Konvolutionaaliset neuroverkot

Teemu Sarapisto

Kandidaatintutkielma
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 30. huhtikuuta 2018

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Teemu Sarapisto

Työn nimi — Arbetets titel — Title

Konvolutionaaliset neuroverkot

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Kandidaatintutkielma	30. huhtikuuta 2018	13

Tiivistelmä — Referat — Abstract

Viimeisen hieman yli kymmenen vuoden aikana voidaan sanoa keinotekoisten neuroverkkojen ja syväoppimisen tehneen läpimurron. Syväoppimisen voidaan katsoa syntyneen jo 40-luvulla, mutta laajamittaiseen sovelluskäyttöön se on tullut vasta viime vuosina, kun sekä riittävä määrä luokiteltua dataa, että riittävästi prosessointitehoa on tullut helposti saataville. Myös algoritmipuolella tapahtuneet edistykset ovat edesauttaneet läpimurtoa. Aikaisemmin koneoppimisen alalla haasteelliseksi osoittautuneissa sovelluskohteissa kuten kuvien sekä puheen sisällön tunnistamisessa keinotekoiset neuroverkot ovat osoittautuneet toistaiseksi ylivoimaisesti parhaiten toimiviksi ratkaisuiksi.

Neuroverkkojen opetuksessa tärkeimpiä menetelmiä ovat gradienttimenetelmä ja takaisinvirtausalgoritmi

Konvolutionaaliset neuroverkot ovat eräänlaisia neuroverkkoja jotka soveltuvat hyvin kuvien ja muiden paikallisuudesta hyötyvien syötteiden käsittelyyn

Avainsanat — Nyckelord — Keywords
neuroverkot, konvoluutio, takaisinvirtausalgoritmi

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

1 Johdanto

Syväoppimisen historia ulottuu 1940-luvulle asti, jolloin kybernetiikan tutkimuksen myötä McCulloch ja Pitts kehittivät mukaansa nimetyn McCulloch-Pitts neuronin, tarkoituksenaan luoda matemaattinen malli jolla kuvailla biologista aivoissa tapahtuvaa oppimista. Heidän kehittelemällään lineaarisella mallilla oli mahdollista tunnistaa kahden syötekategorian välillä kummasta on kyse kategoriat määrittelevien painotuksien avulla, ihmisen joutuessa määrittelemään nämä painot.

Vasta 1950-luvulla kehitettiin ensimmäinen malli joka pystyi oppimaan syötekategorioita kuvaavat painotukset niistä annettujen esimerkkien perusteella, niin kutsuttu perseptroni.

Kiinnostuksen kybernetiikkaan hiivuttua 1960-luvun aikana, seuraavan kerran merkittävää kehitystä tapahtui 80-90-luvulla konnektionismin tuodessa neuroverkkomallit takaisin suosioon. Yksi tärkeimmistä näihin aikoihin tapahtuneista kehityksistä syväoppimisen kannalta oli, kun takaisinvirtausalgoritmi (backpropagation) keksittiin 1986 mahdollistavan monikerroksisten neuroverkkojen harjoittamisen verrattain tehokkaasti.

90-luvun puolivälin jälkeen syväoppiminen eli jälleen hiljaiseloa vuoteen 2006 asti, jonka jälkeen se on ollut jatkuvasti pinnalla tähän päivään asti. Geoffrey Hinton osoitti tällöin syvien uskomusverkkojen (deep belief network) olevan harjoitettavissa tehokkaasti tasoittain ja muut tutkimusryhmät yleistivät tämän harjoitustavan muille syville keinotekoisille neuroverkoille. Näiden tutkimuksien myötä syväoppiminen terminä alkoi yleistyä, termin käytön tarkoituksena korostaa aikaisempaa syvempien verkkojen harjoitettavissa olemista.

Lopullinen syväoppimisen läpimurto tapahtui vuonna 2012 kun suurimman kuvista objektien tunnistamisen kilpailun, ImageNet Large Scale Visual Recognition Challengen (ILSVRC), voitti ensimmäistä kertaa syvä konvolutionaalinen neuroverkko. Voitto tapahtui myös huomattavalla erolla toisen sijan saavuttaneeseen sekä aikaisempien vuosien voittajiin. Tämän jälkeen kilpailun on joka vuosi voittanut syvä konvoluutioverkko, ja nykyään neuroverkot pärjäävät kyseisessä varsin rajoitetussa tunnistamistehtävässä ihmistä paremmin.

Nykyään käytössä olevat ihmisille monimutkaisissakin tehtävissä pärjäävät oppimisalgoritmit ovat pääasiassa samoja kuin jo 80-luvulla käytössä olleet. Jonkin verran muutoksia silloisiin algoritmeihin on tehty syvien verkkorakenteiden harjoittamista helpottavina yksinkertaistuksina, mutta selkeästi suurin syy syväoppimisen tärkeäksi muuttumiseen vasta äskettäin on kuitenkin yhteiskunnan digitalisoitumisen myötä merkittävästi kasvanut helposti saatavilla olevan luokitellun datan määrä, sekä valtavasti kasvanut laskentakapasiteetti, jotka olivat edellytyksiä algoritmien kunnolliselle toiminnalle.

Esimerkkinä tarvittavan harjoitusdatan määrästä konvolutionaalisten

neuroverkkojen harjoittamiseen kuvien luokittelua varten toimii yleensä tuhansia ellei jopa miljoonia kuvien sisällön perusteella etukäteen luokiteltuja kuvia. Esimerkiksi ILSVRC-kilpailun harjoitusdatana käytössä oleva ImageNet sisältää yli 14 miljoonaa luokiteltua kuvaa. (http://image-net.org/about-stats)

Kuvien sisällön tunnistamisen lisäksi syväoppiminen on osoittautunut erittäin hyödylliseksi useissa haasteellisissa sovelluskohteissa, kuten puheen sisällön, liikennemerkkien luokittelun, sekä jalankulkijoiden tunnistamisessa.

Tässä tekstissä käsitellään ensin yleisiä keinotekoisten neuroverkkojen piirteitä, kuten yksittäisen neuronin rakennetta, eteenpäinsyöttävien neuroverkkojen rakennetta, sekä sitä, miten neuroverkkoja harjoitetaan. Tämän jälkeen käsitellään miten konvolutionaaliset neuroverkot eroavat muista neuroverkoista, ja mitä sovelluksia niillä on.

2 Neuroverkkojen rakenne

2.1 Keinotekoinen neuroni

Biologisista vaikuttimistaan huolimatta keinotekoiset neuronit ovat käytännössä (TODO: selitä pelkästään tekstimuodossa re: palaute)

$$x_1, x_2, ..., x_n \mapsto \Sigma w_i x_i \mapsto f(\Sigma w_i x_i + b)$$

kaltaisia matemaattisia funktioita. Ne ottavat vastaan yhden tai useampia syötteitä x_1, x_2, ..., x_n, joista kullekin on asetettu jokin painoarvo w_i. Syötteiden ja painotuksien tulojen summa $\Sigma x_i w_i$ annetaan parametrina aktivaatiofunktiolle f ja tämän funktion arvo toimii neuronin lopullisena ulostuloarvona.

Toisinaan käytetään myös taipumusvakiota (bias) b, joka lisätään syötteiden ja painotuksien tulojen summaan, tarkoituksena säätää neuronin ulostuloarvoja painotuksista riippumatta.

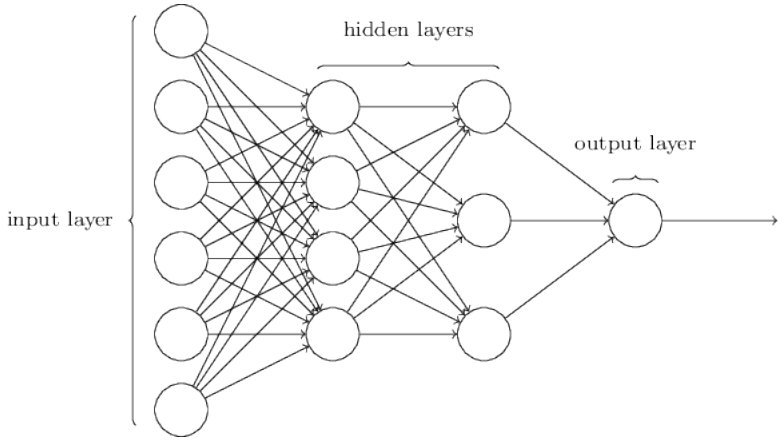
50-luvulla kehitetty ensimmäinen tällainen neuroni, perseptroni, kehitettiin 50-luvulla. Sen syötteet ja ulostuloarvot ovat binäärisiä ja aktivaatiofunktio muotoa

$$f(\Sigma w_i x_i + b) \begin{cases} 0 \text{ jos } \Sigma x_i w_i + b \leq 0 \\ 1 \text{ jos } \Sigma x_i w_i + b > 0. \end{cases} \quad (1)$$

Yksittäisen neuronin tasolla neuronien oppiminen tapahtuu syötteiden painotuksien ja taipumusarvon muuttumisen kautta. Perseptroneja käytettäessä törmätään kuitenkin usein ongelmaan, jossa yksi pieni muutos painotuksissa tai taipumusarvossa johtaa ulostuloarvon vaihtumiseen, joka saattaa aiheuttaa suuria muutoksia ulostuloarvoissa myös koko neuroverkon tasolla. Usein halutaan hienovaraisempia muutoksia ja tällöin käytetään neuroneita joiden syöte- ja paluuarvot voivat olla myös mitä vain reaalilukuja nollan ja

yhden väliltä. Esimerkiksi yksi tällainen laajalti käytössä oleva neuroni on sigmoidinen neuroni, jonka aktivaatiofunktiona toimii sigmoidinen funktio.

2.2 Keinotekoisten neuroverkkojen rakenne



Kuva 1: Tyypillinen neuroverkon rakenne, jossa kaksi piilokerrosta [4]

Yksinkertaisimman verkkorakenteen omaavat eteenpäinsyöttävät neuroverkot muodostetaan tasoittain, jossa jokaisen verkon tason neuronit saavat syötteenään niitä edeltävän tason neuroneiden ulostuloarvot. Poikkeuksena ensimmäinen taso (kuvassa vasemmanpuoleisimpana), joihin verkon syöte koodataan. Esimerkiksi haluttaessa syöttää 64x64 kuva neuroverkolle voidaan syötekerroksena käyttää 64x64 neuronin kerrosta, johon kuvan pikselien väriarvot koodataan. Neuroverkon laskennan lopputuloksena toimii verkon viimeisen tason neuroneiden ulostuloarvot.

Vaikka syväoppimista voidaan harjoittaa myös muutoin kuin keinotekoisilla neuroverkoilla, neuroverkkojen tapauksessa termillä viitataan neuroverkkojen piilokerroksiin ja niiden määrään. Kasvattamalla neuroverkkotasojen sekä tasoissa olevien neuronien määrää neuroverkoilla voidaan mallintaa entistä monimutkaisempia funktioita.

3 Neuroverkkojen harjoittaminen

Eteenpäinsyöttävien neuroverkkojen joissa on yksi piilokerros jossa on tarpeeksi neuroneita on todistettu pystyvän mallintamaan mitä tahansa jatkuvia R^n kompaktien aliavaruuksien funktioita, kunhan tarpeeksi laskentaresursseja on käytettävissä [2]. Neuroverkkojen harjoittamisen voidaan sanoa olevan neuroverkon tekemän virheen minimointia sen approksimoidessa jotakin funktiota.

Tätä neuroverkon tekemän virheen määrää voidaan mitata virhefunktion (error function) avulla. Usein käytetään neliöllistä virhefunktiota

$$E(x) = \frac{1}{2} \sum_{i=1}^{N} \|y(x_i) - a(x_i)\|^2 \tag{2}$$

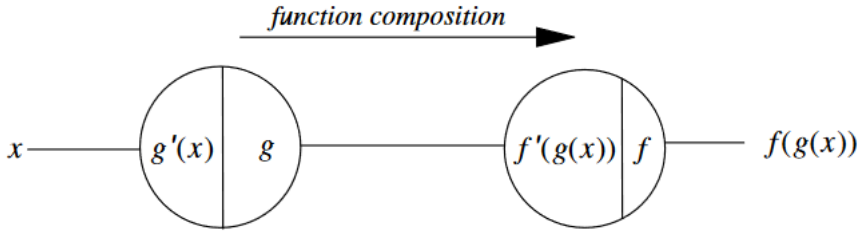
jossa $y(x_i)$ on neuroverkon tulos syötteellä x_i, $a(x_i)$ on jonkin harjoitusaineiston yksikön i:s jäsen, ja N syötteiden määrä.

3.1 Gradienttimenetelmä ja takaisinvirtausalgoritmi

Gradienttimenetelmä (gradient descent) on numeerinen menetelmä joka toimii minkä tahansa derivoitavissa olevan funktion lokaalien minimien etsintään. Käytännössä kuitenkin yleensä riittää, että funktiot ovat suurimmilta osin derivoituvia. Neuroverkkojen yhteydessä gradienttimenetelmä toimii valittaessa neuronien aktivaatiofunktioksi pääosin derivoituvia funktioita, jolloin myös koko neuroverkon tuottamat arvot ja siten virhefunktio ovat derivoituvia. Gradientti kertoo mihin suuntaan funktion arvo laskee nopeimmin, joten kuljettaessa iteratiivisesti tähän suuntaan, kunnes gradientti on tarpeeksi pieni, päädytään lähelle lokaalia minimiä.

Takaisinvirtausalgoritmilla voidaan laskea virhefunktiolle osittaisderivaatta kunkin verkon painon suhteen. Osittaisderivaatoista voidaan muodostaa virhefunktiolle gradientti ja siten soveltaa gradienttimenetelmää. Virhefunktion osittaisderivaatan selvittämiseksi yksittäisen neuronien n_i ja n_j välillä olevan verkon painon w_{ij} suhteen tarvitaan ensin välituloksena virhefunktion osittaisderivaatta neuronin n_j kohdalla.

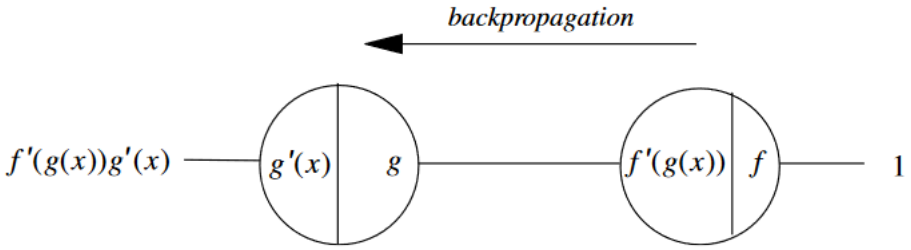
Neuroverkon voidaan ajatella olevan yhdistetty funktio sen neuroneiden funktioista, joten sen derivointiin on kätevää soveltaa ketjusääntöä. Jotta virhefunktion arvot saadaan osaksi verkkoa, lisätään ulostulokerroksen jälkeen yksi näennäinen neuroni joka ottaa ulostulokerroksen neuroneiden ulostulot syötteenään ja laskee niiden perusteella virhefunktion arvon. Nyt virhefunktion derivaattojen laskentaan voidaan hyödyntää seuraavaksi esitettävää verkkomuotoista derivaatan ketjusäännön toteutusta.



Kuva 2: Ketjusääntö ja eteenpäinvirtaus kahden solmun verkossa [5]

Ketjusäännön mukainen derivointi suoritetaan kahdessa osassa, käyden verkko ensin läpi normaaliin suuntaan syötteistä ulostuloarvojen kautta virhefunktion arvoon, ja sen jälkeen takaperin. Takaisinvirtausalgoritmi on saanut nimensä tästä taaksepäin kulkemisesta, jossa virhefunktion arvojen voidaan ajatella virtaavan taaksepäin verkon loppupäästä.

Eteenpäinsyöttövaiheessa jokaisen solmun kohdalla tallennetaan solmun toteuttaman funktion derivaatta. Kuvassa 2 tämä tallennettava arvo näkyy palloina esitettyjen solmujen vasemmassa puoliskossa.



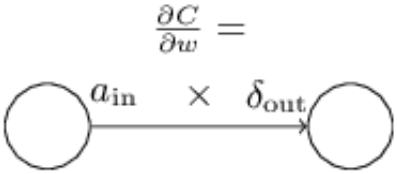
Kuva 3: Ketjusääntö ja takaisinvirtaus kahden solmun verkossa [5]

Takaisinvirtausvaiheessa verkkoa kuljetaan takaperin aikaisempaan nähden. Takaisinvirtaus aloitetaan syöttämällä verkkoon numero yksi, kuljettaen tätä arvoa mukana solmulta toiselle, ja uuteen solmuun saavuttaessa kertomalla mukana kulkeva arvo solmuun tallennetulla, kuvassa 3 solmun vasemmalla puoliskolla näkyvällä arvolla. Näin lopulta esimerkin kahden solmun läpi kulkemisen jälkeen yhdistetty funktio $f(g(x))$ on saatu muotoon $f'(g(x))g'(x)$ joka on funktion $f(g(x))$ derivaatta.

Kuvaamalla virhefunktion derivaattaa neuronin n_j kohdalla merkinnällä δ_j voidaan virhefunktion derivaatta suhteessa painoon w_{ij} esittää muodossa

$$\frac{\partial E}{\partial w_{ij}} = o_i \delta_j,$$

jossa o_i on neuronin n_i ulostuloarvo.



Kuva 4: Virhefunktion derivaatta yksittäisen painon suhteen [4]. TODO: piirrä oma kuva omilla merkinnöillä

Virhefunktion arvon laskeminen jokaiselle syötteelle ja keskiarvon ottaminen ja tämän perusteella takaisinvirtausalgoritmin suorittaminen olisi

erittäin raskasta kun syötteitä voi olla miljoonia kappaleita. Tämän takia käytetään yleensä stokastista gradienttimenetelmää, jossa virhefunktion keskiarvo lasketaan kaikkien syötteiden sijaan joukolle satunnaisesti valittuja syötteitä, ja ajetaan takaisinvirtausalgoritmi tämän tuloksen perusteella.

3.2 Ylisovitus

Suuri haaste neuroverkkojen harjoittamisessa on ylisovitus (overfitting) jossa neuroverkon virhefunktion arvo on harjoitusdatalla saatu erittäin pieneksi, mutta uuden datan kanssa virhefunktio antaa suuria arvoja. Tällöin neuroverkon oppima malli vastaa harjoitusdataa liian tarkkaan, eikä enää suoriudu yleisestä tapauksesta toivotulla tavalla. Ylisovitusta korjaamaan on kehitetty tekniikoita kuten esimerkiksi neuroniyksikköjen pudotus (dropout) jossa harjoitusvaiheessa yksittäisiä neuroneita poistetaan satunnaisesti käytöstä, joka estää yksittäisiä neuroneita naapureineen erikoistumatta tiettyyn datan ominaisuuteen liian tarkasti [6].

3.3 Esimerkki harjoittamisesta

Tässä kappaleessa käymme läpi neuroverkkojen harjoittamisesta esimerkin jossa suoritamme yhden harjoittamiskierroksen yhdellä harjoitusaineiston yksiköllä. Laskemme ensin neuroverkon antaman tuloksen esimerkkisyötteelle satunnaisesti alustetuin painotuksin, ja sen jälkeen suoritamme harjoittamiskierroksen takaisinvirtausalgoritmia ja gradienttimenetelmää hyödyntäen pyrkien näin muuttamaan neuroverkon aluksi antamaa tulosta lähemmäksi harjoitusaineiston yksikköä. Esimerkin neuroverkko on eteenpäinsyöttävä, siinä ei ole taipumusarvoja, ja siinä on kolme kerrosta: syötekerros, täysin yhdistetty piilokerros, sekä täysin yhdistetty ulostulokerros. Kussakin kerroksessa on 2 neuronia. Aktivaatiofunktiona käytetään sigmoidista funktiota

$$\frac{1}{1+e^{-x}}$$

koska se on kivan muotoinen ja yleisesti käytetty? TODO: kuva sigmoidisesta funktiosta?.

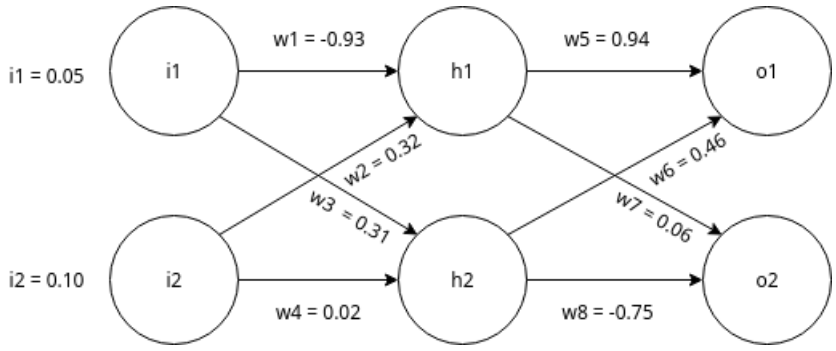
Lasketaan ensin verkon tuottamat ulostuloarvot syötteillä $i_1 = 0,05$ ja $i_2 = 0,10$ sekä väliltä $[-1, 1]$ arvotuilla painotuksilla $w_1 ... w_8$. Piilokerroksen neuroneiden syötteiden summaus tehdään kappaleessa 2.1 esitetyn kaavan $\Sigma x_i w_i$ mukaisesti

$$h_1 syöte = w_1 * i_1 + w_2 * i_2$$

$$h_1 syöte = -0,93 * 0,05 + 0,32 * 0,10 = -0,0145$$

$$h_2 syöte = w_3 * i_1 + w_4 * i_2$$

$$h_2 syöte = 0,31 * 0,05 + 0,02 * 0,10 = 0,0175$$



Kuva 5: Esimerkin neuroverkon syötteet ja painotukset

Ja syöttämällä nämä arvot sigmoidiseen aktivaatiofunktioon saadaan

$$h_1 ulos = \frac{1}{1 + e^{-(-0{,}0145)}} \approx 0,496$$

$$h_2 ulos = \frac{1}{1 + e^{-0{,}0175}} \approx 0,504.$$

Toistaen vastaavat vaiheet ulostulokerrokselle käyttäen piilokerroksen ulostuloarvoja syötteenä saadaan:

$$o_1 syöte = w_5 * h_1 + w_6 * h_2$$

$$o_1 syöte = 0,94 * 0,496 + 0,32 * 0,504 \approx 0,628$$

$$o_2 syöte = w_7 * h_1 + w_8 * h_2$$

$$o_2 syöte = 0,06 * 0,496 + (-0,75) * 0,504 \approx -0,348$$

$$o_1 ulos = \frac{1}{1 + e^{-0.627}} \approx 0,652$$

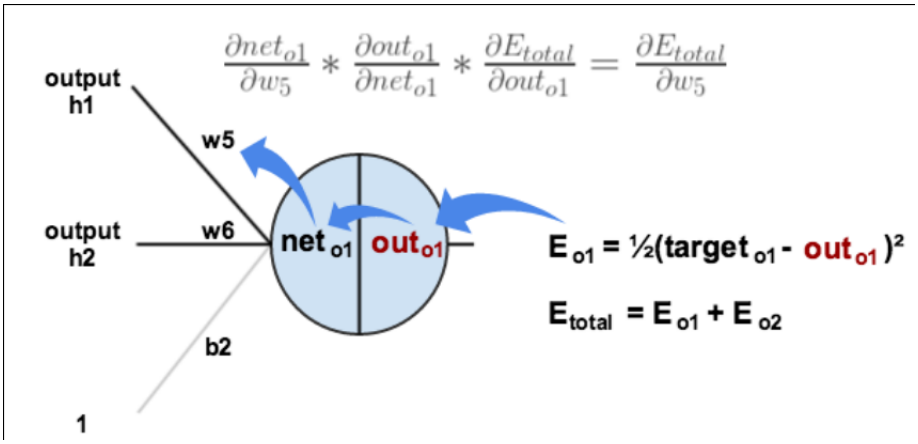
$$o_2 ulos = \frac{1}{1 + e^{-(-0.348)}} \approx 0,414.$$

Kun harjoitusaineistoksi valitaan arvot $o_{harjoitus1} = 0.01$ ja $o_{harjoitus2} = 0.99$ saadaan kaavan 2 mukaisesta virhefunktiosta arvoksi

$$0,5 * ((0,652 - 0,01)^2 + (0,414 - 0,99)^2) = 0,372.$$

Taaksepäinkulkeutumisvaiheessa painon vaikutus koko verkon virheeseen voidaan laskea ketjusäännön avulla

$$\frac{\delta E_{koko}}{\delta w_5} = \frac{\delta E_{koko}}{\delta ulos_{o1}} * \frac{\delta ulos_{o1}}{\delta summa_{o1}} * \frac{\delta summa_{o1}}{\delta w_5}.$$



Kuva 6: TODO piirrä oma kuva, atm: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Lasketaan ensimmäiseksi painon w_5 vaikutus virheeseen. Koska E_{koko} koostuu ulostulokerroksen neuroneiden tekemien virheiden summista, derivoitaessa se tietyn neuronin ulostulon suhteen, muiden neuroneiden virheiden termit putoavat pois. Eli jäljelle jää

$$\frac{\delta E_{koko}}{\delta ulos_{o1}} = 2*\frac{1}{2}(ulos_{o1}-harjoitus_{o1})^{2-1}*1 = ulos_{o1}-harjoitus_{o1} = 0.652-0.01 = 0.651$$

Sigmoidisen aktivaatiofunktion $f(x) = \frac{1}{1+e^{-x}}$ derivaatta on $f'(x) = f(x)(1-f(x))$. Koska $ulos_{o1} = \frac{1}{1+e^{-net_{o1}}}$ niin

$$\frac{\delta ulos_{o1}}{\delta summa_{o1}} = ulos_{o1}(1-ulos_{o1}) = 0.652(1-0.652) \approx 0.227.$$

Koska $summa_{o1} = w5 * out_{h1} + w_6 * out_{h2}$ niin viimeinen tarvittava osa

$$\frac{\delta summa_{o1}}{\delta w_5} = 1 * out_{h1} + w_5 * 0 + 0 = out_{h1}$$

joten $\frac{\delta E_{koko}}{\delta w_5} = 0.651 * 0.227 * 0.496 \approx 0.073$.

Virhettä korjataan yleensä jonkin oppimistahdin (learning rate) mukaisesti, joka usein muuttuu neuroverkon harjoittamisen aikana. Tässä esimerkissä käytetään mielivaltaisesti valittua $\eta = 0.5$ oppimistahtia. Korjaamme nyt w_5 virhettä asettamalla sen uudeksi arvoksi

$$w_5^+ = w_5 - \eta * \frac{\delta E_{koko}}{\delta w_5} = 0.94 - 0.5 * 0.073 = 0.9035.$$

Muiden ulostulokerroksen neuroneiden syötteiden painotukset voidaan laskea vastaavasti. Piilokerroksen painojen vaikutus ja virheen pienentäminen

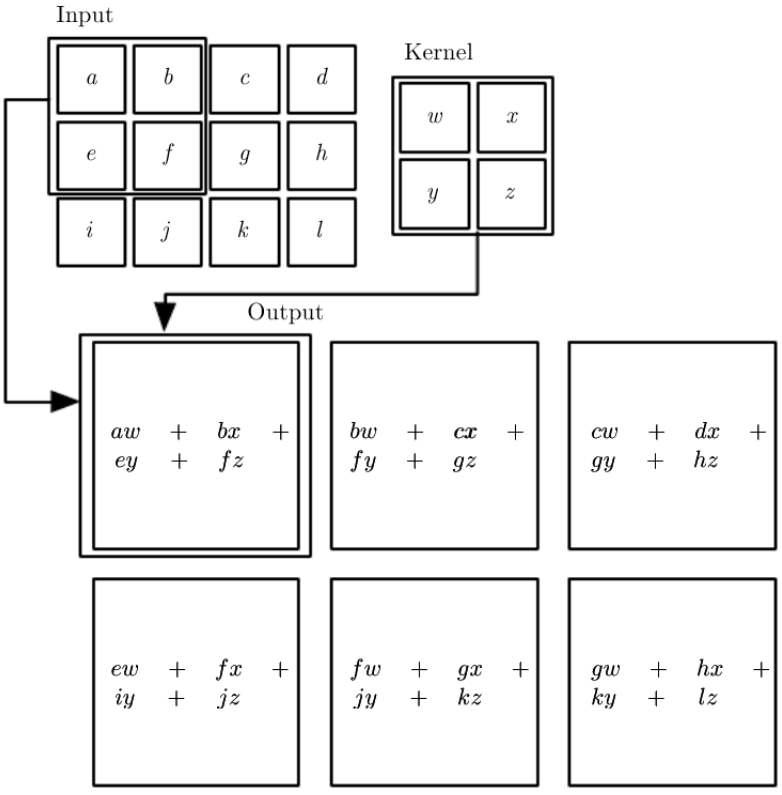
tehdään muuten vastaavasti, mutta laskettaessa kokonaisvirheen muutosta piilokerroksen neuroneiden ulostuloarvojen suhteen, on otettava huomioon piilokerroksen neuroneiden ulostuloarvojen vaikutus kaikkien ulostulokerroksen neuroneiden tekemään virheeseen.

4 Konvolutionaalisten neuroverkkojen rakenne

Konvolutionaalisten neuroverkkojen rakenne eroaa tavanomaisista täysin yhdistetyistä neuroverkoista konvoluutio- ja kokoamiskerroksien (pooling layer) olemassaolon, kerrosten mahdollisen rinnakkaisuuden, sekä ominaisuuskarttojen jaettujen painojen kautta.

(TODO: ReLu) (TODO:lisää kuvia ja selitystä ytimestä, kuten esitelmähommassa puhuttiin)

4.1 Konvoluutiokerrokset



Kuva 7: Konvoluutiokerrosten syötteiden valinta ja ulostulojen muodostuminen [1]

Konvoluutioverkkojen nimi juontaa juurensa matemaattiseen konvoluutioon, sillä kaava jolla konvoluutiokerrosten neuroneiden syötteet $a_{x,y}$ painotuksineen $w_{x,y}$ voidaan esittää matemaattisesti esimerkiksi 5x5 kokoiselle paikalliselle vastaanottavalle kentälle (local receptive field) muodossa

$$\sum_{l=0}^{4} \sum_{m=0}^{4} w_{l,m} a_{j+l,k+m}$$

joka on käytännössä diskreetti konvoluutio, jossa painotukset ovat konvoluution ydin (kernel).

Intuitiivisemmin, konvoluution voidaan ajatella olevan liukuva ikkuna, joka rajoittaa neuronit saamaan kuvan 7 mukaisesti syötteenään vain osan syötekerroksensa ulostuloista, täysin yhdistetyistä neuroverkkokerroksista poiketen. Tämä rajoittuneisuus mahdollistaa neuroneiden erikoistumisen johonkin osa-alueeseen syötteissään, joka on erityisen hyödyllistä haluttaessa käyttää neuroverkkoja syötteiden tutkintaan joissa ilmenee paikallisuutta. Esimerkiksi kuvat ovat erittäin paikallistuneita, pikselien etäisyys toisistaan korreloi vahvasti sen kanssa, liittyykö niiden sisältö toisiinsa.

Kuvasta 7 ilmenee myös toinen yleinen konvoluutiokerrosten piirre: mikäli tehdään vain konvoluutioita joissa ydin mahtuu kokonaan syötekuvaan, konvoluutiokerroksella on vähemmän ulostuloja kuin sisääntuloja. Kuvan tapauksessa nähdään syötematriisin ollessa 3x4, ja ytimen 2x2 kokoinen, ulostulomatriisin kooksi tulee 2x3. Tällä tavalla konvoluutio mahdollistaa myös sen, että konvoluutioverkot voivat ottaa vastaan vaihtelevan kokoisia syötteitä.

4.2 Ominaisuuskartat ja jaetut painot

TODO: lisää kuvia yms. siitä mitä ydin on jne.

Konvoluutiokerroksia on yleensä useita rinnakkain, ja konvoluutiokerrosten neuronit jakavat samassa kerroksessa olevien neuroneiden kesken yhteiset painot (shared weights). Kerroksien yhteiset painot mahdollistavat sen, että kukin kerros oppii tunnistamaan translationaalisesti invariantteja ominaisuuksia syötteistään, ja yksittäisen neuronin voidaan ajatella kertovan löytyykö sen saamien syötteiden alueelta tätä ominaisuutta. Esimerkiksi kuvien tapauksessa ominaisuuskartta joka löytää kuvasta suoria viivoja saattaa olla hyödyllinen, sillä suoria viivoja voi ilmetä useassa eri paikassa kuvassa, ja toisaalta korkeammalla abstraktiotasolla kissa on edelleen kissa, vaikka sitä olisi siirretty muutamia pikseleitä johonkin suuntaan.

Painotuksien jakamisen ansiosta konvolutionaalisten kerroksien ja siten neuroverkkojen parametrien määrä on usein huomattavasti pienempi kuin vain täysin yhdistettyjä kerroksia sisältävissä neuroverkoissa, mikä mahdollistaa tehokkaamman harjoittamisen.

4.3 Kokoaminen ja tehokkuus

Usein konvoluutioverkoissa käytetään heti konvoluutiokerrosten jälkeen kokoamiskerroksia, jotka tekevät yhteenvedon jostakin edeltävästä neuroverkkokerroksen alueesta. Hyvin yleinen maksimikokoamiskerros (max-pooling) antaa ulostuloksi esimerkiksi 2x2 kokoiselle syötteelle suurimman yksittäisen aktivointiarvon alueen neuroneista.

Sekä kokoamiskerrokset että jaetut painotukset yhdessä vähentävät konvolutionaalisten neuroverkkojen tarvittavien harjoitettavien parametrien määrää ja mahdollistavat niiden tehokkaan käytön suurillekkin syötteille.

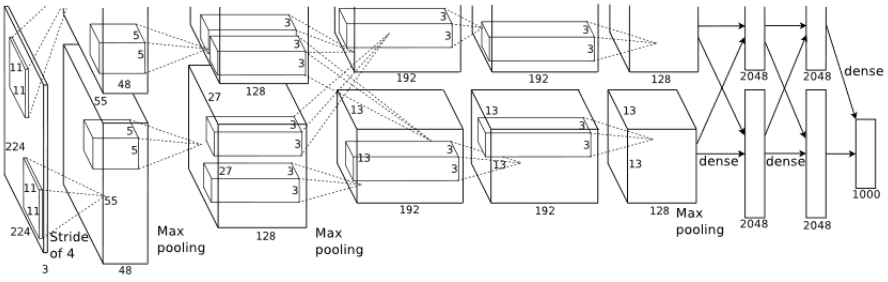
4.4 Täysin yhdistetyt kerrokset

Viimeinen kerros konvolutionaalisista neuroverkoista on yleensä tavallinen täysin yhdistetty verkkokerros, joka samalla litistää (flattening) useaan rinnakkaiseen kerrokseen jakautuneen verkon takaisin yhteen.

5 Konvolutionaalisten neuroverkkojen sovellukset

Konvoluutioverkkoja on käytetty erityisen onnistuneesti kuvien sisällön luokitteluun. Monille kuvia tunnistaville neuroverkoille yhteinen piirre on, että ulostulokerroksessa on yksi neuroni jokaista tunnistettavaa objektiluokkaa kohden, ja neuronin arvo kertoo kuinka todennäköisesti kyseisen luokan objekti löytyy kuvasta.

5.1 Kuvien luokittelu



Kuva 8: ILSVRC-2012 kilpailun voittaneen neuroverkon rakenne [3]

Vuonna 2012 ImageNet kuvantunnistuskilpailun voittaneessa Krizhevsky, Sutskever ja Hintonin (myöhemmin KSH) konvolutionaalisessa neuroverkossa on 7 piilokerrosta, joista 5 ensimmäistä ovat konvoluutiokerroksia, ja 2 viimeistä kerrosta täysin yhdistettyjä kerroksia. KSH:ssa on 1000 neuronin ulostulokerros joka vastaa sen tunnistamaa tuhatta erilaista kuvaluokkaa.

ImageNet kuvamateriaalissa on vaihtelevan kokoisia kuvia, mutta KSH:n luomassa neuroverkon syötekerros oli 3x224x224 neuronin kokoinen. KSH:n ratkaisu syötteen sopivaksi saamiseksi oli skaalata lähdekuvat ensin 256x256 pikselin kokoon, ja tämän jälkeen ottaa kuvista 224x224 osakuvia satunnaisista sijainneista lähdekuvasta, näin laajentaen harjoitusdataa ja vähentäen ylisovitusta.

KSH:n verkossa aktivaatiofunktiona toimi viimeaikoina hyväksi havaittu $R(z) = max(0, z)$ muotoinen Rectified Linear Unitiksi (ReLU) kutsuttu funktio [3].

6 Yhteenveto

Tässä tekstissä käytiin läpi ensin kaikille neuroverkoille yleisiä piirteitä kuten yksittäisen neuronin rakennetta ja sitä, miten yksittäisistä neuroneista muodostetaan eteenpäinsyöttävä neuroverkko. Takaisinvirtausalgoritmista esiteltiin kuinka sen avulla pystytään gradienttimenetelmää hyödyntäen harjoittamaan neuroverkkoja harjoitusaineiston perusteella. Tämän jälkeen käsiteltiin konvolutionaalisten neuroverkkojen eroavaisuuksia muista neuroverkoista ja lopuksi esittelimme käytännön esimerkin konvolutionaalisesta neuroverkosta.

Käsittelemättä jäi ...?

Lähteet

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

[2] Kurt Hornik, Maxwell B. Stinchcombe ja Halbert White: *Multilayer feedforward networks are universal approximators.* Neural Networks, 2(5):359–366, 1989. `https://doi.org/10.1016/0893-6080(89)90020-8`.

[3] Alex Krizhevsky, Ilya Sutskever ja Geoffrey E. Hinton: *ImageNet Classification with Deep Convolutional Neural Networks.* Teoksessa *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, sivut 1106–1114, 2012. `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks`.

[4] Michael A. Nielsen: *Neural Networks and Deep Learning.* Determination Press, 2015. `http://neuralnetworksanddeeplearning.com/`.

[5] Raúl Rojas: *Neural Networks - A Systematic Introduction.* Springer, 1996.

[6] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever ja Ruslan Salakhutdinov: *Dropout: a simple way to prevent neural networks from overfitting*. Journal of Machine Learning Research, 15(1):1929–1958, 2014. `http://dl.acm.org/citation.cfm?id=2670313`.