Niko Tammentie

C#- ja Python-ohjelmointikielten soveltuvuus neuroverkkojen toteuttamiseen

Tietotekniikan kandidaatintutkielma

11. kesäkuuta 2018

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Niko Tammentie

Yhteystiedot: nividata@student.jyu.fi

Työn nimi: C#- ja Python-ohjelmointikielten soveltuvuus neuroverkkojen toteuttamiseen

Title in English: Suitability of C#- and Python- programming languages for implementing Neural Networks

Työ: Kandidaatintutkielma

Sivumäärä: 22+0

Tiivistelmä: Koneoppimista hyödynnetään kaikkialla. Neuroverkot muodostavat koneoppimisen selkärangan, joten luodakseen koneoppimista hyödyntäviä sovelluksia, on ohjelmoijan ymmärrettävä neuroverkkojen toimintaa. Tässä tutkielmassa esitellään neuroverkkojen historiaa ja toimintatapoja ja etsitään syitä sille, miksi Python-ohjelmointikieli on niin suosittu neuroverkkojen toteutukseen. Johtopäätöksenä havaitaan, että muun muassa NumPy-kirjasto sekä Googlen kehittämä TensorFlow-kirjasto puoltavat neuroverkkojen toteutusta Python-kielellä.

Avainsanat: neuroverkko, C#, Python

Abstract: Machine learning is being utilized everywhere. Neural networks form the basis for machine learning, so in order to create machine learning applications, the programmer has to have a grasp of how neural networks function. The purpose of this thesis is to summarize the history of neural networks, explain how neural networks function and find reasons as to why Python is so popular in the implementation of neural networks. This thesis finds that for example NumPy and Google's Tensorflow libraries make Python the preferred language for creating neural networks.

Keywords: neural network, C#, Python

Kuviot

Kuvio 1. Mark 1 perseptroni. Vasemmalla kamera. Keskellä pistoketaulu. Oikealla synapsipainot. (Bishop 2006, s. 196) 4
Kuvio 2. Visualisaatio Perseptronin ja Adalinen kyvyttömyydestä kategorisoida luokkia epälineaarisesti..... 5
Kuvio 3. Monikerroksinen perseptroniverkko. 7
Kuvio 4. Toisin kuin Madeline, Adaline ja Mark 1 Perseptroni, modernit neuroverkot kykenevät ratkaisemaan epälineaarisia ongelmia. 8

Sisältö

1 JOHDANTO 1

2 NEUROVERKKO 3
2.1 Historia 3
2.2 Toimintaperiaate 6

3 TARKASTELTAVAT OHJELMOINTIKIELET 11
3.1 C# 11
3.2 Python 12

4 YHTEENVETO 14

KIRJALLISUUTTA 16

1 Johdanto

Koneoppiminen mahdollistaa järjestelmät, jotka täsmällisen ohjelmoinnin sijaan käyttävät suuria datamääriä toimintamallinsa määrittelyyn. Termin mukaisesti kone siis oppii toimimaan toivotulla tavalla (Hurwitz & Kirsch 2018, s. 1-17).

Neuroverkko on koneoppimisen malli, joka matkii biologisten aivojen toimintatapaa. Neuroverkko koostuu neuroneista ja niitä toisiinsa liittävistä synapseista. Neuroverkkoa opetetaan syöttämällä sille suuret määrät dataa, minkä aikana verkko päivittää automaattisesti synapsiensa arvoja. Synapsien arvojen ollessa oikeat, neuroverkko osaa tulkita dataa, jota se ei ole koskaan ennen nähnyt. Neuroverkot muodostavat nykyisin melkein kaiken koneoppimisen selkärangan. (Hurwitz & Kirsch 2018, s. 1-17)

Hyvänä esimerkkinä koneoppimisen ja neuroverkkojen tehokkuudesta on Google DeepMind -yrityksen kehittämä Go-lautapeliä pelaava tietokoneohjelma AlphaGo, joka maaliskuussa 2016 voitti pelissä ammattilaistason pelaajan. Go-lautapelissä on absurdi määrä mahdollisia pelitilanteita, joten neuroverkkoteknologian hyödyntäminen oli ainoa tapa tämän lopputuloksen saavuttamiseen. Pelitilanteen analysoiminen ja pelistrategian valinta voitiin tehokkaasti automatisoida kulloisenkin pelitilanteen mukaan.

Perinteiset metodit, jotka perustuvat puurakenteen läpikäymiseen eivät kahdesta syystä sopineet Go-lautapelin pelaamiseen: päätöspuusta kasvaa mahdottoman suuri, ja jokaisen pelisiirron voimakkuuden määrittely on vaikeaa. Neuroverkkojen ansiosta AlphaGo pystyy keskittymään tehokkaasti tämänhetkiseen pelitilanteeseen käymättä läpi edellisiä siirtoja.

Neuroverkot ja koneoppiminen ovat tietotekniikan alalla merkitykseltään kasvava konsepti. Neuroverkkoja hyödynnetään yhä useampien ongelmien ratkaisemiseen. Tämän takia moni ohjelmoija tutustuukin aiheeseen ohjelmoimalla oman neuroverkkonsa. Kuten minkä tahansa ohjelman, neuroverkonkin voi luoda tyhjästä millä tahansa ohjelmointikielellä. On havaittavissa, että valtaosa internetistä löyty-

vistä ohjeista neuvoo rakentamaan neuroverkon Python-ohjelmointikielellä. Mitä ominaisuuksia Pythonissa on, joita esimerkiksi C#-ohjelmointikielessä ei ole?

Tämä tutkielma on kirjallisuuskatsaus, joka keskittyy neuroverkkojen määrittelyn lisäksi vertailemaan niiden luomista C#- ja Python-ohjelmointikielissä. Tarkkailun kohteena tulee erityisesti olemaan kielien ominaisuuksien sopivuus neuroverkkojen luomiseen. Tarkoitus on myös selvittää kielten välisiä eroja relevanttien kirjastojen, kuten valmiiden koneoppimiskirjastojen ja matriisilaskentakirjastojen suhteen. Tarkoitus on myös selvittää, miksi suurin osa internetistä löytyvistä neuroverkkoohjeista suosii Pythonia ylitse muiden ohjelmointikielten.

Tässä tutkielmassa keskitytään kuvailemaan neuroverkkojen toimintaperiaatteita konseptitasolla. Pääpaino on neuroverkkojen esittelyssä lukijalle, ja matemaattisia kaavoja esitetään vain vähän. Näin ollen tutkielma sopii antamaan aihepiiristä hyvän yleiskäsityksen myös niille lukijoille, joilla ei ole laajaa matemaattista osaamista tai ohjelmointikokemusta.

Luvussa Neuroverkko esitellään neuroverkko yleisesti, keskittyen ensin neuroverkkojen historiaan ja sen jälkeen monikerroksisen perseptronin toimintaan. Tämän yleisesittelyn tavoitteena on toimia sekä ensikosketuksena neuroverkkoihin, että avata tutkimusongelmaa. Luvun jälkeen lukija ymmärtää mm. miksi luvussa Tarkasteltavat ohjelmointikielet käsitellään kielten välisiä matriisilaskentakirjastoja.

2 Neuroverkko

Käsite neuroverkko on lähtöisin biologiasta aivotutkimuksen saralta, ja sillä tarkoitetaan kaikkien elävien, aivollisten organismien aivoista löytyvää päätöksentekoa säätelevää hermostoa. Puhuttaessa neuroverkoista tietotekniikan kontekstissa, tarkoitamme aivojemme toiminnan malliin perustuvia tietokoneohjelmia, keinotekoisia neuroverkkoja (eng. artifical neural network). Neuroverkko loistaa tiedon kategorisoimisessa, ja sitä usein käytetäänkin erinäisissä tunnistus- ja konenäkösovelluksissa (Hurwitz & Kirsch 2018, s. 1-17). Tässä luvussa tarkastellaan neuroverkkojen historiaa ja tapahtumia, jotka ajoivat ja toisinaan jarruttivat keinotekoisten neuroverkkojen kehitystä.

2.1 Historia

Ensimmäisen askeleen kohti keinotekoisia neuroverkkoja ottivat neuropsykologi ja matemaatikko (McCulloch & Pitts 1943, s.115-133) neuronien toimintaa käsittelevässä tutkielmassaan, jossa he mallinsivat neuroverkkoja virtapiirien avulla. Tutkielmassaan he todistivat, että keinotekoinen neuroverkko pystyisi esittämään mitä tahansa äärellistä loogista lauseketta. Vaikka Warrenin ja Cullochin neuroverkko ei pystynyt oppimaan, heidän työnsä oli astinkivi seuraaville keinotekoisten neuroverkkojen malleille ja paradigmoille (Eberhart & Dobbins 1990, s. 17-18).

Vuonna 1949 kanadalainen psykologi Hebb (1949) osoitti, että neuroneiden välissä olevat hermoliitokset, toiselta nimeltään synapsit, vahvistuvat niitä käytettäessä, ja että neuroverkoissa kaikki data on tallennettu näiden synapsien vahvuuteen.

Psykologi Rosenblatt (1957) kuvaili raportissaan luomaansa vanhinta yhä käytössä olevaa neuroverkkoalgoritmia, ‘perseptronia’ (eng. perceptron). Ensimmäinen implementaatio algoritmista oli IBM 704 - tietokoneelle (Kuvio 1), mutta myöhemmin Rosenblatt rakensi Office of Naval Researchin rahoittaman erikoisvalmisteisen Mark 1 perseptronin. Mark 1 perseptroni sai sisääntulonsa, 400 pikselin kuvan, sisäänrakennetun kameran kautta. Rosenblatt osoitti, että hänen mallinsa pystyi luo-

tettavasti kategorisoimaan käsinkirjoitettuja kirjaimia ja yksinkertaisia geometrisista muotoja, ja että kategoriointi onnistuisi jopa lisätyn kohinan läpi. Malli toimisi myös joidenkin osien verkosta ollessa poissa käytöstä, sillä tieto hajautettaisiin onnistuneesti jäljellejäävään neuroverkkoon. Muun muassa (Garson 1998, s. 3-4) ja muut pitävät Rosenblattia koneellisten neuroverkkojen uranuurtajana.

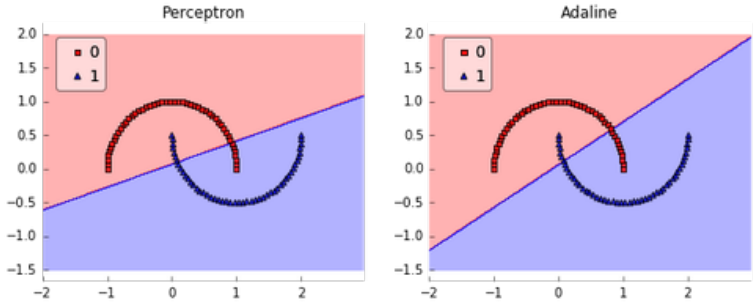
Kuvio 1. Mark 1 perseptroni. Vasemmalla kamera. Keskellä pistoketaulu. Oikealla synapsipainot. (Bishop 2006, s. 196)

Hieman Rosenblattia myöhemmin Stanfordin Yliopiston professori Bernard Widrow yhdessä oppilaansa Marcian Hoffin kanssa (Widrow & Lehr 1993, s.1-3) kehitteli oman keinotekoisen neuroverkkoalgoritminsa, ADALINE:n, eli Adaptive Linear Elementin. Sekä ADALINE että perseptroni toimivat samaan tapaan yksikerroksisina neuroverkkoina, mutta ADALINE oli nopeutensa ja tarkkuutensa vuoksi tehokkaampi. Kehitettyään monikerroksisen version ADALINE:sta, MADALINEN (Many ADALINE), Widrow ja hänen oppilaansa alkoivat kehittää käytännön sovellutuksia ADALINE:lle ja MADALINE:lle. Ensimmäisiin sovellutuksiin kuului mm. puheen- ja hahmontunnistus, sääennustus ja mukautuva ohjainohjelmisto. Ensimmäinen käytännön ongelmaan toteutettu ratkaisu joka on käytössä yhä tänä päivänä on MADALINE:n käyttö kaiun eliminoimiseen puheluista.

Näiden onnistumisten seurauksena odotukset neuroverkkojen suhteen kasvoivat räjähdysmäisesti. (Olazaran 1993, s.341-343) dokumentoi, kuinka New York Times kirjoitti perseptronin julkistamistilaisuuden yhteydessä, että Rosenblattin mukaan myöhempien perseptronien odotetaan osaavan kävellä, puhua, nähdä, kirjoittaa ja lisääntyä. Rosenblattin puheet perseptronien kaikkivoipaisuudesta aiheuttivat välittömän vastareaktion tietyssä osassa tieteellistä yhteisöä. Paisuneet odotukset yh-

dessä uusien sovellutusten ja teknologisten harppausten puutteen kanssa johtivat pettymyksiin ja harhakuvitelmien menetyksiin.

Tutkijat Marvin Minsky ja Seymour Papert julkaisivat vuonna 1968 Rosenblattin perseptroneja kritisoivan kirjan 'Perceptron' Minsky & Papert (1968). Kirjassa kiinnitettiin huomiota mallin kykyyn ratkaista ainoastaan lineaarisesti erotettavissa olevia ongelmia. Minsky ja Papert todistivat, että neuronien määrä sisääntulokerroksessa kasvaisi mielivaltaisen suureksi yrittäessä mallintaa geometrisesti merkityksellisiä luokkia. Täten he osoittivat, että perseptronilla olisi mahdotonta kategorisoida luokkia, jotka liittyisivät toisiinsa merkitsevästi mutta epälineaarisesti. Yleinen mielipide on se, että kirja oli yksi päätekijöistä jo valmiiksi kuihtuvan neuroverkkotutkimuksen rahoituksen pienentämiseen ja yleisen kiinnostuksen vähentymiseen, sekä niin sanotun AI-talven alkuun (Garson 1998, s. 3-4).



Kuvio 2. Visualisaatio Perseptronin ja Adalinen kyvyttömyydestä kategorisoida luokkia epälineaarisesti.

1980-luvun alkupuolella kiinnostus neuroverkkoja kohtaan alkoi taas kasvaa. Psykologit Rumelhart & McClelland (1986) esittelivät neuroverkoille kriittisen konnektionismin käsitteen, ja heidän ansiokseen usein laitetaan 1980-luvun neurolaskentateknologian noususuhdanne. Rumelhart ja McClelland eivät kuitenkaan olleet ainoita, jotka herättelivät uinuvaa teknologiaa. Tiedemies Hopfield (1982)n lähestymistapa ei ollut vain yrittää mallintaa aivojen toimintaa, vaan yrittää luoda hyödyllisiä laitteita. (Paris 2005, s.217-218) kertoo Hopfieldin suureksi ansioksi hänen karismaattisen kykynsä osata selittää, mitä neuroverkot pystyisivät ja eivät pystyisi tekemään vajoamatta teknologian ylimyymiseen.

Samana vuonna Japanin Kiotossa pidettiin konferenssi neuroverkoista yhdessä Yhdysvaltojen kanssa. Konferenssin yhteydessä Japani julisti viidennen sukupolven ponnistuksen neuroverkkojen suhteen. Yhdysvalloissa kasvoi huoli jälkeenjäämisestä, ja pian neuroverkkotutkimus sai taas lisää rahoitusta. Nykyisin neuroverkkoja sovelletaan koneoppimisessa kaikkialla. (Paris 2005, s.217-218)

2.2 Toimintaperiaate

Sana neuroverkko kattaa kymmeniä erilaisia hermoverkon toimintaan perustuvia informaatiokäsittelyn, matematiikan ja laskennan malleja. Tästä kappaleesta alkaen tullaan käsittelemään ainoastaan yhtä eniten käytetyimmistä neuroverkoista, eteenpäin kytkettyä monikerroksista perseptroniverkkoa (eng. multi-layer perceptron). Tämän tutkielman keinotekoisen neuroverkon toimintaa käsittelevissä esimerkeissä käytetään aikaisemmin esiteltyä Rosenblattin kaksikerroksista Mark 1 Persetronia, eli piirrettyjä numeroita tunnistavaa perseptroniverkkoa.

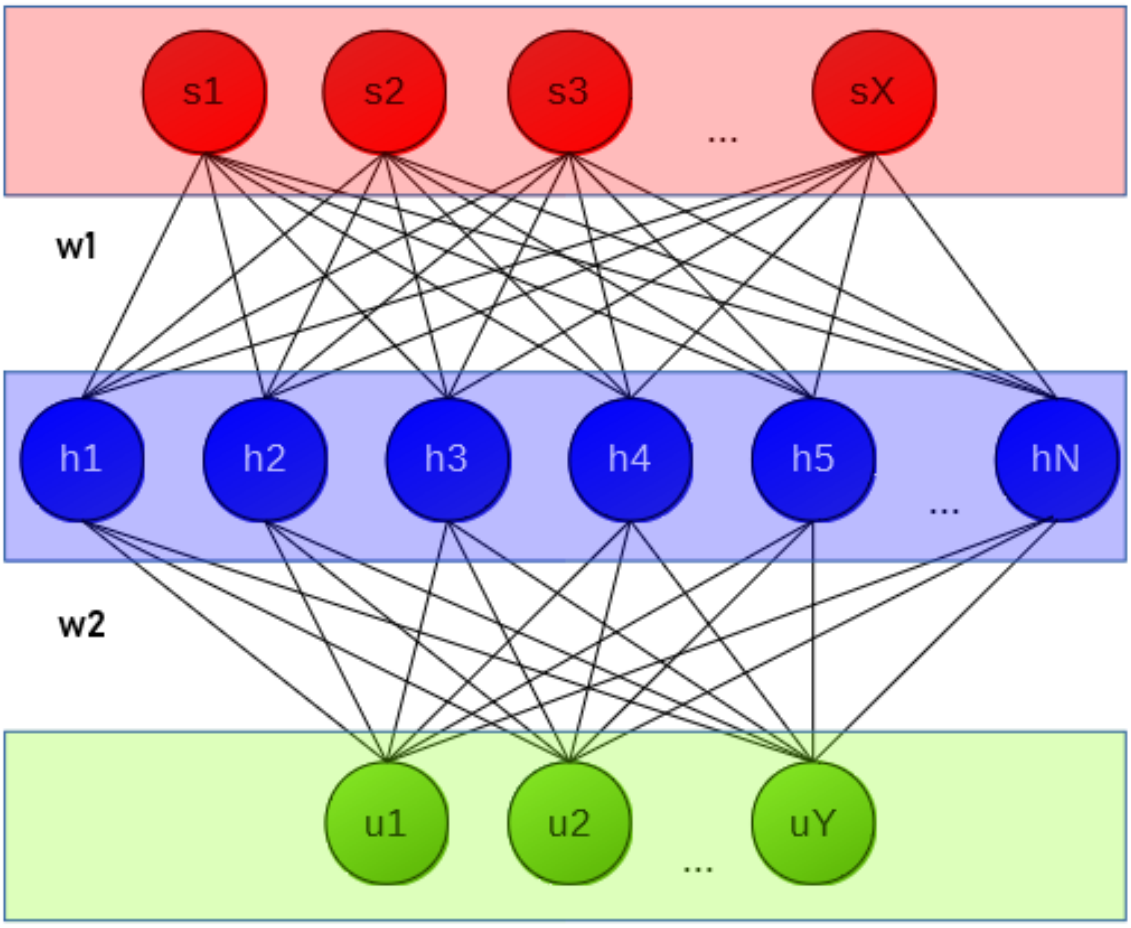
Monikerroksinen perseptroniverkko koostuu nimensä mukaisesti eri kerroksista neuroneja. Kerroksia on kolme: sisääntulokerros, piilotettu kerros ja ulostulokerros. Piilotettuja kerroksia voi olla useita, mutta sisääntulo- ja ulostulokerroksia vain yksi kumpaakin.

Neuroverkolle määritellään sisääntulokerroksen neuronimäärä, joka määrää kuinka suuren määrän tietoa neuroverkko voi ottaa käsiteltäväkseen. Kerroksen jokaisella neuronilla on jokin arvo, jonka pohjalta neuroverkko aloittaa toimintansa. Mark 1 Perseptron käytti 20x20 pikselin valokennoa, eli sisääntuloneuroneita laitteessa oli 400, yksi neuroni jokaista pikseliä kohden. Yksittäisen neuronin arvo määräytyi tässä tapauksessa kuvassa ilmentyvän mustuuden mukaan. Näin verkko pystyi analysoimaan sisääntulevaa tietoa matemaattisesti. Moderneissa neuroverkoissa sisääntulokerros muodostuu yhä samalla tavalla.

Piilotetun kerroksen tehtävä on suorittaa laskutoimituksia ja kuljettaa tietoa sisääntulokerroksesta ulostulokerrokseen. Toisin kuin sisään- ja ulostulokerroksia, piilotettuja kerroksia voi olla useita. Neuronien ja kerrosten määrä on usein ohjelmoijan

arvio tehtävään vaaditusta määrästä, sillä määrän valitsemiseen ei ole tarkkaa matemaattista mallia.

Ulostulokerroksen arvot ovat neuroverkon laskennan lopullinen tulos. Jokainen ulostulokerroksen neuroni käsittelee yhtä mahdollisuutta. Mark 1 Perseptronissa ulostulokerroksessa neuroneita oli 10; yksi neuroni jokaista yksilukuista numeroa kohden. Hypoteettisessa tilanteessa, jossa Mark 1 Perseptron olisi ollut täysin varma siitä, että sille syötetyssä kuvassa on numero 2, olisi sen ulostulokerroksen arvotaulukko näyttänyt $[0, 0, 0.99, 0, 0, 0, 0, 0, 0, 0]$.



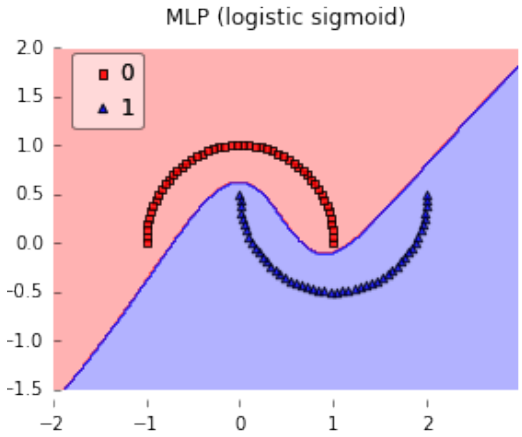
Kuvio 3. Monikerroksinen perseptroniverkko.

Kuvion 3 kaltainen kaavio on yleisin tapa esittää keinotekoisen, monikerroksisen neuroverkon arkkitehtuuria graafisesti. Synapsit liittävät punaisen sisääntulokerroksen jokaisen neuronin jokaiseen sinisen piilotetun kerroksen neuroniin. Sama ta-

pahtuu piilotetun kerroksen neuronien ja ulostulokerroksen neuronien välillä. Neuronien määrät kussakin kerroksessa määräytyvät edellämainituilla tavoilla.

Kuten biologisissa aivoissa, neuronien välillä on kerroksittain yhteyksiä, synapseja, joita kutsutaan keinotekoisista neuroverkoista puhuttaessa yleensä painoiksi (eng. weights). Neuroverkko voidaan opettaa erilaisiin tehtäviin säätelemällä näiden painojen arvoja. Arvojen säätely tapahtuu verkkoa opettamalla, eli syöttämällä verkolle ennaltamerkittyä harjoitusdataa. Verkon arvausta verrataan merkittyyn vastaukseen ja synapsien painoja säädetään arvauksen virheellisyyden perusteella. Toistamalla tätä prosessia suurella määrällä merkittyä harjoitusdataa verkko voidaan opettaa tunnistamaan ja kategorisoimaan sille tuntematonta dataa. Tätä prosessia kutsutaan vastavirta-algoritmiksi (eng. backpropagation).

Vastavirta-algoritmin, ja täten koko neuroverkon toiminnan kannalta tärkeä osa on aktivointifunktio. Aktivointifunktioita on erilaisia, mutta esimerkissä käytetään Sigmoid-aktivointifunktiota. Aktivointifunktion tärkein ominaisuus on sen lineaarittomuus. Täten perseptroni kykenee ratkaisemaan ongelmia, jotka eivät ole lineaarisesti ratkaistavissa, kuten Kuviossa 4 esitetty punaisen ja sinisen kategorian erottaminen omiin luokkiinsa.



Kuvio 4. Toisin kuin Madeline, Adaline ja Mark 1 Perseptroni, modernit neuroverkot kykenevät ratkaisemaan epälineaarisia ongelmia.

Vastavirta-algoritmi voidaan kirjoittaa muodossa

$$\frac{\partial E}{\partial w_{jk}} = -(t_k - o_k) \cdot Sigmoid(\sum_j w_{jk} \cdot o_j)(1 - Sigmoid(\sum_j w_{jk} \cdot o_j)) \cdot o_j \cdot H \tag{2.1}$$

missä t on ulostuloneuronin haluttu ulostulo, o on ulostuloneuronin ulostulo, w on synapsin paino neuronien välillä, k on edeltävän kerroksen neuroni, j on tämän kerroksen neuroni ja H on ohjelmoijan päättämä harjoitusnopeus.

Kaavan osat selitettynä selitettynä auki:

- $\frac{\partial E}{\partial w_{jk}}$ = painon muutos
- $-(t_k - o_k)$ = virhe kerroksen ulostulossa
- $Sigmoid$ = neuroverkkoon valittu aktivointifunktio
- $\sum_j w_{jk} \cdot o_j$ = jokainen tarkasteltavaan ulostuloon vaikuttava ulostulo
- o_j = edellisen kerroksen ulostulo
- H = harjoitusnopeus

Kaavaa sovelletaan neuroverkon jokaisen kerroksen jokaiseen neuroniin jokaisen harjoitusdataelementin läpikäymisen jälkeen. Toteuttaessa kaavaa koneellisesti neuroverkon painot onkin hyvä esittää matriiseilla, eli moniulotteisilla taulukoilla, minkä jälkeen vastavirta-algoritmin käyttö onnistuu helposti synapsien läpi silmukoimalla.

Algoritmin avulla voidaan päivittää neuroverkon painoja. Algoritmia sovelletaan jokaisen iteraation jälkeen jokaiseen verkon painoon. Iteraatioita toistetaan, kunnes ulostulon virhe on riittävän pieni. Tarkemman ohjeen vastavirta-algoritmin toteuttamiseen Python-ohjelmointikielessä voi katsoa esimerkiksi Rashid (2016) kirjasta.

Toteuttaessa neuroverkkoa ohjelmointikielellä tekijä joutuu miettimään seuraavia asioita:

- Synapsit on järkevintä esittää matriiseina ja taulukoina.
- Jokainen synapsipaino täytyy erikseen päivittää. Matriisien läpikäymisen täytyy olla vaivatonta.
- Verkon painojen päivitys onnistuu lineaarialgebraa soveltamalla.

- Neuroverkot käyvät läpi suuret määrät harjoitusdataa. Säikeistämisen pitäisi olla helppoa.

Neuroverkon mielekäs luominen tyhjästä vaatii siis kaksi komponenttia: matriisilaskentaa ja säikeytystä.

3 Tarkasteltavat ohjelmointikielet

Tässä luvussa käsitellään C#- ja Python-ohjelmointikieliä. Tarkasteltavana on sekä kielten perusominaisuudet että saatavilla olevat koneoppimiskirjastot ja tieteelliset laskentakirjastot. Ominaisuuksia ja kirjastoja tarkastellaan nimenomaan neuroverkkojen luomisen kannalta. Kirjastot on valittu helpon löydettävyytensä ja suosionsa perusteella. Käsiteltäväksi valitut kirjastot ovat myös ilmaisia.

3.1 C#

C# on määritelmänsä ECMA-334 (2017) mukaan Microsoftin vuonna 2000 julkaisema, .NET-runkoon kuuluva ohjelmointikieli. ECMA-334 (2017)-standardissa mainitaan, että kielen kehittämisessä tavoitteena on ollut "Yksinkertainen, moderni, yleiskäyttöinen, olio-ohjelmointikieli". C#-kielen ominaisuuksia ovat muun muassa vahva tyypitys, taulukoiden rajojen tarkastaminen, julistamattomien muuttujien käyttöyritysten tarkistaminen ja automaattinen roskienkeruu.

Stackoverflow (2017):n teettämän kyselyn mukaan 33.8% kaikista vastaajista käyttää C#-kieltä jossakin yhteydessä.

Math.NET

Math.NET (2018) on 2009 aloitettu .NET-ympäristöön kehitetty avoimen lähdekoodin tieteellisen laskennan kirjasto. Math.NET on yksi suosituimmista laskentakirjastoista C#-kielelle. Kirjastoa on käytetty useissa projekteissa ja tutkimuksissa. Esimerkiksi Fermisim (2011)-projekti, jonka tavoitteena oli fermin paradoksin mahdollisten ratkaisujen tutkiminen laskennallisen simuloinnin avulla, käytti Math.NET-kirjastoa simulaatioissaan.

Math.NET Numericsista löytyy kaikenkattava luokka lineaarialgebralle. Vektorien ja matriisien luominen on helppoa, ja vektoreille sekä matriiseille löytyy neuroverkoille tärkeä pistetulofunktio.

AForge.NET

AForge.NET (2013) on C#-kielelle kehitetty koneoppimiseen suunnattu .NET-ohjelmistokehys. AForge.NETistä löytyy monipuolinen neuroverkkoluokka. Neuroverkkoluokka muun muassa abstraktoi vastavirta-algoritmin ja aktivointifunktiot funktioiden taakse. AForge.NETillä tämän tutkimuksen esimerkeissä käsitelty monikerroksinen perseptroniverkko on helppo luoda. Koneoppimisen saralla AForge.NET on yksi C#-kielen suosituimmista kirjastoista.

Esimerkkejä AForge.NETin neuroverkkoluokkaa hyödyntävistä projekteista ovat esimerkiksi NeurApp (2017) ja Sinapse (2018).

3.2 Python

Python-ohjelmointikielellä ei ole samankaltaista virallista määrittelyä kuin C#-kielellä. Se on määrittelemätön kieli ja sen lähdekoodi on avoin. Python on monikäyttöinen, korkeatasoinen ja erittäin käyttäjäystävällinen olio-ohjelmointikieli. Helppokäyttöisyytensä vuoksi Pythonia usein suositellaan aloittelijan ensimmäiseksi ohjelmointikieleksi. Toisin kuin C#-kielessä, Pythonissa ei ole vahvaa tyypitystä.

Stackoverflow (2017):n teettämän kyselyn mukaan 31.7% kaikista vastaajista käyttää Python-ohjelmointikieltä jossakin yhteydessä.

NumPy

NumPy (2017) on olennainen kirjasto kaikelle tieteelliselle laskennalle. NumPy on lähtöisin kattavammasta tieteellisestä SciPy-laskentakirjastosta. NumPy hajautettiin omaksi kirjastokseen, sillä ei haluttu, että tehokkaaseen taulukoiden ja matriisien manipuloimiseen tarvitsisi käyttää suurta SciPy-kirjastoa. NumPy on tehokkuutensa, kattavuutensa ja helppokäyttöisyytensä vuoksi varmistanut paikkansa jokaisessa laskentaa vaativassa Python-projektissa.

Neuroverkon luomiseen tyhjästä NumPy on korvaamaton. NumPy sisältää intuitiiviset operaattorit mm. matriisien yhteenlaskuun ja kertomiseen. Kirjastosta löytyy

myös tärkeä pistetulofunktio sekä transponointifunktio. NumPy poistaa tarpeen silmukoida taulukoiden tai matriisien läpi niitä manipuloidessa, mikä nopeuttaa koodin kirjoittamista huomattavasti ja vähentää virheiden mahdollisuutta. Mm. Rashid (2016) käyttää neuroverkossaan ainoastaan NumPya.

TensorFlow

TensorFlow Abadi & Co (2016) on valtavan suosion saavuttanut, Googlen kehittämä koneoppimisen ohjelmistokehys. TensorFlow on pääasiallisesti luotu Pythonille, mutta kirjoittamisen hetkellä TensorFlow tarjoaa rajapinnat myös C++-, Haskell-, Go-, Java- ja Rust-ohjelmointikielille. TensorFlow'ta kehitetään jatkuvasti, ja se on jo käytössä useilla suurilla yrityksillä, kuten Dropboxilla ja Intelillä.

TensorFlow on luultavasti tämän hetken tehokkain ja kattavin koneoppimiskehys. TensorFlow'n omilta sivuilta löytyy valmiit ohjeet mm. monikerroksisen perseptroniverkon luomiseen. Toisin kuin esim. C#-kielen AForge.NET, TensorFlow ei ainoastaan abstraktoi neuroverkon luomiseen tarvittavia tärkeitä komponentteja ja jätä loppua ohjelmoijan tehtäväksi, vaan TensorFlow'lla neuroverkkoja luodessa jokainen askel on mahdollista tehdä TensorFlow'n omilla ominaisuuksilla.

4 Yhteenveto

Tutkimuksen ensisijaisena tavoitteena oli selvittää, miksi suurimmat koneoppimiskirjastot ja koneoppimista käsittelevät kirjat ovat pääasiassa suunnattu Python-ohjelmointikielelle. Toisena tavoitteena oli esitellä neuroverkkojen historiaa ja toimintaa.

Kuten tutkimuksessa mainitaan, Python on muihin kieliin verrattuna monin verroin suositumpi, kun kyse on koneoppimisesta, tässä tapauksessa neuroverkoista. Tutkimuksessa C# toimi esimerkkinä muista kielistä, kun se asetettiin Pythonin kanssa rinnakkain.

Tuloksena on, että Python on suositumpi kieli neuroverkkojen rakentamiseen tyhjästä. Näin on suureksi osaksi NumPy-kirjaston vuoksi. Kuten havaittiin, neuroverkon rakentaminen vaatii paljon lineaarialgebraa. NumPyn ja Pythonin ominaisuuksien ansiosta jopa kompleksin vastavirta-algoritmin toteuttaminen voidaan lyhentää vaivattomasti yhteen riviin. Tämä on käytännössä mahdotonta C#-kielessä. Toki neuroverkon toteuttaminen muilla kielillä ei ole mahdotonta, mutta NumPyn kanssa käytettynä Python tekee siitä helppoa ja nopeaa. Jos kriteerinä olisi, että mitään kirjastoja ei saa käyttää, kielellä ei olisi enää niin paljon väliä. Käytännössä kuitenkin neuroverkkoja ohjelmoidessa on aina lineaarialgebraan erikoistuneet kirjastot apuna.

Pythonin helppokäyttöisyyden ja kirjoitusnopeuden vuoksi teknologiajätti Google valitsi Pythonin koneoppimiskirjasto TensorFlow'n kieleksi. Tämä toi lisää huomiota Pythonille neuroverkkojen kielenä ja houkutteli lisää ohjelmoijia tekemään neuroverkkoja Pythonilla.

Tutkimuksen tulos vastasi odotuksia, mutta samalla kirjoittaja löysi myös uusia näkökulmia. Esimerkiksi esitellyn C#-kielen kirjastot eivät olleet kirjoittajalle entuudestaan tuttuja. Vaikka Math.NET-kirjastoa ei ole yhtä helppo käyttää kuin NumPy-kirjastoa, kirjoittaja aikoo vertailla miten neuroverkkototeutuksen koodi lyhenee käyttämällä Math.NET-kirjastoa.

Tämä tutkimus on yleiskuvaus, ja siinä ei ole otettu huomioon esim. kirjastojen ja kielien performanssieroja. Tutkimuksessa ei myöskään tutkittu Pythonin lisäksi muita kieliä kuin C#-kieltä, vaan C# oli yleistetty edustamaan muita kieliä. Tutkimusta voisi jatkaa ottamalla mukaan lisää ohjelmointikieliä ja tutkimalla niiden kirjastoja.

Kirjallisuutta

Abadi M. & Co. 2016. *TensorFlow: A System for Large-Scale Machine Learning*. Haettu osoitteesta *https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf* [Viitattu 23.4.2018]

AForge.NET. 2013. Haettu osoitteesta *https://code.google.com/archive/p/aforge/* [Viitattu 23.4.2018]

Bishop, C. 2006. *Pattern Recognition and Machine Learning*. Springer Science+Business Media Inc. New York, Yhdysvallat.

Eberhart, R. & Dobbins, R. 1990. *Neural Network PC Tools: A Practical Guide*. Academic Press

ECMA-334. 2017. Haettu osoitteesta *http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf* [Viitattu 23.4.2018]

Fermisim. 2011. Haettu osoitteesta *https://launchpad.net/fermisim* [Viitattu 23.4.2018]

Garson, G. 1998. *Neural Networks: An Introductory Guide for Social Scientists*. SAGE Publications Ltd

Hebb, D. 1949. *The Organization of Behavior: A Neuropsychological Theory*. Haettu osoitteesta *http://s-f-walker.org.uk/pubsebooks/pdfs/The_Organization_of_BehaviorDonald_O._Hebb.pdf* [Viitattu 23.4.2018]

Hopfield, J. 1982. *Neural networks and physical systems with emergent collective computational abilities*. Haettu osoitteesta *https://bi.snu.ac.kr/Courses/g-ai09-2/hopfield82.pdf* [Viitattu 23.4.2018]

Hurwitz, J. & Kirsch, D. 2018. *Machine Learning for dummies*. John Wiley & Sons, Inc.

Math.NET. 2018. Haettu osoitteesta *https://www.mathdotnet.com/* [Viitattu 23.4.2018]

McCulloch, W. & Pitts, W. 1943. *A Logical Calculus Of The Ideas Immanent In Nervous Activity*. Bulletin of Mathematical Biophysics, Volume 5.

Minsky, M. & Papert, S. 1968. *Perceptron*. The MIT Press.

NeurApp. 2017. Haettu osoitteesta *http://www2.arnes.si/%7Eljc3m2/igor/software/NeurApp/* [Viitattu 23.4.2018]

Numpy. 2017. Haettu osoitteesta *http://www.numpy.org/* [Viitattu 23.4.2018]

Olazaran, M. 1993. *A Sociological History of the Neural Network Controversy* teoksessa *Advances in Computers, Nide 37*. Academic Press

Paris D. *Neural Networks for Residential Infrastructure Management* teoksessa Attoh-Okine N. & Ayuub B. (toim) 2005. *Applied Research in Uncertainty Modeling and Analysis*. Springer Science+Business Media Inc. New York, Yhdysvallat

Rashid, T. 2016. *Make Your Own Neural Network*

Rosenblatt, F. 1957. *Report NO. 85-460-1, The Perceptron: A Perceiving And Recognizing Automaton*

Rumelhart, D. & McClelland, J. 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press

Sinapse. 2018. Haettu osoitteesta *https://code.google.com/archive/p/sinapse/* [Viitattu 23.4.2018]

Stackoverflow. 2017. Haettu osoitteesta *Developer Survey Results https://insights.stackoverflow.com/survey/2017* [Viitattu 23.4.2018]

Widrow, B. & Lehr, M. 1993. *Artificial Neural Networks of the Perceptron, Madaline, and Backpropagation Family*.