Aktivointifunktioiden vaikutus neuroverkkojen suorituskykyyn

Aaron Leino

LuK-tutkielma
Joulukuu 2024

Ohjaaja:
FT S.Emet

MATEMATIIKAN JA TILASTOTIETEEN LAITOS

TURUN YLIOPISTO
Matematiikan ja tilastotieteen laitos

Aaron Leino: Aktivointifunktioiden vaikutus neuroverkkojen suorituskykyyn
LuK-tutkielma, 21 s.
Sovellettu Matematiikka
Joulukuu 2024

Tutkielma tarkastelee neuroverkkojen aktivointifunktioiden ominaisuuksia ja niiden vaikutusta neuroverkon oppimisprosessiin. Tarkastelu keskittyy sigmoid-, hyperbolinen tangentti- ja ReLU-funktioihin, jotka ovat yleisesti käytettyjä aktivaatiomekanismeja neuroverkoissa.

Aluksi tutkielmassa esitellään neuroverkkojen rakenne ja historia, minkä jälkeen siirrytään neuroverkon optimointiprosessin tarkasteluun. Seuraavaksi analysoidaan aktivointifunktioiden suotuisia ja epäsuotuisia ominaisuuksia, erityisesti optimoinnin näkökulmasta. Lopuksi käsitellään tarkemmin sigmoid-, hyperbolinen tangentti- ja ReLU-funktiot, ja toteutetaan esimerkki lämpötilan ennustamisesta eteenpäin kytketyllä neuroverkolla, jossa verrataan eri aktivointifunktioiden vaikutuksia. Esimerkistä käy ilmi, että ReLU on tehokkain ja tarkin aktivointifunktio, kun taas hyperbolinen tangentti ja sigmoid vaativat pidemmän koulutusajan saavuttaakseen vastaavan ennustustarkkuuden.

Tutkielman tulokset osoittavat, että aktivointifunktion valinnalla on merkittävä vaikutus neuroverkon suorituskykyyn.

Asiasanat: keinotekoinen neuroverkko, aktivointifunktio, koneoppiminen, ReLU, hyperbolinen tangentti, sigmoid.

Sisällys

1 Johdanto

Koneoppimisen edistysaskeleet ovat tuoneet mukanaan tehokkaita menetelmiä monimutkaisten ja epälineaaristen ilmiöiden mallintamiseen ja ennustamiseen. Yksi merkittävimmistä ja laajimmin tutkituista menetelmistä on keinotekoinen neuroverkko, joka perustuu löyhästi ihmisen aivojen toimintaan [12]. Neuroverkkoja käytetään monilla eri sovellusaloilla, kuten puheentunnistuksessa, kuvantunnistuksessa ja ennakoivassa analytiikassa, ja ne ovat osoittautuneet erittäin toimiviksi työkaluiksi [8]. Odotetaankin, että tulevaisuudessa neuroverkkojen rooli kasvaa ja niiden teknologia kehittyy nopeasti. [19]

Aktivointifunktiot ovat tärkeä osa neuroverkkojen toimintaa. Aktivointifunktio määrittää, miten neuronit aktivoituvat, ja tuo neuroverkkoon epälineaarisuutta, joka on välttämätöntä monimutkaisten suhteiden oppimisessa. Aktivointifunktioilla on erilaisia ominaisuuksia, jotka vaikuttavat merkittävästi niiden käyttökelpoisuuteen erilaisissa tehtävissä. [5] Tässä tutkielmassa tarkastellaan kolmea yleisesti käytössä olevaa aktivointifunktiota: sigmoidia, hyperbolista tangenttia (tanh) ja ReLU-funktiota (Rectified Linear Unit).

Tutkielman tavoitteena on johdattaa lukija neuroverkkojen teoriaan, erityisesti niiden rakenteeseen ja koulutukseen, mitä käsitellään luvussa 2. Sen jälkeen esitellään aktivointifunktioiden suotuisia ja epäsuotuisia ominaisuuksia luvussa 3. Luvussa 4 esitellään sigmoid-, tanh- ja ReLU-funktiot yksityiskohtaisesti ja tarkastellaan niiden keskeisiä ominaisuuksia. Luvussa 5 vertaillaan näiden aktivointifunktioiden suorituskykyä lämpötilan ennustamistehtävässä. Lukijalta oletetaan perustason ymmärrystä lineaarialgebrasta ja reaalianalyysistä.

2 Neuroverkkojen perusteet

Keinotekoinen neuroverkko on tietojen- tai signaalinkäsittelyjärjestelmä, joka koostuu yksinkertaisista käsittelysolmuista, jotka on liitetty kerroksittain toisiinsa [20]. Tässä tutkielmassa neuroverkolla viitataan nimenomaan keinotekoisiin neuroverkkoihin, jollei toisin mainita. Neuroverkon perusteiden ymmärtäminen on edellytyksenä aktivointifunktioiden tarkemmalle tarkastelulle.

2.1 Neuroverkkojen kehitys

Neuroverkkojen kehitys on jakautunut muutamaan eri ajanjaksoon. Ensimmäinen merkittävä edistysaskel otettiin vuonna 1943, kun neurofysiologi Warren McCulloch ja matemaatikko Walter Pitts käsittelivät neuronien mahdollisia toimintamekanismeja julkaisussaan *The Logical Calculus of the Ideas Immanent in Nervous Activity*. Heidän mallinsa perustui hermotoiminnan "kaikki tai ei mitään" -luonteeseen, jossa yksittäinen neuroni joko aktivoitui tai ei aktivoitunut riippuen siitä, ylittikö syöte tietyn kynnysarvon. [19]

Neuroverkot jäivät kuitenkin pitkään taka-alalle koneoppimisen kehityksessä [19]. Tilanne muuttui 2010-luvulla, kun Alex Krizhevsky, Ilya Sutskever ja Geoffrey E. Hinton julkaisivat vuonna 2012 tutkimuksen kuvantunnistukseen suunnitellusta neuroverkosta, AlexNetista. Tässä verkossa oli 650 000 neuronia ja 60 miljoonaa parametria, mikä erotti Alexnetin aiemmista neuroverkkototeutuksista. [15] Myöhemmin parametrien määrä on kasvanut merkittävästi: esimerkiksi vuonna 2020 julkaistussa OpenAI:n GPT-3-mallissa on peräti 175 miljardia parametria [1].

2.2 Neuroverkon rakenne

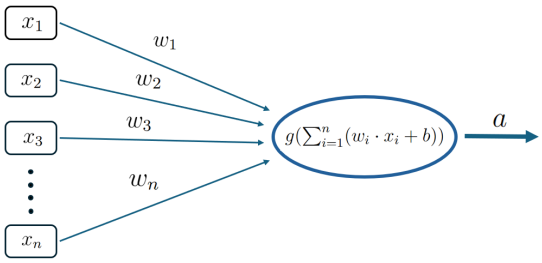
Neuroverkon rakenne muistuttaa käsitteellisellä tasolla ihmisen hermostoa [12]. Erilaisia neuroverkkotyyppejä on olemassa useita, mutta niiden rakenne koostuu suurelta osin samoista elementeistä, kuten neuroneista ja neuronikerroksista [20].

2.2.1 Neuroni

Neuroverkon perusyksiköitä ovat neuronit, joiden tehtävänä on käsitellä annetut syötteet ja välittää tulos neuroverkon seuraavalle neuronille. Lähtökohtaisesti neuronit ovat kaikki samanlaisia, mutta ne voidaan yhdistää toisiinsa useilla eri tavoilla. Yksittäisen neuronin ulostulo a voidaan esittää funktiona

$$a = f(\mathbf{w} \cdot \mathbf{x} + b),$$

jossa $f(\cdot)$ on aktivointifuktio, $\mathbf{w} = (w_1, ..., w_n)$ on painovektori, $\mathbf{x} = (x_1, ..., x_n)$ on syötevektori ja $b \in \mathbb{R}$ on harhatermi (engl. bias), jonka tarkoitus on auttaa säätelemään neuronin aktivointikynnystä. Neuronin rakennetta havainnollistetaan kuvassa 1. [18]



Kuva 1: Neuronin rakenne

2.2.2 Syöte- ja lähtökerros

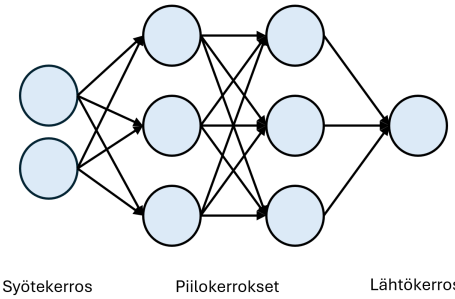
Syötekerros on neuroverkon ensimmäinen kerros, joka vastaanottaa neuroverkolle syötettävän aineiston alkuperäiset arvot. Syötekerroksen neuronit eivät yleensä muokkaa syötettä, vaan välittävät tiedon verkon seuraaville kerroksille. [19] Usein aineisto esikäsitellään, esimerkiksi normalisoimalla, jo ennen syötteen antamista neuroverkolle. Normalisoinnin avulla aineiston arvot sovitetaan sopivalle välille, esimerkiksi 0 ja 1, jolloin arvoja on helpompi verrata toisiinsa. Esikäsittely parantaa verkon laskennallista tehokkuutta ja tarkkuutta, sillä skaalatut arvot auttavat vähentämään oppimisen epävakautta ja mahdollistavat nopeamman optimoinnin. [14]

Vastaavasti lähtökerros on neuroverkon viimeinen kerros, joka laskee syötetylle aineistolle verkon lopullisen ennusteen. Lähtökerroksen rakenne riippuu paljon annetun tehtävän luonteesta esimerkiksi binäärisissä luokittelutehtävissä voidaan käyttää vain yhtä neuronia lähtökerroksessa, kun taas moniluokittelutehtävässä niitä on usein useita. Lähtökerroksen tulosta on toisin sanoen neuroverkon vastauksena annettuun tehtävään. [19]

2.2.3 Piilokerrokset

Syöte- ja lähtökerrosten väliin jää neuroverkon monimutkaisuudesta riippuen yksi tai useampi piilokerros, joissa tapahtuu varsinainen laskennallinen käsittely [19]. Piilokerrokset ovat keskeisiä neuroverkon suorituskyvylle, sillä niiden avulla verkko pystyy oppimaan syötteen ja halutun lopputuloksen välisiä epälineaarisia riippuvuuksia. Niiden määrä ja rakenne määrittävät pitkälti neuroverkon kyvyn käsitellä erilaisia tehtäviä. Lisäksi piilokerrosten ominaisuudet vaihtelevat jonkin verran neuroverkkotyypistä riippuen, mikä vaikuttaa siihen, millaisia ilmiöitä verkko pystyy mallintamaan. Seuraavaksi esitellään lyhyesti kolme yleisesti käytettyä neuroverkkotyyppiä. [6]

Eteenpäin kytketyissä neuroverkoissa (engl. Feedforward Neural Network, lyh. FNN) piilokerrokset sisältävät useita neuroneita. Kerrokset ovat järjestettynä siten, että informaatio etenee syötekerroksesta piilokerroksien kautta lähtökerrokseen yksisuuntaisesti ilman takaisinkytkentöjä. Tätä rakennetta havainnollistetaan kuvassa 2. [2] Tässä tutkielmassa keskitytään pääosin eteenpäin kytkettyihin neuroverkkoihin niiden intuitiivisen rakenteen vuoksi.



Kuva 2: Eteenpäin kytketty neuroverkko

Toinen yleisesti käytössä oleva neuroverkkotyyppi on takaisinkytketty neuroverkko (engl. Recurrent Neural Network, lyh. RNN). Toisin kuin eteenpäin kytketyissä neuroverkoissa, takaisinkytketyissä neuroverkoissa informaatio voi palata edellisiin kerroksiin. Tämän takaisinkytkennän takia neuronit pystyvät säilyttämään suhteensa edellisiin neuroneihin, mikä on erityisen hyödyllistä sekventiaalisen tai aikaperusteisen aineiston käsittelyssä. [2]

Konvoluutioneuroverkko (engl. Convolutional Neural Network, lyh. CNN) on erityisesti kaksiulotteisen aineiston oppimiseen käytetty tyyppi. Konvoluutioverkko eroaa eteenpäin kytketyistä neuroverkoista siten, että se hyödyntää konvoluutiokerroksia ja suodattimia aineiston piirteiden tunnistamiseen. [16] Tässä tutkielmassa ei syvennytä tarkemmin konvoluutioverkkoihin.

2.3 Optimointi

Neuroverkon oppiminen perustuu optimointitehtävän ratkaisemiseen, jossa minimoidaan tappiofunktio painovektorin suhteen. Optimoinnissa käytetään yleisesti takaisinsyöttöalgoritmia. [4]

2.3.1 Tappiofunktiot

Tappiofunktiolle on tyypillistä, että se laskee ennustettujen ja todellisten arvojen erotusta eli käytännössä sitä, kuinka lähellä neuroverkon ennuste on oikeaa vastausta. Monesti tehtävän luonne määrittelee sen millä tavalla tappio lasketaan. Esimerkiksi regressiotehtävissä yleisesti käytetään keskineliövirhettä (engl. Mean Squared Error, lyh. MSE), joka lasketaan seuraavasti:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2,$$

jossa N on havaintojen lukumäärä, y_i on havainnon i todellinen arvo ja $\hat{y}_i$ on neuroverkon havainnolle i ennustama arvo. Vastaavasti luokittelutehtävissä yleisesti käy-

tetään ristientropiaa (engl. Cross-Entropy Loss), joka on muotoa:

$$\text{ristientropia} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \cdot log(\hat{y}_{ij}),$$

jossa N on havaintojen lukumäärä, C on luokkien lukumäärä. Muuttuja y_{ij} on indikaattoriarvo sille, onko havainto i luokassa j vai ei, ja $\hat{y}_{ij}$ on neuroverkon ennustama todennäköisyys, että havainto i kuuluu luokkaan j. Tässä kaavan muoto eroaa hieman keskineliövirheen kaavasta, mikä johtuu luokittelutehtävien luonteesta. [3]

Tarkoituksena on siis minimoida tappiofunktio, jolloin neuroverkon ennusteet olisivat mahdollisimman lähellä todellisia arvoja. Käytännössä tämä tapahtuu gradienttilaskennan avulla, jolloin lasketaan tappiofunktion derivaatat suhteessa neuronien painoihin. [4] Gradientin käsite perustuu osittaisderivaattoihin.

Määritelmä 1. [17] Olkoot $A \subseteq \mathbb{R}^n$, $f : A \to \mathbb{R}$, $\mathbf{x} \in A^\circ$ ja $\mathbf{e} = (e_1, e_2, \ldots, e_n)$, jossa $e_i = 1$, kun $i = j$ ja $e_i = 0$ muulloin. Jos raja-arvo

$$\lim_{h\to 0} \frac{f(\mathbf{x} + he_j) - f(\mathbf{x})}{h}$$

on olemassa, niin se on funktion f *osittaisderivaatta* muuttujan x_j suhteen pisteessä $\mathbf{x}$. Osittaisderivaatasta käytetään tässä tutkielmassa merkintää $\frac{\partial f}{\partial x_j}(\mathbf{x})$.

Osittaisderivaatan laskemisessa tutkitaan funktion muutosta vain yhden muuttujan suhteen, kun muut muuttujat ovat vakioita. Esimerkiksi osittaisderivaatan erotusosamäärässä ainoastaan x_j muuttuu, kun muut vektorin $\mathbf{x}$ komponentit $x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n$ pysyvät vakioina:

$$\frac{\partial f}{\partial x_j}(\mathbf{x}) = \lim_{h\to 0} \frac{f(x_1, \ldots, x_j + h, \ldots, x_n) - f(\mathbf{x})}{h}.$$

Tämän vuoksi osittaisderivaattojen laskemiseen ei tarvitse johtaa erillisiä osittaisderivointisääntöjä, vaan tavallisen derivoinnin säännöt riittävät. Tämä yhteneväisyys tekee osittaisderivoinnista paitsi yksinkertaisempaa myös laskennallisesti tehokasta. [17]

Määritelmä 2. [17] Oletetaan, että funktiolla $f : \mathbb{R}^n \to \mathbb{R}$ on olemassa osittaisderivaatat pisteessä $\mathbf{x} \in \mathbb{R}^n$. Sen osittaisderivaattojen muodostamaa vektoria kutsutaan funkion f *gradientiksi* pisteessä $\mathbf{x}$. Gradientille käytetään merkintää

$$\nabla f(\mathbf{x}) = (\frac{\partial f}{\partial x_1}(\mathbf{x}), \frac{\partial f}{\partial x_2}(\mathbf{x}), \ldots, \frac{\partial f}{\partial x_n}(\mathbf{x})).$$

Samalla tavalla kuin derivaatta määrittelee muutosnopeuden suunnan yhden muuttujan funktiolle, gradientti määrittelee suunnan, jossa useamman muuttujan funktion muutos on nopeinta [17]. Tappiofunktiolle taas gradientti määrittää suunnan,

missä tappio kasvaa eniten. Koska tarkoituksena on saavuttaa tappiofunktion minimi, niin neuroverkon painovektoria Θ päivitetään ottamalla askelia gradientin vastakkaiseen suuntaan:

$$\Theta_{n+1} = \Theta - \eta \nabla C(\Theta),$$

jossa η on askelpituus ja $C(\cdot)$ on tappiofunktio. [13]

Gradienttilaskennassa jokainen uusi painovektori Θ_{n+1} on hieman lähempänä tappiofunktion $C(\Theta)$ minimoivaa painovektoria. Tätä optimointialgoritmia kutsutaan gradienttimenetelmäksi (engl. gradient descent), ja se on yksi yleisimmin käytössä olevista algoritmeista neuroverkkojen koulutuksessa.

2.3.2 Takaisinsyöttöalgoritmi

Takaisinsyöttöalgoritmin tarkoituksena on minimoida tappiofunktio muuttamalla neuroverkon painoja siten, että ennusteen ja todellisen tuloksen välinen virhe pienenee. Käytännössä siis lasketaan jokaisen kerroksen painojen vaikutusta tappiofunktioon, jotta painoja voidaan päivittää oikein. Takaisinsyöttöalgoritmi koostuu neljästä päävaiheesta:

1. Eteenpäin kulkeva laskenta (engl. Forward Pass): Syöte kulkee neuroverkon läpi syötekerroksesta lähtökerrokseen, jolloin neuroverkko laskee ennusteen, jota verrataan oikeaan tulokseen tappiofunktion avulla.

2. Tappiofunktion laskenta: Tehtävästä riippuen määritellään sopiva tappiofunktio $C(\Theta)$, jolla arvioidaan neuroverkon ennusteen tarkkuutta.

3. Gradientin laskenta takaisinsyötöllä: Lasketaan tappiofunktion gradientti suhteessa painoihin.

4. Painojen päivitys: Päivitetään painot valitulla optimointialgoritmilla, esimerkiksi gradienttimenetelmällä, jotta neuroverkon virhe pienenee iteratiivisesti.

Tappiofunktion C arvo kuvaa virhettä lähtökerroksessa, eli kuinka paljon neuroverkon ennuste eroaa oikeasta tuloksesta. Näin ollen lähtökerroksen gradientti osoittaa, mihin suuntaan painoja tulisi päivittää virheen minimoimiseksi. Samalla tavalla piilokerroksen gradientti kertoo, kuinka paljon kyseisen kerroksen painoja tulisi muuttaa, jotta virhe pienenisi. Tämä virheen "kuljettaminen" neuroverkon läpi on keskeinen ajatus takaisinsyöttöalgoritmissa, ja siinä hyödynnetään *ketjusääntöä*. [4]

Lause 1. *(Ketjusääntö) Olkoot $f : A \to B$ ja $g : B \to \mathbb{R}$. Jos funktio f on derivoituva pisteessä $x_0 \in A$ ja funktio g on derivoituva pisteessä $f(x_0) \in B$, niin funktio $g \circ f : A \to \mathbb{R}$ on derivoituva pisteessä x_0 ja*

$$(g \circ f)'(x_0) = g'(f(x_0))f'(x_0).$$

Todistus. Ks. [10] s. 98. □

Käytetään seuraavia merkintojä gradienttien laskennan havainnollistamiseen:

- $\mathbf{x}$: syötevektori,
- y: kohdeulostulo,
- C: tappiofunktio,
- L: kerrosten lukumäärä,
- $W^l = w^l_{jk}$: painomatriisi kerroksien $l-1$ ja l välillä ja w^l_{jk} on kerroksen $l-1$ neuronin j ja kerroksen l neuronin k välinen paino,
- f^l: kerroksen l aktivointifunkiot. Yksinkertaistamista varten saman kerroksen neuroneille on sama aktivointifunktio.
- a^l_j: kerroksen l neuronin j ulostulo,
- z^l: kerroksen l painojen summa,
- δ^l: kerroksen l virhegradientti.

Näin ollen neuroverkko voidaan ilmaista funktiona

$$g(\mathbf{x}) = f^L(W^L f^{L-1}(W^{L-1} \dots f^1(W^1 \mathbf{x})).$$

Koko neuroverkon ulostulo on määritelty rekursiivisesti, jonka vuoksi virheet lasketaan lähtökerroksesta kohti syötekerrosta. Näin ollen kerroksen $l+1$ virhe vaikuttaa edellisen kerroksen l virheeseen. Ensimmäinen askel on laskea lähtökerroksen virhegradientti, joka on tappiofunktion C derivaatta suhteessa lähtökerroksen neuronien ulostuloon a^L:

$$\delta^L = \frac{\partial C}{\partial a^L}.$$

Yleisessä tapauksessa kerroksen l virhegradientin laskemiseen voidaan soveltaa ketjusääntöä:

$$\delta^l = \frac{\partial C}{\partial z^l} = \frac{\partial C}{\partial z^{l+1}} \cdot \frac{\partial z^{l+1}}{\partial z^l}.$$

Tässä $\dfrac{\partial C}{\partial z^{l+1}} = \delta^{l+1}$. Koska $z^{l+1} = W^{l+1} \circ f^l(z^l)$, painotettujen summien derivaatta voidaan hajottaa osittaisderivaattojen avulla:

$$\frac{\partial z^{l+1}}{\partial z^l} = W^{l+1} \circ f^{(l)\prime}(z^l),$$

jolloin kerroksen l virhegradientti on muotoa:

$$\delta^l = \delta^{l+1} W^{l+1} \circ f^{(l)\prime}(z^l). \tag{1}$$

Kun virhegradientit δ^L on laskettu kaikille kerroksille, voidaan palata takaisinsyöttöalgoritmin vaiheeseen 4 painojen päivittämiseksi. Tämän jälkeen neuroverkon painoja säädetään iteratiivisesti suuntaan, joka pienentää tappiofunktion arvoa. Algoritmia toistetaan, kunnes tappiofunktion minimi on saavutettu tai se on riittävän lähellä minimiä.

On erityisen tärkeä ymmärtää takaisinsyöttöalgoritmin suhdetta aktivointifunktioiden ominaisuuksiin, sillä aktivointifunktiot vaikuttavat suoraan virhegradientin laskentaan. Erityisesti aktivointifunktioiden suotuisat ja epäsuotuisat ominaisuudet korostuvat takaisinsyötön aikana, koska algoritmi perustuu niiden derivaattoihin. Aktivointifunktioiden derivaatat voivat joko tukea virhegradientin tehokasta kuljettamista tai hidastaa sitä, mikä vaikuttaa suoraan neuroverkon oppimisprosessin nopeuteen ja luotettavuuteen.

3 Aktivointifunktioiden ominaisuuksista

Aktivointifunktioilla on tärkeä merkitys neuroverkon toiminnassa. Tehokas aktivointifunktio voi nopeuttaa neuroverkon oppimista, kun taas tehtävään huonosti soveltuva funktio voi jopa pysäyttää oppimisen. Aktivointifunktion valinnassa tuleekin ymmärtää tehtävän luonne. Esimerkiksi yksi funktio voi olla tehokas tekstinkäsittelyssä, mutta tehoton sään ennustamisessa. Kuitenkin kaikkia tehokkaita aktivointifunktioita yhdistää yksi keskeinen ominaisuus: epälineaarisuus. [5]

3.1 Epälineaarisuus ja differoituvuus

Epälineaarisuus on yksi tärkeimmistä ominaisuuksista, jota tehokkaalta aktivointifunktiolta edellytetään. Jos neuroverkon käyttämä aktivointifunktio on lineaarinen, silloin koko neuroverkko on vain lineaarinen malli, eikä se pysty oppimaan monimutkaisia riippuvuuksia.

Lause 2. *Olkoon eteenpäin kytketyssä neuroverkossa P kerrosta, ja jokaisessa kerroksessa käytetään lineaarista aktivointifunktiota $f(x) = Bx + C$. Näin ollen koko neuroverkko on lineaarinen malli.*

Todistus. Todistetaan väite induktiolla ja normalisoidaan lineaarinen aktivointifunktio muotoon $f(x) = x$, jolloin funktio on yksinkertaisempi, mutta lineaarisuus säilyy.

1. Perusaskel: Kun $P = 1$, neuroverkossa on vain yksi kerros, jolloin sen ulostulo on $y = f(Wx+b)$, jossa W on painomatriisi ja b on harhatermi. Koska $f(x) = x$, niin $y = Wx + b$, joka on lineaarinen funktio. Näin ollen perusaskel pitää paikkansa.

2. Induktio-oletus: Oletetaan, että väite pitää paikkansa, kun $P = k$.

3. Induktioaskel: Todistetaan, että väite pitää paikkansa myös, kun $P = k + 1$. Induktio-oletuksen mukaan $y^{(k)} = W^{(k)}x+b^{(k)}$, jossa (k) notaatio tarkoittaa k-kerroksisen neuroverkon yhdistettyä painomatriisia ja harhatermiä. Kun $P = k + 1$, uusi ulostulo on

$$\begin{aligned} y^{(k+1)} &= f(W^{(k+1)}y^{(k)} + b^{(k+1)}) \\ &= W^{(k+1)}y^{(k)} + b^{(k+1)} \\ &= W^{(k+1)}(W^{(k)}x + b^{(k)}) + b^{(k+1)} \\ &= (W^{(k+1)}W^{(k)})x + (W^{(k+1)}b^{(k)} + b^{(k+1)}), \end{aligned}$$

   joka on syötteen x lineaarinen funktio.

4. Johtopäätös: Osoitettiin, että väite pätee luvulle $P = 1$, ja että väite pätee luvulle $P = k + 1$, jos se pätee luvulle $P = k$. Näin ollen eteenpäin kytketty neuroverkko on lineaarinen kaikilla kerroksilla P, kun aktivointifunktiot ovat lineaarisia.

□

Vaikka lineaarisuutta koskeva tulos pätee vain eteenpäin kytketyille neuroverkoille, se osoittaa, kuinka neuroverkon kyky oppia monimutkaisia riippuvuuksia heikkenee, jos aktivointifunktiot ovat lineaarisia. Tällöin monikerroksinen neuroverkko voitaisiin pelkistää yhdeksi yleistetyksi lineaariseksi malliksi, mikä rajoittaisi merkittävästi sen laskennallista kapasiteettia ja joustavuutta [11].

Epälineaarisuuden lisäksi on tärkeää, että aktivointifunktio on jatkuvasti differoituva, jolloin takaisinsyöttöalgoritmi pystyy laskemaan virhegradientit ja päivittämään painovektoria. Kuitenkaan kaikissa tapauksissa differoituvuus ei ole ehdoton edellytys aktivointifunktioille jos kohdat, joissa funktio ei ole differoituva, ovat harvinaisia ja ennalta tiedossa. Differoituvuudessa epälineaarisuus tuo aiheuttaa muitakin haasteita neuroverkon optimoinnissa, kuten katoavan gradientin ja aktivointifunktion saturoitumisen [5].

3.2 Saturoituminen ja katoava gradientti

Saturoitumisella tarkoitetaan tässä asiayhteydessä tilannetta, jossa aktivointifunktio $f(x)$ lähestyy ääriarvoaan $\lim_{x\to\infty} f(x) = c$ tai vastaavasti $\lim_{x\to-\infty} f(x) = b$, jossa c, $b \in \mathbb{R}$. Tällöin funktion derivaatta lähestyy nollaa, mikä hidastaa neuroverkon painojen päivitystä tai pysäyttää oppimisen kokonaan. [5]

Saturoitumisen vaikutuksia voidaan havainnollistaa yhtälön (1) avulla. Kun aktivointifunktion derivaatta $f'(x)$ lähestyy nollaa, niin kerroksen l virhegradientti lähestyy nollaa eikä algoritmi silloin pysty päivittämään kerroksen l painoja tehokkaasti ja oppiminen hidastuu.

Saturoituminen siis johtaa tilanteeseen, jossa $f'(x) < 1$, mikä aiheuttaa ongelmia erityisesti syvissä neuroverkoissa, joissa on useita piilokerroksia. Esimerkiksi yhtälön (1) mukaan jokaisen kerroksen l virhegradienttia laskettaessa kerroksen l aktivointifunktio derivoidaan ja tulos kerrotaan kerroksen $l+1$ virhegradientilla. Näin ollen ensimmäisen kerroksen virhegradientti sisältää kaikkien myöhempien kerrosten aktivointifunktioiden derivaatat. Jos esimerkiksi jokaisen kerroksen l aktivointifunktion derivaatta $f'(z) = 0.5$, niin ensimmäisen kerroksen virhegradientti pienenee eksponentiaalisesti arvoksi 0.5^L, kun kerrosten määrä L kasvaa. Tätä ilmiötä kutsutaan katoavan gradientin ongelmaksi (vanishing gradient problem) [5].

4 Aktivointifunktioiden vertailu

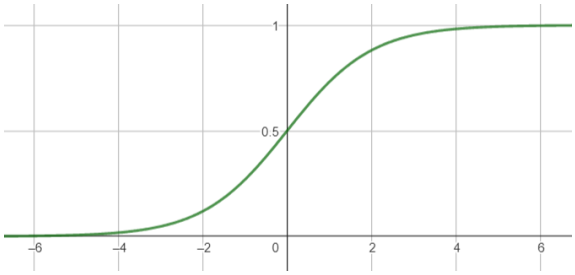
Havainnollistetaan aktiivointifunktioiden ominaisuuksia kolmella yleisesti käytössä olevalla aktivointifunktiolla: sigmoidilla, hyperbolisella tangentilla ja ReLU-funktiolla.

4.1 Sigmoid

Sigmoid-funktio on erityisesti binääriluokittelussa yksi yleisimmin käytetyistä aktivointifunktioista. Nimenomaan neuroverkkojen asiayhteydessä tarkoitetaan funktiota

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

jossa $x \in \mathbb{R}$ ja $\sigma(x) \in (0, 1)$. Funktion kuvaajaa havainnollistetaan kuvassa 3. [5]



Kuva 3: Sigmoid-funktion kuvaaja

Sigmoid-funktion hyödyt tulevat esiin erityiseti binääriseen luokitteluun perustuvissa tehtävissä maalijoukon luonteen takia. Funktio on selkeästi epälineaarinen, joka mahdollistaa perusluonteisten riippuvuuksien muodostamisen aineistosta. Lisäksi sigmoid on jatkuvasti derivoituva, mikä on suotavaa erityisesti takaisinsyöttöalgoritmissa. [5]

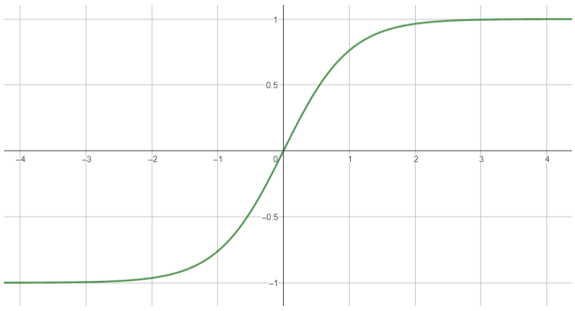
Sigmoid-funktiolla on kuitenkin myös haittoja. Esimerkiksi funktio saturoituu, kun $x \to \infty$ tai $x \to -\infty$, jolloin $\sigma(x) \to 1$ tai $\sigma(x) \to 0$. Näin ollen sigmoidin derivaatta lähestyy nollaa, mikä voi johtaa gradientin katoamiseen. Tämän ongelman lieventämiseksi aineisto yleensä normalisoidaan tietylle välille, jotta aktivointifunktion syöte ei ole liian pieni tai suuri [14].

4.2 Hyperbolinen tangenttifunktio

Matematiikassa hyperbolisilla funktioilla tarkoitetaan tavallisia trigonometrisia funktioita, jotka ovat määritelty hyperboliselle tasolle. Hyperbolinen sini on määritelty $\sinh x = \frac{e^x - e^{-x}}{2}$ ja vastaavasti hyperbolinen kosini $\cosh x = \frac{e^x + e^{-x}}{2}$. Näin ollen hyperbolinen tangentti johdetaan hyperbolisen sinin ja kosinin osamäärästä:

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{\frac{e^x - e^{-x}}{2}}{\frac{e^x + e^{-x}}{2}} = \frac{e^{2x} - 1}{e^{2x} + 1},$$

jossa $x \in \mathbb{R}$ ja $\tanh(x) \in (-1, 1)$. Funktion kuvaajaa havainnollistetaan kuvassa 4. [5]



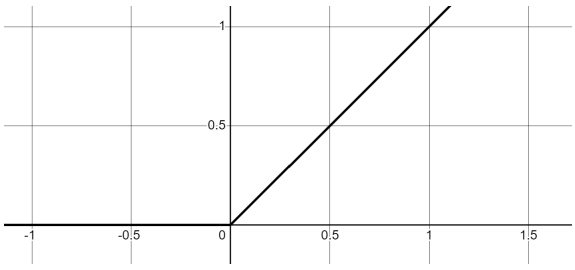
Kuva 4: Hyperbolisen tangenttifunktion kuvaaja

Geometrisesti hyperbolinen tangentti muistuttaa sigmoidia, mikä tarkoittaa, että funktiot jakavat tiettyjä ominaisuuksia keskenään. Ensinnäkin tanh on epälineaarinen, mutta funktion heikkoutena on myös saturoituminen ja gradientin katoaminen, kun x kasvaa tai pienenee merkittävästi.

Tanh on ainoa tämän tutkielman kolmesta aktivointifunktiosta, jonka arvojoukkoon kuuluu myös negatiivisia arvoja. Monesti halutaankin, että aktivointifunktio reagoi sekä positiivisiin, että negatiivisiin syötteisiin. Tanh-funktion symmetrinen ulostulo soveltuu erityisesti sekventiaalisen datan, kuten sanojen tai aikasarjojen, käsittelyyn, joissa seuraava tila riippuu edellisestä. Esimerkiksi tekstinkäsittelyssä tanh voi mallintaa sanojen vahvistavia ja kumoavia merkityksiä, kuten "ei" ja "on".

4.3 ReLU

Rectified Linear Unit (ReLU) on yksi yleisimmin käytetyistä aktivointifunktioista. Sen suotuisat ominaisuudet ovat hyödyllisiä erityisesti syvissä neuroverkoissa. Funktion matemaattinen määritelmä on $\mathrm{ReLU}(x) = \max(0, x)$, jossa $x \in \mathbb{R}$ ja $\mathrm{ReLU}(x) \in [0, \infty)$. [5] Kuvassa 5 havainnollistetaan ReLU-funktion kuvaajaa, joka eroaa geometrisesti aiemmin käsitellyistä funktioista.



Kuva 5: ReLU-funktion kuvaaja

ReLU-funktion selkeän epälineaarisuuden lisäksi se on yksinkertaisuutensa takia laskennallisesti hyvin tehokas. Koska ReLU-funktion derivaatta positiivisilla syötteillä

on 1, sen gradientti ei myöskään häviä suurilla syötteillä samalla tavalla kuin sigmoidin tai hyperbolisen tangenttifunktion gradientti. Tämän takia ReLU on soveltuu erityisen hyvin syville neuroverkoille.

Vaikka ReLU ei saturoidu samalla tavalla kuin sigmoid tai hyperbolinen tangentti, sen käyttöön liittyy muita ongelmia. Jos neuronit saavat jatkuvasti negatiivisia syötteitä, niiden ulostulo jää aina nollaksi. Tämä voi johtaa tilanteeseen, jossa kyseiset neuronit eivät enää osallistu oppimiseen, mikä tunnetaan kuolleiden neuroneiden ongelmana (engl. dead neurons). [5] ReLU-funktio ei myöskään pysty luomaan neuronikerrosten välisiä negatiivisia suhteita positiivisen arvojoukkonsa takia.

ReLU-funktio on myös ainoa tämän tutkielman aktivointifunktioista, joka ei ole jatkuvasti differoituva. Tämä voidaan osoittaa laskemalla ReLU-funktion toispuoleisten derivaattojen raja-arvot pisteessä $x = 0$ [10]. Koska

$$\lim_{x\to 0^-} \frac{max(0,x) - 0}{x - 0} = \lim_{x\to 0^-} \frac{0}{x} = 0$$

ja

$$\lim_{x\to 0^+} \frac{max(0,x) - 0}{x - 0} = \lim_{x\to 0^-} \frac{x}{x} = 1,$$

niin ReLU-funktion erotusosamäärällä ei ole raja-arvoa origossa, joten ReLU ei ole jatkuvasti derivoituva. Kun neuroverkkoa rakennettaan, ReLU-funktion derivaatta voidaan erikseen määritellä nollaksi origossa, jolloin laskennallisesti neuroverkon optimointi ei häiriinny.

5 Neuroverkon rakentaminen ja analysointi

Käytännön sovelluksen tavoitteena on havainnollistaa neuroverkon toimintaperiaatteita sekä tuoda esiin, kuinka eri aktivointifunktiot vaikuttavat neuroverkon suorituskykyyn lämpötilan ennustustehtävässä. Kyseessä on yksinkertainen eteenpäin kytketty neuroverkko, joka soveltuu hyvin demonstraatioon ja tarjoaa selkeän esimerkin neuroverkon toiminnasta. Mallin ennustetarkkuutta arvioidaan käyttämällä kolmea eri aktivointifunktioita, ja tulosten avulla tarkastellaan, miten funktiot vaikuttavat verkon oppimisprosessiin ja ennustetarkkuuteen. Neuroverkko rakennetaan Pythonin TensorFlow-kirjaston avulla.

5.1 Rakenne

Neuroverkko koostuu seitsemän neuronin syötekerroksesta, neljästä piilokerroksesta, joissa jokaisessa on kymmenen neuronia, sekä yhden neuronin lähtökerroksesta. Tällainen rakenne on riittävän yksinkertainen, jotta malli pysyy hallittavana, mutta tarjoaa silti tarpeeksi laskennallista kapasiteettia tehokkaaseen oppimiseen ja lämpötilan ennustamiseen.

Sovelluksessa vertaillaan kolmea erilaista neuroverkon mallia, joilla kullakin on erilaiset aktivointifunktiot syöte- ja piilokerroksissa. Ensimmäisessä mallissa aktivointifunktioina käytetään ReLU-funktiota, toisessa sigmoid-funktiota ja kolmannessa hyperbolista tangenttia. Jokaisessa mallissa lähtökerroksen aktivointifunktio on kuitenkin lineaarinen, $f(x) = x$, jotta neuroverkkojen ulostulot voivat olla jatkuvia. Lineaarinen aktivointifunktio lähtökerroksessa on myös laskennallisesti yksinkertainen ratkaisu ja soveltuu hyvin regressiotehtäviin.

5.2 Aineisto ja koulutus

Aineistona käytetään New Yorkin vuoden 2019 säätietoja [9]. Aineistossa on siis jokaisen päivän havainnot, eli niiden lukumäärä on 365. Muuttujat ovat päivämäärä, päivän suurin lämpötila, päivän matalin lämpötila, päivän keskilämpötila, lämmitystarveluku, jäähdytystarveluku, sademäärä, sataneen lumen määrä ja lumen määrä tuumissa. Aluksi aineistosta poistetaan lumeen liittyvät muuttujat tehtävän yksinkertaistamiseksi ja erotellaan päivämäärät kuukausiin ja viikonpäiviin, jotta neuroverkko voi oppia niistä monimutkaisempia suhteita. Lisäksi sademääräsarakkeen sisältämät T-arvot, jotka tarkoittavat hyvin pientä sademäärää (engl. trace), on muutettu arvoksi 0,001. Tällöin sademääräsarakkaeen arvot ovat numeerisia, mutta kuitenkin vaikuttavat mallissa. Tämän jälkeen aineisto jaetaan koulutus- ja validaatioaineistoihin 80% ja 20% painotuksella. Lisäksi aineisto sovitetaan välille $[0, 1]$, jotta erot muuttujien välisessä painotuksessa olisivat tasaisemmat. Sovitettuja arvoja ei palauteta takaisin alkuperäisiin arvoihin, koska esimerkin tarkoituksena on havainnollistaa aktivointifunktion eroavaisuuksia, eikä neuroverkon soveltuvuutta käytäntöön.

Neuroverkko optimoidaan takaisinsyöttöalgoritmilla ja tappiofunktiona käytetään keskineliövirhettä, joka soveltuu hyvin regressiotehtäviin. Koulutus kestää 50 epook-

kia, joista jokainen vastaa yhtä koulutusaineiston täydellistä kierrosta. Keskineliövirheen lisäksi koulutuksessa seurataan absoluuttista keskivirhettä (engl. Mean Absolute Error, lyh. MAE), joka lasketaan seuraavasti:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|.$$

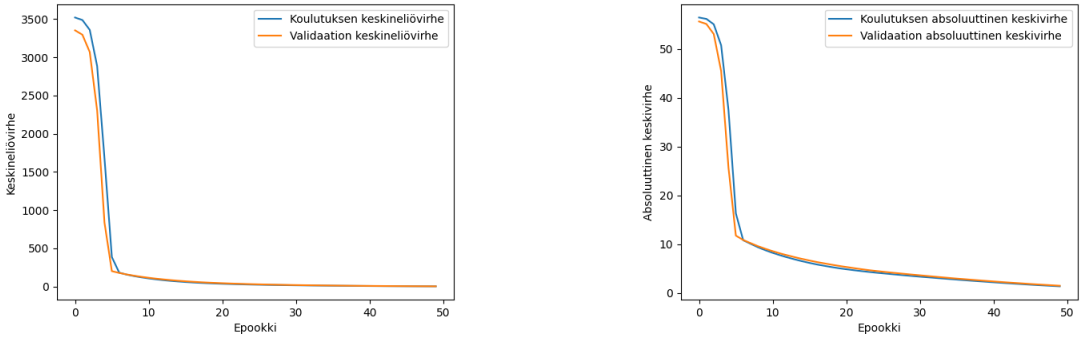
Absoluuttinen keskivirhe on intuitiivisesti helpompi ymmärtää kuin keskineliövirhe, joka havainnollistaa hyvin, kuinka kaukana ennuste on oikeasta vastauksesta.

5.3 Mallien vertailu

Vertailuparametrina käytetään validaatioaineiston ennustetarkkuutta eli keskineliövirhettä (MSE) ja absoluuttista keskivirhettä (MAE). Näiden mittareiden avulla voidaan arvioida kunkin mallin ennustustarkkuutta ja suhteellista virhettä validaatioaineistossa. Lisäksi vertailussa otetaan huomioon oppimisnopeuden erot mallien välillä.

5.3.1 ReLU-malli

ReLU-aktivointifunktiota käyttävä malli menestyi erinomaisesti sovitetulla aineistolla. Validaatioaineistolla mallin absoluuttinen keskivirhe oli 1,4524 ja keskineliövirhe 3,0284. [7] Kuvasta 6 nähdään, kuinka mallin virheet pienenevät nopeasti jo noin kuuden epookin kohdalla, mikä osoittaa, että malli oppii tehokkaasti ilman merkittävää ylioppimista. Lisäksi absoluuttisen keskivirheen arvo on lähellä keskineliövirheen neliöjuurta, mikä viittaa siihen, että malli ei tee suuria ennustusvirheitä validaatioaineistossa. Tämä johtuu siitä, että MSE korostaa suuria virheitä voimakkaammin kuin MAE, joka painottaa kaikkia virheitä tasaisesti.



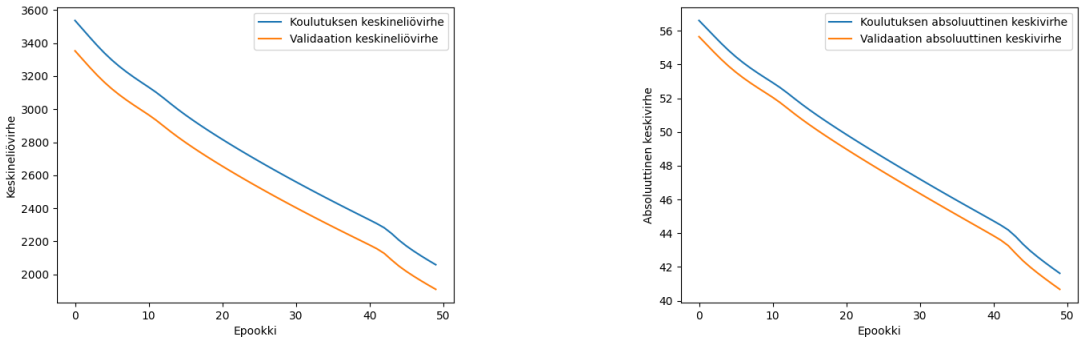
(a) MSE (b) MAE

Kuva 6: ReLU-mallin virheet suhteessa epookkien määrään [7]

Nämä tulokset korostavat ReLU-funktion laskennallista tehokkuutta ja sen kykyä säilyttää gradientti suurillakin syötearvoilla. Lisäksi regressiotehtävään sovitettu eteenpäinkytketty neuroverkko vaikuttaa olevan erityisen toimiva yhdistelmä ReLU-aktivointifunktion kanssa.

5.3.2 Sigmoid-malli

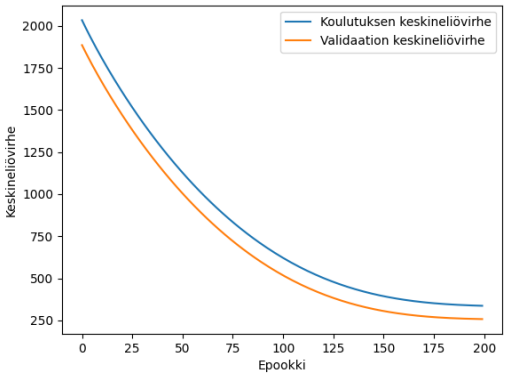
Sigmoid-funktiota käyttävä malli ei menestynyt hyvin sovitetulla aineistolla. Validaatioaineistolla mallin absoluuttinen keskivirhe oli 40,6728 ja keskineliövirhe oli 1909,9554, mikä viittaa suuriin ennustusvirheisiin. [7] Erityisesti kuvasta 7 nähdään, että oppimisprosessi etenee lähes lineaarisesti epookkien määrän lisääntyessä, mikä poikkeaa merkittävästi ReLU-mallin lähes eksponentiaalisesta oppimisnopeudesta (kuva 6).



(a) MSE (b) MAE

Kuva 7: Sigmoidimallin virheet suhteessa epookkien määrään [7]

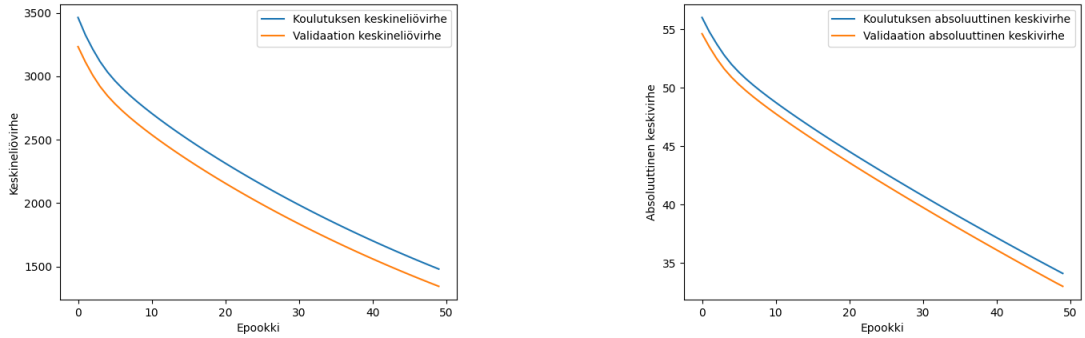
Virheiden pienentämiseksi koulutusta jatkettiin 200 epookkiin, jolloin absoluuttinen keskivirhe pieneni arvoon 13,7019 ja keskineliövirhe arvoon 257,9259 [7]. Kuvasta 8 nähdään, että validaatioaineiston keskineliövirhe lähestyy arvoa noin 250. Tässä vaiheessa oppiminen hidastuu huomattavasti, mikä saattaa johtua sigmoid-funktion luontaisesta saturoitumisesta. Vaikka virheet pienenivät merkittävästi suuremmalla epookkien määrällä, sigmoidifunktion käyttö ei vaikuta tehokkaalta ratkaisulta tässä tehtävässä.



Kuva 8: 200:n epookin sigmoidimallin keskineliövirhe [7]

5.3.3 Hyperbolinen tangentti

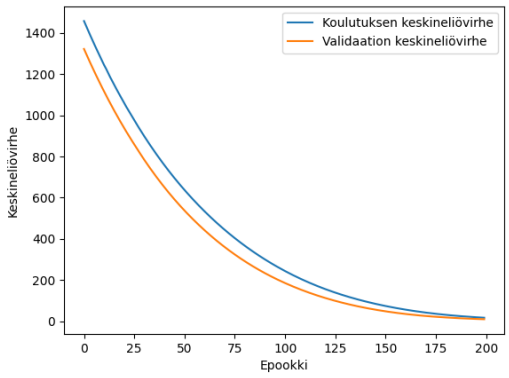
Hyperbolista tangenttifunktiota käyttävä malli saavutti aluksi absoluuttiseksi keskivirheeksi 33,0145 ja keskineliövirheeksi 1344,6893, mikä osoittaa kohtalaista suorituskykyä sovitetulla aineistolla [7]. Kuvasta 9 voidaan havaita, että mallin oppimisprosessi näyttää lineaariselta, mikä viittaa tasaisempaan oppimisvauhtiin ilman äkillisiä muutoksia.



(a) MSE (b) MAE

Kuva 9: Tanh-mallin virheet suhteessa epookkien lukumäärään [7]

Koulutusta jatkettiin 200 epookkiin suurehkojen aloitusvirheiden vuoksi. Tämä pienensi absoluuttista keskivirhettä arvoon 1,3735 ja keskineliövirhettä arvoon 9,8383 [7]. Kuvasta 10 nähdään, että virheet lähestyvät nollaa epookkien määrän kasvaessa, mikä viittaa siihen, että malli oppii jatkuvasti lisää aineistosta.



Kuva 10: 200:n epookin tanh-mallin keskineliövirhe [7]

Hyperbolista tangenttifunktiota käyttävä malli osoittautui varsin hyväksi vaihtoehdoksi lämpötilan ennustamiseen. On kuitenkin huomionarvoista, että absoluuttinen keskivirhe on merkittävästi pienempi kuin keskineliövirheen neliöjuuri. Tämä viittaa siihen, että malli tekee joitakin suuria yksittäisiä ennustusvirheitä, joita MSE korostaa enemmän kuin MAE. Tämä voi johtua esimerkiksi aineiston poikkeavista havainnoista tai siitä, että malli on ylisovittunut koulutusaineistoon.

5.3.4 Yhteenveto

Tulosten perusteella ReLU-malli oli tässä tehtävässä tehokkain vaihtoehto. Se saavutti huomattavasti pienemmän absoluuttisen keskivirheen vain kymmenellä epookilla verrattuna sigmoid-malliin, joka tarvitsi 200 epookkia päästäkseen vastaavaan tarkkuuteen. Tämä ero johtuu erityisesti ReLU tehokkaasta gradienttilaskennasta, joka ei kärsi saturoitumisesta, toisin kuin sigmoidifunktio.

Hyperbolinen tangenttifunktio osoittautui myös kohtuullisen hyväksi vaihtoehdoksi, mutta sen ennustetarkkuus oli hieman ReLU-funktiota huonompi. Tanh-malli oli kuitenkin huomattavasti tarkempi kuin sigmoid-malli, mikä selittyy osittain tanh-funktion laajemmalla arvojoukolla $(-1, 1)$ ja symmetrisyydellä nollan suhteen [5]. Laajemman arvojoukkonsa avulla tanh-mallin on mahdollista oppia aineiston piirteitä monipuolisemmin kuin sigmoid-malli, joka on rajoitettu välille $(0, 1)$.

Tämä vertailu havainnollistaa, kuinka tärkeää on ymmärtää tehtävän luonne ja valita oikea aktivointifunktio sen mukaan. Eteenpäinkytketyllä neuroverkolla regressiotehtävissä ReLU-funktio osoittautui selkeästi parhaaksi kolmen vaihtoehdon joukosta. Kuitenkin toisissa tehtävissä, kuten kuvantunnistuksessa tai tekstinkäsittelyssä, paras aktivointifunktio voisi olla erilainen riippuen tehtävän vaatimuksista ja aineiston rakenteesta.

6 Johtopäätökset

Tässä tutkielmassa tarkasteltiin sigmoid-funktion, hyperbolisen tangenttifunktion ja ReLU-funktion ominaisuuksia sekä niiden soveltuvuutta lämpötilan ennustamistehtävään neuroverkon avulla.

Geometrisesti sigmoid- ja tanh-funktiot ovat samankaltaisia, mikä näkyy myös niiden ominaisuuksissa, kuten gradientin katoamisessa. ReLU-funktio sen sijaan erottuu geometrisesti näistä kahdesta: se säilyttää gradientin suuremmilla syötearvoilla, mikä mahdollistaa tehokkaamman oppimisen ilman samanlaista saturoitumisongelmaa. ReLU-funktiolla on kuitenkin omat haasteensa, kuten kuolleiden neuronien ongelma, mikä ei välttämättä näin yksinkertaisessa neuroverkossa vielä aiheuta ongelmia.

Lämpötilan ennustamistehtävässä ReLU-funktio osoittautui ylivertaiseksi. Sen tarkat ennusteet saavutettiin jo lyhyellä koulutusajalla, mikä tekee siitä tehokkaan vaihtoehdon tämän tyyppisissä regressiotehtävissä. Tanh-funktio tarvitsi pidemmän koulutusajan, mutta suoriutui hyvin myöhemmissä vaiheissa, vaikka ylisovittamisen riski kasvoi pidemmän koulutuksen aikana. Sigmoid-funktion suorituskyky jäi merkittävästi alle tanh- ja ReLU-funktioiden tasosta huolimatta epookkien määrän kasvattamisesta.

Esimerkin tarkoituksena ei kuitenkaan ollut löytää parasta aktivointifunktiota tietyssä tehtävässä, vaan havainnollistaa, kuinka merkittävästi aktivointifunktion valinta voi vaikuttaa neuroverkon suorituskykyyn jopa yksinkertaisessa verkossa.

Viitteet

[1] Nefi Alarcon: *OpenAI Presents GPT-3, a 175 Billion Parameters Language Model*. NVIDIA Developer, https://developer.nvidia.com/blog/openai-presents-gpt-3-a-175-billion-parameters-language-model/, 13.11.2024.

[2] Ethem Alpaydin: *Machine Learing, Revised and Updated Edition*. The MIT Press, Cambridge, Massachusetts, 2021.

[3] Dave Bergmann, Cole Stryker: *What is a loss function?*. IMB, https://www.ibm.com/think/topics/loss-function, 13.11.2024.

[4] Dave Bergmann, Cole Stryker: *What is backpropagation?*. IBM, https://www.ibm.com/think/topics/backpropagation, 13.11.2024.

[5] Bin Ding, Huimin Qian, Jun Zhou: *Activation functions and their characteristics in deep neural networks*. IEEE, Shenyang, China, 2018.

[6] Sanjay Dutta: *Understanding the Number of Hidden Layers in Neural Networks: A Comprehensive Guide*. Medium, https://medium.com/@sanjay_dutta/understanding-the-number-of-hidden-layers-in-neural-networks-a\protect\penalty\z@-comprehensive-guide-0c3bc8a5dc5d, 13.11.2024.

[7] Esimerkkityön Python-komennot. Github, https://github.com/leinoaar/new_york_weather_RNN, 14.11.2024.

[8] Mohit Gupta: *Introduction to Data in Machine Learning*. GeeksforGeeks, 13.11.2024.

[9] New York City Weather Data 2019. Kaggle, https://www.kaggle.com/datasets/alejopaullier/new-york-city-weather-data-2019?resource=download, 6.11.2024.

[10] Petteri Harjulehto, Riku Klén, Mika Koskenoja: *Analyysiä reaaliluvuilla*. Unigrafia, Helsinki, 2022.

[11] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani: *An Introduction to Statistical Learning: with Applications in R, Second Edition*. Springer, 2021.

[12] Eric R. Kandel, John D. Koester, Sarah H. Mack, Steven A. Siegelbaum: *Principles of Neural Science, Sixth Edition* . McGraw-Hill Education, New York, NY, 2021.

[13] Khan Academy, Gradient descent,

https://www.khanacademy.org/math/multivariable-calculus/applications-of-multivariable-derivatives/optimizing-multivariable-functions/a/what-is-gradient-descent, 13.11.2024.

[14] Wei Hao Khoong: *Why Scaling Your Data Is Important*. Medium, `https://medium.com/codex/why-scaling-your-data-is-important-1aff95ca97a2`, 13.11.2024.

[15] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton: *ImageNet Classification with Deep Convolutional Neural Networks*. Communications of the ACM, 2012.

[16] Jakub Kufel, Katarzyna Bargieł-Łączek, Szymon Kocot, Maciej Koźlik, Wiktoria Bartnikowska, Michał Janik, Łukasz Czogalik, Piotr Dudek, Mikołaj Magiera, Anna Lis, Iga Paszkiewicz, Zbigniew Nawrat, Maciej Cebula, Katarzyna Gruszczyńska: *What Is Machine Learning, Artificial Neural Networks and Deep Learning?—Examples of Practical Applications in Medicine*. MDPI, 2023.

[17] Jyrki Lahkonen: *Vektorilaskenta*. Turun yliopisto, 2023.

[18] Josh Patterson, Adam Gibson: *Deep learning: a practitioner's approach*. O'Reilly Media, 2017.

[19] Khalid Saeed, Władysław Homenda: *Computer Information Systems and Industrial Management: 15th IFIP TC8 International Conference, CISIM 2016 Vilnius, Lithuania, September 14–16, 2016 Proceedings*. Springer, Vilna, Liettua, 2016.

[20] Anthony Zachnich: *Neural Networks for Intelligent Signal Processing*. World Scientific Publishing Company, 2003.