Markus Lappalainen

KONEOPPIMISTYÖKALUJEN TOIMINTAPERIAATTEET

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2023

TIIVISTELMÄ

Lappalainen, Markus
Koneoppimistyökalujen toimintaperiaatteet
Jyväskylä: Jyväskylän yliopisto, 2023, 28 s.
Tietojärjestelmätiede, kandidaatintutkielma
Ohjaaja(t): Taipalus, Toni

Koneoppimisalgoritmit ja neuroverkot ovat nykyään osa jokapäiväistä elämäämme, ja niiden tuoma kehitys on mullistanut yhteiskunnan montaa osa-aluetta. Nopean kehityksen vuoksi asiaan perehtymätön henkilö tuskin yleensä edes huomaa käyttävänsä koneoppimiseen perustuvia teknologioita. Koneoppiminen, ja varsinkin neuroverkot, ovat nykyään niin monimutkaisia, että niiden toimintaa voi olla vaikeaa, tai jopa mahdotonta ymmärtää. Tutkimus toteutettiin kuvailevana kirjallisuuskatsauksena. Tutkimuksessa annetaan ensin lyhyt yleiskatsaus koneoppimisesta lukijan ymmärryksen tueksi. Tutkimuksen päätarkoitus, sekä tutkimusongelma, oli selvittää koneoppimistyökalujen toimintaperiaatteet. Tutkimusongelmaa selvitettiin tutustumalla koneoppimistyökalujen toimintaan, etenkin niiden opetusvaiheeseen, jonka aikana luodaan edellytykset niiden päätöksenteolle. Tutkimuksessa selvitettiin joidenkin suosittujen koneoppimistyökalujen toimintaperiaatteita, sekä esitellään ne helppolukuisessa ja helposti ymmärrettävässä muodossa, ilman matemaattisia kaavoja. Koneoppimistyökalujen perustoimintaperiaatteeksi löydettiin virhefunktiot, ja niiden minimoiminen. Virhefunktiot esittävät koneoppimistyökalun ennusteen ja toteutuneen tapahtuman välistä eroa, joten virhefunktion minimoiminen on koneoppimisen ydintavoite. Keinot virhefunktioiden minimoimiseksi riippuu käsiteltävän koneoppimistyökalun piirteistä. Tarkasteltaessa valittujen koneoppimistyökalujen optimointiongelmia, paljastui yleisimmäksi keinoksi gradienttimenetelmään perustuvat iteratiiviset optimointialgoritmit. Tutkimuksen aikana löytyi myös muita optimointiongelmia, joita ei pystytä ratkaisemaan gradienttimenetelmällä. Tutkimuksen aikana selvisi myös koneoppimisen perustuvan vahvasti matematiikkaan, erityisesti lineaarialgebraan sekä derivointiin.

Asiasanat: Koneoppiminen, koneoppimisalgoritmit, neuroverkot, tekoäly, optimointi.

ABSTRACT

Lappalainen, Markus
The operating principles of machine learning tools
Jyväskylä: University of Jyväskylä, 2023, 28 pp.
Information Systems, Bachelor's Thesis
Supervisor(s): Taipalus, Toni

Machine learning algorithms and neural networks are a ubiquitous part of our everyday lives, and their recent development has revolutionized many areas of society. Due to the rapid pace of development, a person who is not familiar with the subject may not even realize that they are using technologies based on machine learning. Machine learning, especially neural networks, are now so complex that understanding the logic behind their decision can sometimes be impossible. The study was conducted as a descriptive literature review. The study begins by providing a brief overview of machine learning to support the reader's understanding. The main objective of the study, and the research question, was to clarify the basic principles of machine learning tools. This was done by examining the operation of machine learning tools, particularly their training phase, during which the conditions for their decision-making are established. The study examined the operating principles of some popular machine learning tools and presented them in an easy-to-understand form, without mathematical formulas. The basic operating principle of machine learning tools was found to be cost functions and their minimization. Cost functions measure the difference between the prediction of the machine learning tool and the actual outcome, so minimizing the cost function can be seen as the primary goal of machine learning. The method for minimizing a cost function depends on the characteristics of the machine learning tool being used. When examining the optimization methods of the studied machine learning tools, iterative optimization algorithms based on the gradient descent algorithm were found to be the most common approach. The study also identified other optimization problems that cannot be solved by the gradient descent algorithm. At the time of writing the study, it became clear that machine learning is heavily based on mathematics, especially linear algebra, and differentiation.

Keywords: Machine learning, machine learning algorithms, neural networks, artificial intelligence, optimization.

KUVIOT

SISÄLLYS

1 JOHDANTO

Viime vuosien aikana koneoppimisesta on tullut arkielämässä jatkuvasti läsnä oleva voima, joka on mahdollistanut monenlaista teknologista kehitystä lähes jokaisella elämämme osa-alueella. Kehitystä on tapahtunut esimerkiksi itseohjautuvissa ajoneuvoissa, luonnollisen kielen käsittelyssä, hahmontunnistuksessa, sekä lukemattomissa muissa teknologioissa. Koneoppimisen tutkimuksessa on tällä hetkellä käynnissä suuri buumi, jonka vetävinä voimina toimii erityisesti käytettävissä olevan datan eksponentiaalinen kasvu, laskentatehon helppo saatavuus, sekä viimeaikaiset edistykset alalla, kuten syvät neuroverkot.

Koneoppiminen onkin tällä hetkellä erittäin aktiivisesti tutkittu ala, ja trendi ei vaikuta olevan hidastumaan päin, vaan sitä pidetään yleisesti yhtenä lupaavimmista tietotekniikan tutkimusalueista. Koneoppimisen tämänhetkinen tutkimustyö kohdistuu pääasiassa uusiin teknologioihin, kuten syviin neuroverkkoihin. Neuroverkot eivät kuitenkaan ainakaan vielä pysty täysin syrjäyttämään perinteisiä koneoppimistyökaluja. Useissa tuoreissa tutkimuksissa todetaan perinteisten koneoppimisalgoritmien päihittävän neuroverkot tarkkuudellaan, nopeudellaan, sekä tarvittavan harjoitusdatan määrän vähyydellä. Onkin siis tärkeää, että tutkimuksessa ei sokaistuta uusista löydöksistä ja käsitellä vain uusimpia teknologioita, vaan otetaan huomioon myös vanhojen teknologioiden vahvuudet.

Tässä kandidaatintutkielmassa tutustutaan ensin lyhyesti koneoppimiseen yleisesti, selvittämällä sen taustaa sekä nykytilaa. Yleiskatsauksessa myös esitellään koneoppimisen eri tyypit, joiden perusteella tutkielman rakenne muodostuu. Yleiskatsauksen jälkeen paneudutaan itse tutkimusongelmaan, eli selvitetään koneoppimistyökalujen toimintaperiaatteet. Tutkimusongelmaa selvitetään tutustumalla perinteisten koneoppimistyökalujen toimintaan, ja selvittämällä miten nämä työkalut päätyvät ratkaisuihinsa. Perinteisten koneoppimisalgoritmien lisäksi käsitellään neuroverkkojen toimintaa pilkkomalla niiden rakenne osiin. Tämän jälkeen tarkastellaan miten löydetyt osat muodostavat kokonaisuuden, joka lopulta kykenee suoriutumaan sille annetuista tehtävistä.

Tutkimus toteutetaan kuvailevana kirjallisuuskatsauksena. Menetelmä valikoitui sen vapaamuotoisuuden vuoksi, joka helpottaa täysin uuteen aiheeseen

tutustumista. Tutkimusongelman laajuus ei ollut ennen tutkimuksen aloittamista täysin tiedossa, joka myös puolsi menetelmän valintaa, sillä laajuutta oli helppo rajata myös tutkimuksen edetessä. Tutkimuksessa käytetty kirjallisuus koostuu pääasiassa tieteellisistä vertaisarvioiduista julkaisuista. Joidenkin koneoppismistyökalujen toimintaa selvennettiin myös teknisten oppaiden avulla. Kirjallisuutta valittaessa kriteereinä käytettiin muun muassa julkaisujen tuoretta ikää, sillä käsiteltävä aihe on jatkuvassa murroksessa nykyisen buumin vuoksi. Joitakin vanhempia teoksia käytettiin selventämään koneoppimisen perusteita, sillä tuoreemmassa kirjallisuudessa jotkin asiat koettiin itsestään selviksi, jolloin niitä ei välttämättä avattu riittävän tarkasti. Muita kriteereitä oli esimerkiksi luotettavat julkaisualustat sekä viittausten määrä. Joissakin valinnoissa edellä mainituista kriteereistä on saatettu joustaa, jos teoksen tekijät ovat tieteenalalla hyvin tunnettuja, tai teos on jollain tapaa kytköksissä luottamusta herättävään instituutioon.

Ennen kun aletaan käsittelemään itse koneoppimista, on hyvä vielä esitellä lyhyesti joitakin tärkeitä käsitteitä, jotka eivät välttämättä ole itsestään selviä. Koneoppimisen käsitteistössä on paljon synonyymeja, ja näille käsitteille voi olla useita käännöksiä, joten tässä vaiheessa on hyvä selventää tutkimuksessa käytettyjen termien merkitykset.

Dataobjekti

Dataobjektit ovat yksittäisiä esiintymiä datassa. Esimerkiksi jos käsiteltävä data koostuu henkilöistä, on jokainen henkilö yksittäinen dataobjekti.

Attribuutti

Attribuutit ovat dataobjektien muuttujia. Esimerkiksi henkilön attribuutteja voisi olla sukupuoli, ikä tai pituus. Attribuutteja kutsutaan myös datan ulottuvuuksiksi tai muuttujiksi.

Virhefunktio

Virhefunktio on koneoppimisen kontekstissa mekanismi, joka selvittää kuinka suuri koneoppijan tekemän ennusteen ja todellisen tuloksen välinen ero on. Koneoppijan tehtävä on lähtökohtaisesti aina minimoida virhefunktio, esimerkiksi gradienttimenetelmällä.

Gradienttimenetelmä

Gradienttimenetelmä on yksi yleisimpiä keinoja virhefunktion minimoimiseksi koneoppimisessa. Gradienttimenetelmässä etsitään painokertoimia, joilla virhefunktio saadaan minimoitua. Käytännössä gradienttimenetelmässä käydään iteratiivisesti läpi virhefunktion derivaattoja eri painokertoimilla, kunnes virhefunktio ei enää pienene (Shrestha & Mahmood, 2019).

2 KONEOPPIMINEN YLEISESTI

Koneoppiminen voidaan määritellä yksikertaisesti koneiden opettamiseksi oppimaan, jonka jälkeen ne pystyvät suoriutumaan tietyistä tehtävistä autonomisesti (Coelho & Richert, 2015). Koneoppimisen olennaisin hyöty on vähentää ihmisen työtaakkaa luomalla analyyttisiä malleja automattisesti, jotka ilman koneoppimista jouduttaisiin ohjelmoimaan erikseen vastaamaan jokaisen käyttökohteen erityispiirteitä (Janiesch, Zschech & Heinrich, 2021). Koneoppiminen ei ole uusi tieteenala, vaan termi koneoppiminen tuli tutuksi ensimmäisen kerran jo vuonna 1959, jolloin IBM:n Arthur Samuel todisti tutkimuksellaan sen, että koneoppimisohjelma pystyy itseoppimaan ja päihittämään tekijänsä tammi-lautapelissä (Samuel, 1959).

Koneoppimisen ytimessä on käsiteltävä data. Aiemmin koneoppimiselle oli selkeät lähtökohdat ja tavoitteet, kuten lautapelien selkeät säännöt ja niissä voittaminen, tai yksinkertaisten syy-seuraussuhteiden osoittaminen. Nykyään dataa kerääntyy huomattavasti aiempaa enemmän, eikä sen tulkinta tai hyödyntäminen välttämättä ole yhtä yksinkertaista.

2.1 Massadata

Kun käsitellään nykyaikaista koneoppimista, on tärkeä ottaa huomioon massadata, sillä nyky-yhteiskunta tuottaa enenevissä määrin massadatan muottiin sopivaa dataa. Yleisesti massadatalla tarkoitetaan suurta määrää kerättyä digitaalista tietoa. Massadatalle on useita erilaisia määritelmiä, mutta massadatan tärkeimmät piirteet ovat sen suuri määrä, monipuolisuus, vaihtelevuus, sekä nopeus.

Katal, Wazid ja Goudar (2013) esittelevät nämä piirteet seuraavanlaisesti: Massadatan määrällä he tarkoittavat saatavilla olevan datan määrää, jonka kasvu on nykyään erittäin nopeaa, erityisesti sosiaalisen median vuoksi. Monipuolisuudella he tarkoittavat datan olevan monenlaista, ja monessa eri muodossa Erilaista dataa saadaan esimerkiksi sosiaalisesta mediasta, lokitiedoista, sähköposteista,

sensoridatasta, sekä monesta muusta lähteestä. Vaihtelevuudella Katal ja kumppanit tarkoittavat datan tuotannon vaihtelua eri aikoina. Esimerkiksi erilaiset tapahtumat voivat aiheuttaa piikkejä datantuotannossa sosiaalisessa mediassa. Nopeudella he taas tarkoittavat sitä, että data liikkuu jatkuvasti ja nopeasti. Esimerkiksi sensoridata liikkuu jatkuvasti sitä varastoivaan tietokantaan, mutta datan määrän takia ei välttämättä pystytä olettamaan, että data pysyy paikallaan tarpeeksi kauan, jotta sitä pystyttäisiin käsittelemään (Katal ym., 2013). Näiden piirteiden perustella massadataa kuvaillaan sellaiseksi dataksi, jota on vaikeaa tai mahdotonta käsitellä tai hyödyntää tehokkaasti perinteisten työkalujen tai menetelmien avulla (Chen & Lin, 2014).

Chenin ja Linin huomio on koneoppimisen kannalta merkittävä, sillä se kiteyttää koneoppimisen kehittämisen ja tutkimisen tärkeyden. Massadata onkin koneoppimisen tulevaisuuden keskiössä, sillä modernit koneoppimistyökalut pystyvät käsittelemään massiivisia datamääriä. Kun käytössä on kehittyneitä moderneja koneoppimistyökaluja, kerätty massadata ei mene hukkaan, vaan sitä voidaan hyödyntää lähes jokaisella yhteiskunnan alalla. Koneoppimisesta onkin tullut jälleen yksi IT-alan kuumimpia aiheita juuri massadatan vuoksi.

2.2 Koneoppimisen jaottelu

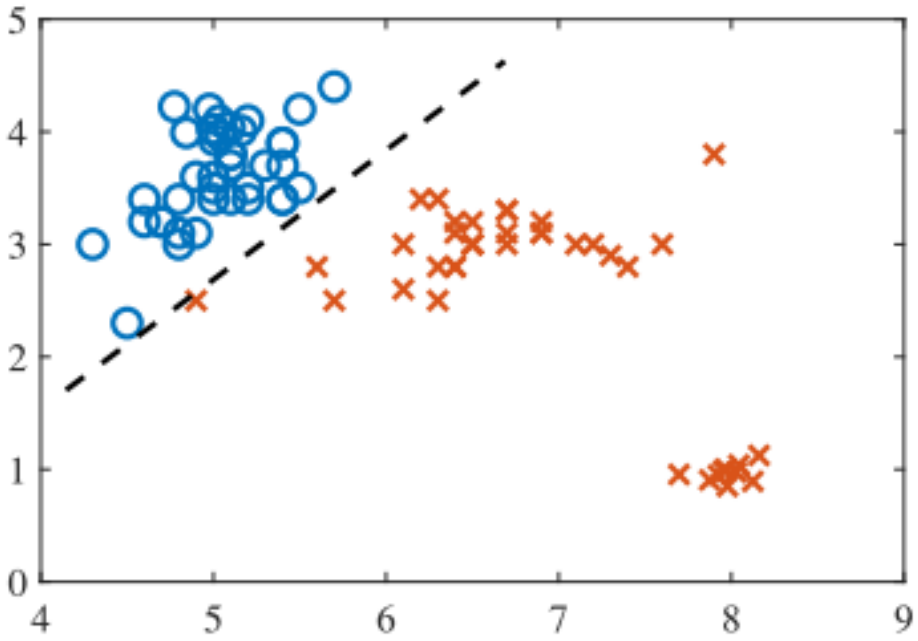
Koneoppimisen eri tyyppien välillä on suuria eroja, esimerkiksi massadatan ominaisuuksien vuoksi sitä ei pystytä käsittelemään tehokkaasti perinteisin koneoppimistyökaluin, vaan sen käsittelemiseen tarvitaan huomattavasti tehokkaampia ja monimutkaisempia keinoja, kuten syviä neuroverkkoja (Chen & Lin, 2014). Simeonen (2018) artikkelin mukaan yhteistä kaikille koneoppimislajeille on se, että ne pyrkivät automatisoimaan data-analyyttisen mallin luomisen käyttämällä tarpeeksi suurta määrää harjoitusdataa. Harjoitusdatan perusteella kone rakentaa mallin, jonka avulla se suoriutuu sille annetusta tehtävästä itsenäisesti (Simeone, 2018).

Tässä tutkimuksessa koneoppiminen jaetaan neljään kategoriaan: perinteisen koneoppimiseen alalajeihin, eli ohjattuun-, ohjaamattomaan-, ja vahvistusoppimiseen, sekä koneoppimisen alahaaraan, neuroverkkoihin. Neuroverkkojen voidaan katsoa yhdistelevän perinteisen koneoppimisen alalajeja, joten koen niiden erottelun omaksi kokonaisuudekseen järkeväksi.

2.2.1 Perinteinen koneoppiminen

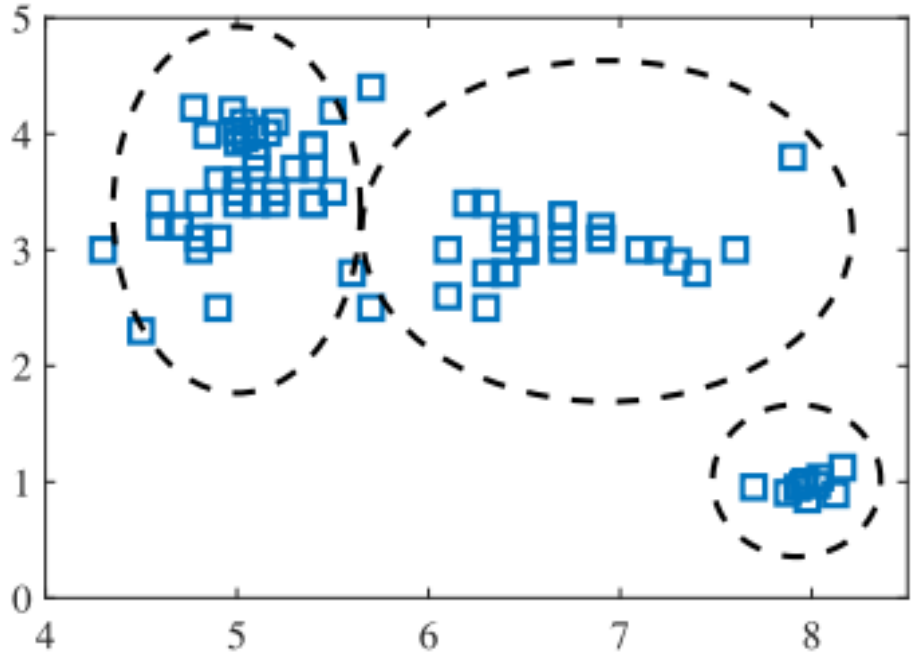
Perinteinen koneoppiminen voidaan jakaa kolmeen eri tyyppiin: Ohjattuun oppimiseen, ohjaamattomaan oppimiseen, sekä vahvistusoppimiseen (Janiesch ym., 2021). Tämä jaottelu perustuu algoritmien oppimisvaiheeseen, eli siihen minkälaista harjoitusdataa tai ohjausta algoritmit tarvitsevat, ennen kuin ne ovat valmiita suoriutumaan annetusta tehtävästä autonomisesti. Jaottelu myös erittelee algoritmien käyttötarkoituksen, esimerkiksi luokitteluun, regressioon tai klusterointiin.

Ohjattu oppiminen on koneoppimisen yksinkertaisin muoto. Ohjatussa oppimisessa analyyttinen malli luodaan käyttämällä harjoitusdatana pareja, jotka koostuvat dataobjekteista ja niitä vastaavista tulosteista (Simeone, 2018). Yksinkertaisimmillaan harjoitusdatan dataobjektit voivat olla yksittäisiä arvoja, ja tulosteena on näitä arvoja vastaavat luokat, jolloin kyseessä on luokittelualgoritmi (Janiesch ym., 2021). Yksi ohjatun oppimisen käyttötapa on luokittelu, kuten kuvion 1 esimerkki, jossa harjoitusdata koostuu syötteenä olevista koordinaateista, ja tulosteena harjoitusdatassa on näitä vastaavat luokat, eli ympyrät ja rastit (Simeone, 2018). Tässä esimerkissä koneoppimisalgoritmi luo harjoitusdatan avulla matemaattisen funktion, eli kuviossa 1 esitetyn katkoviivan, jonka alapuolelle sijoittuvat syötteet saavat luokakseen rastin ja yläpuolelle sijoittuvat syötteet saavat luokakseen ympyrän.



KUVIO 1 Ohjattu oppiminen (Simeone, 2018, s. 2)

Ohjaamattomassa oppimisessa harjoitusdata koostuu pelkästään dataobjekteista, joita ei ole luokiteltu valmiiksi, eikä niille ole odotettuja tuloksia (Simeone, 2018). Ohjaamattomassa oppimisessa etsittyä tulosta ei siis tarvitse tietää ennalta, vaan algoritmin on tarkoitus löytää datasta esimerkiksi mielenkiintoisia yhteyksiä tai poikkeavuuksia. Esimerkiksi klusteroinnissa datasta etsitään klustereita, joissa dataobjektien ominaisuudet ovat lähellä toisiaan (Janiesch ym., 2021). Esimerkiksi kuviossa 2 on käytetty samaa harjoitusdataa kuin edellisessä esimerkissä, mutta luokittelualgoritmin sijaan on käytetty klusterointialgoritmia, joka on löytänyt datasta kolme eri klusteria, jotka on merkitty kuviossa 2 katkoviivoin.



KUVIO 2 Ohjaamaton oppiminen (Simeone, 2018, s. 2)

Vahvistusoppiminen voidaan käsittää ohjatun ja ohjaamattoman oppimisen välimuodoksi, sillä vahvistusoppimisessa on samoja piirteitä kuin ohjaamattomassa oppimisessa, mutta sen toimintaa ohjataan palautteen avulla (Simeone, 2018). Vahvistusoppimisessa koneoppimisalgoritmille annetaan harjoitusvaiheessa järjestelmän nykytila, tavoitetila, sekä lista sallituista toiminnoista, jonka jälkeen algoritmi etsii parhaimman ratkaisun kokeilemalla eri keinoja päätyä tavoitetilaan, ja valitsee lopulta sen keinon, mikä saa parhaimman palautteen (Janiesch ym., 2021). Vahvistusoppimisen käyttökohteita on esimerkiksi videopelien tekoälyn kehittäminen, sekä kamerasyötteiden hyödyntäminen robotiikassa (Arulkumaran, Deisenroth, Brundage & Bharath 2017).

2.2.2 Neuroverkot

Perinteistä koneoppimista ajankohtaisempi aihe on neuroverkot, jotka ovat tämänhetkisen koneoppimistrendin keskiössä. Vaikka neuroverkot ovat nousseet uudeksi trendiksi, ei idea neuroverkoista ole uusi. Neuroverkkoja on montaa eri tyyppiä, ja aihe onkin vähintään yhtä laaja, kuin perinteinen koneoppiminen.

Shrestan ja Mahmoodin (2019) mukaan neuroverkot ovat koneoppimistekniikoita, jotka ovat saaneet nimensä ihmisaivojen mukaan, ja joiden toimintamekanismi perustuu löyhästi ihmisaivojen tapaan käsitellä signaaleja. Artikkelissa kerrotaan yleisesti neuroverkkojen olevan syötekerrokseen, piilokerrokseen, sekä tuloskerrokseen jaettujen laskentayksiköiden, eli neuronien, muodostama verkko. Syötekerros nimensä mukaisesti syöttää käsiteltävän tiedon piilokerrokseen, jossa neuroverkon laskenta tapahtuu. Piilokerroksen laskennan tulos

lähetetään tuloskerrokseen, joka tekee lopullisen ratkaisun saamiensa tuloksien perusteella. (Shrestha & Mahmood, 2019)

Edellisessä kappaleessa esimerkkinä toiminut kolme kerrosta käsittävä neuroverkko on kuitenkin hyvin yksinkertainen, ja syvät neuroverkot pystyvät huomattavasti monimutkaisempiin tehtäviin. Syvät neuroverkot ovat neuroverkkoja, jotka koostuvat useammasta kuin yhdestä piilokerroksesta (Mohammadi, Al-Fuqaha, Sorour & Guizani, 2018). Syvän neuroverkon kerrosten ja neuronien perustoimintaperiaate on sama, kuin yksinkertaisten neuroverkkojen. Syvien neuroverkkojen vahvuus on niiden kyky käsitellä suuria määriä monipuolista dataa, jonka vuoksi ne soveltuvat erinomaisesti tehtäviin, jotka vaativat ongelman pilkkomista osiin (Chen, Challita, Saad, Yin & Debbah, 2019). Tästä voikin huomata, että syvät neuroverkot vastaavat massadatan tuomiin haasteisiin, kuten datan monipuolisuuteen ja sen suureen määrään.

Syvien neuroverkkojen käyttöä kutsutaan usein koneoppimisen sijaan syväoppimiseksi, mutta syväoppiminen lukeutuu silti koneoppisen piiriin. Neuroverkkojen oppimisvaiheeseen liittyy myös käsittelemäni ohjattu-, ohjaamaton- ja vahvistusoppiminen, mutta neuroverkkojen monipuolisuuden vuoksi yksi neuroverkko voi käyttää ja yhdistellä useita eri oppimistapoja (Janiesch ym., 2021).

3 KONEOPPIMISTYÖKALUT

Perinteiset koneoppimisalgoritmit eivät suinkaan ole vanhentunutta teknologiaa neuroverkkojen ilmaantumisen takia, vaan ne voivat monessa tapauksessa suoriutua paremmin, kuin neuroverkot. Esimerkiksi Borisov ja kanssatutkijat (2022) havaitsivat tutkimuksessaan koneoppimisalgoritmien suoriutuvan neuroverkkoja paremmin, kun käsiteltävä data oli taulukoitua. Tutkimuksessa kuitenkin ilmeni yksi koneoppimisalgoritmien heikkous, sillä erittäin suurta datamäärää käsiteltäessä neuroverkot alkoivat suoriutumaan perinteisiä algoritmeja paremmin, vaikka data oli taulukoitua (Borisov ym., 2022). Zhang ja Ling (2018) taas todistivat tutkimuksellaan sen, että koneoppimisalgoritmeilla voidaan saada pienilläkin määrillä opetusdataa melko tarkkoja tuloksia.

Johtopäätöksenä näistä kahdesta tutkimuksesta voi todeta, että perinteiset koneoppimisalgoritmit ovat neuroverkkoja parempi vaihtoehto ainakin silloin, kun käsiteltävä datamäärä on riittävän pieni, sekä käsiteltävä data on sopivassa muodossa. Toinen tärkeä etu perinteisissä algoritmeissa on niiden ymmärrettävyys, sillä ne ovat usein tarpeeksi yksinkertaisia, jotta ihminen pystyy ymmärtämään miten tulokseen on päädytty (Janiesch ym., 2021). Kyky ymmärtää logiikka algoritmin päätöksen taustalla voi olla tärkeää varsinkin silloin, kun väärä johtopäätös voi aiheuttaa mittavaa haittaa, esimerkiksi lääketieteessä tai kriittisten riskien hallinnassa liiketoiminnassa.

3.1 Ohjattu oppiminen

Ohjatussa oppimisessa harjoitusdata koostuu syötteistä, ja niitä vastaavista tulosteista, joiden avulla luodaan malleja, joilla ratkaistaan luokittelu- ja regressio-ongelmia (Simeone, 2018). Artikkelin mukaan regressio-ongelmissa etsitään syötteitä vastaavia tuloksia, jotka ovat jatkuvia lukuja. Vastaavasti luokitteluongelmissa etsittävät tulokset ovat epäjatkuvia, eli niiden tulokset voivat olla vain tiettyjä arvoja, esimerkiksi kyllä tai ei. Simeone summaa artikkelissaan ohjatun

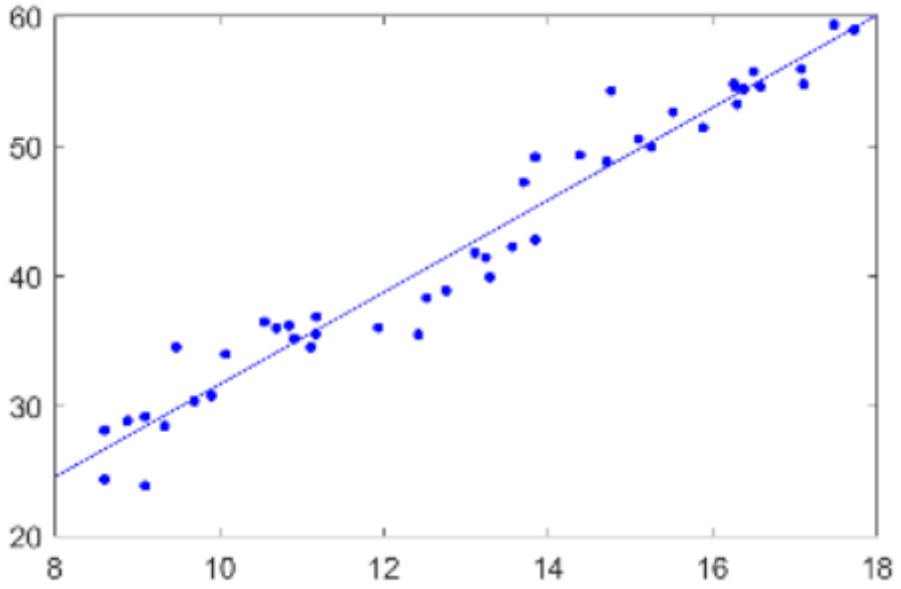
oppimisen tavoitteeksi luoda harjoitusdatan perusteella ennustava malli, jonka avulla pystytään ennustamaan syötteitä vastaavia tuloksia (Simeone, 2018).

Yksinkertainen esimerkki regressio-ongelmasta voisi olla jonkin tuotteen hinnan arvioimista yhden attribuutin mukaan. Harjoitusdatan syötteenä voisi olla asuntojen etäisyys kaupungin keskustasta, ja tätä vastaavana tulosteena näiden asuntojen neliöhinta. Harjoitusdatasta muodostetun mallin avulla pyrittäisiin ennustamaan muiden asuntojen hintaa. Luokitteluongelmassa voitaisiin pyrkiä esimerkiksi arvioimaan asunnon soveltuvuutta henkilölle, jolla ei ole käytössään autoa. Tällöin harjoitusdatan syötteenä voisi olla etäisyys lähimmälle linja-autopysäkille tai metroasemalle, ja vastaavana tulosteena olisi kyllä tai ei, perustuen harjoitusdatassa oleviin aiempien asunnonhakijoiden vastauksiin. Nämä esimerkit edellyttäisivät asunnon hinnan tai soveltuvuuden perustuvan vain yhteen attribuuttiin, joka harvemmin toteutuu oikean maailman käyttötapauksissa.

Ohjatun oppimisen ongelmia voidaan ratkoa monin eri algoritmein, joista jotkin soveltuvat vain regressio- tai luokitteluongelmien ratkomiseen, kun taas joillain algoritmeilla pystytään ratkomaan molempia. Eri algoritmit soveltuvat tiettyihin ongelmiin toisia paremmin, tai niiden suorituskyky voi riippua esimerkiksi käsiteltävän datan määrästä.

3.1.1 Lineaarinen regressio

Lineaarinen regressio on yleinen tapa regressio-ongelmien ratkaisemiseksi. Yksinkertaisimmillaan siinä luodaan malli, jonka mukaan syötteen ja tulosteen suhde on johdonmukainen, jolloin suhdetta voidaan esittää suorana viivana, eli hypoteesifunktiona (Shanthamallu, Spanias, Tepedelenlioglu & Stanley, 2017). Koneoppimisen tehtävä lineaarisessa regressiossa on etsiä hypoteesifunktiolle vakiotermi sekä kulmakerroin, joilla hypoteesifunktio, eli ennustefunktio, on mahdollisimman tarkka (Rustam ym., 2020). Käytännössä tämä tarkoittaa virhefunktion minimoimista, eli koneoppija etsii vakiotermille ja kulmakertoimelle arvot joilla ero arvioiden ja todellisten arvojen välillä ovat keskimäärin pienimmillään (Rustam ym., 2020). Keren He ja Cuiwei He (2021) esittävät virhefunktioksi harjoitusdatan ja ennusteen keskineliösummien välisen erotuksen, eli keskimääräisen etäisyyden hypoteesifunktion ja dataobjektien välillä. Heidän artikkelin mukaan tämän virhefunktion minimoimiseksi voidaan käyttää gradienttimenetelmää (He & He, 2021). Lopputuloksena on kuvion 3 kaltainen hypoteesifunktio, jonka avulla voidaan ennustaa tulevia syötteitä vastaavat tulokset.



KUVIO 3 Yhden attribuutin lineaariregressio (Shanthamallu ym., 2020, s. 2)

Usein regressio-ongelmissa kuitenkin käsitellään sellaista dataa, joissa tulokseen vaikuttaa useampi kuin yksi attribuutti. Vaikka attribuutteja on useita, voidaan ongelman ratkaisemiseksi silti käyttää lineaarista regressiota. Kun tulokseen vaikuttaa useampi kuin yksi attribuutti, tarvitsee koneoppijan selvittää vakiotermin ja kulmakertoinen lisäksi myös eri attribuuttien painokertoimet, jonka jälkeen tuloksen arviointiin käytetään painotettujen syötteiden lineaarikombinaatiota (Shanthamallu ym., 2017). Vaihtoehtoisesti datan attribuutteja voidaan ennen käsittelyä vähentää, esimerkiksi myöhemmin käsiteltävän pääkomponenttianalyysin avulla.

3.1.2 Logistinen regressio

Logistista regressiota käytetään luokitteluongelmien ratkaisuun, sillä siis ennustetaan jatkuvien lukujen sijaan diskreettejä tuloksia. Maalouf (2011) kertoo artikkelissaan luokittelun tapahtuvan arvioimalla todennäköisyyttä sille, että objekti kuuluu tarkasteltavaan luokkaan. Käytännössä todennäköisyys on jatkuva luku välillä 0–1, mutta luokitteluongelmassa objekti joko kuuluu tai ei kuulu luokkaan, joten todennäköisyyden ollessa alle 50 %, objektin ei arvioida kuuluvan luokkaan, ja vastaavasti sen ollessa yli 50 %, arvioidaan sen kuuluvan luokkaan (Maalouf, 2011). Vaikka lopputulos on binäärinen, voidaan logistisen regression avulla luokitella objekteja useampaan kuin kahteen luokkaan. Shanthamallun ja kumppaneiden (2017) artikkelissa mainitaan yksi vastaan kaikki toteutus, jossa jokaiselle luokalle tehdään oma luokittelija, ja objekti luokitellaan korkeimman todennäköisyyden perusteella.

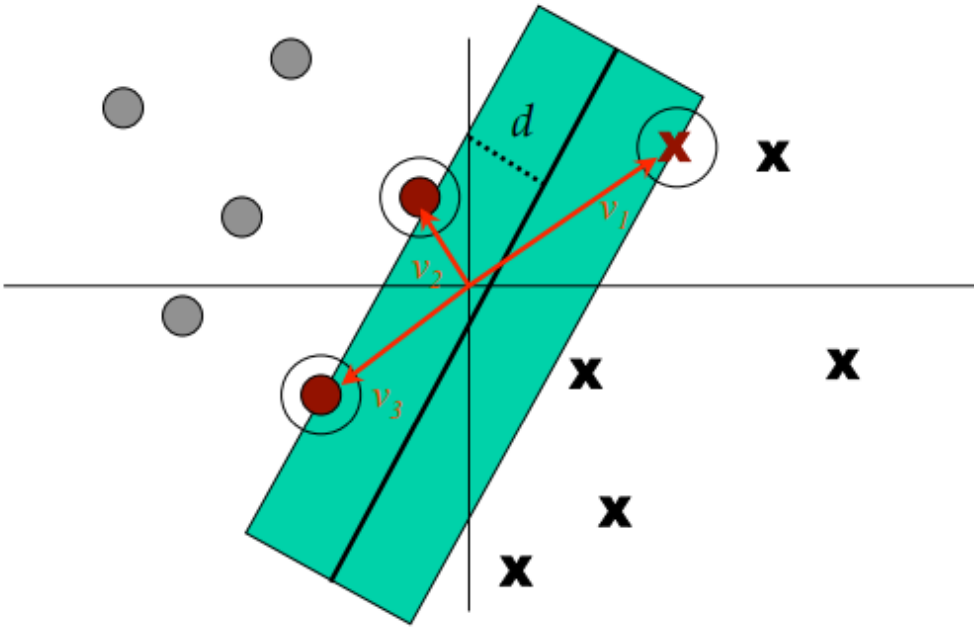
Keren He ja Cuiwei He (2021) kuvailevat tutkimuksessaan logistisen regression toimintaa näin: Aluksi tulokseen vaikuttaville parametreille määritetään painokertoimet, ja näiden parametrien painotettu summa vastaa sitä kuinka

suurella todennäköisyydellä objekti kuuluu tarkasteltavaan luokkaan. Artikkelissa huomautetaan arvioinnin kohteen olevan todennäköisyys, jonka takia tulokset muutetaan logistisen funktion avulla välille 0–1, eli välille ei tapahdu koskaan - tapahtuu aina. Artikkelin mukaan koneoppijan tehtävä logistisessa regressiossa on löytää parametreille painokertoimet, joilla todennäköisyysarviot vastaavat mahdollisimman hyvin toteutuneita tapauksia. Artikkelissa kerrotaan optimoinnin, eli virhefunktion minimoimisen, tapahtuvan gradienttimenetelmän avulla. Ennen kuin gradienttimenetelmää voidaan käyttää, muutetaan optimointiongelma konveksiksi logaritmisen uskottavuusfunktion avulla. (He & He, 2021)

3.1.3 Tukivektorikone

Tukivektorikoneita käytetään pääasiassa luokitteluongelmien ratkaisemiseksi, mutta ne kykenevät myös ratkomaan regressio-ongelmia. Tukivektorikoneista käsittelen vain niiden ensimmäistä ja yksinkertaisinta mallia perusidean ymmärtämiseksi. Mammone, Turchi ja Christiani (2009) kertovat artikkelissaan tukivektorikoneen yksinkertaisimmaksi käyttötavaksi datan lineaarisen luokittelun. Artikkelin mukaan tukivektorikoneen ydintehtävä on tällöin löytää hypertaso, joka erottaa luokat toisistaan mahdollisimman selkeästi (Mammone ym., 2009). Hypertaso on aliavaruus, jolla on yksi ulottuvuus vähemmän, kuin mitä käsiteltävällä datalla on, eli kaksiulotteisessa koordinaatistossa hypertaso on suora viiva (Shanthamallu ym., 2017).

Berwickin (2003) mukaan kaksiulotteisessa ongelmassa koneoppijan tehtävä on siis sijoittaa luokkien väliin suora, ja maksimoida suoran etäisyys lähimmistä dataobjekteista. Tällöin lopputuloksessa hypertaso erottaa luokat siten, että hypertason molemmilla puolilla kulkee yhdensuuntaiset viivat, jotka koskettavat lähimpiä dataobjekteja (Berwick, 2003). Itse tukivektorit, joiden mukaan algoritmi on saanut nimensä, piirtyvät niiden dataobjektien välille, jotka ovat lähimpänä hypertasoa (Kuvio 4). Koneoppijan ratkaistavaksi jää yhtälörajoittunut optimointiongelma, joka voidaan ratkaista Lagrangen kertoimien avulla (Mammone ym., 2009). Lagrangen kertoimien käyttöä ei käsitellä tarkemmin, mutta niiden avulla yhtälörajoittunut optimointiongelma voidaan ratkaista osittaisten derivaattojen avulla (Berwick, 2003).



KUVIO 4 Tukivektorikoneen hypertaso ja tukivektorit (Berwick, 2003, s. 7)

3.1.4 Päätöspuut

Päätöspuut ovat joukko koneoppimisalgoritmeja, joilla luodaan haarautuva sarja kysymyksiä, joiden avulla pyritään määrittämään dataobjektin luokka tai numeerinen arvo. Erilaisilla päätöspuilla pystytään siis ratkomaan luokittelu- ja regressio-ongelmia, mutta niiden kaikkien ydintoimintaperiaate on sama. Rokach ja Maimon (2005) antavat artikkelissaan kattavan kuvauksen päätöspuiden rakenteesta ja toimintaperiaatteesta. Artikkelissa kerrotaan päätöspuiden koostuvan solmuista, joista ensimmäistä solmua kutsutaan juureksi. Juuressa tapahtuu päätöspuun ensimmäisen testi, jonka tuloksen perusteella päätetään mihin solmuun algoritmi etenee seuraavaksi. Seuraava solmu sisältänee taas testin, jolloin sitä kutsutaan testisolmuksi. Kun päätöspuussa edetään tarpeeksi pitkälle, jotta testin perusteella voidaan luokitella tai antaa dataobjektille tietty arvo, päättelyprosessi päättyy päätössolmuun, joka luokittelee tai määrittää dataobjektin arvon (Rokach & Maimon, 2005). Valmis päätöspuu on siis visuaalisesti esitettynä vuokaavio, joka esittää päätöksenteon algoritmia. Koneoppimisen kannalta mielenkiintoisempaa on se, miten koneoppija, eli kasvattaja, päättää puun rakenteen.

Rokach ja Maimon (2005) käsittelevät artikkelissaan yleisintä kasvatuskeinoa, eli ylhäältä-alaspäin kasvatusta. Tällaiset kasvattajat aloittavat juurisolmusta, ja etsivät jokaisella tasolla yhden tai useamman attribuutin, joiden perusteella data jaetaan kahteen tai useampaan osaan. Artikkelissa mainitaan myös joidenkin kasvattajien kasvatusvaiheen olevan kaksivaiheinen, eli niihin voi kuulua karsintavaihe. Jos algoritmiin kuulu karsintavaihe, puun kasvatuksen jälkeen algoritmi käy solmut läpi, mahdollisesti poistaen solmuja, jos niistä on kriteerien mukaan enemmän haittaa, kuin hyötyä (Rokach & Maimon, 2005).

Tarkastellaan päätöspuiden toimintaa CART-algoritmin, eli luokittelu ja regressiopuun avulla. Rokach ja Maimon (2005) kertovat CART:in rakentavan binääripuun, eli jokainen solmu haarautuu alaspäin kahdeksi solmuksi. Jakokriteerinä se käyttää kahtiajakokriteeriä, ja puun valmistuessa, algoritmi karsii heikoimpia solmuja hintakompleksisuuteen perustuen. Kuten algoritmin nimestä voi päätellä, pystyy se suoriutumaan luokittelun lisäksi myös regressiosta (Rokach & Maimon, 2005). Rokachin ja Maimon artikkelissa jätettiin käsittelemättä CART:in ennustemallin ohjausparametrit, joiden avulla voidaan ohjata puun rakennetta halutunlaiseksi. Sarkarin, Patelin, Madaanin ja Maitin (2016) tutkimuksessa luetellaan mahdollisiksi ohjausparametreiksi esimerkiksi puun maksimisyvyys, havaintojen minimimäärä päätössolmuissa, sekä puun kokoon vaikuttava kompleksisuusparametri. Sarkarin ja kumppaneiden (2016) sekä Rokachin ja Maimon (2005) artikkeleissa molemmissa mainitaan CART:in olevan ahne algoritmi, eli se etsii juurisolmusta alkaen parasta ratkaisua, välittämättä ongelmista mitä ahneudesta voi myöhemmin koitua. CART:in optimointi tapahtuu kahdella tapaa, riippuen käytetäänkö sitä regressioon vai luokitteluun. Luokittelussa optimoinnilla pyritään Rokachin ja Maimon (2005) mukaan saavuttamaan mahdollisimman korkea puhtaus, eli tavoitellaan jakoa, jossa kaikki osajoukon dataobjektit ovat mahdollisimman samankaltaisia. Käytännössä tämä tapahtuu laskemalla Gini-kerroin jokaisen eri attribuutin perusteella tehdylle jaolle, ja valitsemalla jakajaksi attribuutti, jonka perusteella luotujen osajoukkojen eriarvoisuus on mahdollisimman pieni. Vastaavasti artikkelista selviää, että regressiopuun optimoinnissa minimoidaan solmussa tapahtuvan jaon keskineliövirhe (Rokach & Maimon, 2005). Käytännössä tämä tarkoittaa, että algoritmi etsii kahtiajakokohdan, josta jakamalla aiheutuu keskimäärin vähiten virhettä harjoitusdataan verrattuna.

3.2 Ohjaamaton oppiminen

Ohjaamattomassa oppimisessa käsiteltävä data on luokittelematonta, eli data koostuu vain syötteistä, joille ei ole mitään odotettua tulostetta (Simeone, 2018). Ohjaamattomassa oppimisessa voidaan esimerkiksi etsiä harjoitusdatasta rykelmiä, joissa syötteiden arvot ovat lähellä toisiaan, tätä kutsutaan klusteroinniksi. Toinen yleinen ohjaamaton menetelmä on datan ulottuvuuksien vähentäminen, eli vaikuttavien attribuuttien vähentäminen. Ohjaamattoman oppimisen keinoilla voidaan myös etsiä datasta poikkeavuuksia.

3.2.1 K-keskiarvot-klusterointi

K-keskiarvot algoritmi on yksi käytetyimpiä ohjaamattoman oppimisen algoritmeja. Algoritmista on tehty useita variaatioita, joista käsittelen pääasiassa alkuperäistä algoritmia, eli kehittäjänsä mukaan nimettyä Lloydin k-keskiarvot algoritmia. Algoritmin nimi kuvastaa sen toimintaa hyvin, sillä kirjain K kuvaa

klustereiden määrää, ja klusterointi tapahtuu dataobjektien etäisyyksien keskiarvojen avulla.

Algoritmin toimintaperiaate on hyvin yksinkertainen. Shanthamallu ja muut (2017) kirjoittavat artikkelissaan algoritmin käytön vaativan aluksi tiedon siitä, kuinka moneen klusteriin data halutaan jakaa. Seuraavaksi valittu määrä tulevia klustereiden keskipisteitä asetetaan satunnaisten dataobjektien kohdalle. Tämän jälkeen kaikki dataobjektit liitetään niitä lähimpänä olevaan keskipisteeseen, jolloin muodostuu haluttu määrä klustereita (Shanthamallu ym., 2017). Nämä klusterit ovat kuitenkin aluksi hyödyttömiä, sillä ne on jaettu täysin satunnaisten pisteiden mukaan, jota voisi kutsua algoritmin alustamiseksi. Itse algoritmin toiminta voidaan Kanungon ja tutkijatovereiden (2002) artikkelin mukaan jakaa kahteen vaiheeseen. Ensimmäisessä vaiheessa algoritmi laskee klustereihin kuuluvien dataobjektien keskiarvon, ja siirtää klustereiden keskipisteet keskipisteeseen. Toisessa vaiheessa algoritmi klusteroi datan uudestaan määrittelemällä dataobjektit kuulumaan klustereihin, joiden keskipiste on niitä lähimpänä (Kanungo ym., 2002). Näitä kahta vaihetta toistetaan niin kauan, kunnes keskipisteiden paikka ei enää muutu (Shanthamallu ym., 2017). Alkuperäinen Lloydin k-keskiarvot algoritmi siis toistaa kahta yksinkertaista laskutoimitusta, kunnes virhefunktio on minimoitu. Algoritmin virhefunktio on dataobjektien ja niiden lähimpien keskipisteiden välisten etäisyyksien neliöiden summa (Kanungo ym., 2002).

Kanungo ja muut (2002) mainitsevat artikkelissaan yhden tärkeän piirteen, jota ei mainita Shanthamallun ja kumppaneiden (2017) artikkelissa lainkaan. Kanungo ja muut (2002) huomauttavat, että algoritmin lopputulos on aina paikallinen minimi, mutta ei välttämättä globaali minimi. Käytännössä tämä johtuu siitä, että algoritmi minimoi virhefunktion vain yhdestä satunnaisesti valitusta alkutilanteesta. Jos haluttaisiin varmistaa globaalin minimin löytyminen, täytyisi algoritmin käydä sama prosessi läpi jokaisesta mahdollisesta alkutilanteesta, joka olisi erittäin työlästä. Hyvän tuloksen saavuttamiseksi algoritmia käytetään usein monta kertaa peräkkäin, jonka jälkeen voidaan valita paras paikallinen minimi. Vaihtoehtoisesti voidaan käyttää jotakin algoritmin eri varianttia, joka pyrkii löytämään globaalin minimin, tai ainakin paremman paikallisen minimin. Tarkempaan ratkaisuun pyrkivä algoritmi olisi kuitenkin todennäköisesti huomattavasti hitaampi.

3.2.2 Pääkomponenttianalyysi

Pääkomponenttianalyysin ydintavoite on pienentää tarkasteltavan datan attribuuttien määrää, kuitenkin siten, että data säilyttää mahdollisimman paljon varianssia. Varianssin säilyttäminen tarkoittaa käytännössä sitä, että vaikka attribuuttien määrä vähenee, datasta häviää mahdollisimman vähän hyödyllistä informaatiota.

Abdin ja Williamsin (2010) mukaan datan valmistelu pääkomponenttianalyysia varten sisältää kaksi toimenpidettä: Data yleensä keskitetään siten, että jokaisen attribuutin keskiarvo on 0, joka toteutetaan laskemalla attribuuttien ilmentymien keskiarvo, ja vähentämällä tämä keskiarvo attribuutin jokaisesta

ilmentymästä. Toinen toimenpide on eri attribuuttien standardisointi, joka toteutetaan jakamalla jokaisen attribuutin jokainen ilmentymä omalla normillaan, jolloin kaikki attribuutit saadaan samaan skaalaan (Abdi & Williams, 2010). Smith (2002) käy pääkomponenttianalyysin läpi yksinkertaisella esimerkillä, jossa hän jakaa prosessin kuuteen osaan. Ensimmäiset kaksi vaihetta liittyivät juuri esiteltyyn datan valmisteluun. Kolmas vaihe Smithin oppaan mukaan on kovarianssimatriisin laskenta, josta selviää attribuuttien väliset korrelaatiot. Oppaan neljännessä vaiheessa lasketaan kovarianssimatriisille ominaisarvot, sekä ominaisvektorit. Viidennessä vaiheessa ominaisvektorit järjestetään matriisiin siten, että suurimman ominaisarvon omaava ominaisvektori on matriisissa päällimmäisenä, joka on myös löytämämme pääkomponentti. Smithin oppaan viimeisessä vaiheessa muunnetaan alkuperäinen data vastaamaan löytämiämme ominaisvektoreita, joka tapahtuu kertomalla alkuperäinen datamatriisi ominaisvektorimatriisilla, jossa ominaisvektorit ovat sarakkeissa. Lopputulokseksi saadaan dataobjektien pääkomponenttiarvot, eli alkuperäisten attribuuttien lineaarikombinaatiot (Smith, 2002).

Pääkomponenttianalyysin ymmärtämisen kannalta on tärkeintä, että ymmärtää viidennen vaiheen, jossa käsitellään ominaisvektoreita, tai tarkemmin omaisvektoreiden ominaisarvoja. Abdi ja Williams (2010) kertovat artikkelissaan pääkomponenttien hyvyyden mittariksi sen, miten ison osan kokonaisvarianssista se selittää. Kaikkien ominaisarvojen summa vastaa kaikkien attribuuttien varianssia, joten laskemalla viidennessä vaiheessa ominaisarvojen osuudet ominaisarvojen summasta, saamme selville kuinka suuren osan kokonaisvarianssista kukin attribuutti selittää (Abdi & Williams, 2010). Tämä on tärkeää ymmärtää, koska kuten aluksi mainittiin, tavoitteemme on vähentää tarkasteltavien attribuuttien määrää, mutta säilyttää mahdollisimman korkea varianssi. Laskettuamme esimerkiksi viiden eri attribuutin varianssit, josta huomaisimme ensimmäisen kahden selittävän lähes kaiken varianssin, voisimme muuntaa alkuperäisen datan vain kahdella ensimmäisellä ominaisvektorilla, jolloin pienentäisimme attribuuttien määrän viidestä kahteen, kuitenkaan menettämättä paljoa varianssia.

Koneoppimisen kontekstissa pääkomponenttianalyysin toimintaperiaate ei eroa esitetystä tilastotekniikasta juurikaan, ja sen kuulumisen koneoppimisen alle voisi kyseenalaistaa. Pääkomponenttianalyysiä kuvaillaan ohjaamattomaksi datan esikäsittelykeinoksi, joten tutkimuksessa se sisällytettiin ohjaamattoman oppimisen alle, vaikkei siinä ole suoranaista oppimista. Pääkomponenttianalyysilla on kuitenkin merkittävä rooli koneoppimisessa, sillä se on yksi käytetyimpiä keinoja datan attribuuttien vähentämiseksi, joka on usein tarpeellinen toimenpide datan valmistelussa koneoppimista varten.

3.3 Vahvistusoppiminen

Vahvistusoppimisen erityispiirre on harjoitusdatan luonne, joka koostuu järjestelmän nykytilasta, sallituista toiminnoista, sekä tavoitetilasta.

Vahvistusoppiminen eroaa merkittävästi ohjatusta- ja ohjaamattomasta oppimisesta, sillä sen käyttö liittyy lähinnä suljettuihin ympäristöihin, kuten videopeleihin, tai robotin toimintaympäristöön.

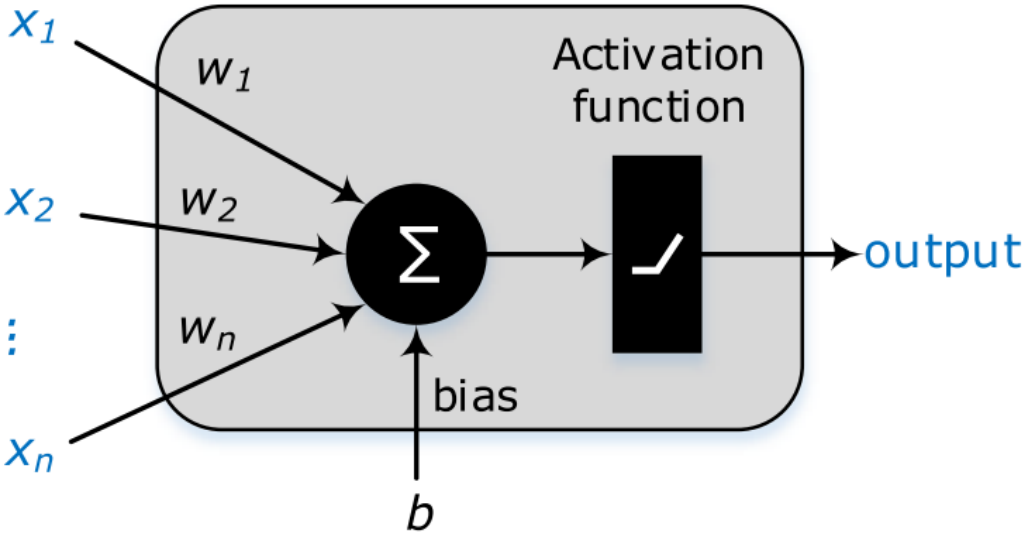
Arulkumaranin ja kumppaneiden (2017) artikkelissa kerrotaan vahvistusoppimisen perusideaksi oppimisen perustuvan vuorovaikutukseen. Artikkelissa vuorovaikutuksella tarkoitetaan vahvistusoppimisagentin, eli itsenäisesti toimivan ja oppivan tietokoneohjelman, läpikäymää prosessia, jossa se oppii ympäristöltään saaman palautteen perusteella. Koneoppimisprosessiin liittyy ympäristön nykytilan havainnoiminen, agentin toiminta ympäristön nykytilan perusteella, ja ympäristöltä saatavat palkkiot toiminnan jälkeen. Artikkelissa kerrotaan näiden kolmen toiminnon riittävän optimaalisen ratkaisun löytämiseksi (Arulkumaran ym., 2017). Vahvistusoppimisen optimointi on siis melko yksinkertainen prosessi, jossa erilaisia vaihtoehtoja kokeillaan iteratiivisesti niin kauan, kunnes löydetään paras mahdollinen sarja toimintoja. Vahvistusoppimisessa optimointiongelma ei siis ole virhefunktion minimoimista, vaan arvofunktion maksimoimista (Arulkumaran ym., 2017). Arvofunktion maksimointi kuitenkin tarkoittaa samalla virheen minimoimista, joten yleisesti voidaan tässäkin tapauksessa puhua virhefunktion minimoimisesta.

3.4 Neuroverkot

Neuroverkkojen toiminta on perinteisiä koneoppimisalgoritmeja huomattavasti joustavampaa, ja niiden voidaankin katsoa yhdistelevän useita konsepteja perinteisestä koneoppimisesta (Janiesch ym., 2021). Sen sijaan, että neuroverkot selvittäisivät yhtä ongelmaa, kuten esimerkiksi yhden virhefunktion minimointia, niiden päätöksenteko pohjautuu useiden neuronien laskentatulosten yhdistelyyn. Tämän takia neuroverkot kykenevät huomattavasti monipuolisempien ongelmien ratkaisemiseen, kuin perinteiset koneoppimisalgoritmit, mutta varjopuolena tälle on neuroverkkojen mustalaatikkomaisuus, eli usein emme kykene selvittämään miten neuroverkko on päätynyt ratkaisuunsa (Janiesch ym., 2021). Neuroverkkoarkkitehtuureja on useita, mutta niiden kaikkien toimintaperiaate perustuu muutaman konseptin ympärille. Eri arkkitehtuurit suoriutuvat tietynlaisista tehtävistä toisiaan paremmin, mutta lähtökohtaisesti kaikilla neuroverkkoarkkitehtuureilla pystytään selvittämään kaikenlaisia ongelmia, mutta niiden tarkkuus ja teho kärsivät tehtävissä, joihin niiden arkkitehtuuri ei sovellu (Janiesch ym., 2021).

Shrestha ja Mahmood (2019) kertovat neuroverkkojen koostuvat yhteen kytketyistä laskentayksiköistä, eli neuroneista, jotka on jaettu syöte-, piilo-, ja tuloskerroksiin. Artikkelissa kerrotaan neuronien olevan kytketty edeltävän ja seuraavan tason neuroneihin. Neuroneiden rakennetta kuvataan kuviossa 5, jossa x kuvastaa edelliseltä tasolta saatua syötettä, w kuvastaa syötteen painokerrointa, ja b kuvastaa vakiotermiä. Shrestan ja Mahmoodin (2019) artikkelin mukaan jokaisessa neuronissa syötteet kerrotaan niitä vastaavilla painokertoimilla, summataan ne, ja lisätään niihin neuronin vakiotermi. Kun neuroni on summannut

syötteet, laskennan tulos ajetaan aktivointifunktion läpi (kuvio 5), joka muuntaa tuloksen siihen muotoon, missä se lähetetään seuraavalle tasolle (Shrestha & Mahmood, 2019). Aktivointifunktioita on monenlaisia, esimerkeiksi Shrestha ja Mahmood (2019) antavat muun muassa sigmoidifunktion sekä ReLU-funktion. Sigmoidifunktio tuli tutuksi jo logistista regressiota käsitellessä, eli sen avulla voidaan muuttaa kaikki arvot välille 0–1, vaihtoehtoisesti sitä voidaan käyttää myös todennäköisyyden mittarina, tai binäärisenä ei-kyllä luokittelijana. ReLU-funktio taas on erittäin yksinkertainen ja helposti optimoitava aktivointifunktio, joka aktivoituu vain positiivisilla arvoilla, eikä aktivoiduttuaan muunna lähetettävää arvoa lainkaan (Shrestha & Mahmood, 2019).



KUVIO 5 Neuroverkon piilokerroksen neuroni (Mohammadi ym., 2018, s. 7)

Vaikka neuroverkkojen rakenne on täysin erilainen kuin muiden koneoppimistyökalujen, on niiden opettamisvaihe melko samankaltainen. Mohammadi ja kumppanit (2018) kuvaavat neuroverkkojen opetusprosessia artikkelissaan seuraavanlaisesti: Syötekerros asettaa käsiteltävälle harjoitusdatalle satunnaiset painotukset, ja lähettää sen ensimmäiselle piilokerrokselle. Tämän jälkeen kaikki piilokerrokset vuorollaan asettavat syötteille painotukset, lisäävät niihin vakiotermit, suorittavat laskentansa, ja lähettävät syötettä eteenpäin, kunnes kaikki kerrokset on käyty läpi. Lopulta neuroverkon viimeinen kerros, eli tuloskerros, tekee lopullisen arvion. Tätä arviota verrataan harjoitusdataan, ja virhefunktion avulla määritetään, kuinka lähellä arvio oli todellista tulosta (Mohammadi ym., 2018).

Kuten perinteisessä koneoppimisessa, on neuroverkkojen optimointiongelma virhefunktion minimoiminen. Neuroverkkojen optimoinnin erityispiirre on laskennan tapahtuminen useassa vaiheessa, ja tämän erityispiirteen ratkaisemiseksi käytetään yleensä jotain vastavirta-algoritmia. Vastavirta-algoritmin

toiminta perustuu gradienttimenetelmään, eli myös neuroverkkojen virhefunktion minimoiminen tehdään yleensä derivoimalla (Shrestha & Mahmood, 2019). Neuroverkoissa eri tasojen derivointi tapahtuu ketjusäännön avulla, joka on keino derivoida yhdistettyjä funktioita. Neuroverkon optimointi tapahtuu Mohammadin ja kanssatutkijoiden (2018) artikkelin mukaan ensin suorittamalla neuroverkon laskenta satunnaisin painokertoimin ja vakiotermein. Kun laskenta on suoritettu ensimmäisillä arvoilla, lasketaan virhefunktiolla ennustetun ja toteutuneen tuloksen välinen ero. Seuraavaksi artikkelin mukaan virhefunktion tulos syötetään kerros kerrokselta vastavirta-algoritmilla takaisin syötekerrokseen asti, muuttaen matkalla neuronien painokertoimia ja vakiotermejä gradienttimenetelmään perustuvan laskennan keinoin. Tätä prosessia toistetaan niin kauan, että virhefunktion tulos saadaan alle määritetyn virhemarginaalin (Mohammadi ym., 2018).

4 YHTEENVETO

Tutkittava aihe oli koneoppimistyökalut sekä niiden toiminta. Aiheen ulkopuolelle jäi siis koneoppiminen ilmiönä, joka kattaisi esimerkiksi hyödyt, haitat, sekä vaikutukset yhteiskuntaan. Koneoppiminen on tällä hetkellä erittäin aktiivinen tutkimuksen kohde, mutta tutkimus keskittyy pääasiassa uusiin teknologioihin sekä niiden hyödyntämiseen. Aihetta käsittelevässä tuoreessa tutkimuksessa monien asioiden ymmärtämistä pidetään itsestäänselvyytenä, joka voi vaikeuttaa aiheeseen perehtymistä. Tämän tutkimuksen päämotivaattori oli tarve ymmärtää koneoppimisen perusteet, ja sen vuoksi tutkimus toimii hyvänä lähtökohtana aihepiiristä kiinnostuneille.

Koska tutkimuksen tavoitteena oli selvittää perusteita uusien löydöksien tekemisen sijaan, keskityttiin tutkimuksessa pääasiassa vanhempiin koneoppimistyökaluihin. Yksinkertaisuutensa vuoksi vanhempien työkalujen toimintaa oli helpompi tulkita, joten ne soveltuivat tutkimuksen aiheeseen paremmin. Vaikka käsiteltävänä oli pääasiassa vanhempia työkaluja, tutkimus auttaa ymmärtämään myös uudempien koneoppimistyökalujen toimintaa. Koneoppimistyökaluista on vuosien saatossa tullut aina vain monimutkaisempia, mutta siitä huolimatta niiden toiminnan perusidea ei ole juuri muuttunut. Tutkimuksen tutkimusongelma oli selvittää koneoppimistyökalujen toimintaperiaatteet. Toimintaperiaatteilla tarkoitetaan keinoja, joita käyttämällä koneoppija suoriutuu sille annetusta tehtävästä ja päätyy johonkin lopputulokseen.

Tutkimus toteutettiin kuvailevana kirjallisuuskatsauksena. Lähdekirjallisuutena toimi pääasiassa vertaisarvioidut tieteelliset artikkelit, mutta ajoittain tukeuduttiin myös muun muassa oppikirjoihin sekä teknisiin oppaisiin. Aluksi tehtiin lyhyt yleiskatsaus koneoppimisen perusteista, jotta lukijalla olisi peruskäsitys koneoppimisesta ennen koneoppimistyökalujen yksityiskohtaisempaa tarkastelua. Työkalujen toimintaperiaatteisiin tutustuttiin pääasiassa lähdekirjallisuudessa esiintyneiden matemaattisten kaavojen avulla. Näistä kaavoista pystyi yleensä erottamaan jonkin selkeän ongelman, jonka ratkaisemiseksi tarvittaisiin jotakin matemaattista menetelmää. Tutkimuksen tulokset esiteltiin lukijaystävällisesti ilman matemaattisia kaavoja.

Tutkimuksen merkittävin löydös oli se, että koneoppiminen on usein jonkin perinteisen tilastotieteellisen menetelmän suorittamista iteratiivisesti niin kauan, että saavutetaan haluttu lopputulos tai tarkkuus. Näiden tilastotieteellisten menetelmien sisältä löytyi usein jokin matemaattinen optimointiongelma. Useimmiten tämä optimointiongelma oli löytää arvot painokertoimille sekä vakiotermeille, joiden avulla ennusteen ja toteutuneen tapahtuman välinen ero, eli virhefunktio on pienimmillään. Keinot näiden arvojen selvittämiseksi riippuu siitä, millaista optimointiongelmaa selvitetään. Kiteytetysti koneoppimistyökalujen toiminta perustuu matemaattiseen optimointiin, jota suoritetaan iteratiivisesti tietokoneen laskentatehon avulla.

Tutkimuksessa löydettyjen optimointiongelmien ratkaiseminen perustuu vahvasti matematiikkaan, varsinkin lineaarialgebraan sekä derivointiin. Lineaarialgebraa sovelletaan koneoppimisessa muun muassa silloin, kun vakinaistetaan arvoja, pohditaan datan ulottuvuuksia, tai suoritetaan lineaarista regressiota. Lineaarialgebraa käytetään tavalla tai toisella kaikissa koneoppimistyökaluissa, joita tutkimuksen aikana käsiteltiin. Lineaarialgebran tärkeys tuli esille varsinkin pääkomponenttianalyysin yhteydessä, joka perustuu lähes täysin lineaarialgebraan. Derivoinnin havaittiin usein olevan avainasemassa virhefunktion minimoimisessa, eli silloin kun etsitään gradienttimenetelmän avulla funktion matalinta pistettä. Virhefunktion minimointi tarkoittaa siis ennustavan mallin tarkkuuden maksimointia, joka lopulta on kaiken koneoppimisen ydintavoite. Ilman koneoppimista matemaattinen optimointi on erittäin työlästä, jonka takia toistoja pystyttäisiin tekemään huomattavasti vähemmän, joka vuorostaan heikentäisi ennusteiden tarkkuutta. Tutkimuksen löydöksillä voidaankin alkuperäisen tutkimustavoitteen lisäksi perustella myös koneoppimisen tärkeyttä yleisesti.

Tämän kirjallisuuskatsauksen aikana löytyi useita mielenkiintoisia jatkotutkimusaiheita. Tutkimuksen rajallinen pituus rajoitti käsiteltävien koneoppimistyökalujen määrää, joten tutkimuksessa ei pystytty tutustumaan kuin pieneen joukkoon työkaluja. Jatkotutkimuksissa voitaisiin tutkia muita koneoppimistyökaluja tai keskittyä tutkimaan yksittäisten työkalujen toimintaa syvällisemmin. Syvällisempi tutkimus voisi käsitellä koneoppimistyökalujen suorituskykyä, tai kuinka ne soveltuvat käsittelemään erityyppisiä datakokonaisuuksia. Tässä tutkimuksessa todettiin optimointiongelmien ratkaisemisen olevan koneoppimisen keskiössä. Näitä optimointiongelmia ratkaistaan erilaisten optimointialgoritmien avulla, joiden tutkiminen olisi looginen jatkumo tälle tutkimukselle. Tässä tutkimuksessa ei myöskään tutustuttu neuroverkkoihin yleistasoa syvällisemmin. Kaikki edellä mainitut jatkotutkimusaiheet olisivat soveltuvia sekä mielenkiintoisia myös neuroverkkojen kontekstissa.

LÄHTEET

Abdi, H. & Williams, L. J. (2010). Principal Component Analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, *2*(4), 433–459. https://doi.org/10.1002/wics.101

Arulkumaran, K., Deisenroth, M. P., Brundage, M. & Bharath, A. A. (2017). Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Processing Magazine*, *34*(6), 26–38. https://doi.org/10.1109/MSP.2017.2743240

Berwick, R. (2003). *An Idiot's guide to Support vector machines (SVMs)*. Massachusetts Institute of Technology.

Borisov, V., Leemann, T., Sessler, K., Haug, J., Pawelczyk, M. & Kasneci, G. (2022). Deep Neural Networks and Tabular Data: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 1–21. https://doi.org/10.1109/TNNLS.2022.3229161

Chen, M., Challita, U., Saad, W., Yin, C. & Debbah, M. (2019). Artificial Neural Networks-Based Machine Learning for Wireless Networks: A Tutorial. *IEEE Communications Surveys & Tutorials*, *21*(4), 3039–3071. https://doi.org/10.1109/COMST.2019.2926625

Coelho, L. P. & Richert, W. (2015). *Building machine learning systems with Python: get more from your data through creating practical machine learning systemw with Phython* (Second edition). Packt Publishing.

He, K. & He, C. (2021). Housing Price Analysis Using Linear Regression and Logistic Regression: A Comprehensive Explanation Using Melbourne Real Estate Data. *2021 IEEE International Conference on Computing (ICOCO)*, 241–246. https://doi.org/10.1109/ICOCO53166.2021.9673533

Janiesch, C., Zschech, P. & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, *31*(3), 685–695. https://doi.org/10.1007/s12525-021-00475-2

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R. & Wu, A. Y. (2002). An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(7), 881–892. https://doi.org/10.1109/TPAMI.2002.1017616

Katal, A., Wazid, M. & Goudar, R. H. (2013). Big data: Issues, challenges, tools and Good practices. *2013 Sixth International Conference on Contemporary Computing (IC3)*, 404–409. https://doi.org/10.1109/IC3.2013.6612229

Maalouf, M. (2011). Logistic regression in data analysis: an overview. *International Journal of Data Analysis Techniques and Strategies*, *3*(3), 281. https://doi.org/10.1504/IJDATS.2011.041335

Mammone, A., Turchi, M. & Cristianini, N. (2009). Support vector machines. *WIREs Computational Statistics*, *1*(3), 283–289. https://doi.org/10.1002/wics.49

Mohammadi, M., Al-Fuqaha, A., Sorour, S. & Guizani, M. (2018). Deep Learning for IoT Big Data and Streaming Analytics: A Survey. *IEEE Communications Surveys & Tutorials*, *20*(4), 2923–2960. https://doi.org/10.1109/COMST.2018.2844341

Rokach, L. & Maimon, O. (2005). Top-Down Induction of Decision Trees Classifiers—A Survey. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, *35*(4), 476–487. https://doi.org/10.1109/TSMCC.2004.843247

Rustam, F., Reshi, A. A., Mehmood, A., Ullah, S., On, B.-W., Aslam, W. & Choi, G. S. (2020). COVID-19 Future Forecasting Using Supervised Machine Learning Models. *IEEE Access*, *8*, 101489–101499. https://doi.org/10.1109/ACCESS.2020.2997311

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, *3*(3), 210–229.

Shanthamallu, U. S., Spanias, A., Tepedelenlioglu, C. & Stanley, M. (2017). A brief survey of machine learning methods and their sensor and IoT applications. *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1–8. https://doi.org/10.1109/IISA.2017.8316459

Shrestha, A. & Mahmood, A. (2019). Review of Deep Learning Algorithms and Architectures. *IEEE Access*, *7*, 53040–53065. https://doi.org/10.1109/ACCESS.2019.2912200

Simeone, O. (2018). A Very Brief Introduction to Machine Learning With Applications to Communication Systems. *IEEE Transactions on Cognitive*

Communications and Networking, *4*(4), 648–664. https://doi.org/10.1109/TCCN.2018.2881442

Smith, L. (2002). *A tutorial on Principal Components Analysis* (OUCS-2002-12). University of Otago, Department of Computer Science.

Xue-Wen Chen & Xiaotong Lin. (2014). Big Data Deep Learning: Challenges and Perspectives. *IEEE Access*, *2*, 514–525. https://doi.org/10.1109/ACCESS.2014.2325029