SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6346

Konvolucijski modeli za jednooku predikciju dubine scene

Filip Oreč

Zagreb, lipanj 2019.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

Hvala

SADRŽAJ

1. Uvod

Predikcija dubine scene je još davno poznati problem. Konvencionalni prikazi kao što su slika ili video prikazuju trodimenzionalni svijet u dvije dimenzije. Naime, time gubimo informaciju o trećoj dimenziji koja sadrži informaciju o dubini scene. Iako je dvodimenzionalna reprezentacija dovoljna u većini primjena, nekad nam je potrebna trodimenzionalna reprezentacija. Percepcija dubine proizlazi iz raznih znakova ili naznaka o dubini. Obično se dijele na binokularne znakove koji sadrže informacije u tri dimenzije i mogu se vidjeti s dva oka i monokularne znakove koji sadrže informacije u dvije dimenzije i mogu se vidjeti s jednim okom.

Tijekom godina razvile su se razne tehnike za predikciju dubine poput stereoskopije koja se oslanja na binokularne znakove. Korištenjem dvije slike iste scene koje su dobivene iz malo drugačijih kuteva, moguće je izračunati udaljenost od objekta. Aplikacijama koje uključuju razumijevanje scene, 3D modeliranje i slično jako je bitna informacija o dubini kad gore navedena organizacija podataka i tehnike nisu dostupni. U tom slučaju koristi se jednooka predikcija dubine scene koja predstavlja loše postavljen (engl. *ill-posed*) problem, jer iz jedne dvodimenzionalne slike može nastati beskonačno mnogo različitih trodimenzionalnih scena. Za ljude ovo ne predstavlja veliki problem, jer možemo jako dobro iskoristiti monokularne znakove, ali za računala predstavlja ogroman problem za riješiti s velikom preciznosti i malom uporabom resursa. Zbog navedenog razloga, u zadnje vrijeme upotrebljavaju se konvolucijski modeli koji uče odnos između piksela boje i dubine, što je i predmet ovog rada. Detaljnije ću obraditi konvolucijsku neuronsku mrežu predloženu u [2], te ju implementirati i primijeniti na na problem jednooke predikcije dubine scene.

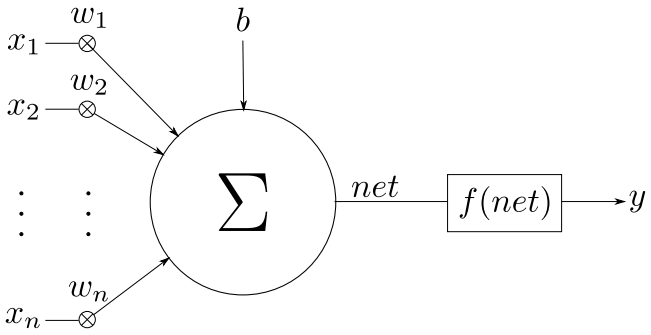
2. Umjetna neuronska mreža

Nastanak umjetnih neuronskih mreža u početku je bio inspiriran biološkim neuronima i neuronskim mrežama, ali daljnjim razvojem su se odvojile od biološke povezanosti i postale stvar inženjerstva. Definiraju se procesne jedinice zvane umjetni neuroni koji se međusobno povezuju te grade umjetne neuronske mreže.

Ovaj rad se bavi samo unaprijednim neuronskim mrežama (engl. *feedforward neural networks*). Cilj unaprijedne neuronske mreže je aproksimirati neku funkciju f^*, na način da definira preslikavanje $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$ i nauči vrijednost parametara $\boldsymbol{\theta}$ kako bi najbolje aproksimirala ciljanu funkciju f^*. Ovakvi modeli zovu se unaprijedni jer nema veza koje izlaze iz modela vraćaju nazad na ulaze.

2.1. Umjetni neuron

Warren McCulloch i Walter Pitts 1943. godine konstruirali su matematički model neurona kakav je prikazan na slici 2.1.



Slika 2.1: Umjetni neuron

Umjetni neuron prima više ulaza $x_1, x_2, ..., x_n$ koji tvore ulazni vektor $\mathbf{x}$. Taj ulazni vektor može biti stvarni ulaz ili izlaz iz nekog prethodnog neurona. Za svaku vrijednost

x_i ulaznog vektora $\mathbf{x}$ postoji vrijednost w_i koju nazivamo težina (eng. *weight*). To je vrijednost koja predstavlja utjecaj ulaza x_i na neuron. Svaki ulaz x_i množi se s odgovarajućom težinom w_i što daje umnožak $x_i \cdot w_i$. Vrijednosti $w_1, w_2, ..., w_n$ tvore vektor težina $\mathbf{w}$. Još se definira i vrijednost b koja označava pomak (engl. *bias*). Sve primljene vrijednosti se sumiraju prema izrazu 2.1.

$$net = \sum_{i=1}^{n} x_i \cdot w_i + b \tag{2.1}$$

Izraz 2.1 može se zapisati u matričnom obliku:

$$net = \mathbf{w}^\top \cdot \mathbf{x} + b \tag{2.2}$$

Pri tome ulazni vektor $\mathbf{x}$ i vektor težina $\mathbf{w}$ imaju dimenzije N$\times$1, gdje N predstavlja broj ulaza u neuron, dok je pomak b skalar. Rezultat sumiranja net dalje se predaje kao ulaz prijenosnoj funkciji koja određuje konačni izlaz o neurona prema izrazu 2.3.

$$y = f(net) = f(\mathbf{w}^\top \cdot \mathbf{x} + b) \tag{2.3}$$

Gdje f predstavlja prijenosnu funkciju.

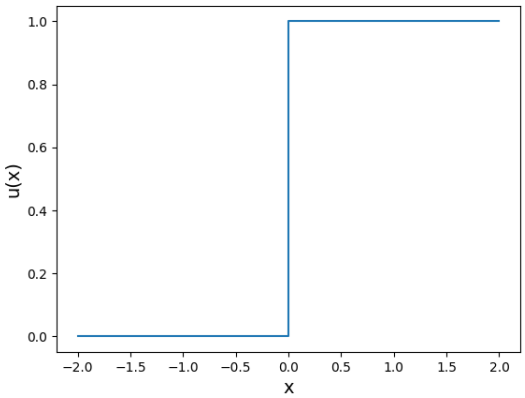
2.1.1. Prijenosne funkcije

Glavna zadaća prijenosne funkcije je pretvorba ulaznih vrijednosti u izlaznu vrijednost. Pri tome se mogu koristiti različite vrste funkcija ovisno o arhitekturi mreže. Danas postoji više prijenosnih funkcija koje se češće koriste.

Prva od njih je funkcija skoka i definirana je izrazom 2.4.

$$u(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{2.4}$$

Bitno svojstvo ove funkcije je prekid u točki $x = 0$, te zbog toga nije diferencijabilna u toj točki. Naime, to svojstvo onemogućava korištenje algoritama učenja koji se temelje na gradijentu. Izgled funkcije prikazan je na slici 2.2.

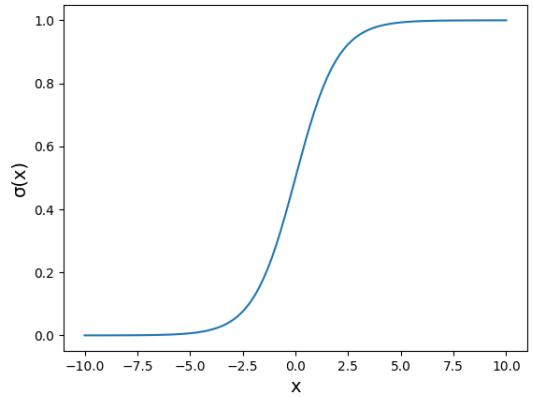


Slika 2.2: Funkcija skoka

Sigmoidalna funkcija ima jako dobra svojstva te se zbog toga često primjenjuje. Smješta bilo koji realni broj na interval $[0, 1]$. Definirana je izrazom 2.5.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$

Ova funkcija ima dosta sličnosti s funkcijom skoka. Naime, ako je x neki veliki pozitivan broj onda je $e^{-x} \approx 0$, i izlaz iz neurona je blizu 1. S druge strane ako je x neki veliki negativni broj onda $e^{-x} \to \infty$, i izlaz iz neurona je blizu 0. To se jako dobro može primijetiti na grafu funkcije prikazanom na slici 2.3.



Slika 2.3: Sigmoidalna funkcija

Velika prednost sigmoidalne funkcije u odnosu na funkciju skoka je činjenica da je derivabilna, što omogućava korištenje algoritama učenja koji se temelje na gradijentu. Derivacija ove funkcije dana je izrazom 2.6.

$$\frac{d\sigma(x)}{dx} = \sigma(x) \cdot (1 - \sigma(x)) \tag{2.6}$$

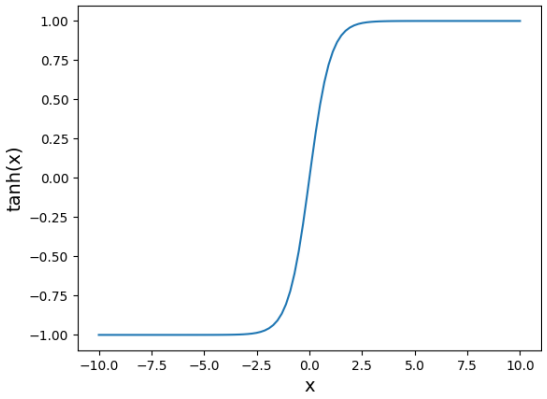
Na krajevima funkcija jako slabo reagira na promjene i zato će gradijent u tim dijelovima imati male vrijednosti. Ako mreža uči pomoću lokalnih gradijenata, u tom slučaju učenje staje, jer gradijenti ne mogu napraviti značajne promjene.

Funkciju tangens hiperbolni moguće je dobiti izravno iz sigmoidalne funkcije. Definicija je dana izrazom 2.7.

$$tanh(x) = 2\sigma(2x) - 1 \tag{2.7}$$

Ova prijenosna funkcija smješta bilo koji realni broj na interval $[-1, 1]$. Prednost je što je derivabilna, ali kao i sigmoidalna funkcija na krajevima jako slabo reagira na promjene, te ako je vrijednost ulaza u funkciju u tom području, učenje staje, što se može vidjeti na grafu funkcije 2.4. Derivacija funkcije dana je izrazom 2.8.

$$\frac{dtanh(x)}{dx} = 1 - tanh^2(x) \tag{2.8}$$



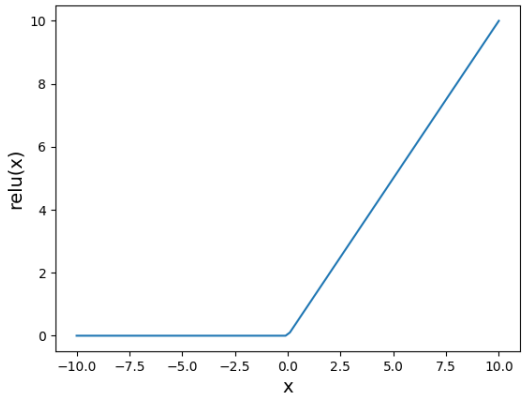
Slika 2.4: Tangens hiperbolni

Zadnjih nekoliko godina sve se češće koristi ReLU (*Recitified Linear Unit*) prijenosna funkcija. Određena je izrazom 2.9.

$$relu(x) = max(0, x) \tag{2.9}$$

Ova prijenosna funkcija sve vrijednosti veće od 0 propušta, dok vrijednosti manje od 0 uopće ne propušta, što se može vidjeti na grafu funkcije 2.9.
Derivacije ove funkcije dana je izrazom 2.10.

$$\frac{drelu(x)}{dx} = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{2.10}$$

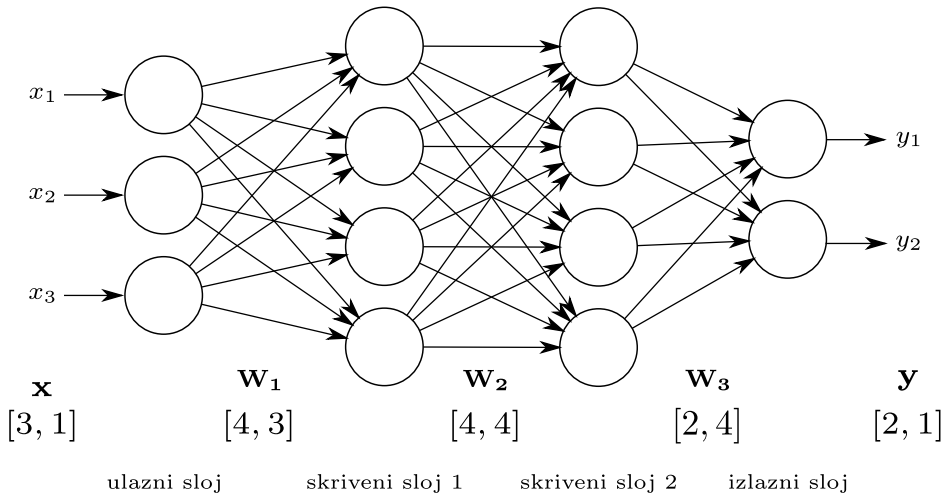


Slika 2.5: ReLU

Razlog zbog kojeg se sve češće primjenjuje je brzina računanja. Ali i ova prijenosna funkcija ima nedostatak. Iz izraza 2.10 se vidi da je za sve negativne brojeve derivacija jednaka 0, što predstavlja problem za algoritme učenja temeljene na gradijentu. Neuron kojemu je suma ulaza, odnosno net negativan, se zbog toga neće moći prilagođavati.

2.2. Arhitektura umjetne neuronske mreže

Umjetna neuronska mreža sastoji se od više umjetnih neurona koji su međusobno povezani i grupirani u slojeve. Na slici 2.6 je prikazana neuronska mreža koja ima četiri sloja. Prvi sloj je ulazni i sastoji se od tri neurona, a zadnji sloj je izlazni i sastoji se od dva neurona. Svaki sloj koji se nalazi između ulaznog i izlaznog zove se skriveni sloj.



Slika 2.6: Umjetna neuronska mreža

Ovakva mreža zove se još i potpuno povezana mreža, jer je izlaz iz svakog neurona u jednom sloju povezan sa svim neuronima u idućem sloju. Arhitektura neuronske mreže opisuje kompoziciju funkcija. Na primjer, mrežu sa slike 2.6 možemo opisati kompozicijom funkcija $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$, gdje $f^{(1)}$ predstavlja prvi skriveni sloj, $f^{(2)}$ drugi skriveni sloj, a $f^{(3)}$ izlazni sloj. Pri tome su funkcije $f^{(1)}$, $f^{(2)}$, $f^{(3)}$ definirane izrazima:

$$\mathbf{h_1} = f^{(1)}(\mathbf{x}; \mathbf{W_1}, \mathbf{b_1}) = g_1(\mathbf{W_1} \cdot \mathbf{x} + \mathbf{b_1}) \tag{2.11}$$

$$\mathbf{h_2} = f^{(2)}(\mathbf{h_1}; \mathbf{W_2}, \mathbf{b_2}) = g_2(\mathbf{W_2} \cdot \mathbf{h_1} + \mathbf{b_2}) \tag{2.12}$$

$$\mathbf{y} = f^{(3)}(\mathbf{h_2}; \mathbf{W_3}, \mathbf{b_3}) = g_3(\mathbf{W_3} \cdot \mathbf{h_2} + \mathbf{b_3}) \tag{2.13}$$

Gdje g_1, g_2 i g_3 predstavljaju prijenosne funkcije u pojedinim slojevima, a matrice $\mathbf{W_1}$, $\mathbf{W_2}$ i $\mathbf{W_3}$ predstavljaju matrice težina pojedinih slojeva. Vektori $\mathbf{b_1}$, $\mathbf{b_2}$ i $\mathbf{b_3}$ predstavljaju pomak za svaki sloj.

2.3. Učenje umjetne neuronske mreže

Učenje neuronske mreže je proces nalaženja optimalnih parametara $\boldsymbol{\theta}$, kako bi neuronska mreža što bolje aproksimirala ciljnu funkciju. Ciljna funkcija se zaključuje na osnovu uzoraka iz skupa za učenje koji se predočavaju neuronskoj mreži. Svaki uzorak je par koji se sastoji od ulaza u mrežu i željenog izlaza. Ovakav način učenja se zove nadzirano učenje (engl. *supervised learning*).

Kako bi odredili koliko dobro mreža aproksimira ciljnu funkciju koristimo funkciju gubitka. Optimiziranjem funkcije gubitka neuronska mreža se uči.

2.3.1. Funkcija gubitka

Kako bi se moglo ostvariti učenje, potrebno je definirati funkciju gubitka, koja ovisi o parametrima $\boldsymbol{\theta}$, tj. o težinama $\mathbf{W}$ i pomacima $\mathbf{b}$. Kao i prijenosna funkcija, funkcija gubitka se može definirati na više načina.

Najčešće funkcije gubitka za regresijske probleme su $\mathcal{L}_1$ i $\mathcal{L}_2$ funkcije gubitka. Definirane su sljedećim izrazima:

$$\mathcal{L}_1 = \sum_{i=0}^{n} |y_i - \hat{y}_i| \tag{2.14}$$

$$\mathcal{L}_2 = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 \tag{2.15}$$

gdje y_i predstavlja ispravnu vrijednost, $\hat{y}_i = f(\mathbf{x})$ procijenjenu vrijednost, a n broj podataka.

Za probleme klasifikacije se $\mathcal{L}_1$ i $\mathcal{L}_2$ funkcije gubitka ne koriste toliko često. Za ovaj problem se pretežito koristi unakrsna entropija (engl. *cross- entropy loss*). Definirana je izrazom 2.16.

$$\mathcal{L}_{CE} = -\sum_{i=0}^{n} y_i \cdot log(\hat{y}_i) \tag{2.16}$$

2.3.2. Gradijentni spust

Optimiziranjem funkcije gubitka parametri neuronske mreže se "štimaju" i mreža uči. Za optimiziranje funkcije gubitka koristi se gradijentni spust. To je iterativni optimizacijski algoritam za pronalaženje minimuma funkcije.

Ako je funkcija $y = f(x)$, gdje su x i y realni brojevi, funkcija koja se optimizira, onda je derivacija te funkcije $f'(x) = \frac{dy}{dx}$. Tada je aproksimacija funkcije $f(x)$ Taylorovim razvojem prvog reda:

$$f(x + \Delta x) \approx f(x) + f'(x) \cdot \Delta x \tag{2.17}$$

Ako se u tu aproksimaciju uvrsti pomak u smjeru gdje funkcija pada, odnosno u smjeru negativne derivacije dobiva se:

$$f(x - \epsilon \cdot sign(f'(x))) \approx f(x) - f'(x) \cdot \epsilon \cdot sign(f'(x)) \tag{2.18}$$

Vidi se da je $f(x - \epsilon \cdot sign(f'(x)))$ manje od $f(x)$ za dovoljno mali ϵ. Dakle, ako se x iterativno pomiče malim koracima sa suprotnim predznakom derivacije, može se doći do minimuma funkcije.

Za funkcije više varijabli koriste se parcijalne derivacije i gradijent funkcije. Parcijalna derivacija $\frac{\partial}{\partial x_i} f(\mathbf{x})$ određuje koliko se f promijeni ako se jedino x_i poveća u točki $\mathbf{x}$. Gradijent funkcije $\nabla f(\mathbf{x})$ je vektor koji sadrži sve parcijalne derivacije funkcije. Dan je izrazom 2.19.

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n} \right] \tag{2.19}$$

Aproksimacija funkcije više varijabli Taylorovim razvojem prvog reda je dana sljedećim izrazom:

$$f(\mathbf{x} + \Delta \mathbf{x}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \Delta \mathbf{x} \tag{2.20}$$

Ako se u danu aproksimaciju uvrsti pomak u smjeru najbržeg pada funkcije, odnosno u smjeru negativnog gradijenta funkcije dobiva se:

$$f(\mathbf{x} - \epsilon \cdot \nabla f(\mathbf{x})) \approx f(\mathbf{x}) - \epsilon \cdot \nabla f(\mathbf{x})^\top \nabla f(\mathbf{x}) \tag{2.21}$$

gdje ϵ predstavlja mali pozitivan parametar koji se češće zove korak učenja. Kako je $\nabla f(\mathbf{x})^\top \nabla f(\mathbf{x}) \geq 0$, funkcija $f(\mathbf{x})$ će se smanjivati sve dok ne dođe do minimuma gdje je $\nabla f(\mathbf{x}) = \mathbf{0}$. Za sljedeću iteraciju $\mathbf{x}$ se ažurira izrazom 2.22 i postupak se ponavlja za točku $\mathbf{x}'$.

$$\mathbf{x}' = \mathbf{x} - \epsilon \cdot \nabla f(\mathbf{x}) \tag{2.22}$$

Ako funkcija f predstavlja funkciju gubitka $\mathcal{L}$ onda se svakim korakom gradijentnog spusta ažuriraju se parametri neuronske mreže izrazima:

$$\mathbf{w}' = \mathbf{w} - \epsilon \cdot \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \mathbf{b}) \tag{2.23}$$

$$\mathbf{b}' = \mathbf{b} - \epsilon \cdot \nabla_{\mathbf{b}} \mathcal{L}(\mathbf{w}, \mathbf{b}) \tag{2.24}$$

Ako je skup podatak za učenje velik, računanje gradijenta za sve podatke odjednom može biti sporo. Postupak se može ubrzati korištenjem stohastičkog gradijentnog spusta. Korak gradijentnog spusta se računa samo za dio skupa za učenje (engl. *batch*). Prolazak kroz cijeli skup za učenje naziva se epoha. Ovaj način također omogućava i bolje izbjegavanje lokalnih minimuma.

2.3.3. Optimizacija

Učenje neuronske mreže gradijentnim spustom može se ubrzati raznim optimizacijskim metodama.

Jedna od tih metoda je momentum. Definira se novi vektor $\mathbf{v}$, te se mijenja jednadžba za ažuriranje parametara gradijentnog spusta na sljedeći način:

$$\begin{aligned} \mathbf{v}' &= \alpha \mathbf{v} - \epsilon \nabla \mathcal{L} \\ \mathbf{w}' &= \mathbf{w} + \mathbf{v}' \end{aligned} \tag{2.25}$$

gdje je $\alpha \in [0, 1)$ hiperparametar koji modelira prigušenje koje omogućuje zaustavljanje u minimumu. Najčešće se inicijalizira na 0.9. Vektor $\mathbf{v}$ predstavlja brzinu kretanja prema minimumu, što je gradijent veći, brzina će se sve više povečavati.

2.3.4. Regularizacija

2.3.5. Propagacija pogreške unatrag

Propagacija pogreške unatrag (engl. *back-propagation*) je algoritam za učinkovito računanje gradijenata funkcije putem rekurzivne primjene pravila ulančavanja. Dakle, za neku funkciju $f(\mathbf{x})$ gdje je $\mathbf{x}$ ulazni vektor, algoritam propagacije pogreške unatrag

računa gradijent funkcije f u točki $\mathbf{x}$, tj. $\nabla f(\mathbf{x})$. Za slučaj učenja neuronske mreže gradijentim spustom, funkcija f će odgovarati funkciji gubitka $\mathcal{L}$, a ulazni vektor $\mathbf{x}$ će se sastojati od podataka za učenje, težina $\mathbf{W}$ i pomaka $\mathbf{b}$ neuronske mreže.

U konačnici, to znači računanje parcijalnih derivacija $\partial\mathcal{L}/\partial w_{jk}^l$ i $\partial\mathcal{L}/\partial b_j^l$, gdje w_{jk}^l označava težinu koja povezuje k-ti neuron u $(l-1)$-om sloju s j-tim neuronom u l-tom sloju. Slično, b_j^l označava pomak j-tog neurona u l-tom sloju. Za računanje ovih parcijalnih derivacija uvodi se pogreška δ_j^l neurona j u sloju l:

$$\delta_j^l = \frac{\partial\mathcal{L}}{\partial net_j^l} \tag{2.26}$$

gdje net_j^l predstavlja izlaz iz neurona prije primjene aktivacijske funkcije definiran jednadžbom 2.1. Za pogrešku neurona δ_j^L u izlaznom sloju vrijedi:

$$\delta_j^L = \frac{\partial\mathcal{L}}{\partial y_j^L}\frac{\partial y_j^L}{\partial net_j^L} = \frac{\partial\mathcal{L}}{\partial y_j^L}f'(net_j^L) \tag{2.27}$$

gdje y_j^L predstavlja izlaz iz neurona, a f prijenosnu funkciju definirane jednadžbom 2.3. Jednadžba 2.27 može se zapisati u matričnom obliku:

$$\delta^L = \nabla_y\mathcal{L} \odot f'(net^L) \tag{2.28}$$

gdje $\odot$ predstavlja umnožak matrica gdje se elementi na istim mjestima množe. Primjenom ove jednadžbe mogu se dobiti pogreške samo za neurone izlaznog sloja, ali poznavanjem pogrešaka nekog sloja mogu se dobiti pogreške prethodnog sloja. Pogreške skrivenih slojeva se izračunavaju prema izrazu:

$$\delta^l = ((w^{l+1})^\top\delta^{l+1}) \odot f'(net^l) \tag{2.29}$$

Jednadžbama 2.28 i 2.29 može se izračunati pogreška δ^l za svaki sloj mreže. Prvo se izračuna pogreška izlaznog sloja δ^L pomoću jednadžbe 2.28, zatim se pomoću jednadžbe 2.29 izračuna pogreška δ^{L-1}, zatim pogreška δ^{L-2} i sve tako do ulaznog sloja mreže.

Parcijalna derivacija s obzirom na pomak odgovara upravo pogrešci neurona:

$$\frac{\partial\mathcal{L}}{\partial b_j^l} = \delta_j^l \tag{2.30}$$

dok je parcijalna derivacija s obzirom na težinu neurona:

$$\frac{\partial\mathcal{L}}{\partial w_{jk}^l} = y_k^{l-1}\delta_j^l \tag{2.31}$$

Jednadžbama 2.30 i 2.31 izračunava se ovisnost funkcije gubitka o pojedinim parametrima, što je i cilj algoritma propagacije pogreške unatrag.

3. Konvolucijska neuronska mreža

4. Zaključak

Zaključak.

LITERATURA

[1] Ian Goodfellow, Yoshua Bengio, i Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[2] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, i Nassir Navab. Deeper depth prediction with fully convolutional residual networks. *CoRR*, abs/1606.00373, 2016. URL http://arxiv.org/abs/1606.00373.

[3] Marko Čupić. Umjetna inteligencija, umjetne neuronske mreže. http://java.zemris.fer.hr/nastava/ui/ann/ann-20180604.pdf.

Konvolucijski modeli za jednooku predikciju dubine scene

Sažetak

Sažetak na hrvatskom jeziku.

Ključne riječi: Ključne riječi, odvojene zarezima.

Title

Abstract

Abstract.

Keywords: Keywords.