K. Q. Weinberger, *On Calibration of Modern Neural Networks*, Proceedings of the 34th International Conference on Machine Learning,

Proceedings of Machine Learning Research, sv. 70, PMLR, 2017, str. 1321–1330, https://proceedings.mlr.press/v70/guo17a.html.

[11] D. H. Hubel i T. N. Wiesel, *Receptive Fields of Single Neurons in the Cat's Striate Cortex*, Journal of Physiology **148** (1959), br. 3, 574–591.

[12] G. James, D. Witten, T. Hastie i R. Tibshirani, *An Introduction to Statistical Learning*, Springer, New York, 2013.

[13] S. Linnainmaa, *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*, Disertacija, University of Helsinki, 1970.

[14] D. J.C. MacKay, *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, Cambridge, UK, 2003.

[15] W. S. McCulloch i W. H. Pitts, *A Logical Calculus of Ideas Immanent in Nervous Activity*, Bulletin of Mathematical Biophysics **5** (1943), br. 4, 115–133.

[16] M. Minsky i S. Papert, *Perceptrons: An Introduction to Computational Geometry*, MIT Press, Cambridge, MA, 1969.

[17] F. Rosenblatt, *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*, Psychological Review **65** (1958), br. 6, 386–408.

[18] I. Rubinstein i R. Con, *Improved Upper and Lower Bounds on the Capacity of the Binary Deletion Channel*, 2023, https://arxiv.org/abs/2305.07156.

[19] D. E. Rumelhart, G. E. Hinton i R. J. Williams, *Learning representations by back-propagating errors*, Nature **323** (1986), br. 6088, 533–536.

[20] C. E. Shannon i W. Weaver, *The Mathematical Theory of Communication*, University of Illinois Press, Urbana, IL, 1949.

[21] J. T. Springenberg, A. Dosovitskiy, T. Brox i M. Riedmiller, *Striving for Simplicity: The All Convolutional Net*, International Conference on Learning Representations (ICLR), 2015, https://arxiv.org/abs/1412.6806.

[22] H. Tavakoli, *New Capacity Upper Bounds For Binary Deletion Channel*, 2021, https://arxiv.org/abs/2103.11904.

[23] P. J. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, Disertacija, Harvard University, Cambridge, MA, 1974.

Sažetak

U ovom radu dajemo pregled osnovnih pojmova i koncepata teorije informacija i dubokog učenja. Rad započinjemo definiranjem najvažnijih mjera teorije informacija u diskretnom slučaju. Nakon toga, obrađujemo dva najvažnija teorema čitavog područja: teorem o kodiranju izvora i teorem o kodiranju kanala. Nakon toga uvodimo mjere u neprekidnom slučaju i pokazujemo poveznicu s diskretnim slučajem. U drugom dijelu rada bavimo se dubokim učenjem. Dajemo kratki povijesni pregled počevši od prvih neuronskih modela. Ilustriramo ekvivalentnost nekih klasičnih modela strojnog učenja s modelima s jednim neuronom. Posebnu pažnju posvećujemo unaprijednim i konvolucijskim neuronskim mrežama te prolazimo kroz algoritam unazadne propagacije. Za kraj, uspoređujemo točnost različitih modela na problemima klasifikacije slika. Tu naglašavamo da kao funkciju gubitka neuronskih modela koristimo unakrsnu entropiju, veličinu iz teorije informacija.

Summary

In this paper, we provide an overview of the basic concepts and principles of information theory and deep learning. We begin by defining the most important measures of information theory in the discrete case. After that, we discuss two of the most important theorems in the entire field: the source coding theorem and the channel coding theorem. We then introduce measures in the continuous case and demonstrate the connection with the discrete case. In the second part of the paper, we focus on deep learning. We provide a brief historical overview starting from the earliest neural models. We illustrate the equivalence of some classical machine learning models with single-neuron models. We pay special attention to feedforward and convolutional neural networks and go through the backpropagation algorithm. Finally, we compare the accuracy of different models on image classification problems. Here we emphasize that the loss function used by the neural models is cross-entropy, a quantity from information theory.

Životopis

Rođen sam 1. listopada 1999. godine u Čakovcu. Osnovnu školu pohađao sam u Domašincu, a 2014. godine upisujem smjer tehničke gimnazije u Tehničkoj školi Čakovec. Nakon završetka srednje škole, godine 2018. na Prirodoslovno-matematičkom fakultetu u Zagrebu upisujem sveučilišni preddiplomski studij, smjer Matematika. U jesen 2021. godine stječem titulu sveučilišnog prvostupnika matematike te na istom fakultetu upisujem diplomski sveučilišni studij Matematička statistika.