Logistička i softmaks regresija (opis metoda, implementacija, primena nad podacima, komparativna analiza) i njihova primena kao aktivacione funkcije u neuronskim mrežama

Student: Danilo Veljović, broj indeksa: 1120

Novembar 6, 2020

Sadržaj

Glava 1

Uvod

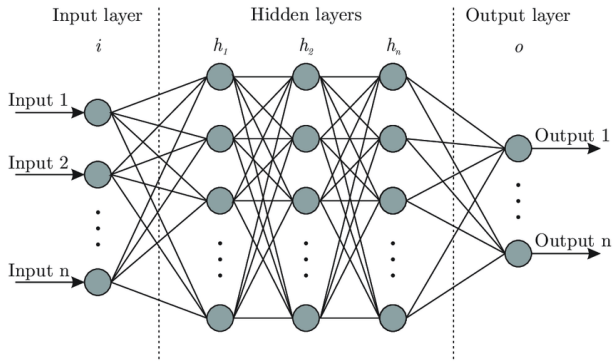
Termini poput veštačke inteligencije i mašinskog učenja su sve popularniji u savremenoj kulturi. Preko medija sve češće čujemo nagoveštaje vodećih naučnika iz ove oblasti da će sistemi koji koriste veštačku inteligenciju zameniti ljude na mnogim radnim mestima. Priča se da će prvo nestati najrepetitivniji poslovi poput vozača kamiona, radnika u skladištima, prodavaca i dr. Treba postaviti suštinsko pitanje zašto je to tako. Zašto će prvo rutinski poslovi nestati prvi? Odgovor na to pitanje daje prave smernice kada neko pokušava da shvati šta je to mašinsko učenje.

Mašinsko učenje vuče korene iz statistike. Slobodno rečeno, mašinsko učenje se može shvatiti kao primena statističkih metoda nad velikom količinom podataka, da bi se izvukle nekakve pravilnosti iz podataka. Te pravilnosti uočava model i opisuje ih različitim statističkim terminima. Model se može shvatiti kao parametrizovana funkcija koja može da „uči" iz podataka. Podaci se u mašinskom učenju dele na trening i test skupove. Trening skupovi imaju određene vrednosti za atribute i imaju izlaznu labelu klase kojoj pripadaju ili neke kontinualne vrednosti koju uzimaju. Model uzima jedan po jedan podatak iz trening skupa, ili čitavu grupu, i proba da napravi predikciju koje bi izlazne vrednosti mogle da odgovaraju tim ulazima. Kada vidi koliko njegova procena odudara od vrednosti labele za taj podatak, on se ispravi svoje parametre. Ovaj proces se naziva treniranje modela. Kada model optimizuje svoje parametre, i kada se odradi evaluacija modela, prelazi se na testiranje. Kada se model testira njemu se daju instance podataka koje nikad nije video. Za te instance on sada predviđa izlazne labele.

Posle ovog kratkog objašnjenja procesa mašinskog učenja, lako se može naslutiti odgovor na prethodno postavljeno pitanje o poslovima koje će prvo „progutati" inteligentni sistemi. Uzmimo primer radnika koji radi u skladištu i pomera kutije od mesta A do mesta B. Robot može da posmatra radnika koji nosi kutije i da zaključi da radnik na mestu A podigne kutiju, nosi je do mesta B i tu je spušta. To je njegov „trening skup" podataka. Kod ovakvog tipa inteligentih sistema najčešće se koriste konvolucione neuronske mreže za računarski vid, pomoću kojih sistem prima podatke o tome šta se dešava oko njega. Princip ostaje isti, sistem preko kamera, prima ulazne podatke, i ovo služi kao trening skup podataka. Trenira određen model, i kada se završi taj proces on može da izvede zadatak prenosa kutija. Ako se od sistema zahteva da radi nešto novo, slično prethodnom, on će probati to da uradi na jedan način, i u zavisnosti od ishoda „učiće" o tome gde je pogrešio, odnosno optimizovaće model.

Tehnike mašinskog učenja su srž inteligentnih sistema. Neki od važnijih modela mašinskog učenja su:

- *Klasifikacija* predviđa klasu koja odgovara podatku
- *Regresija* predviđa kontinualnu vrednost koja odgovara podatku
- *Klasterizacija* je tehnika nenadgledanog učenja. Za razliku od tehnika nadgledanog učenja, o kojima je do sad bilo reči, kod nenadgledanog učenja nemamo informaciju o tome koji podatak ima kakvu klasu/vrednost. Zadatak klasterizacije je da one podatke koji su „slični" (mera sličnost se definiše kao parametar tehnike) grupiše zajedno.
- *Neuronske mreže* su tehnika koja je donela revoluciju u svet mašinskog učenja i omogućila stvari poput računarskog vida i procesuiranja prirodnih jezika. Ideja za ovu tehniku dobijena je iz ideje ljudskog mozga. Cela neuronska mreža se sastoji iz veštačkih neurona, koji su međusobno povezani. Na slici 1.1 dat je prikaz jedne neuronske mreže. Neuronska mreža se sastoji iz tri sloja, *ulaznog* sloja, mnoštvo *skrivenih* slojeva i *izlaznog sloja*. Ideja kod funkcionisanja neuronske mreže je da početni slojevi prepoznaju jako male delove sistema, npr kod računarskog vida prepoznaju ivice, kod procesuiranja prirodnih jezika prepoznaju pojedina slova. Skriveni slojevi prepoznaju složenije pojave, npr geometrijske figure ili tela, ili reči i rečenice. Na kraju izlazni neuroni se aktiviraju ako je prepoznato nešto. Primera radi, mreža može da se trenira da prepoznaje slike mačaka ili da prepoznaje da li je neka rečenica nosi pozitivna ili negativna osećanja. Ako mreža prepozna sliku mačke, ili ako je određena rečenica ima pozitivna osećanja, izlazni neuron se aktivira.



Slika 1.1: Arhitektura neuronske mreže

Jedna od tehnika koja se koristi za klasifikaciju se naziva *logistička regresija*. Iako u nazivu sadrži reč *regresija* ova tehnika se ne koristi za predviđanje kontinualne vrednosti, već za predviđanje da li neki podatak pripada nekoj klasi ili ne. U nastavku rada biće reči o matematičkim osnovama logističke regresije, o funkciji koja je vrlo slična logističkoj regresiji i vrlo često se koristi u sličnim situacijama - *softmaks regresiji*. Implementiraćemo logističku i softmaks regresiju u programskom jeziku Python, primeniti je nad skupom podataka i evaluirati njihove performanse i napraviti komparativnu analizu obe tehnike. Na kraju rada biće dat prikaz primene logističke i softmaks regresije kao aktivacione funkcije („*threshold functions*") kod neuronskih mreža.

Glava 2

Teorijske osnove

2.1 Kratak pregled potrebnih pojmova iz verovatnoće i statistike

Kako bi se pravilno razumela logistička regresija potrebno je pre toga dobro poznavati njene matematičke osnove. U nastavku je data kratka rekapitulacija potrebnih pojmova iz verovatnoće i statistike koji su potrebni za njeno razumevanje. Da bi se lakše svi pojmovi dati u nastavku, razmatrani su samo u okvirima diskretnih slučajnih promenljivih.

2.1.1 Verovatnoća

Verovatnoća predstavlja odnos između broja ishoda u kome je neki događaj ispunjen i broja svih mogućih ishoda.

$$P = \frac{broj\ pozitivnih\ ishoda\ događaja}{broj\ svih\ mogućih\ ishoda} \tag{2.1}$$

Primer 1. Primer jednog događaja može biti bacanje novčića. Pozitivan ishod je kada se padne glava. Broj svih mogućih ishoda u jednom bacanju je 2 (može se pasti ili pismo ili glava). Kada zamenimo ove vrednosti u prethodnu jednačinu dobijamo:

$$P = \frac{1}{2} = 0.5 \tag{2.2}$$

Iz jednačine se vidi da je verovatnoća da u jednom bacanju novčića dobijemo glavu 0.5, odnosno 50 %.

2.1.2 Šansa

Šansa predstavlja odnos između verovatnoće da se događaj desio i verovatnoće da se događaj nije desio.

$$šansa = \frac{P(događaj\ se\ desio)}{P(događaj\ se\ nije\ desio)} = \frac{p}{1\text{-}p} \tag{2.3}$$

Ako se vratimo na primer 1 bacanja novčića, vidimo da je šansa da se događaj desi, odnosno da se padne glava jednak:

$$šansa(glava) = \frac{0.5}{0.5} = 1, tj.1{:}1 \tag{2.4}$$

Zaključujemo da je šansa da se padne glava ista kao i šansa da se ne padne glava, tj 1 prema 1.

Odnos šansi se definiše kao:

$$odnos\ šansi = \frac{šansa_1}{šansa_1} \tag{2.5}$$

Primer 2. Uzmimo dva novčića, jedan koji je fer i drugi za koji se zna da je verovatnoća da se padne glava 0.7. Naći odnos šansi ova dva novčića za događaj kada se padne glava.

Za fer novčić važi:

$$P(glava) = \frac{1}{2} = 0.5 \tag{2.6}$$

$$šansa(glava)_{fer} = \frac{0.5}{0.5} = 1, tj.1:1 \tag{2.7}$$

Za „nefer" novčić važi:

$$P(glava) = 0.7 \tag{2.8}$$

$$šansa(glava)_{nefer} = \frac{0.7}{0.3} = 2.333 \tag{2.9}$$

Odnos šansi definišemo kao:

$$odnos\ šansi = \frac{šansa(glava)_{nefer}}{šansa(glava)_{fer}} = \frac{2.333}{1} = 2.333 \tag{2.10}$$

Vidi se da su šanse da se padne glava na „("nefer) novčiću veća nego da se dobije glava na „("fer) novčiću 2.333 puta.

2.1.3 Bernulijeva raspodela

Bernulijevu raspodelu je najlakše razumeti kroz Bernulijev eksperiment. Bernulijev eksperiment je eksperiment koji se dešava jednom i koji kao rezultat može da ima uspeh (obično obeležen 1) ili neuspeh (obično obeležen 0) tj ima samo dve moguće posledice. Najbolji primer za to je bacanje novčića. Dve moguće posledice su da se padne ili pismo ili glava. Uzmimo da se uspešnim događajem smatra da se pala glava pri bacanju. Odavde sledi da ako se padne pismo, to smatramo neuspehom. Izvedemo eksperiment jednom i ako se padne glava smatramo da se desio uspešan događaj, ako se padne glava smatramo da se desio neuspešan događaj.

Definišemo slučajnu promenljivu X = broj uspeha pri bacanju novčića i $X \sim Bernoulli(p)$. Moguće vrednosti koje slučajna promenljiva može da ima (obeležavamo ih sa x) su 0 ili 1. Nula se dobija ako se pri jednom bacanju nije desio uspešan događaj, odnosno 1 ako se desio uspešan događaj. Obeležavamo to sa x = 0, 1.
Verovatnoća da se desio uspešan događaj je ujedno i jedini parametar distribucije, i obeležava se sa p.

$$P(X = 1) = p \tag{2.11}$$

Jedan parametar se prosleđuje jer imamo samo jedan eksperiment. Odavde se može primetiti da je Bernulijeva raspodela specijalan slučaj Binomne raspodele za $n = 1$.
Verovatnoća da se desio neuspešan događaj je:

$$P(X = 0) = 1 - p = q \tag{2.12}$$

Za primer fer bacanja novčića, verovatnoća uspeha je:

$$P(X = 1) = 0.5 \tag{2.13}$$

Verovatnoća neuspeha je data u jednačini

$$P(X = 0) = 0.5 \tag{2.14}$$

Očekivana vrednost (srednja vrednost) se za diskretne slučajne promenljive koje imaju Bernulijevu raspodelu dobija kao:

$$\mu = E[X] = \sum_{x_\mathrm{i} \in \mathcal{X}} x_\mathrm{i} p(X = x_\mathrm{i}) = 1 * p + 0 * q = p \tag{2.15}$$

gde je X slučajna promenljiva, x moguće vrednosti slučajne promenljive i $\mathcal{X}$ je skup vrednosti koje slučajna promenljiva može da ima.

Varijansa (mera koja pokazuje meru odstupanja od srednje vrednosti) se dobija kao:

$$\begin{aligned} Var[X] = \sigma^2 = \\ &= Cov[X, X] = \\ &= E[(X - \mu)(X - \mu)] = E[(X - \mu)^2] = \\ &= \sum_{x_\mathrm{i} \in \mathcal{X}} (x_\mathrm{i} - \mu)^2 p(X = x_\mathrm{i}) = \\ &= (0 - p)^2 * q + (1 - p)^2 * p = p^2 * q + q^2 * p = pq(p + q) = pq \end{aligned} \tag{2.16}$$

Standardna devijacija predstavlja koren iz varijanse:

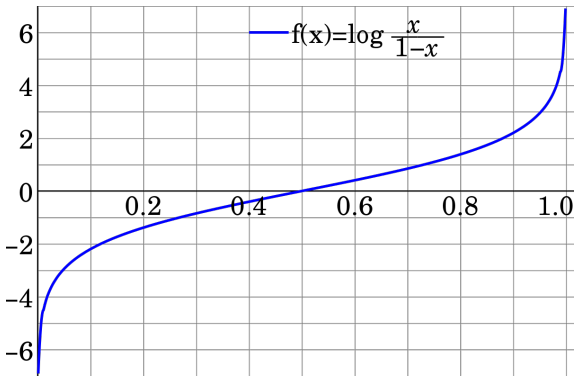
$$\sigma = \sqrt{(Var[X])} = \sqrt{pq} \tag{2.17}$$

2.2 Logistička regresija

Logistička regresija je tehnika koja se koristi za klasifikaciju. Najčešće primenu nalazi u binarnoj klasifikaciji. Pomoću nje se određuje da li neki podatak pripada nekoj klasi ili ne. Da bi se neki skup podataka mogao modelirati logističkom regresijom, on mora imati Bernulijevu raspodelu. U skladu s tim se očekuje da se na izlazu mogu pojaviti samo dve vrednosti. Nula na izlazu znači da podatak ne pripada klasi, dok jedinica znači da podatak pripada klasi.

2.2.1 Procena verovatnoće

Slično linearnoj regresiji, logistička regresija procenjuje nepoznatu verovatnoću linearne kombinacije elemenata ulaznog vektora. Nepoznatu verovatnoću procenjuje logističkom funkcijom. Nakon procene verovatnoće određene linearne kombinacije elemenata ulaznog vektora, mapira tu verovatnoću na 0 ili 1. Ako je procenjena verovatnoće p veća od 0.5 daje izlaz 1, odnosno 0, ako je verovatnoća manja od 0.5. Procenjena verovatnoća p se često obelažava sa $\hat{p}$.



Slika 2.1: Logit funkcija

Skup ulaznih podataka kod logističke regresije, bar u slučaju mašinskog učenja, predstavljen je kao matrica $\theta^T \mathbf{x}$, gde je $\mathbf{x} = \mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_\mathrm{n}$ a $\theta = \theta_1, \theta_2, ..., \theta_\mathrm{n}$, odnosno $\mathbf{x}$ je vektor atributa podataka na ulazu, a θ su odgovarajuće težine svakog od atributa.

Potrebno je naći vezu između nezavisnih promenljivih na ulazu i izlaza tako da, kao kod Bernulijeve raspodele, izlaz bude ili 0 ili 1. Ta veza se predstavlja *logit* funkcijom. Logit funkcija mapira linearnu kombinaciju nezavisno promenljivih na ulazu na izlaz . Logit funkcija predstavlja prirodni logaritam šanse. Logit funkcija je podobna jer ima samo jedan ulazni parametar koji ce kasnije biti verovatnoća p

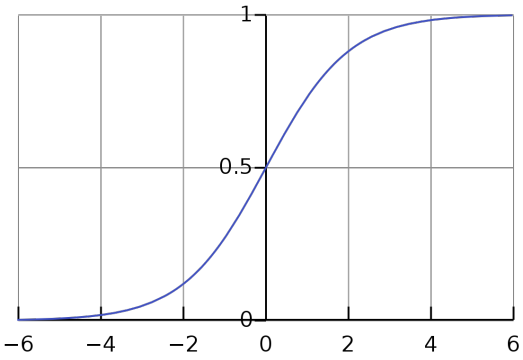
$$\mathit{logit}(p) = ln\left(\frac{p}{1\text{-}p}\right) \tag{2.18}$$

Logit funkcija je data na slici 2.1. Domen funkcije je interval $(0, +1)$, i definisana je svuda sem u 0 i 1. Kodomen funkcije je interval $(-\infty, +\infty)$. Pošto je potrebno da nama argumenti mogu da uzmu bilo koju realnu vrednost, a da izlaz bude u intervalu između 0 i 1, potražićemo inverznu funkciju ovoj. Ta funkcija se naziva *logistička* funkcija. Izvođenje je dato u jednačini 2.12. Finalni oblik logističke funkcije dat je u jednačini 2.13.

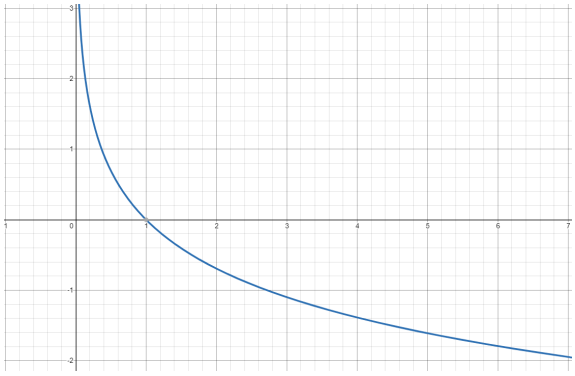
$$\begin{aligned} ln\left(\frac{p}{1\text{-}p}\right) &= \alpha, \alpha \in \mathbb{R} \\ \frac{p}{1-p} &= e^\alpha \\ p &= e^\alpha(1-p) \\ p + pe^\alpha - e^\alpha &= 0 \\ p(1+e^\alpha) - e^\alpha &= 0 \\ p &= \frac{e^\alpha}{1+e^\alpha} \end{aligned} \tag{2.19}$$

$$\mathit{logit}^{-1}(\alpha) = h_\theta(\alpha) = \sigma(\alpha) = \frac{e^\alpha}{1+e^\alpha} = \frac{1}{1+e^{-\alpha}} \tag{2.20}$$

Grafik logističke funkcije dat je na slici 2.2. Vidimo da je sada domen funkcije $(-\infty, +\infty)$, a kodomen $(0, +1)$. Ovakva funkcija se naziva *sigmoidnom* funkcijom (takođe postoji naziv i „S"-kriva) i ona daje broj uvek između 0 i 1.
Kada logistička regresija nađe verovatnoću $\hat{p}$ kao funkciju ulaznih parametara , ona



Slika 2.2: Logistička funkcija



Slika 2.3: Grafik funkcije -ln(x)

na osnovu te verovatnoće može da da predikciju da li taj podatak pripada pozitivnoj klasi.

$$\hat{y} = \begin{cases} 1, & \hat{p} > 0.5 \\ 0, & \hat{p} < 0.5 \end{cases} \tag{2.21}$$

2.2.2 Treniranje modela i funkcija gubitka

Cilj treniranja modela je da se podese parametri vektora θ tako da model daje visoku verovatnoću za pozitivne instance (y = 1) i nisku verovatnoću za negativne instance (y = 0). Funkcija gubitka se koristi kada se optimizuju parametri modela. Funkcija gubitka mapira neku realnu vrednost, najčešće je to nekakva razlika predviđene vrednosti i stvarne vrednosti labele neke instance, u realan broj. Taj realan broj predstavlja cenu. Cilj optimizacije modela je da minimizira funkciju gubitka. Početni izgled funkcije gubitka za logističku regresiju je dat u jednačini .

$$c(\theta) = \begin{cases} -ln(\hat{\text{p}}), & y = 1 \\ -ln(1-\hat{\text{p}}), & y = 0 \end{cases} \tag{2.22}$$

Na slici 2.3 je dat grafik funkcije $-ln(x)$. Kada $x \to 0$, tada $-ln(x) \to \infty$. Zbog toga će cena biti velika ako model proceni verovatnoću koja teži nuli za pozitivne instance. Takođe biće jako velika ako model proceni verovatnoću blizu jedinice za negativne instance. Ako je x u okolini jedinice, tada će -ln(x) imati vrednost oko nule. Odnosno ako cena će biti 0 ako je procenjena verovatnoća blizu 0 za negativne

instance, ili blizu 1 za pozitivne instance. Funkcija 2.22 se može takođe napisati kao:

$$c(\theta) = [\ yln(\hat{p}) + (1-y)ln(1-\hat{p})] \tag{2.23}$$

Odnosno za sve elemente ulazne matrice dobijamo konačni oblik, dat u jednačini 2.24.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[\ y^i ln(\hat{p}^i) + (1-y^i)ln(1-\hat{p}^i)] \tag{2.24}$$

Konstanta $\frac{1}{m}$ se koristi da bi se našla prosečna vrednost svih grešaka. Jednačina zatvorenog oblika koja optimalne izračunava vrednosti θ vektora nije poznata. Međutim ova funkcija jeste konveksna, tako da se optimizacionim algoritmom opadajućeg gradijenta može naći globalni minimum funkcije. Za globalni minimum funkcije gubitaka svi parametri vektora θ imaju optimalne vrednosti, odnosno najmanja je razlika između pravih labela i procenjenih labela. Kada se stigne do globalnog minimuma nije više moguće smanjivati razliku između procenjene i realne vrednosti.

U nastavku biće dat postupak dobijanja gradijenta funkcije 2.24. Biće dat postupak dobijanja parcijalnog izvoda po promenljivoj θ_j. Gradijent funkcije je specijalan slučaj Jakobijeve matrice, kada je funkcija skalarna. U opštem slučaju gradijent skalarne diferencijabilne funkcije više promenljih je vektorsko polje ∇f čija vrednost u tački p je vektor čije komponente su parcijalni izvodi f u p. Odnosno za f: $\mathbb{R}^\mathrm{n} \to \mathbb{R}$, njen gradijent je definisan kao: ∇f: $\mathbb{R}^\mathrm{n} \to \mathbb{R}^\mathrm{n}$ je definisan u tački $p = (\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_\mathrm{n})$ u n-to dimenzionalnom prostoru kao vektor:

$$\nabla f(p) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \\ \frac{\partial f}{\partial x_2} \\ . \\ . \\ . \\ \frac{\partial f}{\partial x_\mathrm{n}} \end{pmatrix} \tag{2.25}$$

Gradijent vektor se može protumačiti kao „pravac i stopa najbržeg rasta" funkcije. Ako je gradijent funkcije u tački p nenulti, pravac gradijenta je pravac u kome funkcija najbrže raste od tačke p. Intenzitet gradijenta je stopa rasta u tom pravcu. Gradijent je nulti vektor u tački samo ako je to stacionarna tačka, tj. ako je to tačka ekstremuma. Kasnije će biti potrebno da iterativnim metodama smanjujemo vrednost funkcije i za ovo će nam biti potreban negativan gradijent funkcije, umesto pozitivnog.

Izvod logističke funkcije

U nastavku je dat izvod logističke funkcije $\sigma(x) = \frac{1}{1+e^{-x}}$.

$$\begin{aligned} \frac{\partial(\sigma(x))}{\partial x} &= \frac{0*(1+e^{-x}) - (1)*(e^{-x}*(-1))}{(1+e^{-x})^2} \\ &= \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1-1+e^{-x}}{(1+e^{-x})^2} = \frac{1+e^{-x}}{(1+e^{-x})^2} - \frac{1}{(1+e^{-x})^2} \\ &= \frac{1}{1+e^{-x}} * \left(1 - \frac{1}{1+e^{-x}}\right) = \sigma(x)(1-\sigma(x)) \end{aligned} \tag{2.26}$$

Izvod funkcije gubitaka

U nastavku je dat izvod funkcije gubitaka 2.24.

Korak 1: Nalaženje diferencijala složene funkcije:

$$\begin{aligned}\frac{\partial(J(\theta))}{\partial\theta_{\mathrm{j}}} &= -\frac{1}{m}\left(\sum_{i=1}^{m}\left[\ y^i * \frac{1}{h_\theta(x^i)} * \frac{\partial(h_\theta(x^i)}{\partial\theta_{\mathrm{j}}}\right] + \sum_{i=1}^{m}\left[\ (1-y^i) * \frac{1}{1-h_\theta(x^i)} * \frac{\partial(1-h_\theta(x^i)}{\partial\theta_{\mathrm{j}}}\right]\ \right) \\ &= -\frac{1}{m}\left(\sum_{i=1}^{m}\left[\ y^i * \frac{1}{h_\theta(x^i)} * \sigma(z)(1-\sigma(z)) * \frac{\partial(\theta^T x)}{\partial\theta_{\mathrm{j}}}\right]\right) \\ &\quad -\frac{1}{m}\left(\sum_{i=1}^{m}\left[\ (1-y^i) * \frac{1}{1-h_\theta(x^i)} * -\sigma(z)(1-\sigma(z)) * \frac{\partial(\theta^T x)}{\partial\theta_{\mathrm{j}}}\right]\ \right)\end{aligned} \tag{2.27}$$

Korak 2: Zamena sigmoidne funkcije i nalaženje diferencijala argumenta:

$$\begin{aligned}\frac{\partial(J(\theta))}{\partial\theta_{\mathrm{j}}} &= -\frac{1}{m}\left(\sum_{i=1}^{m}\left[\ y^i * \frac{1}{h_\theta(x^i)} * \sigma(z)(1-\sigma(z)) * \frac{\partial(\theta^T x)}{\partial\theta_{\mathrm{j}}}\right]\right) \\ &\quad -\frac{1}{m}\left(\sum_{i=1}^{m}\left[\ (1-y^i) * \frac{1}{1-h_\theta(x^i)} * -\sigma(z)(1-\sigma(z)) * \frac{\partial(\theta^T x)}{\partial\theta_{\mathrm{j}}}\right]\ \right) \\ &= -\frac{1}{m}\left(\sum_{i=1}^{m}\left[\ y^i * \frac{1}{h_\theta(x^i)} * h_\theta(x^i)(1-h_\theta(x^i)) * {x_{\mathrm{j}}}^i\right]\right) \\ &\quad -\frac{1}{m}\left(\sum_{i=1}^{m}\left[\ (1-y^i) * \frac{1}{1-h_\theta(x^i)} * -h_\theta(x^i)(1-h_\theta(x^i)) * {x_{\mathrm{j}}}^i\right]\ \right)\end{aligned} \tag{2.28}$$

Napomena: z = $\theta^T\mathbf{x}$

Korak 3: Uprošćavanje množenjem:

$$\begin{aligned}\frac{\partial(J(\theta))}{\partial\theta_{\mathrm{j}}} &= \\ &= -\frac{1}{m}\left(\sum_{i=1}^{m}\left[\ y^i * (1-h_\theta(x^i)) * {x_{\mathrm{j}}}^i - (1-y^i) * h_\theta(x^i) * {x_{\mathrm{j}}}^i\right]\ \right) \\ &= -\frac{1}{m}\left(\sum_{i=1}^{m}\left[\ y^i - y^i * h_\theta(x^i) - h_\theta(x^i) + y^i * h_\theta(x^i)\right]\ {x_{\mathrm{j}}}^i\right) \\ &= -\frac{1}{m}\left(\sum_{i=1}^{m}\left[\ y^i - h_\theta(x^i)\right]\ {x_{\mathrm{j}}}^i\right)\end{aligned} \tag{2.29}$$

2.3 Softmaks regresija

Logistička regresija se može generalizovati tako da podržava višeklasnu klasifikaciju, bez potrebe da se trenira i kombinuje više binarnih klasifikatora. Ovakav tip

regresije se naziva *softmax regresija*, ili *multinomna logistička regresija.*

Kada se softmaks regresiji prosledi instanca $\mathbf{x}$, prvo se za tu instancu izračunava vrednost $s_{\mathrm{k}}(x)$ za svaku klasu k. Zatim se procenjuje verovatnoća za svaku klasu primenom *softmaks funkcije* nad vrednostima $s_{\mathrm{k}}(x)$. Jednačina za izračunavanje vrednosti $s_{\mathrm{k}}(x)$ je data na slici 2.30

$$s_{\mathrm{k}}(x) = \mathbf{x}^T \theta^{(k)} \tag{2.30}$$

U jednačini se može primetiti da svaka klasa ima svoj poseban vektorski parametar $\theta^{(k)}$. Svi ovi vektori se pamte kao redovi u matrici parametara Θ.. Kada se izračuna vrednost za instancu $\mathbf{x}$ može se proceniti verovatnoća $\hat{p}_{\mathrm{k}}$ da instanca pripada klasi k. To se radi tako što se vrednosti $s_{\mathrm{k}}(x)$ prosleđuju kao parametri softmaks funkciji (slika 2.31). Softmaks funkcija računa $e^{s_{\mathrm{k}}(x)}$ za svaku vrednost $s_{\mathrm{k}}(x)$ koju jedna instanca ima. Zatim svaku od vrednosti normalizuje. Ako imamo k klasa po kojima vršimo klasifikaciju, rezultat koji se dobija je k verovatnoća, po jedna za svaku klasu kojoj se vrši klasifikacija, za svaku instancu u skupu podataka.

$$\hat{p}_{\mathrm{k}} = \sigma(\mathbf{s(x)})_{\mathrm{k}} = \frac{e^{s_{\mathrm{k}}(x)}}{\sum_{j=1}^{K} s_{\mathrm{j}}(x)} \tag{2.31}$$

Oznake u jednačini 2.31: K broj klasa po kojima se vrši klasifikacija, $\mathbf{s(x)}$ je vektor koji sadrži softmaks rezultate za svaku klasu, a $\sigma(\mathbf{s(x)})_{\mathrm{k}}$ je procenjena verovatnoća da ta instanca $\mathbf{x}$ pripada klasi k

Kao kod logističke regresije, i softmaks regresija kao izlaz će imati oznaku klase koja ima najveću verovatnoću za tu instancu kao što se vidi u jednačini eq:softmaxprediction.

$$\hat{y}_{\mathrm{k}} = \underset{k}{\operatorname{argmax}}\, \sigma(\mathbf{s(x)})_{\mathrm{k}} = \underset{k}{\operatorname{argmax}}\, s_{\mathrm{k}}(\mathbf{x}) = \underset{k}{\operatorname{argmax}}((\theta^{(k)})^T)\mathbf{x}) \tag{2.32}$$

Operator *argmax* vraća vrednost promenljive koja maksimizuje funkciju. U ovoj jednačini vraća vrednost k koja maksimizuje procenjenu verovatnoću $\sigma(\mathbf{s(x)})_{\mathrm{k}}$. Softmaks regresija vraća jednu klasu po instanci i stoga bi je trebalo koristi samo sa klasama koje su međusobno isključive.

Cilj treniranja modela je da on daje visoku verovatnoću za klase kojima trening instance pripadaju i nisku verovatnoću za sve ostale klase. Minimizacija funkcije gubitaka (koja se još naziva i *cross-entropy*) će dati takav rezultat. Ona „kažnjava" model kada da nisku verovatnoću za klasu kojoj trening instanca pripada. Cross-entropy funkcija se često koristi kada se meri koliko dobro skup procenjenih verovatnoća odgovara skupu klasa kojima instance stvarno pripadaju.

$$J(\Theta) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} {y_{\mathrm{k}}}^{(i)} ln({\hat{p}_{\mathrm{k}}}^{(i)}) \tag{2.33}$$

Oznake u jednačini2.33: ${y_{\mathrm{k}}}^{(i)}$ je verovatnoća da li i-ta instanca pripada klasi k. U praksi generalno može biti jednaka 1 ili 0 jer instanca može samo da pripada ili ne pripada, nema delimičnog pripadanja. Za K = 2 (kada postoje samo dve klase), ova funkcija gubitka je jednaka funkciji gubitka kod logističke regresije.

Gradijent vektor ove funkcije gubitaka po $\theta^{(k)}$ je dat u jednačini 2.34.

$$\nabla_{\theta^k} J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} (\hat{p}_{\mathrm{k}}{}^{(i)} - y_{\mathrm{k}}{}^{(i)}) \mathbf{x}^i \tag{2.34}$$

Sad se može izračunati gradijent vektor za svaku klasu, onda da se iskoristi metoda opdajućeg gradijenta da se nađu parametri u matrixi Θ koji minimizuju funkciju gubitaka.

Glava 3

Implementacija

3.1 Logistička regresija

3.2 Softmaks regresija

Glava 4

Analiza rezultata

Glava 5

Primena kod neuronskih mreža