LAPPEENRANNAN-LAHDEN TEKNILLINEN YLIOPISTO
School of Engineering Science
Laskennallisen tekniikan koulutusohjelma
Kandidaatintyö

Aleksi Surakka

Neuroverkko-oppimistyöpajan toteutus LUT Junior Universityä varten

Ohjaaja: Tutkijaopettaja Arto Kaarna

TIIVISTELMÄ

Lappeenrannan-Lahden teknillinen yliopisto

School of Engineering Science

Laskennallisen tekniikan koulutusohjelma

Aleksi Surakka

Neuroverkko-oppimistyöpajan toteutus LUT Junior Universityä varten

Kandidaatintyö

2020

26 sivua, 8 kuvaa, 2 taulukkoa

Ohjaaja: Tutkijaopettaja Arto Kaarna

Avainsanat: Neuroverkko; Koneoppiminen; Python; LUT Junior University;

Kandidaatintyössä toteutettiin LUT Junior Universityä varten neuroverkon toimintaa visualisoiva sovellus. Työpajan kohderyhmänä on 8-luokkalaiset oppilaat. Sovelluksessa luotu neuroverkko jätettiin tarkoituksen mukaisesti hyvin yksinkertaiseksi, jotta sen hahmottaminen ja toiminta koituisi 8-luokkalaisille mahdollisimman helpoksi. Luodussa sovelluksessa oppilaat luokittelevat vaatekappaleita kolmen syötteet avulla, jotka kuvaavat vaatekappaleen ominaisuuksia, ja säätävät neuroverkon painokertoimia, jotta luokittelu olisi todenmukainen. Syötteiden antamisen jälkeen neuroverkko ajetaan syötteillä ja määritetyillä painokertoimilla, jonka jälkeen sovellus antaa painokertoimille palautetta. Palautteen avulla oppilas saa suuntaa, miten painokerrointa tulisi säätää jotta luokitteluvirhe saataisiin minimoitua. Työpajan sovellus toteutettiin Python ohjelmointikielellä hyödyntäen numeerista taulukkolaskentakirjastoa sekä graafista käyttöliittymäkirjastoa.

Työpajan kokonaisuutta testattiin yliopisto-oppilaista koostuvasta testiryhmästä etäyhteyden avulla. Testiryhmän palautteen perusteella luotu sovellus koettiin hyvin visualisoivan neuroverkon toimintaa ja jatkossa työpajaa järjestetään 8-luokkalaisille, joilta tullaan keräämään sovelluksen vaatimaa palautetta.

Sisältö

Symboli- ja lyhenneluettelo

b	Vakiotermi (Bias)
u	Syötteiden ja painokertoimien summa
v	Neuronin summa
$w = [w_1, w_2, \ldots, w_i]$	Painokerroinvektori
$x = [x_1, x_2, \ldots, x_i]$	Syötevektori
y	Neuronin ulostulo
∇E	Virhefunktion gradientti
$\varphi(\cdot)$	Aktivaatiofunktio
$\varphi(v)$	Sigmoid-funktio
$\varphi'(v)$	Sigmoid-prime
LUT-yliopisto	Lappeenrannan-Lahden teknillinen yliopisto LUT

1 JOHDANTO

1.1 Tausta

LUT Junior University - Lappeenranta on LUT-yliopiston ja Lappeenrannan koulujen välinen koulutusyhteistyö. Toiminta käynnistyi lukuvuoden 2018-2019 aikana, mistä lähtien se on toiminut koulutusyhteistyössä lasten ja nuorten parissa [1]. Junior University toteuttaa oppimista sekä toimintaa esikoululaisille, kolmas-, viides- ja kahdeksasluokkalaisille sekä lukiolaisille. Junior Universityn koordinaattori Kati Koikkalainen kertoo Lappeenrannan Uutisissa "Yhtenä tarkoituksena Junior Universitylla on lisätä kiinnostusta matematiikkaa kohtaan, näyttämällä mihin se taipuu arkipäivässä" [2].

Oppiminen on yksi tärkeimmistä tekijöistä, joka tekee tietoteknisestäjärjestelmästä älykkään. Tietokoneiden toiminnat on hyvin rajoittunut siihen, mihin tarkoitukseen ne on ohjelmoitu. Tietokoneiden hyödyllisyyttä voidaan tehostaa koneoppimisella. Koneoppimisella tarkoitetaan uusien tehtävien tekemistä ilman tehtävään vaativaa ohjelmointia. Koneoppimisen osa-alueet voidaan jakaa kolmeen eri kategoriaan, jotka ovat ohjattu oppiminen, ohjaamaton oppiminen ja vahvistusoppiminen. Ohjattua oppimista käytetään esimerkiksi luokittelutehtävissä.

Neuroverkot ovat koneoppimisen osa-alue. Neuroverkkojen yleistymisen on saanut aikaiseksi laskentatehon kehittyminen, niin prosessoreissa kuin näytönohjaimissa. Ohjatussa oppimisessa neuroverkkoja voidaan opettaa valmiilla aineistolla ja neuroverkko voi sopeutua uusiin tuntemattomiin ympäristöihin muokkaamalla synapsisia painoarvojaan tuottaen tarpeellisen toiminnon matkien aivojen toimintaa. Neuroverkot ovat parhaiten sopeutuneet hahmontunnistukseen ja ensimmäiset mallit kehitettiin jo vuonna 1954 Marvin Minskyn väitöskirjassa [3].

LUT-yliopiston koulutusohjelmat järjestävät koulutusohjelmaansa liittyviä työpajoja kahdeksasluokkalaisille. Laskennallisen tekniikan ja analytiikan koulutusohjelman työpajaa haluttiin laajentaa entisestään luomalla uusi koneoppimiseen liittyvä oppimisympäristö neuroverkoille, jota työpajaan osallistuvat oppilaat pääsevät käyttämään. Työpajassa koululaiset pääsevät tutustumaan neuroverkon toimintaan yksinkertaisen luokitteluongelman avulla.

1.2 Tavoitteet ja rajaus

Työn tavoitteena oli suunnitella ja toteuttaa laajennus LUT Junior Universityn laskennallisen tekniikan ja analytiikan koulutusohjelman työpajaa varten. Työpajan kohderyhmänä toimii kahdeksasluokkalaiset oppilaat. Työpajassa keskeinen tavoite on lisätä koululaisten kiinnostusta matematiikkaan, sovelluskehittämiseen ja laskennallisiin menetelmiin sekä näyttää, kuinka näitä asioita voidaan hyödyntää käytännössä esimerkiksi koneoppimisessa. Työpajaa päätettiin laajentaa tekemällä sovellus, jossa perehdytään neuroverkon toimintaan.

Työpajaa laajentava sovellus päätettiin toteuttaa tekemällä neuroverkkojen avulla oleva yksinkertainen luokittelusovellus. Sovelluksen tavoitteena oli esitellä neuroverkkojen sisäinen toiminta tavalla, jota kahdeksasluokkalainen voisi ymmärtää. Sovelluksessa oletettiin neuroverkkojen olevan tuntematon käsite kahdeksasluokkalaisille, joten luokittelutehtävä rajattiin vain kolmeen erilliseen luokkaan ja kolmeen eri syötteeseen. Sovelluksessa luokittelu täytyi olla yksinkertainen, jonka kahdeksasluokkalaiset opiskelijat osaisivat tehdä. Luokkiin kuuluvat ominaisuudet, eli syötteet, pitivät myös olla oppilaille helposti hahmotettavissa. Luokittelu päätettiin tehdä vaatekappaleista, koska se olisi läheisesti elämään liittyvä aihe kahdeksasluokkalaisille.

Sovellus toteutettiin Python-ympäristössä hyödyntäen ilmaisia laskenta- sekä käyttöliittymäkirjastoja. Sovelluksessa käyttäjä antaa luokitteluun vaatekappaletta vastaavat ominaisuudet syötteenä ja säätää neuroverkon painoarvoja, jotta ulostulona olisi ominaisuuksia vastaava vaateluokka. Syötteet rajattiin olevan binäärijärjestelmän mukaisia eli nolla tarkoittaa ominaisuuden puuttumista ja yksi tarkoittaa sisältymistä.

1.3 Raportin rakenne

Tämän kandidaatintyön raportti sisältää kappaleessa 2 yleistä tietoa koneoppimisesta ja sen osa-alueesta neuroverkoista sekä menetelmistä, mitä hyödynnettiin toteutuksessa. Kappale 3 sisältää katsauksen nykyiseen työpajaan ja sen toimintaan sekä myös uuden työpajan kokonaisuuden. Samassa kappaleessa käydään läpi myös kehitetty sovellus ja sen käyttö sekä sovelluksen testaus ja testiryhmän kokemukset. Lopuksi tuodaan esille sovelluksen toimivuus ja erityiset huomiot työpajan toimivuudessa kappaleessa 4.

2 KONEOPPIMINEN JA NEUROVERKOT

2.1 Koneoppiminen

Tietokoneet toimivat lukemalla niille kirjoitettua koodia ja toteuttavat näin lukuisia eri algoritmeja. Tietokoneiden käytös on täysin riippunut siitä, mihin tarkoitukseen tietokone on ohjelmoitu. Ongelmana tietokoneissa, verrattuna ihmisiin, on niiden puute oppia uutta. Tietokoneiden käyttötarkoitus ja hyödyllisyys on merkittävästi suurempi, jos tietokoneet oppivat kokemuksen kautta parantaen toimintojaan. Koneoppiminen tarkoittaa uuden tiedon oppimista tietokoneessa, joka voi olla tietokoneen itse tekemää tehtävän optimointia tai uuden tehtävän toteuttamista ilman käsin tehtyä ohjelmointia.

Koneoppiminen jaetaan useaan eri osa-alueeseen, joista jokainen soveltuu eri tehtäville. Osa-alueet jaetaan ohjattuun oppimiseen, ohjaamattomaan oppimiseen ja vahvistusoppimiseen [4]. Yhdistämällä ohjattua ja ohjaamatonta oppimista voidaan myös muodostaa puoliohjatun oppimisen osa-alue. Ohjatulla oppimisella tarkoitetaan aineistosta oppimista, jolloin kone opetetaan opetusaineistolla erillisessä opetusvaiheessa. Opetuksen jälkeen koneen suorituskykyä testataan uudella aineistolla, joka sisältää vain syötteet. Ohjattua oppimista voidaan hyödyntää regressioanalyysissä ja luokittelussa.

Toinen osa-alue on ohjaamaton oppiminen, jossa aineistolla ei ole luokkatunnuksia. Ohjaamaton oppiminen ei hyödynnä ohjatun oppimisen palautejärjestelmää, jolloin virhettä ei voida määritellä. Ohjaamattoman oppimisen tavoitteena on löytää aineistosta yhtäläisyyksiä, joiden perusteella aineisto voitaisiin ryhmitellä löydetyn mallin perusteella. Sen käyttökohteita on dimensioiden vähentäminen ja klusterointi.

Puoliohjattu oppiminen on ohjatun ja ohjaamattoman oppimisen välivaihe. Sitä opetetaan niin luokitetulla kuin luokittamattomalla aineistolla. Puoliohjatussa oppimisessa aineiston jakautuminen täytyy olla tunnettu, jotta luokittamatonta aineistoa voidaan tukea luokitellun aineiston kanssa. Puoliohjatun oppimisen tavoitteena on saada parempi luokittelutarkkuus kuin pelkästään hyödyntämällä luokiteltua aineistoa.

Viimeinen esitettävä osa-alue on vahvistusoppiminen. Vahvistusoppimisessa kone oppii virheistään ja pyrkii parhaaseen suorituskykyyn. Suorituskyvyn mittaamiseen tulee kuitenkin määritellä agentille palautejärjestelmä, jossa agentti palkitaan hyvästä toiminnasta, mutta myös rangaistaan huonosta toiminnasta ympäristössään. Näiden avulla agentti opetetaan suosimaan hyviä toimintoja ja ajan kanssa agentti oppii välttämään virheitä. Vahvistusoppimista

voidaan hyödyntää esimerkiksi peleissä [5] ja tietokoneen resurssien hallintoon [6].

Nilsson N. esitteli kirjassa Learning Machines koneoppimisen alkeellista tutkimusta ja koneoppimisen ensimmäisiä tehtäviä eli hahmontunnistusta [7]. Kirjassa esiteltiin, että hahmontunnistus on onnistunut tiedon luokittelun tai järjestelyn avulla. Yksi esimerkki oli sään ennustaminen, jossa hyödynnettiin lukuisten sääasemien antamia tietoja ilmanpaineesta ja ilmanpaineen muutoksista. Tietojen perusteella pystyttiin ennustamaan sateen todennäköisyyttä tulevalta päivältä luokittelemalla mitatut tiedot kahteen kategoriaan, joista ensimmäinen osoitti sadetta ja toinen kumosi sateen. Ennustusta verrattiin todelliseen tulokseen ja ennustuksen haluttiin olevan todellisen tuloksen mukainen.

Nilsson esitteli myös monia muita luokitteluun liittyviä koneoppimisen tehtäviä. Luokitteluongelmissa yhteistä oli syötettävä tieto, jonka perusteella voitiin tilastollisen tiedon avulla luoda vastaus, jonka luokittelija palautti. Taulukossa 1 on esitetty erilaisia luokittelutehtäviä ja niiden tarvitsemia syötteitä sekä palautettuja tuloksia.

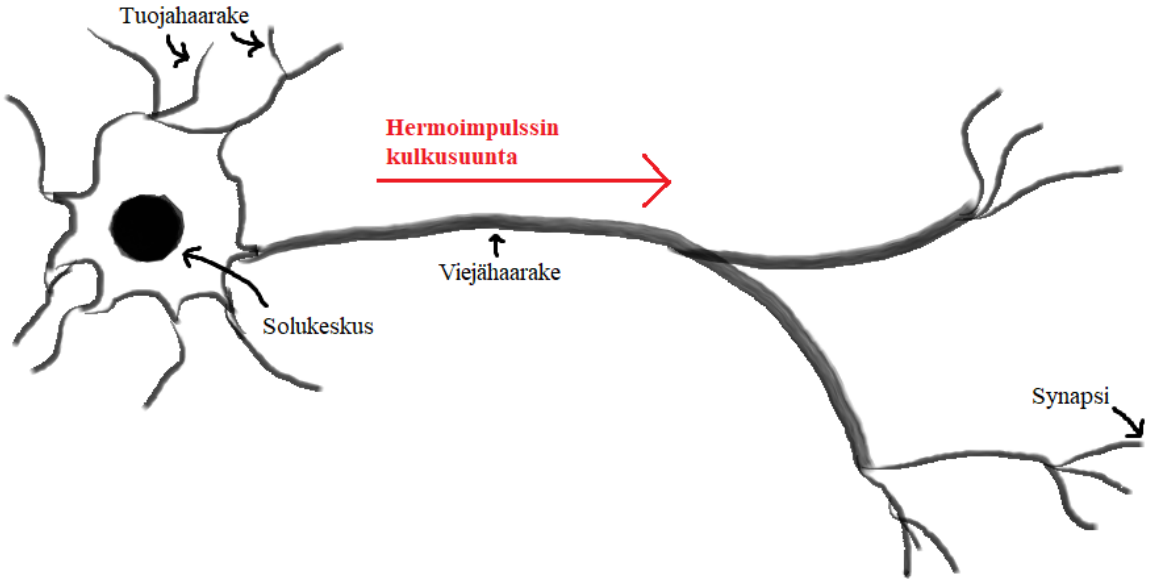
Taulukko 1. Esimerkkejä luokitteluongelmista [7].

Tehtävä	*Syöte*	*Vastaus*
Sään ennustaminen	Säämittaukset	Sääennuste
Käsin kirjoitetun merkin tunnistus	Optinen signaali	Merkki
Lääketieteellinen diagnoosi	Oireet	Sairaus
Puheentunnistus	Akustinen aaltomuoto	Puhe
Puhujatunnistus	Akustinen aaltomuoto	Puhujan nimi
Valokuvan luokittelu	Digitaalinen kuva	Valokuva

2.2 Neuroverkot

Aivot ovat ihmisen hermostollisen toiminnan keskus. Aivot muodostuvat hermokudoksesta. Hermokudokset sisältävät hermosoluja eli neuroneita, joissa tietojenkäsittely tapahtuu. Neuroneita yhdistää hermoliitokset eli synapsit, joiden kautta hermoimpulssit voivat kulkea. Neuronit voivat kuolla ja uusia synapsisia liitoksia voi syntyä. Aivot voivat näin kehittyä ja sopeutua uusiin ympäristöihin. Neuroverkkojen ideana on matkia aivojen oppimiskykyä. Kuvassa 1 on esitetty hermosolun rakenne yksinkertaisesti.

Tietokoneissa keinotekoisetneuroverkot on pohjimmiltaan suunniteltu matkimaan aivojen ta-



Kuva 1. Hermosolun eli neuronin rakenne [8].

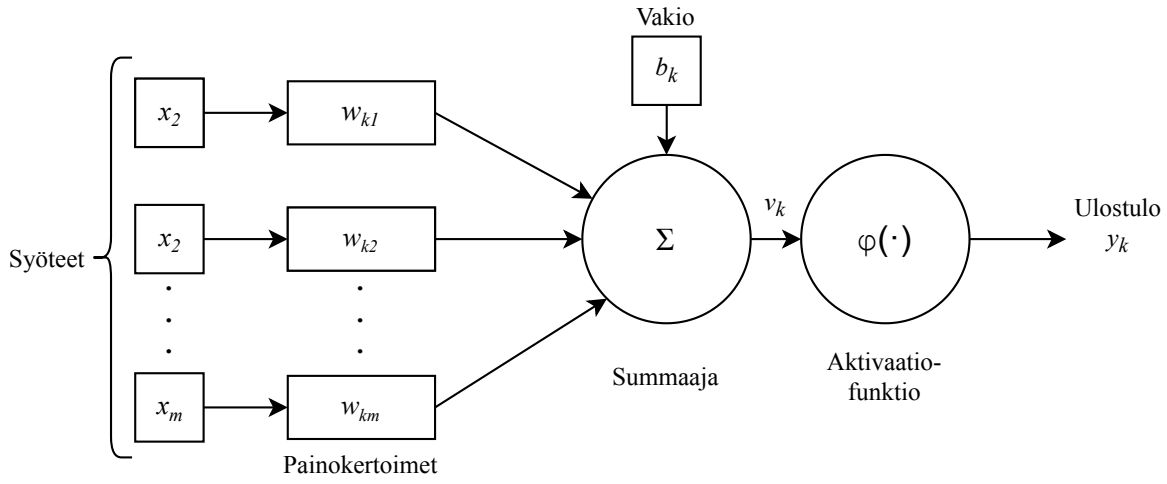
paa ratkaista tietty tehtävä tai tehtävään liittyvä toiminto [9]. Tietokoneiden lineaarinen laskenta on kallista ajalle ja muistille. Neuroverkot saavuttavat hyvän suorituskyvyn muodostumalla suurista määristä yksinkertaisista käsittely-yksiköistä, eli neuroneista, jotka ovat yhdistyneet synapsiliitosten avulla. Synapsiliitokset sisältävät myös oman uniikin kytkentäkohtaisen painokertoimen (englannista synaptic weight), joka toteuttaa oppimisen. Painokertoimien arvot muuttuvat, kun neuroverkko oppii, matkien aivojen oppimista.

Neuroni on tärkeimmistä neuroverkon tiedonkäsittelyn lähteistä. Kuva 2 esittää neuroverkon neuronia. Neuroni muodostuu kolmesta elementistä, joita ovat:

1. Synapsi, joilla jokaisella on oma painokerroin. Synapsissa syötteen arvo x_m kerrotaan painokertoimella w_{km}.
2. Summaaja, joka summaa syötteet keskenään. Summaan lisätään myös vakiotermi b_k, joka muuttaa summan arvoa, joko vähentämällä tai lisäämällä riippuen summan olevan positiivinen tai negatiivinen.
3. Aktivaatiofunktio $\varphi(\cdot)$, joka laskee summan epälineaarisen kuvauksen. Aktivaatiofunktio rajoittaa ulostulon amplitudia tiettyyn rajalliseen arvoon.

Kuvan 2 neuronin k ulostulo saadaan

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{1}$$



Kuva 2. Neuroni neuroverkossa. [9]

ja

$$y_k = \varphi(u_k + b_k) \tag{2}$$

missä $x_1, x_2, \ldots, x_m$ ovat syötteitä, $w_{k1}, w_{k2}, \ldots, w_{km}$ ovat neuronin k synapsiset painokertoimet, u_k on syötteiden lineaarisen yhdistelmän ulostulo, b_k on vakiotermi (bias), $\varphi(\cdot)$ on aktivaatiofunktio ja y_k on neuronin ulostulo. Vakiotermin b_k käyttö luo affiinikuvauksen ulostulolle u_k eli

$$v_k = u_k + b_k \tag{3}$$

Vakiotermi b_k on neuronin k ulkoinen parametri. Vakiotermi nähdään ulkoisena tekijänä lausekkeessa 2. Vakiotermi voidaan kuitenkin ottaa omaksi syötteeksi yhdistämällä lausekkeet 1, 2 ja 3 seuraavasti:

$$v_k = \sum_{j=0}^{m} w_{kj} x_j \tag{4}$$

ja

$$y_k = \varphi(v_k) \tag{5}$$

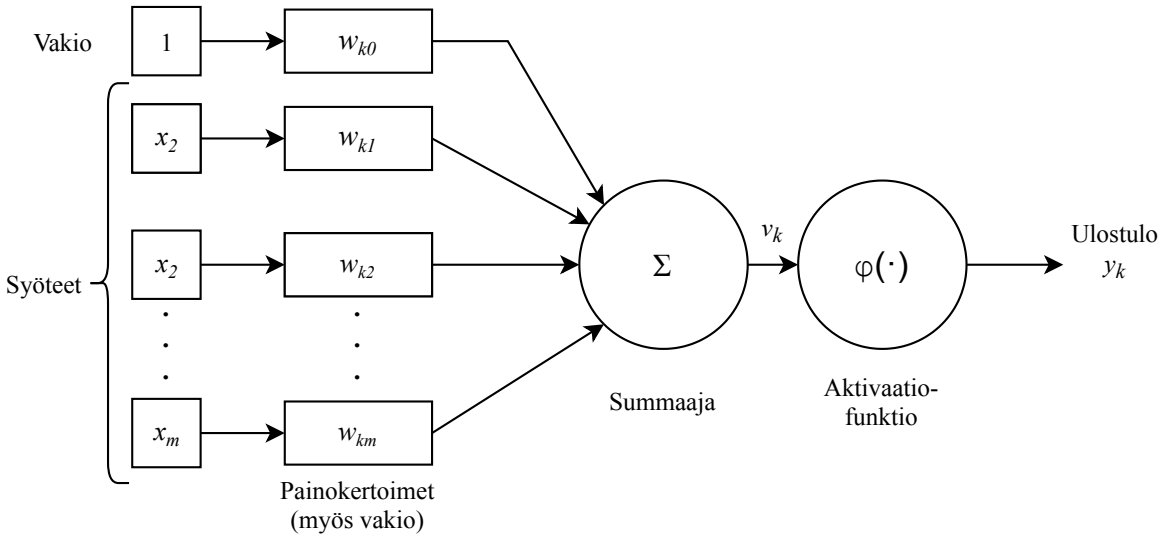
Lausekkeen (4) uusi synapsi saa nyt syötteeksi

$$x_0 = +1 \tag{6}$$

ja painokertoimeksi

$$w_{k0} = b_k \tag{7}$$

Hyödyntäen edellisiä kaavoja, voidaan neuroni k kuvata kuvan 3 tavalla. Kuvassa 3 nähdään, että vakiotermin vaikutus on otettu huomioon luomalla uusi syöte x_0, joka saa arvoksi +1 sekä lisäämällä uusi synapsinen painokerroin w_{k0}, joka saa arvoksi b_k.



Kuva 3. Vaihtoehtoinen neuroni neuroverkossa. [9]

Aktivaatiofunktio $\varphi(v)$ määrittää neuronin ulostulon. Yksi yleisimmistä aktivaatiofunktioista on sigmoid-funktio, koska se sopii malleihin, joissa ulostulona on ennuste [10]. Sigmoid-funktio normalisoi neuronin summan. Sigmoid-funktio on epälineaarinen aktivaatiofunktio. Epälineaariset funktiot voivat mallintaa vastemuuttujia, mitkä vaihtelevat epälineaarisesti selitettävillä muuttujillaan. Ilman epälineaaristen aktivaatiofunktioiden käyttöä, monikerroksisetkin neuroverkot käyttäytyisivät kuin yksinkertaiset perceptron-neuronit, koska monikerroksien neuronien summaus tuottaisi vain lineaarisen funktion. Sigmoid-funktio on muotoa

$$\varphi(v) = \frac{1}{1+\exp(-v)} \tag{8}$$

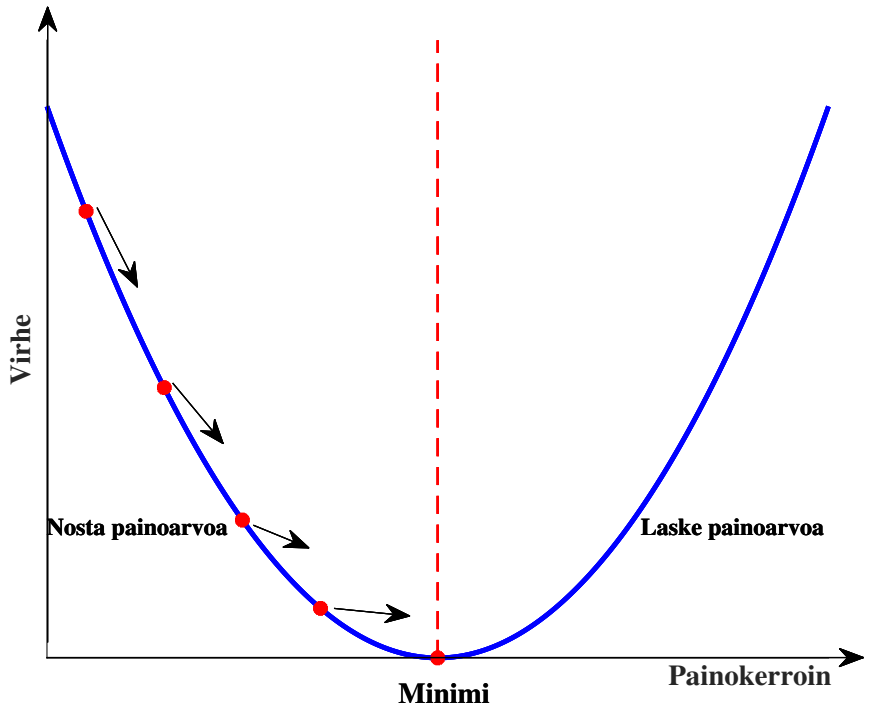
Neuroverkon suurin tehtävä on oppia ja muokata painokertoimia saavuttaakseen tarvittavan tehtävän. Neuroverkko voi oppia saaduista ennalta määrätyistä tiedoista, jolloin kyseessä on ohjattua oppimista. Ohjatussa oppimisessa opetusaineisto sisältää syötteen ja halutun ulostulon. Opetuksen jälkeen neuroverkkoa testataan aineistolla, joita neuroverkko ei ole ennen nähnyt. Opetus voidaan toteuttaa usealla eri algoritmilla, mutta esimerkkinä se voidaan toteuttaa laskemalla virheen neliö, joka saadaan halutun ulostulon ja saadun ulostulon erotuksesta. Virheen laskennan jälkeen neuroverkko säätää painokertoimia, jotta virhettä saataisiin minimoitua. Virheen minimointiin voidaan käyttää gradienttia, jotta halutun virheen minimoimisen jyrkin suunta tiedettäisiin.

Neuroverkko voi ylioppia opetusaineiston, jolloin opetusaineistosta laskettu virhe on erittäin pieni, mutta testiaineistosta laskettu virhe on suuri, jonka seurauksena neuroverkko on hyödytön uudelle aineistolle. Tämän ilmiön välttämiseksi neuroverkon opetusvaiheeseen tulee asettaa lopettamiskriteeri. Yksi suosituimmista lopettamiskriteereistä on aikainen pysäyttäminen [11]. Aikaisessa pysäyttämisessä valitaan pienimmän validointivirheen antama epoch-

määrä. Epoch on se ajanjakso, kun koko opetusaineisto käy kertaalleen neuroverkon opetusvaiheen lävitse. Valittu epoch-määrä on yleensä se kohta, jota edeltää validointivirheen kasvu.

Myötävirta-algoritmi (englanniksi feedforward propagation) on neuroverkon ensimmäinen vaihe, jolloin neuroverkolle annetaan syöte, jolloin syötettä käsitellään verkon piilotetuissa kerroksissa ja lopulta neuroverkko antaa tuloksen ulostulona. Tässä vaiheessa neuroverkonkerrokset eivät ole tietoisia ulostulon todenmukaisuudesta.

Vastavirta-algoritmi (englanniksi back-propagation) on gradienttiin perustuva opetusvaihe, jossa yhden opetustiedon perusteella neuroverkko päättää, kuinka painokertoimia ja vakiotermejä tulisi säätää. Termien säätäminen opetustiedosta tapahtuu lisäämällä tai vähentämällä säädettäviä arvoja löytääkseen, mitkä ovat tärkeimmät suhteet painokertoimilla, jotka eniten vähentävät virhettä. Vastavirta-algoritmi on ytimessään ilmaisu virhefunktion E osittaisderivaatoista $\partial E/\partial w$ ja $\partial E/\partial b$ [12]. Painoarvojen säätämisellä neuroverkko pyrkii löytämään virheen globaaliminimin. Virhefunktion gradientti ∇E antaa nopeimman kasvun suunnan. Ottamalla virhefunktion gradientista vastavektori $-\nabla E$ saadaan virhefunktion nopeimman vähenemisen suunta. Kuvassa 4 on visualisoitu, kuinka painoarvojen säätäminen tapahtuu vastavirta-algoritmissa.



Kuva 4. Vastavirta-algoritmin painokertoimien säätäminen visualisoitu.

Virhefunktion gradientin ∇E ollessa negatiivinen, tulee painokertoimen arvoa nostaa virheen

minimoimiseksi. Jos virhefunktion gradientti ∇E on positiivinen, tulee painokertoimen arvoa laskea virheen minimoimiseksi.

3 NEUROVERKKO-OPPIMISTYÖPAJAN TOTEUTUS

3.1 Katsaus nykyiseen työpajaan

Kasvontunnistus- ja liikedatankeruusovellusten toteutus LUT Junior University työpajaa varten on kandidaatintyö, jonka on tehnyt Matti Martikainen 2019 [13]. Kandidaatintyössä toteutettiin kaksi sovellusta laskennallisen tekniikan työpajaa varten, jotka olivat kasvontunnistus- ja liikedatankeruusovellus. Työpajan kohderyhmänä toimii kahdeksasluokkalaiset oppilaat.

Kasvontunnistussovelluksessa hyödynnettiin konenäön ja hahmontunnistuksen menetelmiä. Sovellus kuvasi käyttäjän kasvot ja vertasi kasvokuvaa julkisuuden henkilöiden kasvokuviin. Käyttäjiä kannustettiin vaihtamaan kasvojen asentoa ja peittämään jonkin osan kasvoista, jolloin pystyttiin tutkimaan tunnistuksen toimivuutta. Lisäksi pohdittiin, mitkä yhtenäiset kasvonpiirteet yhdistivät käyttäjän julkkikseen.

Toinen työpajan tehtävistä liittyi hahmontunnistukseen liikesignaalien avulla. Työpajan tehtävä liittyi data-analyysiin. Työ toteutettiin Raspberry Pi -laitteiston avulla, johon oli ohjelmoitu testiohjelma, joka mittasi ja tallensi antureista saatuja ilmanpaineita sekä kiihtyvyyksiä. Työssä osallistujat jaettiin kahteen ryhmään, joissa molemmissa oli kyseinen laite. Ensimmäinen ryhmä käveli laitteiston kanssa LUT-yliopiston tiloissa, jonka jälkeen laitteisto luovutettiin toiselle ryhmälle. Toisen ryhmän tavoitteena oli jäljittää ensimmäisen ryhmän kulkema reitti hyödyntäen laitteen palauttamia ilmanpaine- ja kiihtyvyyskuvaajia sekä referenssikuvaajia.

Kuitenkaan työssä ei päästy tutkimaan kohderyhmän kokemuksia, mutta työpajan toiminta testattiin yliopisto-opiskelijoista muodostuneessa ryhmässä. Testiryhmän kokemukset olivat positiiviset ja erityisesti hyvänä asiana koettiin, että demot esittelivät laskennallisen tekniikan monialaisuutta. Erityisenä huomiona testiryhmässä todettiin, että työpajan asia tulisi esittää rajatusti ja kahdeksasluokkalaisten ymmärryksen tasolla.

3.2 Neuroverkon oppimistyöpaja

Monet päätöksenteko-ongelmat ovat luokitteluongelmia. Luokitteluongelmat ratkaistaan tilastollisen tiedon ja hahmontunnistuksen avulla. Näitä metodeja voidaan toteuttaa koneoppimisen ja neuroverkkojen avulla. Neuroverkkojen kehittämisen aktiivisuus on huomattavas-

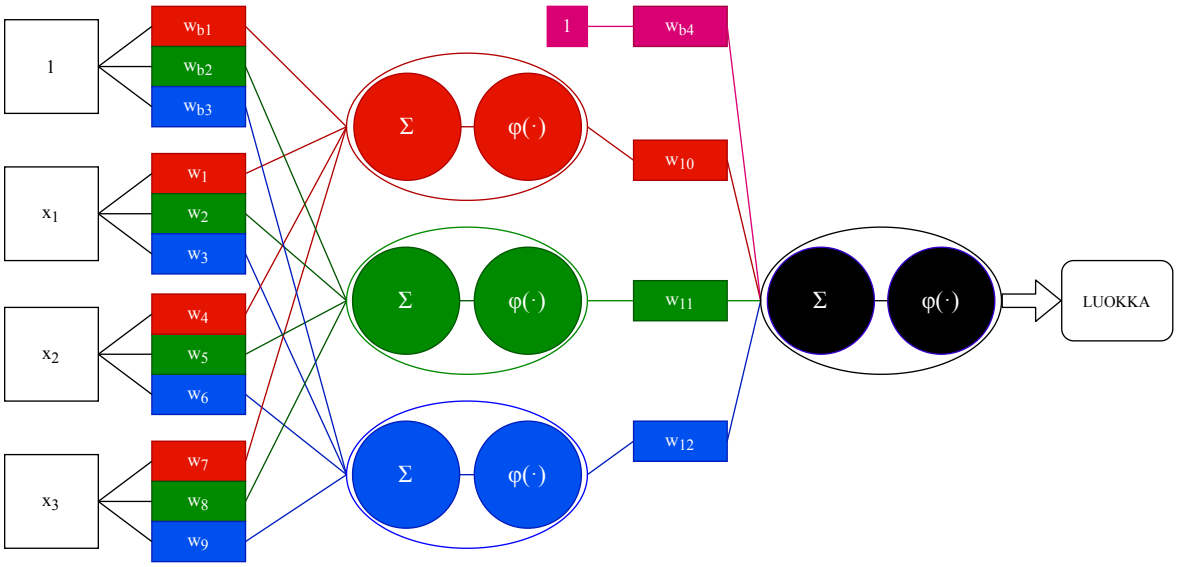
ti noussut onnistuneiden luokitteluongelmien seurauksena, joissa hyödynnetään piilotettuja yksiköitä ja vastavirta-algoritmin opetusmenetelmää [14].

Työpajan kehittäminen toteutettiin luomalla sovellus, jossa hyödynnettiin neuroverkkoa ratkaisemaan luokitteluongelman. Sovellus ei pelkästään ratkaise luokitteluongelmaa omatoimisesti vaan sovelluksessa käyttäjän tavoitteena on antaa kolme ominaisuutta, jotka kuuluvat johonkin tiettyyn luokkaan sekä säätää synapsisia painokertoimia, jotta ulostulona saataisiin oikea luokka, matkien neuroverkon vastavirta-algoritmin opetusta. Sovellus on tarkoitettu visualisoimaan sekä johdattamaan käyttäjä neuroverkon toimintaperiaatteeseen. Ennen sovelluksen esittämistä työpajassa osallistujat johdannottaan neuroverkkojen aiheeseen ja tulevaan sovellukseen PowerPoint-esitelmän avulla.

Neuroverkkomallin valinnassa haluttiin pysyä mahdollisimman yksinkertaisessa kokonaisuudessa. Keskeinen tavoite oli pitää säädettävien painokertoimien lukumäärä mahdollisimman pienenä, jotta sovelluksen käyttäjä ei kokisi painokertoimien säätämistä turhauttavana. Neuroverkkomallissa myös haluttiin pitää mukana yksi piiloneuronikerros. Näillä rajoitteilla neuroverkko pysyi myös visuaalisesti miellyttävämpänä sekä helpommin hahmotettavana. Näitten takia neuroverkolla ei voida taata parasta suorituskykyä.

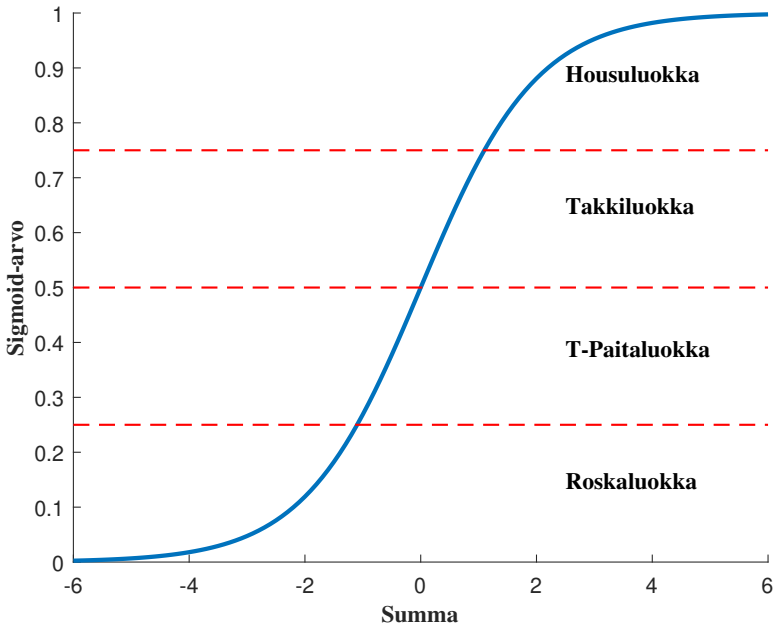
Luokitteluongelmaan käytetään kolmea eri käyttäjän antamaa syötettä lähtötilanteessa. Luokitteluongelman neuroverkko on esitetty kuvassa 5. Kuten kuvassa 3, syöte x_0 sisältää vakiotermit $b_1 - b_3$. Nyt jokaisella syötteellä on piilotetun neuronikerroksen verran painokertoimia. Ensimmäiset kolme neuronia käyttävät ulostulossa Sigmoid-funktiota. Viimeisestä neuronista saadaan ulostulona luokittelun tulos Sigmoid-funktiolla ja itsemääritetyillä luokkatunnusrajoilla.

Kuvan 5 neuroverkossa x_i ovat annettuja syötteitä eli ominaisuuksia. Syötteet x_i ovat binäärilukuja, jolloin binääriluku nolla vastaa ominaisuuden puuttumista ja yksi vastaa ominaisuuden olemassaoloa. Säädettävät painokertoimet ovat $w_1 - w_{12}$ ja bias-painokertoimiet $w_{b1} - w_{b4}$. Summaajat on merkitty Σ -symbolilla ja aktivaatiofunktio $f(summa)$. Neuroverkon ulostulo muutetaan sitä vastaavaksi luokaksi luokittelun lopussa ja luokka palautetaan käyttäjälle erillisessä tekstilaatikossa.



Kuva 5. Neuroverkko luokitteluongelmassa.

Luokittelu tapahtuu jakamalla sigmoid-funktiosta saatu arvo neljään eri luokkatunnukseen. Luokittelua on havainnollistettu kuvassa 6. Luokkatunnusrajat otetaan sigmoid-funktiosta, jotta neuroverkkomallissa painokertoimien määrä pysyisi vähäisenä. Tämän avulla voidaan myös rajata uloimmankerroksen neuronien lukumäärä neljästä vain yhteen. Neljän ulostulon tapauksessa, jokainen ulostulo olisi linkitetty yhteen luokkatunnukseen, jolloin ulostulo vastaisi todennäköisyyttä, että kyseinen luokkatunnus olisi ennustettu luokka. Kuvassa 6 olevat rajat luokkatunnuksille on esitetty taulukossa 2.

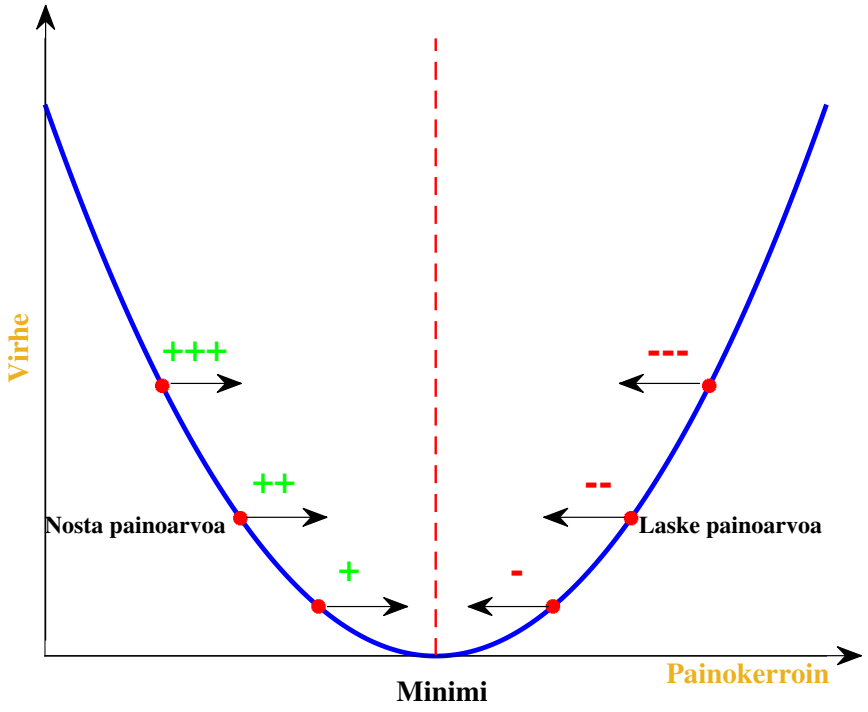


Kuva 6. Luokkatunnusrajat sigmoid-funktiosta.

Taulukko 2. Luokkatunnusrajat luokittelusovelluksessa

$\varphi(\cdot)$	Luokkatunnus
> 0.75	Housut
> 0.5 ja ≤ 0.75	Takki
> 0.25 ja ≤ 0.5	T-Paita
< 0.25	Tuntematon

Jotta painokertoimien säätely ei olisi liian vaativa tehtävä, lisätään sovellukseen palautejärjestelmä, joka antaa suuntaa oikeista painokertoimista. Palautejärjestelmä on havainnollistettu kuvassa 7.



Kuva 7. Palautejärjestelmä virheen minimointiin.

Kuvassa 7 $+++/---$ tarkoittaa, että painokerrointa on kasvatettava/vähennettävä runsaasti ($\pm$ 1.5), $++/--$ tarkoittaa, että painokerrointa on kasvatettava/vähennettävä hieman vähemmän ($\pm$ 1.0), $+/-$ tarkoittaa pientä muutosta painokertoimiin ($\pm$ 0.5) ja $=$ tarkoittaa tarkkaa painokertoimen arvoa ($\pm$ 0.1).

3.3 Suunnittelu- ja ohjelmointiympäristö

Neuroverkkosovelluksen kehityksessä käytetään Python-ohjelmointikieltä. Python-kieli on sopiva neuroverkon luomiseen tyhjästä suurien avointen lähdekoodien kirjastojen takia kuten *numpy*, *PyQt* ja *TensorFlow*. Opetusympäristönä Python toimii myös ideaalisesti, sillä luotuja sovelluksia voidaan suorittaa monilla eri käyttöjärjestelmillä ilman erillistä kääntämisprosessia.

Neuroverkkosovellusta varten ei käytetä valmiita koneoppimisen kirjastoja kuten *TensorFlow*, sillä luotu neuroverkko täytyy olla kuin kristallilaatikko eli kaikki sen sisällä tapahtuvat toiminnot täytyvät olla näkyvissä ja haluttuja arvoja täytyy pystyä tarkastelemaan ja muokkaamaan. Näitten takia neuroverkon toiminta toteutetaan *numpy*:llä ja graafinen käyttöliittymä *PyQt*:llä.

Numpy on Pythonin numeeriseen laskennan kirjasto, joka on luotu nopeaan vektoreiden ja matriisien käsittelyyn. Neuroverkkoa opetetaan laskemalla vastavirta-algoritmissa useasta syötevektorista arvoja, joten *numpy* on ideaalinen kirjasto tähän tarkoitukseen.

*PyQt*5 on Pythonin graafisten käyttöliittymien kirjasto, jonka luomia käyttöliittymiä voidaan suorittaa useissa käyttöjärjestelmissä ilman erillisiä toimintoja. *PyQt*5 käyttöliittymiä voidaan luoda käsinkirjoittamalla koodia tai hyödyntämällä $PyQt5-tools$ kirjaston kanssa tulevaa Qt Designer sovellusta. Qt Designer antaa käyttäjän suunnitella graafisia käyttöliittymiä raahaamalla ja pudottamalla haluttuja toimintoja valmiiseen pohjaan niin kutsutulla "mitä näet on mitä saat"−periaatteella [15]. Ohjelma on aloittelijoille ystävällisempi tapa luoda käyttöliittymiä, sillä ohjelma tuottaa valmiin koodin käyttäjän lisäämistä toiminnoista. Qt Designerin luomat *.ui*-tiedostot voidaan muuntaa *.py*-tiedostoiksi ajamalla esimerkiksi Windowsin komentokehotteessa:

```
>pyuic5 -x tiedosto.ui -o tiedosto.py
```

missä *tiedosto.ui* tarkoittaa Qt Designerin luomaa käyttöliittymätiedostoa ja *tiedosto.py* tarkoittaa käyttäjän nimeämää Python-tiedostoa. Tämän jälkeen käyttäjä voi lisätä ohjelmalle sisäisiä toimintoja muokkaamalla *.py*-tiedostoa.

3.4 Sovelluksen toteutus

3.4.1 Sovelluksen sisäinen toiminta

Sovelluksessa käytetty neuroverkko on muokattu Samay Shamdasanin luomasta yksinkertaisesta neuroverkkosovelluksesta, joka on saatavilla Enlight sivuston oppimateriaalista [16]. Enlight on alusta, jossa kuka tahansa voi oppia koodaamaan hauskojen projektien yhteydessä. Shamdasanin neuroverkkoa on muokattu, jotta sillä voitaisiin suorittaa luokittelu taulukon 2 mukaisesti. Neuroverkkoon on lisätty myös bias, bias-kertoimet ja bias-kertoimien opetus. Lisäyksenä on myös edellä esitettyyn palautejärjestelmää varten oleva piilotettu neuroverkko sekä sen toiminnat. Lopuksi neuroverkko-olioon lisättiin myös *getWeights*1, *getWeights*2, *setWeights*1 ja *setWeights*2 joiden avulla painoarvot voidaan tuoda ja muokata käyttöliittymän avulla.

Kuvasta 5 nähdään, että neuroverkolla on yksi ulostulokerros, yksi piilokerros ja kolme syötettä. Syötteet $x_1 - x_3$ ovat syötevektorin $\vec{x}$ alkioita, painokertoimet $w_1 - w_9$ sekä $w_{b1} - w_{b3}$ muodostavat painokerroin vektorin $\vec{w}_1$, piilokerroksen summaajat ovat Z matriisin alkioita, Z_2 matriisi on Z vektorin sigmoid-funktion tulokset, painokertoimet $w_{10} - w_{12}$ ja w_{b4} ovat $\vec{w}_2$ vektorin alkioita ja Z_3 on ulostulokerroksen neuronin summa. Ulostulo o on nyt sigmoid-funktiosta saatu tulos, joka saa luokkatunnuksensa kuvan 7 ja taulukon 2 mukaisesti.

Neuroverkon antamaa palautejärjestelmää varten on luotu sovelluksessa kutsuttu piilotettu-neuroverkko. Se toimii kopioimalla sen hetkiset painokertoimet ja opettaa kopioituja painokertoimia vastavirta-algoritmilla. Opetusaineiston syötteet eli matriisi X sisältää 7 syötevektoria $\vec{x}$ ja jokaisella syötevektorilla on myös oma ulostulo o, jotka muodostavat matriisin Y. Lähtöarvot ovat

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} \\ x_1^{(3)} & x_2^{(3)} & x_3^{(3)} \\ x_1^{(4)} & x_2^{(4)} & x_3^{(4)} \\ x_1^{(5)} & x_2^{(5)} & x_3^{(5)} \\ x_1^{(6)} & x_2^{(6)} & x_3^{(6)} \\ x_1^{(7)} & x_2^{(7)} & x_3^{(7)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{9}$$

ja halutut ulostulot

$$Y = \begin{bmatrix} o^{(1)} \\ o^{(2)} \\ o^{(3)} \\ o^{(4)} \\ o^{(5)} \\ o^{(6)} \\ o^{(7)} \end{bmatrix} = \begin{bmatrix} 1 \\ 0.66 \\ 0.38 \\ 0.12 \\ 0.12 \\ 0.12 \\ 0.66 \end{bmatrix} = \begin{bmatrix} Housut \\ Takki \\ T-Paita \\ Tuntematon \\ Tuntematon \\ Tuntematon \\ Takki \end{bmatrix} \tag{10}$$

Laskenta aloitetaan laskemalla summaaja matriisin Z arvot syötteiden X ja sovelluksen tuottamilla satunnaisilla painoarvoilla W_1 sekä bias arvolla b. Piilokerroksen arvot Z lasketaan kaavalla

$$Z = X \cdot W_1[0:3] + b \cdot W_1[3] \tag{11}$$

Laskemalla piilokerroksen Z arvoista sigmoid muunnoksen saadaan arvot piilokerroksen Z_2 arvot. Toistamalla kaavan 11 laskenta Z_2 arvoilla ja painokertoimilla W_2 saadaan ulostulokerroksen arvot eli

$$Z_3 = Z_2 \cdot W_2[0:3] + b \cdot W_2[3] \tag{12}$$

Muuntamalla Z_3 arvot sigmoid-funktiolla saadaan ennustetut luokkatunnusarvot $\hat{Y}$. Seuraavaksi sovellus aloittaa vastavirta-algoritmin. Algoritmin vaiheet ovat seuraavat:

1. Lasketaan ulostulokerroksen virheen marginaali laskemalla odotettujen arvojen Y ja ennustettujen arvojen $\hat{Y}$ erotus.

2. Lasketaan ulostulokerroksen sigmoid-funktiosta saatu derivaatta o_{delta}. Derivaattojen summista määritetään bias-kertoimien muutokset laskemalla sarakkeiden summat.

3. Hyödynnetään vaiheen 2 derivaattaa löytääkseen piilokerroksen Z_2 vaikutusta ulostulon virheeseen. Laskettua virhettä kutsutaan Z_{2error}.

4. Lasketaan piilokerroksen sigmoid-funktiosta saatu derivaatta z_{2delta}. Derivaattojen summista määritetään bias-kertoimien muutokset laskemalla sarakkeiden summat.

5. Säädetään piilokerroksen painoarvoja W_1 esittämällä pistetulo syötteiden X^T ja Z_{2delta}. Piilokerroksen bias-kertoimiin summataan Z_{2delta} sarakkeiden summat. Painoarvot W_2 säädetään esittämällä pistetulo Z_2^T ja o_{delta} kanssa. Uloimpaan bias-kertoimeen summataan o_{delta} sarakkeen summat.

Sigmoid-funktion derivaatta kutsutaan nimellä sigmoid-prime, joka antaa muutoksen tahdin tai kaltevuuden aktivaatiofunktiossa summan suhteen. Sigmoid-prime esitetään kaavalla

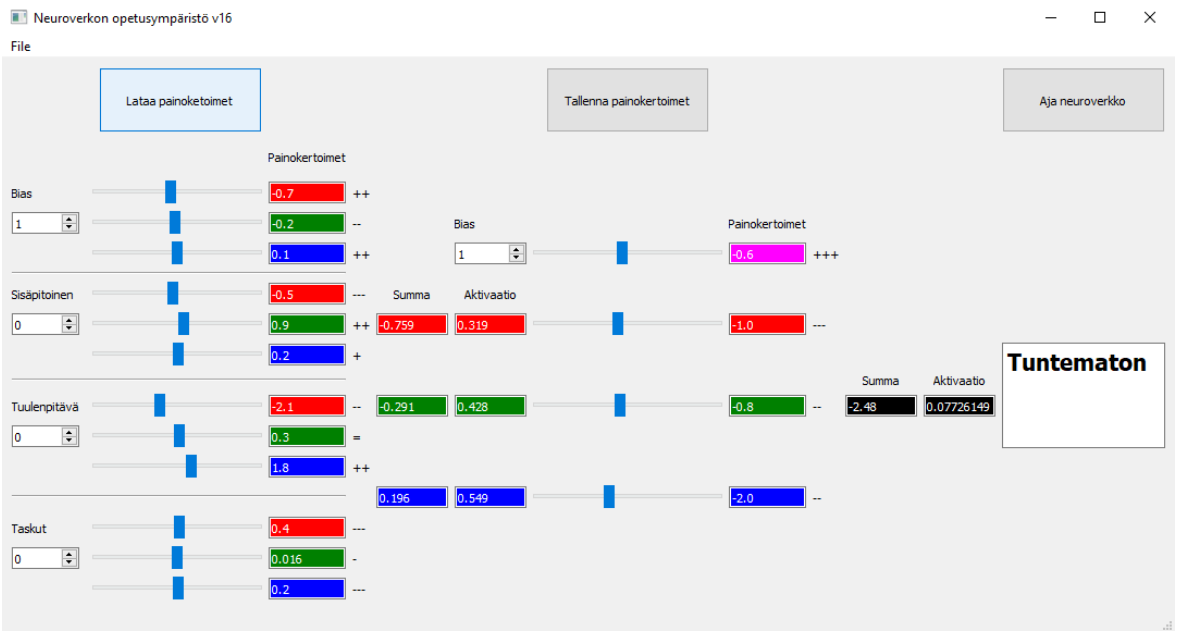
$$\varphi'(s) = \varphi(s) * (1 - \varphi(s)) \tag{13}$$

Python-kielellä neuroverkon virheenlaskenta ja painojen säätäminen on muotoa

```
def backward(self, X, y, o):
self.o_error = y - o
self.o_delta = self.o_error*self.sigmoidPrime(o)
self.o_delta_bias = np.sum(self.o_delta, axis=0)

self.z2_error = self.o_delta.dot(self.W2.T)
self.z2_delta = self.z2_error[:,:3]*self.sigmoidPrime(self.z2)
self.z2_delta_bias = np.sum(self.z2_delta, axis=0)

self.W1[:3] += X.T.dot(self.z2_delta)
self.W1[3] += self.z2_delta_bias

self.W2[:3] += self.z2.T.dot(self.o_delta)
self.W2[3] += self.o_delta_bias
```

3.4.2 Sovelluksen käyttöliittymä

Kuvassa 8 on sovelluksen käyttöliittymä. Käyttöliittymässä on kolme painiketta ylhäällä. 'Lataa painokertoimet' tuo käyttöliittymään neuroverkon satunnaisesti alustamat painokertoimet. Käyttäjä voi nyt alkaa muuntamaan neuroverkon painokertoimia sinisten liukureiden avulla tai kirjoittamalla painokertoimelle arvon. Muutokset tallennetaan 'Tallenna painokertoimet' nappulasta, jolloin käyttöliittymä lukee arvot ja muokkaa neuroverkon painokertoimiksi. Tämän jälkeen käyttäjä ajaa neuroverkon painokertoimilla painamalla 'Aja neuroverkko'. Neuroverkon ajettua, käyttöliittymä antaa jokaiselle painokertoimelle palautteen sekä luokittelun tuloksen, joka näkyy käyttöliittymässä oikeassa tekstilaatikossa. Käyttäjä voi nyt lähteä muuntamaan palautteen mukaisesti painokertoimia ja tarkistaa luokittelun todenmukaisuutta muokkaamalla vasemmalla olevia ominaisuuksia ja pohtimalla antoiko luokittelu oikean luokkatunnuksen. Sovellusta voidaan myös ajaa näppäimistön avulla, missä CTRL+L = 'Lataa painokertoimet', CTRL+S = 'Tallenna painokertoimet' ja CTRL+R = 'Aja neuroverkko'. Sovelluksessa voidaan myös antaa neuroverkon itse säätää neuroverkon painokertoimia painamalla 'CTRL+T'. Pikanäppäinten toiminnot löytyvät myös sovelluksen vasemmasta ylänurkasta valitsemalla 'File'.



Kuva 8. Neuroverkon opetusympäristö-sovelluksen käyttöliittymä (Windows 10).

Sovelluksessa on näkyvillä jokaisen neuronin summaus ’Summa’ -tekstin alapuolella sekä sigmoid-funktiosta saatu tulos ’Aktivaatio’ -tekstin alapuolella. Neuroverkkoa on hahmotettu värien avulla, jotka ovat kuvan 5 mukaisesti yhdistetty. Värien avulla pyritään auttamaan käyttäjää yhdistämään, mitkä painokertoimet muuntavat minkäkin neuronin tulosta.

Sovelluksen käyttö testattiin toimivan Windows 10 käyttöjärjestelmällä sekä myös Ubuntu 16.04. Kuvan 8 käyttöliittymä voi olla erilainen riippuen käyttöjärjestelmästä.

3.5 Neuroverkkotyöpajan testaus

Työpajan kokonaisuutta testattiin kolmen henkilön ryhmässä, joista kaksi olivat laskennallisen tekniikan toisen vuoden opiskelijoita ja yksi oli kolmannen vuoden sähkötekniikan opiskelija. COVID-19 pandemian tuoman poikkeustilan takia, työpaja toteutettiin etäyhteyden kautta. Testiryhmän osallistujista kaksi eivät olleet ennestään tietoisia neuroverkon toimintaperiaatteista, mutta tiesivät sen liittyvän koneoppimiseen. Yksi osallistuja ei ennestään tiennyt neuroverkoista mitään.

Työpaja järjestettiin pitämällä johdanto luokitteluun ja neuroverkkoihin PowerPoint esityksen avulla. Johdannon jälkeen osallistujat avasivat työpajan sovelluksen ja ohjeistuksen avulla sovelluksen toiminnot esitettiin osallistujille. Ohjeistuksen jälkeen osallistujat omatoimisesti käyttivät sovellusta vaihtamalla syötteiden ja painokertoimien arvoja, jotta luokittelusta

tulisi todenmukainen. Sovelluksen käyttöä tuettiin näyttämällä kuvia 5 ja 7, jotta osallistujat hahmottaisivat neuroverkon mallin ja palautejärjestelmän. Testattu työpajakokonaisuus kesti noin 20 minuuttia ja työpajan suunniteltu kesto ilman etäyhteyttä koettaisiin kestävän noin 15 minuuttia, jolloin työpajan pitäjän on helpompi esittää asia ja seurata toimintaa.

Testiryhmän antamat risut ja ruusut työpajasta:

1. Gradientti koettiin liian vaikeana asiana esittää. Ehdotuksena parantamiseen annettiin, että se pitäisi jättää kokonaan pois ja korvata gradientti sanomalla, että pyritään löytämään mahdollisimman pieni virhe.

2. Myös kuvassa 7 näkyvä paraabeli koettiin olevan testiryhmän osallistujien mielestä kahdeksasluokkalaiselle liian vaikeaksi hahmottaa, jos funktiokuvaajat eivät ole tuttuja. Kahdeksasluokkalaisten funktioiden opetus muisteltiin olevan vain mekaanista laskentaa, jolloin tuloksia ei jääty tulkitsemaan millään tavalla. Asia voitaisiin esittää mäkenä, jonka pohjalle yritettäisiin päästä.

3. Sovelluksen toimintojen esittely, ennen kuin oppilaat itse käyttivät sitä, koettiin myös olevan tärkeä asia. Asiat kuten painokertoimien tarkoitus sekä niiden vaikutus, kun niitä säädetään, koettiin tärkeänä esittää ensin.

4. Testiryhmä halusi myös nähdä ennen sovelluksen käyttöä, miltä näyttäisi sovellus, jossa painokertoimet antaisivat todenmukaisen luokittelun. Tämän avulla oppilaat tietäisivät mitä he yrittävät sovelluksella tehdä.

5. Hyvänä asiana koettiin johdannossa, että aivoja verrattiin neuroverkkoihin ja näin asia yhdistetään nopeasti. Myös johdanto luokitteluun vaatekappaleiden avulla koettiin hyväksi esimerkiksi, mitä neuroverkoilla voitaisiin tehdä.

6. Yksi asia, mikä muistui esityksen jälkeen, oli neuroverkon arkkitehtuuri.

4 JOHTOPÄÄTÖKSET

Työssä toteutettiin itse rakennettu ohjelmisto, jonka avulla neuroverkon matemaattista toimintaa luokittelutehtävän avulla havainnollistettiin. Ohjelmisto on suunniteltu LUT Junior University -työpajaa varten, joka pidetään kahdeksasluokkalaisille oppilaille. Työn haasteena pysyi alusta lähtien esittää asia mahdollisimman yksinkertaisena, joka myös todettiin viime vuoden työpajan toteutuksessa. Työpajan toimintaa testattiin etäyhteyden kautta kolmen henkilön testiryhmällä. Testiryhmältä saadun palautteen perusteella sovellus pystyi hahmottamaan neuroverkon toimintaa. Sovellusta tukeva johdanto PowerPoint-esitelmä koettiin myös tärkeänä osana ennen sovelluksen esittämistä. Kritiikkinä työpaja kokonaisuudessa tuli esille matemaattisten termien käyttö, kuten gradientti ja virheen minimointi, joka huomioitiin työpajan johdannon aikana.

Neuroverkkojen esittäminen kahdeksasluokkalaisen ymmärryksenä on vaativa tehtävä. Matematiikkaa neuroverkon takana ei voida esittää kahdeksasluokkalaiselle 15 minuutissa. Pelkän matematiikan esittäminen työpajassa voi koitua oppilaille liian haasteelliseksi ymmärtää tai jopa tylsänä. Näitten takia sovelluksen neuroverkko on hyvin minimaalinen malli ja suorituskyvyltään sitä ei voida pitää parhaana mahdollisena ratkaisuna esitettyyn luokittelutehtävään. Sovellus on suunniteltu visualisoimaan neuroverkkoa ja neuroverkon opetusvaihetta ja näihin tavoitteisiin testiryhmän saadun palautteen perusteella päästiinkin.

Luotua neuroverkkotyöpajaa tullaan järjestämään LUT-yliopiston laskennallisen tekniikan Junior Universityn työpajassa 8-luokkalaisille, minkä avulla uusi työpaja tulee saamaan sen vaatimaa palautetta omalta kohderyhmältään. Palautteen avulla työpajan sovellusta ja kokonaisuutta voidaan tarpeen mukaan kehittää.

Lähteet

[1] LUT. *LUT Junior University - Lappeenranta*. 2020. URL: https://www.lut.fi/yhteistyo-ja-palvelut/palvelut-oppilaitoksille/lut-junior-university-lappeenranta (viitattu 09. 02. 2020).

[2] Lappeenrannan Uutiset. *LUT Junior University pilotoi ensi lukuvuonna – "Pyrimme antamaan valmiuksia korkeakouluopintoihin"*. 24. helmikuuta 2018. URL: https://www.lappeenrannanuutiset.fi/artikkeli/601607-lut-junior-university-pilotoi-ensi-lukuvuonna-pyrimme-antamaan-valmiuksia (viitattu 09. 02. 2020).

[3] P. D. Wasserman ja T. Schwartz. "Neural networks. II. What are they and why is everybody so interested in them now?" *IEEE Expert* vol. 3.nro 1 (1988), s. 10–15. DOI: 10.1109/64.2091.

[4] Expert System Team. *What is Machine Learning? A definition*. Expert System. 6. maaliskuuta 2017. URL: https://expertsystem.com/machine-learning-definition/ (viitattu 05. 05. 2020).

[5] David Silver et al. "Mastering the game of Go without human knowledge". *Nature* nro 550 (lokakuu 2017), s. 354–359. DOI: https://doi.org/10.1038/nature24270.

[6] Hongzi Mao, Mohammad Alizadeh, Ishai Menache ja Srikanth Kandula. "Resource Management With deep Reinforcement Learning". *Proceedings of the 15th ACM Workshop on Hot Topics in Networks* (Atlanta GA USA 2016), s. 50–56.

[7] Nils J Nilsson. *Learning Machines: Foundations of trainable pattern-classifying systems*. McGraw-Hill Book Company, 1965.

[8] *10. Hermosolu. Hermosolun rakenne*. www.shutterstock.com. URL: https://peda.net/valkeakoski/opetuspalvelut/pk/tyry/oppiaineet/biologia/bi-toimela/ihminen/hermosolu (viitattu 17. 05. 2020).

[9] Simon S. Haykin. *Neural networks and learning machines*. 3rd ed. New York: Prentice Hall, 2009. 906 s.

[10] Gupta Dishashree. *Fundamentals of Deep Learning - Activation Functions and their use*. Analytics Vidhya. 30. tammikuuta 2020. URL: https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/ (viitattu 16. 02. 2020).

[11] Ian Goodfellow, Yoshua Bengio ja Aaron Courville. *Deep learning*. Vol. 22. MIT press, 2016.

[12] Michael A Nielsen. *Neural networks and deep learning*. Vol. 2018. Determination press San Francisco, CA, USA: 2015.

[13] Matti Martikainen. "Kasvontunnistus- ja liikedatankeruusovellusten toteutus LUT Junior University työpajaavarten". Kandidaatintyö. Lappeenrannan-Lahden teknillinen yliopisto LUT, 2019.

[14] Sholom M Weiss, Ioannis Kapouleas ja JW Shavlik. "An empirical comparison of pattern recognition, neural nets and machine learning classification methods". *Readings in machine learning* (1990), s. 177–183.

[15] The Qt Company Ltd. *Qt Designer Manual*. 2020. URL: https://doc.qt.io/qt-5/qtdesigner-manual.html (viitattu 28. 04. 2020).

[16] Samay Shamdasani. *Build a Neural Network*. An introduction to building a basic feed-forward neural network with backpropagation in Python. 4. lokakuuta 2017. URL: https://enlight.nyc/projects/neural-network/ (viitattu 18. 02. 2020).