Hukari Iida

Jokien pinnan korkeuden muutoksen ennustaminen

Tekniikan ja innovaatiojohtamisen akateeminen yksikkö
Energia- ja informaatiotekniikka
Diplomityö
Ohjelmistotekniikka

Vaasa 2022

VAASAN YLIOPISTO

Tekniikan ja innovaatiojohtamisen akateeminen yksikkö

Tekijä: Hukari Iida

Tutkielman nimi: Jokien pinnan korkeuden muutoksen ennustaminen

Tutkinto: Diplomi-insinööri

Oppiaine: Ohjelmistotekniikka

Työn ohjaaja: Lampinen Jouni

Valmistumisvuosi: 2022 **Sivumäärä:** 72

TIIVISTELMÄ:

Tässä diplomityössä käsitellään jokien veden korkeuden muutosten ennustamista käyttäen MLP-neuroverkkoa ennustamisen välineenä. Tutkimus rajattiin vain yhden joen ja tärkeimpien ympäristössä ilmeentyvien muuttujien vaikutuksen tarkasteluun. Joen pinnan muutoksiin vaikuttavat tekijät, jotka ovat ihmiseen sidonnaisia, kuten patovesien juoksuttaminen, ei otettu huomioon. Tavoitteena oli luoda toimiva, muutaman kerroksen neuroverkko, joka kykenisi ennustamaan annetuista muuttujista joen veden pinnan muutokset annetun virhemarginaalin rajoissa.

Työn suorittaminen alkoi tutustumalla esitetyn aiheen toimintaan eli joen virtaaman syntyyn ja vaikutuksiin. Tämä oli tärkeää, jotta kyetään ymmärtämään käsillä olevan ongelman rakennetta, ja jotta ongelma voidaan esittää muuttujien avulla neuroverkolle. Neuroverkkojen rakenteeseen ja taustalla olevaan matematiikan tutustumisen jälkeen voitiin siirtyä itse annetun ongelman: kuinka voidaan neuroverkolla ennustaa joen veden pinnan muutoksia, ratkaisemiseen. Ongelman ratkaisuksi soveltuvaa neuroverkkoa kuvattiin Python- ohjelmointikielellä. Neroverkon luomisen jälkeen sitä testattiin Yhdysvaltain Suurien Järvien alueelta kerätyllä datalla. Data käsiteltiin, jotta se olisi mahdollisimman tarkoituksen mukaista ja sisältäisi mahdollisimman vähän ei-toivottuja piirteitä. Datan käsittelyssä valituista muuttujista luotiin datakehys, joka jaettiin kahteen osaan: opettamis- ja testausjoukkoihin. Neuroverkkoon syötettiin ensimmäisenä opetusjoukko, jonka avulla verkon oppimista ohjaavia painoja säädettiin. Opetusjoukon jälkeen syötettiin testijoukko. Testijoukon on tarkoitus paljastaa onko luotu neuroverkko kyennyt oppimaan ennustamaan joen muutoksia.

Työn tulokset olivat odotusten suuntaisia. Opetusvaiheessa verkko osoitti kykyä yleistää ilmiö kohtalaisesti, ja kykeni ennustamaan testijoukkoa oikein. Vaikka tulokset olivatkin kohtalaisen hyviä annetulla datalla, tulisi neuroverkolle syöttää alkuperäisen aineiston lisäksi myös muilta alueilta kerättyä dataa. Myös neuroverkon rakennetta ja koodausta tulisi tarkastella paremmin. Kuitenkin voidaan todetta, että MLP neuroverkkoja voidaan käyttää jokien pinnan korkeuden ja virtaaman ennustamiseen. Ennustamisen kannalta on tärkeää valita aineisto ja muuttujat huolella sekä kiinnittää huomiota käytettävien funktioiden valintaan.

AVAINSANAT: Neuroverkot, ennustaminen, jokien pinnan muutokset, tulva

Sisällys

Kuvat

Kuviot

Taulukot

Lyhenteet

MLP	MultiLayer Perceptron
ReLu	Rectified Linear Unit
SVR	Support Vector Regression

1 Johdanto

Jokien pinnan korkeus on aina vaihdellut. Toisinaan ongelmana on ollut liian vähäinen vesimäärä, toisinaan tulviminen. Ilmastonmuutoksen edetessä, jokien pinnan vaihtelut ovat voimistuneet sään ääri-ilmiöiden, kuten rankkasateiden seurauksena. Jokien ympärille rakentaminen ja rantojen reunustaminen, luonnollisten tulva-alankojen muokkaaminen sekä patoaminen ovat myös vaikuttaneet jokiin edistäen tulvimista. Joen pinnan nousun ennustamisella kyetään parantamaan tulvavaroitusten luotettavuutta, sekä tehostamaan muun muassa tekoaltaiden ja patojen juoksutussuunnitelmia.

Tässä raportissa tutkitaan sitä, että kyetäänkö jokien pinnan korkeutta ennustamaan neuroverkon avulla. Neuroverkoista on valittu tarkasteluun MLP-verkko. Neuroverkko on kerättyyn dataan pohjautuva metodi, joka kykenee ennustamaan nopeasti ja tarkasti sisään tulevasta datasta vasteen eli tässä tapauksessa joen pinnan korkeuden muutoksen. Joessa virtaavaan veden määrään ja sitä kautta veden pinnan korkeuteen vaikuttaa lukuisia tekijöitä ihmisen toiminnasta aina pohjavesialtaisiin. Vaikka erityisesti ihmisen toiminnalla on merkittävä vaikutus jokiin ja niiden virtaamaan, muun muassa patoamalla päähaaroja, niiden tarkastelu on jätetty vain teoreettiselle tasolle. Pääpaino on tulvan syntyyn vaikuttavien mekanismien sekä kyseisen ilmiön ymmärtäminen. Käytettävän aineiston osalta tämä tarkoittaa, että sellaiset muuttujat, jotka ilmentävät ihmisen toiminnan vaikutuksia on rajattu ulkopuolelle.

Tavoitteena oli luoda toimiva, muutaman kerroksen ja muutaman sisääntuloneuronin muodostama neuroverkko, joka kykenisi ennustamaan annetuista muuttujista joen veden pinnan muutokset. Ennustettava joki on St. Clair Yhdysvaltain Suurien Järvien alueella. Neuroverkolla on tarkoitus pyrkiä ennustamaan joen virtaaman muutosta, jonka muutos kertoo joen pinnan korkeuden muutoksista: virtaaman kasvaessa on syytä olettaa, että myös joen veden pinta nousee. Tarkasteltava maantieteellinen alue rajattiin koskemaan vain St. Clair -jokea ja siihen vahvasti sidoksissa olevien Detroit -joen ja St. Clair -järven välittömässä läheisyydessä olevia alueita. Tarkoituksena

oli valita korkeintaan kolme muuttujaa alueelta, joita käytettäisiin St. Clair -joen virtaaman ennustamiseen.

Työn ensimmäisessä osassa esitellään joen pinnan muutoksiin vaikuttavia tekijöitä, sekä kuinka joen pinnan muutokset vaikuttavat ympäröivään ekosysteemiin. Joen biomin ymmärtäminen auttaa ongelman kuvaamisessa neuroverkolle. Ensimmäisessä osassa myös kerrotaan lyhyesti malleista, joita vedenpinnan korkeuden ennustamiseen on käytetty. Toisessa osassa on kuvattu neuroverkon yleistä toimintaa ja sen taustalla olevaa matematiikkaa. Tässä osassa on tarkoitus antaa lukijalle yleiskäsitys neuroverkoista, jolloin muun muassa verkon toiminnalle tärkeitä aktivointifunktioita sekä gradienttien merkitystä ei käsitellä tarkemmin.

Neuroverkon suunnittelu sekä rakentaminen ja testaus on kuvattu kappaleissa 5 ja 6. Näissä kappaleissa on pyritty kertomaan toteutuksesta ja valintoihin johtaneista seikoista pääpiirteittäin. Kappaleessa 6 annetaan myös esimerkkejä käytetystä koodista ja selitetään sen toimintaa yleisellä tasolla. Itse koodi löytyy kokonaisuudessaan työn lopusta liitteenä 1.

2 Jokien pinnan korkeuden muutokset

Selkeimmin jokien virtaaman muutos havaitaan veden pinnan nousuna tai laskuna. Molemmat ilmiöt vaikuttavat jokea ympäröivään ekosysteemiin ja ihmisten toimintaan. Joen pinnan nousu aiheuttaa ongelman jokea ympäröivälle asutukselle ja teollisuudelle muun muassa tuhoten rakennuksia ja teitä, sekä huuhtomalla irtainta omaisuutta. Vuonna 2016 Seine-joen pinta nousi yli viisi metriä. Näin voimakas muutos aiheutui kun joen ja sen sivuhaarojen yläjuoksuilla koettiin samanaikaisesti rankkasateita (Kuusisalo, 2016).

2.1 Veden pinnan nousuun vaikuttavat tekijät

Joelle on tyypillistä virtaavan veden määrän vaihtelu sään ja vuodenaikojen mukaan. Ihmisen toiminta on kuitenkin muuttanut paikoin veden virtaamista pato- ja muilla rakennelmilla.

2.1.1 Luonnontilainen ympäristö

Yleisin veden pinnan nousuun vaikuttava tekijä on säätilan muuttuminen yhdessä muiden tekijöiden kuten maaperän laadun sekä pohjaveden määrän kanssa. Joessa oleva vesi muodostuu, kun satava vesi ei ehdi höyrystyä takaisin ilmaan vaan imeytyy pohjavedeksi tai valuu suoraan jokeen. Paikoitellen myös pohjavesi nousee jokeen.

Ilman ollessa kylmä ja kostea sadannasta suurin osa imeytyy joko maahan tai siirtyy valuntana vesialueisiin. Alueilla, joissa maan pinnan kasvillisuus on matalaa tai maaperän läpäisykyky on korkea, sadevesi saattaa imeytyä hyvinkin nopeasti pohjavedeksi. Taasen vuoristoalueilla ja seuduilla, joissa maanpinta on paljas tai jäinen, sadevesi saattaa virrata kaukaakin joelle. Niillä alueilla, joissa pääsee muodostumaan jäätä, voi syntyä jää- ja hyydepatoja. Jääpato on ilmiö, jossa jokeen kasautuu jäätä

rajoittamaan joen virtaamista. Yleisimmin jääpatoja tavataan keväisin, jolloin jäänlähdön aikaan jäälautat kasautuvat ja nostavat veden pintaa (Ympäristöhallinnon yhteinen verkkopalvelu, 2013). Hyydepato taasen on alijäähtyneestä vedestä muodostuneiden jääkiteiden muodostama veden pohjaan tai vesirakenteisiin muodostunut patouma (Ympäristöhallinnon yhteinen verkkopalvelu, 2013). Niinä talvina, jolloin lumen vesiarvo on korkea, sulava lumi voi nostaa nopeastikin veden määrää.

Muodostunut pohjavesi, joka ei enää mahdu pohjavesialtaaseen, valuu pinnanmuotoja noudattaen alamäkeen ja päätyy lopulta jokeen tai muuhun ympäröivää maastoa matalampaan kohtaan nostaen näin veden pintaa ja virtaamaa. Mikäli veden määrä joessa ylittää sen kuljetuskyvyn, virtaama kasvaa ja vaarana on joen tulviminen. Mereen virtaavien jokien veden pinnan muutokseen vaikuttaa myös rantaa kohden lyövä aallokko (Han & Shi, 2008).

2.1.2 Rakennettu ympäristö

Tiheästi rakennetuilla alueilla, kuten kaupungeissa, yllä kuvattu luonnollinen veden kierto on häiriintynyt. Kaupungeissa on vain vähän rakentamatonta pinta-alaa, kuten puistoja ja joutomaata, jonka kautta satava vesi pääsisi imeytymään maahan tai johon vesi pääsisi kerääntymään aiheuttamatta vahinkoa. Vaikka sadeveden höyrystyminen onkin rakennetuilla aluilla suurempaa ja hulevesi ohjataan viemäriverkoston avulla pois kaduilta, se ei riitä poistamaan paikallisen rankkasateen vaikutusta. Pullonkaulaksi saattaa muodostua juurikin viemäriverkosto. Satava vesi täyttää nopeasti kadut ja nostaa mukanaan jätettä. Tämä jäte, yhdessä veden kanssa, täyttää viemäriputkiston ja viemäriverkoston kapasiteetti ylitetään. Tästä seuraa katujen tulviminen. Hulevesiviemäreissä virtaava vesi ei ole sekään ongelmatonta; yleisimmin se johdetaan jokeen tai vastaavaan paikkaan, jonka veden pinta nousee huleveden seurauksena ja aiheuttaa jälleen tulvimista.

Oman ongelmansa urbaanien alueiden tulvimiseen tuovat joet. Suurista kaupungeista muun muassa Pariisi, Pietari ja Lontoo ovat rakennettu jokien varsille. Tälle rakentamiselle on aikanaan ollut hyvät perustelut: joki on toiminut kulkuväylänä ja porttina maailmalle. Samalla kun joki on tuonut vaurautta sitä ympäröivälle yhteisölle, se on aina uhannut tulvimisellaan rannan asutusta. Kaupunkien kasvaessa, rakennettu ympäristö on levittäytynyt joen vartta pitkin. Jokien varsia on kivetty ja pengerretty, jokia on ohjattu uusiin uomiin, sivuhaaroja on kuivattu ja niin edes päin. Näistä kaikista toimista on seurannut ettei vedelle ole enää esimerkiksi tulvatasankoja tai joen sivuhaaroja, joihin se pääsisi virtaamaan. Tällöin kaikki vesi "pakkautuu" pääjokeen ja virtaama nousee nostaen samalla itse joen pintaa.

Rakennettuun ympäristöön kuuluvat myös pato- ja allasrakennelmat. Tällaisten rakennelmien tarkoituksena on pyrkiä hallitsemaan veden virtausta, jotain tiettyä syytä varten. Lisää padoista on kerrottu kappaleessa 2.3.2.

2.2 Veden pinnan korkeuden mallintamisen syyt

Hydrologialla tarkoitetaan tiedettä, joka tutkii vettä, sen ominaisuuksia ja tiloja ilmakehässä, maan pinnalla ja maan pinnan alla (Ympäristöhallinnon yhteinen verkkopalvelu, 2013). Hydrologiset mallit taasen ovat matemaattisia esityksiä, joilla pyritään kuvailemaan hydrologista kiertoa. Veden kierto luonnossa; sateen synnystä, valumisesta meriveteen aina haihtumiseen, on monimutkainen prosessi lukuisine muuttujineen. Tästä johtuen on vaikeaa luoda mallia, joka kykenee kuvaamaan koko kiertoa (Devi ja muut, 2015). Kyseiset mallit onkin kehitelty tiettyä päämäärää varten, esimerkiksi ennustamaan jokien virtaaman nopeuden muutosta tai pohjaveden pinnan korkeutta. Yhteistä näille kaikille malleille on kuitenkin pyrkimys ennustamaan tulevia tapahtumia. (Mallintamismenetelmiä esitellään laajemmin kappaleessa 4: Aikaisempi tutkimus.)

Veden pinnan korkeuden mallintaminen on edellytys tehokkaalle tulvan hallinnalle ja tulvasuunnittelulle. Veden pinnan korkeuden ennustamisella voidaan esimerkiksi ajoittaa tulvan hillitsemiseksi rakennettujen patoaltaiden juoksuttaminen niihin ajankohtiin, jolloin joen virtaama on matalimmillaan ja näin välttää hätäjuoksuttamisen aiheuttamat ongelmat alajuoksulla. Myös tulvavaroitusten oikea aikainen antaminen vähentää syntyviä vahinkoja.

Veden pinnan korkeuden arviointia käytetään avuksi vesirakenteiden huollossa ja suunnittelussa. Arviointia käytetään hyväksi myös ennustaessa ilmastonmuutoksen vaikutuksia veden kiertoon, esimerkiksi mallintamalla sademäärien muutoksia ja niiden vaikutusta viljelyalueisiin.

2.2.1 Tulva

Luonnontilaisessa ympäristössä jokea ympäröivä ekosysteemi on sopeutunut veden pinnan muutoksiin, jopa siinä määrin että on syntynyt ekosysteemejä, jotka ovat riippuvaisia joen jaksoittaisesta tulvimisesta. Yksi tällainen ekosysteemi on tulvatasanko. Tulvivan joen mukana kulkeva liete nousee tulvatasangoille ja toimii lannoitteena. Veden määrän kasvaessa, esimerkiksi tulvan seurauksena, virtausnopeus nousee aiheuttaen eroosiota rantapenkereissä. Tämä eroosio voi johtaa rantapenkereen sortumiseen. Muita luonnolle aiheutuvia ongelmia ovat elinympäristöjen tuhoutuminen sekä eliölajien tuhoutuminen.

Joen pinnan muutoksen, erityisesti pinnan nousun, ennustamisen yhtenä tarkoituksena voidaan pitää tulvan ennustamista. Tulvan ennustamisen seurauksena ehditään antamaan tulvavaroitus niille alueille, joihin kyseinen tulva kohdistuu. Aikainen varoitus palvelee paitsi yksittäisiä ihmisiä myös virkamiehiä, sillä näin ehditään aloittaa ajoissa tulvan aiheuttamien vahinkojen ehkäisy. Vahinkojen ehkäisyn keinoina mainittakoon muun muassa evakuoinnit sekä esineiden ja asioiden siirtäminen veden ulottumattomiin.

Tulvan aiheuttamat vahingot voidaan jakaa kahteen ryhmään: ensisijaiset ja toissijaiset vahingot. Ensisijaisia vahinkoja ovat suoraan tulvinnasta aiheutuvia materiaalivahinkoja sekä kuolemia. Tulva vie mennessään rakennuksia ja siltoja, hävittää teitä ja voimalaitoksia sekä kasaa jätettä. Ihmisiä ja karjaa menetetään hukkumiskuolemissa, rakennusten sortuessa sekä katastrofin aiheuttamassa joukkopaniikissa. Toissijaisiin vahinkoihin kuuluvat taloudelliset vahingot ja henkiset vahingot. Tulvasta jää usein jäljelle vain sen mukana kulkeutunutta irtainta materiaalia: kiviä, autoja, rakennusten osia ja niin edelleen. Jotkin alueet ovat saattaneet peittyä liejuun ja muuhun hienojakoiseen ainekseen. Maatalousmaa on saattanut myrkyttyä kelvottomaksi kemikaalitehtaitten tuhoutumisen tai kaivosjätteen seurauksena. Myös saastuneen pohjaveden pääsy pintavesiin aiheuttaa saastumista. Vesijohto- ja sähköverkosto on tuhoutunut. Ihmisille ja eläimille ei ole saatavana puhdasta vettä ja taudit kuten kolera ja punataudit leviävät helposti. Ihmiset ovat myös traumatisoituneet kokemastaan: menettäneet omaisiaan ja joutuneet kohtaamaan ruhjoutuneita ruumiita. Tulvan aiheuttamat toissijaiset vahingot ovat pitkä kestoisia ja lamauttavat alueen usein täysin.

Tulva kuuluu niihin onnettomuuksiin, joka kyetään ennustamaan hydro-meteorologisesta datasta, kuten sadannasta ja pohjaveden määrästä. Tulvat ovat myös yleensä sidottuja vuoden aikoihin ja niiden aiheuttamiin muutoksiin veden kierrossa. Voidaankin ajatella tietyn joen tulvivan suurimmalla todennäköisyydellä silloin kuin "se on aina ennenkin tulvinut". Kuitenkin nykyinen tilanne, jossa ilmastonmuutos voimistaa sään ääri-ilmiötä, myös tulvien ennustaminen vaikeutuu. Tulvan voidaan ajatella olevan myös ääri-ilmiö itsestään. Tietoa tulvahuipuista tarvitaan suunnitellessa patoja, siltoja, evakuointeja sekä tulvatasankoja.

2.2.2 Patorakennelmat ja vesivoimalaitokset

Vesistöjen säännöstelyllä hallitaan vedenkorkeutta ja virtaamaa padoiksi kutsutuilla rakennelmilla. Padoiksi luetaan: tulvapenkereet, kaivospadot, jätepadot,

vesivoimalarakenteet sekä vesistöpadot (Isomäki ja muut, 2018, s.5). Säännöstelyllä halutaan palvella vesivoimatalouden, tulvansuojelun, vedenhankinnan, kalatalouden sekä virkistyskäytön tarpeita.

Vesistöpatoja rakentaessa, täytyy kyetä ennustamaan virtaama, joka aiheuttaa suurimman juoksutustarpeen. Tämä suurin juoksutustarve vastaa usein alueella esiintyvän tulvan (niin kutsuttu mitoitustulva) todennäköisintä virtaamaa. Esimerkiksi suurien vesistöpatojen mitoitustulva on tulva, joka toistumistodennäköisyys on kerran 5000–10000 vuodessa (Isomäki ja muut, 2018, s.60). Mitoitustulvan määrittelyyn voidaan käyttää valuma-alueen arvoja. Muita mitoitukseen vaikuttavia seikkoja ovat altaan ominaisuudet, yläpuoliset patorakennelmat sekä säännöstelyn tyyppi. (Isomäki ja muut, 2018, s.60). Mitoitustulvan avulla padotusaltaat mitoitetaan siten, että vedenkorkeus ei tulvatilanteessakaan ylitä padon turvallista vedenkorkeutta. Turvallisena veden korkeutena pidetään patorakenteen yläpinnan korkeutta. Esimerkiksi maapadoilla tämä on tiivisteosan yläpinnan korkeus (Isomäki ja muut, 2018, s.9–10).

Väärälle vesimäärälle mitoitettu patorakennelma voi vaurioitua veden korkeuden ylittäessä turvallisen kynnyksen, jolloin padottu aine pääsee virtaamaan padon läpi. Pahimmillaan pato sortuu kokonaan ja synnyttää tulva-aallon, joka aiheuttaa riskin ihmishengille, terveydelle sekä vaaran ympäristölle ja omaisuudelle. Tulva-aalto paitsi tuhoaa rakennuksia ja aiheuttaa vaaraa sen vaikutusalueella oleville ihmisille. Ympäristölle erittäin haitallisia ovat jäte- ja kaivospatojen onnettomuudet, joissa tulva-aalto muodostuu jätemateriaalista. Tämä voi aiheuttaa suojellun alueen, harvinaisten lajien sekä vesistöjen pilaantumisen vaaran. (Isomäki ja muut, 2018, s.18). Tulva-aalto poikkeaa aiemmin esitellystä sään aiheuttamasta tulvasta siinä, että se on luonteeltaan äkillisempi ja tuhovoimaltaan suurempi, joskin tuhoalue voi olla hyvinkin rajattu. Siinä missä kausittaista tulvaa kyetään ennustamaan ja tulvavahinkojen rajoitustoimet voidaan ajoittaa pitkälle aikavälille, padon sortuminen voi tapahtua nopeastikin. Tällöin evakuointeihin ja varotoimenpiteisiin ei ole riittävästi aikaa. Äkillisen padon sortumavaaran voi esimerkiksi aiheuttaa juoksutusluukkujen toimintahäiriö (Isomäki

ja muut, 2018, s.12) tai samassa vesistössä ylempänä olevan patorakennelman sortuminen aiheuttaa alempana olevan padon sortumisen.

Ennustamalla yläjuoksujen vesistöjen pinnanmuutoksia ja sateen vaikutusta patoaltaaseen voidaan onnettomuusvaaraa kyetä ennakoimaan. Jos epäillään padotuskyvyn riittävyyttä vettä voida ohjata muualle esimerkiksi peltoaukeille etukäteen (Isomäki ja muut, 2018, s.13), jolloin padon sortuminen ja siitä aiheutuvat vaaratilanteet voidaan välttää.

Patojen ja juoksutusaltaiden toiminta muuttaa merkittävästi alavirran alueen ekosysteemiä, vaikuttamalla jokiin ja jokiin yhteydessä oleviin järviin. Maailmalla jokien luonnollista virtaamista on muutettu pato- ja allasrakennelmilla. Tällaisilla rakennelmilla on luonnollisesti vaikutus joen ylä- ja alajuoksujen elämään, sillä rakennelmat muuttavat muun muassa tulvien ilmestymistä ja kestoa sekä veden virtaamaa. Veden pinnan muutokset sekä muutos kuivan-kostean kausissa esimerkiksi vähentävät eliölajien määrää ja vesistöjen makrofyyttien monimuotoisuutta (Hu ja muut, 2018). Jotta patojen ja juoksutusaltaiden haittoja jokien ekosysteemille voidaan vähentää tulee veden virtaamaa sulkujen läpi voida hallita. Tehokkaaseen hallintaan tarvitaan jokien virtaaman muutokset lyhyellä aikavälillä.

Vesivoimalaitokset ovat osa tulvasuojelua. Näillä rakennelmilla kyetään säännöstelemään alajuoksulle virtaavan veden määrää patojen tavoin. Suomesta esimerkkinä voidaan mainita Iijoki ja siinä sijaitsevat viisi voimalaitosta, joiden kautta vettä juoksutetaan altaista tulvahuippuina ja jonka altaat toimivat vesivarastoina kuivana kautena. Voimalaitosalueilla sijaitsee niin kutsuttuja säännöstelyaltaita, joihin tulvavesi kerätään. Näistä altaista vettä voidaan laskea hallitusti jokeen ohi voimalaitoksen tai vesi voidaan ohjata laitoksen energiantuotantoon. Voimalaitosten altaat antavat myös aikaa tulvahuipusta ilmoittamiseen sekä mahdollisiin toimenpiteisiin, viivyttämällä tulvahuippua. Vesivoimalaitosten patorakennelmat ovat toisaalta myös riski tavanomaista voimakkaimpien sateiden sekä tulvien aikaan. Laitoksen patoihin ja

altaisiin kertynyt vesi luo paineen reunoja kohden, mikä saattaa aiheuttaa patoaltaan murtumisen että veden hallitsemattoman syöksymisen alajuoksulle.

Tulvilla on myös vaikutusta vesivoimalaitoksen patorakennelmien säännöstelyyn. Selkeimmin tulvien aiheuttama säännöstelyn tarve näkyy vesivoimalaitosten juoksutusaltaiden kohdalla. Voimalaitoksen energiantuotanto on riippuvainen monesta eritekijästä: vedenpinnan korkeudesta, sääilmiöistä sekä alajuoksulla olevien vesistöjen ekosysteemin turvaamisesta ja virkistyskäytöstä. Laitoksen yläjuoksulla sijaitsevat säännöstelyjärvet ja -altaat täyttyvät sateiden sekä tulvien seurauksena. Näin varastoitunutta vettä voidaan käyttää voimalaitoksenpyörittämiseen, jolloin voidaan ajatella veden varastoituvan altaisiin sähköntuotantoa varten. Tällöin voidaan puhua säätövoimasta eli sähköntuotantoa, jolla voidaan tasata nopeasti sähkönkulutuksen ja tuotannon epätasapainoa. Toisaalta mikäli joudutaan turvautumaan runsaaseen ohijuoksutukseen, menetetään tehoja sillä ohijuoksutus vähentää putouskorkeutta (mikä yhdessä voimalaitoksen läpi menevästä vesimäärästä, vaikuttaa laitoksen tehoon). (Kervinen, 2012; Pohjolan Voima)

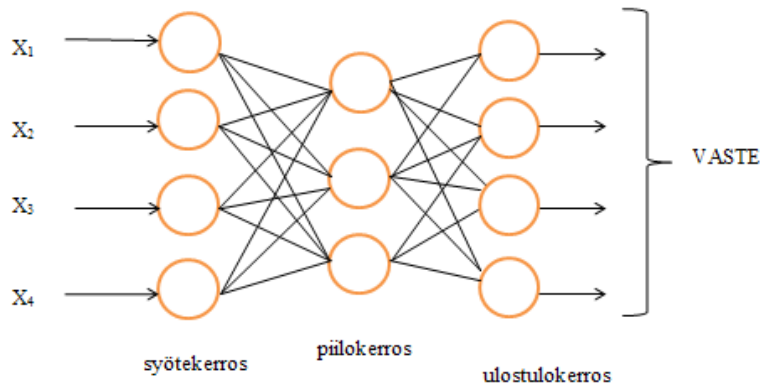
3 Neurolaskenta

Neurolaskennassa funktio esitetään neuroverkoksi kutsutulla mallilla. Neuroverkossa laskutehtävän hoitavat niin kutsutut neuronit ja niistä rakentuvat piilokerrokset. Ulostulokerroksen tuottama vaste kuvastaa funktion arvoa.

3.1 Neuroverkot

Neuroverkot (*ANN artificial neural networks*) ovat keinotekoisia malleja, joiden tarkoituksena on matkia biologista hermoverkkoa. Päämääränä on kyetä löytämään isoissa tietomäärissä olevia säännönmukaisuuksia ja soveltaa löydettyjä säännönmukaisuuksia dataan. Siinä missä biologinen hermoverkko kuvaa elävää olentoa, neuroverkko on matemaattisen funktion esitys. Sekä luonnollisella verkolla, että keinotekoisella, on molemmilla kyky oppia esimerkistä. (Graupe, 2013, s.1–6)

Kuten biologinen hermoverkko, myös keinotekoinen neuroverkko rakentuu neuroneista, joiden välillä on yhteys (kuvio 1). Käsiteltävä data saapuu verkon ensimmäisiin neuroneihin eli niin kutsuttuun syöte- eli sisääntulokerrokseen.



Kuvio 1. Neuroverkon rakenne.

Syötekerroksen neuronit käsittelevät saadun datan ja siirtävät sen seuraavalle kerrokselle. Tätä seuraavaa kerrosta kutustaan piilokerrokseksi ja niitä voi olla useita. Piilokerros käsittelee datan ja siirtää sen lopulta ulostulokerrokselle. Ulostulokerroksesta saatavaa tulosta kutsutaan vasteeksi. Kerrosten väliset yhteydet ovat usein kerrosten neuronien välisiä yhteyksiä. Yksinkertaisin yhteystyyppi on niin kutsuttu tiivisyhteys, jossa tarkasteltavan kerroksen jokaisella neutronilla on yhteys jokaiseen aikaisemman kerroksen neutroniin. (Han & Shi, 2008; Graupe, 2013, s.1–6)

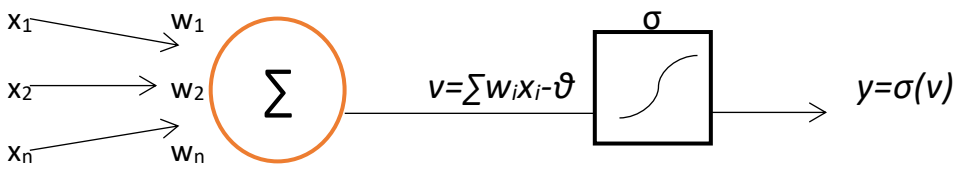
Itse neuroni on yksikkö, joka ottaa vastaan syötteen eli sisääntulon ja tuottaa tästä sisääntulosta ulostulon eli vasteen. Vaste saadaan laskemalla sisääntulokerroksen tai piilokerroksen syötteiden painotettu summa, johon lisätään vakiotermi eli kynnysarvo ja tämä summa käsitellään aktivointifunktiossa. Summan laskemisessa käytetyt painoarvot ja vakioarvot aiheuttavat sen että neuroverkko käyttäytyy ennakoitavalla tavalla. Opetusvaiheessa syntyvä verkko kuvastaa datasta löytyneitä säännönmukaisuuksia ja riippuvuuksia eli on yleistänyt sille annetun datakehyksen. Tästä johtuen neuroverkko kykenee antamaan siedettävän arvauksen vasteeksi, sellaisillakin arvoilla, joita ei ole aiemmin verkolle esitelty. (Han & Shi, 2008; Graupe, 2013 s.1–6)

Neuroverkot voidaan jakaa verkon koon mukaan "mataliin" ja syviin neuroverkkoihin (*DNN Deep Neural Network).* Matalissa verkoissa on vain kolme kerrosta neuroneita: sisääntulokerros, piilokerros sekä ulostulo. Syvissä verkoissa on näiden kerrosten lisäksi vielä kaksi tai useampi piilokerros. Verrattuna mataliin verkkoihin, syvät verkot ovat tarkempia ennustuksissaan. (Graupe, 2013, s.1–6)

3.1.1 Perceptronit ja oppiminen

Perceptron on neuroni, joita voidaan yhdistää toisiinsa verkkomaiseksi rakenteeksi. Yhdistellessä perceptroneja kuvion 2 (s. 20) kaltaiseksi rakenteeksi, saadaan aikaan MLP-verkko (*MultiLayer Perceptron)*. Yksinkertaisimmillaan MPL-verkko koostuu yhdestä sisääntulokerroksesta, piilokerroksesta ja ulostulokerroksesta. MLP-verkossa

syöte on reaalilukuvektori $x= [x_1,x_2 \dots x_n]$, jonka jokaiselle arvolle x_i perceptron antaa painoarvon w_i. Tämän jälkeen neuroni laskee syötteen painotettu summan $v= \sum w_i k_i - \vartheta$, missä ϑ on vakiotermi. Perceptronin (kuvio 2) tuottama vaste $\sigma(v)$ on aktivointifunktion σ tuottama arvo summalle v. (Graupe, 2013, s.11–12,17–18)



Kuvio 2. Perceptronin rakenne.

MLP-verkon oppiminen perustuu perceptronien antamien painojen arvoihin. Aluksi painojen koko arvioidaan vastaamaan jotakin arvoa, mikäli aloitusarvoa ei ole ennalta määrätty. Koska painot kuvaavat neuronien välistä yhteyttä, alun painoarvoja muutetaan opettamisen aikana. Yhteyden ollessa vahva, seuraavan kerroksen neuroni todennäköisesti aktivoituu syötteestä ja tämä neuronien aktivoituminen jatkuu ulostulokerrokseen asti. Negatiivinen paino taasen vaimentaa neuronin ulostuloa. Opettamisvaiheessa mallin ennustamia ulostuloja vertaillaan annettuihin vastaavilla syötteillä saatuihin oikeisiin ulostuloihin. Mikäli verkon laskema arvo ja annettu ulostulo eroavat toisistaan kynnysarvon verran, syntyy niin kutsuttu virhe. Verkko huomioi jokaisen virheen ja muuttaa tarvittaessa painoja tarkoituksena minimoida kyseinen havaittu virhe. Oppimisnopeuteen ja tarkkuuteen vaikuttaa piilokerrosten määrä. Parhaiten toimiva kerrosten määrä löytyy useimmiten yrityksen ja erehdyksen kautta. (Han & Shi, 2008; Karimi ja muut, 2012; Graupe, 2013, s.11–12,17–18)

Jokaiseen neuroverkon kerrokseen lisätään niin kutsuttu biasneuroni. Tämän neuronin avulla on mahdollista siirtää perceptron -neuronin ulostulofunktiota

koordinaatistossa ylös, alas tai sivuille. (Ulostulofunktio saadaan aktivointifunktion avulla. Katso seuraava kappale). Funktion siirrolla on mahdollista saada haluttu ulostulo. Biasneuronille annetaan usein arvoksi 1,0. (Graupe, 2013, s.11–12,17–18)

3.1.2 Hyperparametrit

Hyperparametrit määrittelevät neuroverkon rakenteen, oppimisnopeuden ja kuinka verkon osat toimivat. Näille parametreille annetaan arvo manuaalisesti, eikä opetettava verkko muuta niitä. Hyperparametrit ovat usein sellaisia muuttujia tai vastaavia, joita on vaikea optimoida tai sen arvoa ei voida opettaa opetusjoukon avulla. Mikäli kuitenkin halutaan käsitellä sellaisia hyperparametreja, joiden optimaalinen arvo saadaan kerätystä aineistosta (esimerkiksi kapasiteetti kts 3.4.3) voidaan opetusjoukosta irrottaa erillinen validointijoukko. Tällä joukolla voidaan arvioida verkon virhettä opetuksen jälkeen ja sen aikana, jolloin hyperparametrejä voidaan päivittää tarvittaessa. (Graupe, 2013, s.1–6,11–12,17–18)

3.2 Aktivointifunktio

Aktivointifunktio määrittelee verkon ulostulon sekä laskennallisen tehokkuuden. Ennen kuin neuronin ulostulo voidaan siirtää seuraavan kerrokseen tai verkon lopulliseksi vasteeksi, painotettu summa siirretään aktivointifunktioon. Syvissä neuroverkoissa syötteen kulkiessa verkon läpi, se käsitellään erikseen jokaisen neuronin aktivointifunktiossa, kunnes ulostulokerros tuottaa siitä vasteen. Aktivointifunktio antaa ulostulonaan arvoja väliltä 0 ja 1, tai -1 ja 1. (Graupe, 2013, s.18–21,37)

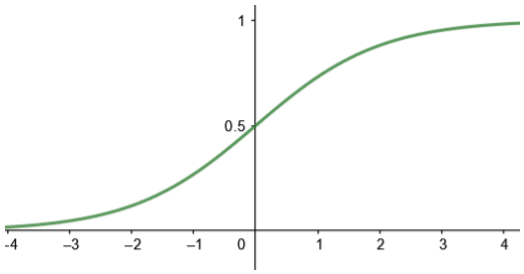
Aktivointifunktio vaikuttaa muun muassa virhefunktion minimointiin sekä verkon oppimisnopeuteen. Virhefunktion minimoinnissa (esimerkiksi backpropagation algoritmi) käytetään apuna aktivointifunktion derivaattaa. Mikäli tällaista derivaattaa ei ole saatavilla, täytyy virhefunktion minimoinnissa käyttää ei-gradienttiin perustuvia

keinoja. Neuroverkon oppiminen on nopeinta silloin, kun aktivointifunktiolla on samoja ominaisuuksia kuin funktiolla, jota halutaan mallintaa. (Graupe, 2013, s.18–21,37)

3.2.1 Sigmoid-funktio

Sigmoid- eli logistinen funktio, on aidosti kasvava sekä rajoitettu funktio, jolla voidaan kuvata S-mallista käyrää koordinaatistossa (kuvio 3).

$$A = \frac{1}{1+e^{-x}} \quad (1)$$



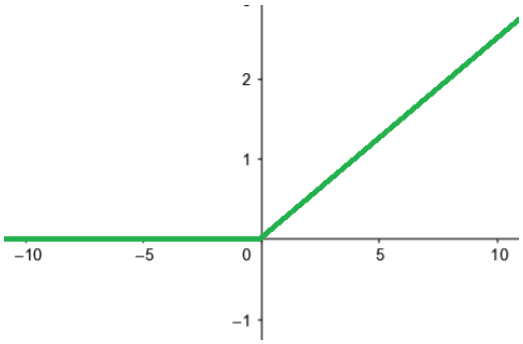
Kuvio 3. Sigmoid-funktion esitys koordinaatistossa.

Sigmoid-funktio ei ole symmetrinen nollan suhteen. Funktio kasvaa ja vähenee hyvin hitaasti eli se reagoi hitaasti vastaaviin x:n muutoksiin. Tästä johtuen verkko, jossa käytetään sigmoid-funktiota, oppii hitaasti. Sigmoid-funktion arvot ovat tyypillisesti välillä [0,1]. Tästä ominaisuudesta johtuen funktiota käytetään usein ulostulokerroksessa luokittelemaan esimerkiksi esineitä. (Graupe, 2013, s.19; Bernico, 2018, s.8)

3.2.2 ReLu

ReLu eli *Rectified Linear Unit* käytetään usein neuroverkon piilokerroksissa. ReLu-funktiolla saadaan aikaan epälineaarisuutta verkon toiminnassa.

$$A(x) = \max(0, x) \tag{2}$$



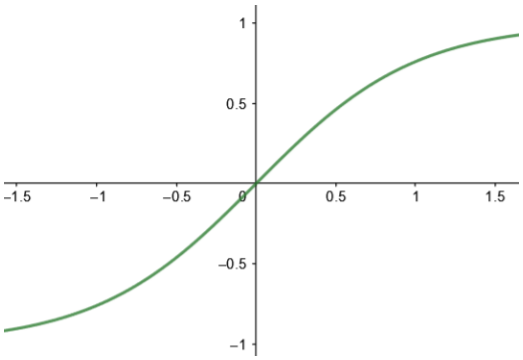
Kuvio 4. ReLu -funktion esitys koordinaatistossa.

ReLun selkeitä huonoja puolia ovat ettei se derivoidu nollassa sekä että funktion arvot ja derivaatat ovat nollia negatiivisilla arvoilla (kuvio 4). Tästä seuraa että, joidenkin neuronien painot muuttuvat oppimisvaiheessa nollaksi eli neuronit eivät aktivoidu. Tähän ongelmaan on kehitetty niin sanottu vuotava ReLu, jossa negatiivinen puoli on kerrottu jollain pienellä luvulla. Tästä seuraa että negatiiviseen suuntaan kuljettaessa tapahtuu kyllä aktivointia, mutta se on huomattavasti vähäisempää verrattuna positiiviseen suuntaan. (Bernico, 2018, s. 10)

3.2.2 Hyperbolinen tangentti

Hyperbolinen tangentti on sigmoid-funktion tapaan aidosti kasvava ja jatkuva funktio. Sigmoid-funktiosta poiketen se on symmetrinen nollan suhteen. Hyperbolinen tangentti saa arvoja välillä [-1,1] (kuvio 5). Hyperbolista tangenttia voidaan käyttää tilanteissa, joissa syötteiden arvot asettuvat selkeästi, joka positiivisiksi, negatiivisiksi tai neutraaleiksi. (Bernico, 2018, s.9–10)

$$A = \frac{1-e^{-2x}}{1+e^{-2x}} \quad (3)$$



Kuvio 5. Hyperbolisen tangentin esitys koordinaatistossa.

3.3 Virhefunktion pienentäminen

Neuroverkon painojen minimointi merkitsee virhefunktion (*loss function, cost function*) pienenemistä. Virhefunktio kuvaa neuroverkossa esiintyvää virhettä ja sitä voidaan käyttää kuvaamaan verkon tarkkuutta annetussa tehtävässä. Alhainen virhefunktion arvo kertoo neuroverkon kykenevän mallintamaan eli oppineen ilmiön hyvin. Virhefunktion minimoimiseen voidaan käyttää useita eri menetelmiä, mutta tässä työssä esitellään backpropagation (*vastavirta*) algoritmi sekä gradienttimenetelmä. Muita mahdollisia keinoja virhefunktion pienentämiseen olisi esimerkiksi brute force -taktiikka, jossa verkon painoja lähdetään muuttamaan kokeilemalla. Tämä tapa on erityisen työläs jo pienillä verkoilla. (Bernico, 2018, s.11)

Virhefunktion pienentäminen tapahtuu neljässä vaiheessa. Ensimmäisessä vaiheessa verkko alustetaan ja verkko saa satunnaiset painoarvot. Tämän jälkeen verkon sisääntuloon syötetään harjoitusdata. Data kuljetetaan verkon läpi (*forward pass*) ulostulolle asti. Lopulta saadaan ensimmäinen ennuste. Toisessa vaiheessa virheen arvo lasketaan ensimmäisen ennusteen ja opetusjoukon arvojen välillä. Kolmannessa vaiheessa algoritmi lähtee käsittelemään verkon painoja virheen pienentämiseksi.

Viimeisessä vaiheessa verkon painot päivitetään. Verkon painot voidaan päivittää tietyin väliajoin esimerkiksi joka viidennen otoksen jälkeen. (Bernico, 2018, s.11–12)

3.3.1 Backpropagation algoritmi

Backpropagation algoritmi (kuva 1) on yleisin neuroverkkojen opettamiseen käytettävä algoritmi. Tämä algoritmi mahdollistaa oppimisen syöttämällä virheen takaisin verkolle ja muuttamalla painojen arvoa tavoitteena pienentää virhettä (Han & Shi, 2008). Backpropagation algoritmissa verkolle syötetään datapiste, joka tuottaa verkossa vasteen. Vasteen arvoon vaikuttavat neuroneissa olevien painojen sen hetkiset arvot. Ensimmäinen vaste on usein täysin satunnainen (Han & Shi, 2008; Graupe, 2013, s.59–63; Bernico, 2018, s.12). Saatua vastetta vertaillaan vastaavien syötteiden oikeaan ulostuloon. Vertailusta saadaan oikean ulostulon ja verkon vasteen välinen virhe. Tämä virhe välitetään takaisin verkolle ja siihen perustuen painoja muutetaan:

$$E(w) = \frac{1}{2}\sum_{k=1}^{n}(t_k - y_k)^2 \quad (4)$$

,missä $E(w)$ = virhefunktio
w = kerrospainot w_{ij} sisältävä matriisi
t_k =haluttu vaste
y_k =verkon antama vaste syötteelle x_k.

Painojen muutoksella on tarkoitus vähentää syötteiden aiheuttaman vasteen ja halutun ulostulon välistä virhettä. Edellä kuvattua prosessia toistetaan jokaiselle opetusjoukon pisteelle. Opetusjoukkoa käydään läpi uudestaan ja uudestaan kunnes saatu virhe on pienempi kuin virheelle asetettu kynnysarvo tai kunnes opetusjoukko on käyty halutun monta kertaa läpi. (Han & Shi, 2008; Graupe, 2013, s.59–63; Bernico, 2018, s.12)

Backpropagation algoritmi:

1. Alustetaan verkon kaikki painot w_{ij}
2. Annetaan verkolle sisääntulovektori X
   ja sitä vastaava ulostulo
3. Lasketaan ulostulo Y vektorille X
4. Manipuloidaan neuroneiden painoja:

$$w_{ij}(t+1) = w_{ij}(t) + \mu\delta_i X_i,$$

   missä μ on oppimisnopeus,
   t on vaste,
   j on neuroni, jos j on ulostulokerroksessa niin:

$$\delta_i = Y_i(1 - Y_i)(t_i - Y_i)$$

   jos j on piilokerroksessa niin:

$$\delta_i = X_i(1 - X_i)\sum_k \delta_k w_{jk}$$

5. Painojen manipulointi lopetetaan kun Y ja X vastaavan ulostulon välinen virhe alittaa virheen raja-arvon

Kuva 1. Backpropagation algoritmin pseudokielinen toiminta. (mukaillen Graupe, 2013, s.59–63; Bernico, 2018, s.12–15)

3.3.2 Gradienttimenetelmä

Gradienttimenetelmässä (*gradient descent*) funktion gradientti antaa kohdan, jossa funktion kasvu on jyrkimmillään, sen vastavektoria tarkastelemalla löydetään kohta, jossa funktion arvon pieneminen on nopeinta. Menetelmässä verkon alkupainot ovat satunnaisia. Tarkoituksena on testata yksittäisiä painoja muuttamalla milloin saadaan ulostulo, joka vastaa parhaiten toivottua vastetta. Tarkoituksena on siis juurikin virhefunktion minimoiminen, mutta lisäksi yritetään välttää tilannetta, jossa virhefunktion liiallinen pieneminen aiheuttaisi ylisovittamista (katso kappale 3.4.1 Ylisovittaminen). Tällaista pistettä, missä virhefunktion arvo on pienemmillään, kutsutaan

paikalliseksi minimiksi. Minimi löytyy vastavektorin osoittamasta suunnasta. (Bernico, 2018, s.12–13)

Gradienttimenetelmässä opetusjoukon yksittäistä jäsentä eli alkiota x^i vastaa virhe $\varepsilon_i(x^i)$. Tämä virhe lasketaan jokaisen alkion jälkeen, jolloin virhefunktio on koko opetusjoukon virheiden summa:

$$\varepsilon = \frac{1}{N}\sum_{i=1}^{N} \varepsilon_i \quad (5)$$

,missä N on alkioiden lukumäärä
ε_i on opetusjoukon alkion aiheuttama virhe.

Opetusjoukon läpikäynnin jälkeen, verkon parametrit päivitetään. Käytännössä tämä tarkoitta, että opetusjoukon tiedot ovat kerralla "muistissa". Tästä aiheutuu, erityisesti isoilla opetusjoukoilla, verkko oppii hitaasti. Yleisin gradienttimenetelmän ongelma on päätyminen liian pieneen paikalliseen minimiin, jolloin saatu lopputulos ei ole paras mahdollinen. Gradientti menetelmää käytetään usein yhdessä backpropagation algoritmin kanssa, jolloin syötettä vastaavat virhefunktion arvot lasketaan backpropagation algoritmin avulla, jonka jälkeen gradienttimenetelmää käytetään apuna korjattaessa neuroni kohtaisia parametrejä. (Bernico, 2018, s.12–13)

3.4 Neuroverkon opettaminen

Neuroverkon opettamisen tarkoituksena on luoda malli, joka kykenee oppimaan halutun ilmiön ja yleistämään oppimansa uuteen dataan. Opettamisen aikana yritetään löytää tasapainovinouman (*bias*) ja varianssin välillä. Vinoumalla mitataan, kuinka luotu verkko kykenee mallintamaan opetusjoukon. Testattaessa uudella datajoukolla, varianssi kertoo, kuinka hyvä verkon kyky luokitella on. (Nielsen, 2015)

Neuroverkon oppimiskykyyn vaikuttaa selkeimmin verkon kerrosten lukumäärä ja neuronien lukumäärä. Itse oppimiskyky on neuroverkon painojen valinta

mahdollisimman optimaalisesti. Oppimisnopeus taasen kertoo painojen muutoksen suuruuden. Liian suuri oppimisnopeus saattaa tehdä verkosta epävakaan. Liian nopean oppimisnopeuden vaarana on myös, että optimaalinen kohta voidaan ohittaa. Oppimisnopeuden taasen ollessa liian pieni, on oppiminen paitsi hidasta ja vaarana on prosessin jumiutuminen tiettyyn kohtaan, yleensä paikalliseen minimiin.

Mallin opettamista varten aineisto voidaan jakaa testi- ja opetusjoukkoon. Opetusjoukko on yleisemmin yli puolet koko aineistosta. Opetusjoukko voidaan vielä lisäksi jakaa useampaan osaan. Opetusjoukon jakamisen tarkoituksena on testata mallin toimivuutta lisäämällä ajoittain toisesta opetusjoukosta mallille uutta dataa käytettävän opetusjoukon mukaan. Testijoukko pidetään opetusjoukosta erillään, ja sitä käytetään vain valmiin mallin testaamiseen. Tarkoituksena on saada aikaan verkko, joka on kyennyt yleistämään tutkittavan ilmiön ja toimii riittävän tarkasti myös uudelle datalle. (Nielsen, 2015)

3.4.1 Ylisovittaminen

Ylisovittaminen (ylioppiminen) on verkon liiallista mukautumista opetusjoukkoon eli verkon voidaan ajatella oppineen esimerkiksi opetusjoukon luokkarajat liian tarkasti, jolloin verkko ei enää kykenekään käsittelemään testijoukkoa sallitun virheen rajoissa. Ylisovittamista voidaan välttää kasvattamalla opetusjoukkoa tai muuntelemalla joukkoa lisäämällä siihen dataa. Käytettävän datamäärän ollessa rajallinen voidaan myös käyttää niin kutsuttua ennen aikaisen pysähdyksen tapaa. Tässä tavassa verkon opettaminen lopetetaan heti kun sen tarkkuus lähtee pienenemään. Ideana on että verkon tarkkuuden tarkastelu sopivien osajoukkojen välein, auttaa havaitsemaan sen pisteen missä ylioppiminen alkaa. (Bernico, 2018, s.64; Nielsen, 2015)

Dataan kohdistuvien menetelmien lisäksi voidaan puuttua myös itse neuroverkon rakenteeseen yrittäessä poistaa ylisovittamisen ongelmaa. Neuroneita voidaan poistaa väliaikaisesti verkosta ja palauttaa myöhemmässä vaiheessa. Neuroneiden

poistamisen jälkeen verkon opettamista jatketaan tavalliseen tapaan. Opettamisen päätteeksi neuronit palautetaan verkkoon ja mahdollisesti seuraavat neuronit poistetaan uuden opetuskierroksen ajaksi. Testausvaiheessa verkko on jälleen kokonainen. (Nielsen, 2015)

3.4.2 Alisovittaminen

Alioppimisessa verkko ei kykene mallintamaan opetusjoukkoa. Ongelmaan voidaan puuttua lisäämällä neuronien määrää. Tällöin verkon ennustamisprosessi monimutkaistuu ja alioppiminen vähenee. Neuronien määrän lisäksi, myös sisääntulojen määrää voidaan kasvattaa saman vaikutuksen aikaan saamiseksi. (Nielsen, 2015; Elmusrati, 2018). Myös käytettävän datamäärän kasvattaminen, jolloin käsiteltävä aineisto vastaa paremmin tutkittavaa ilmiötä, voi auttaa vähentämään alioppimisen vaaraa (Goodfellow ja muut ,2016, s. 109).

3.4.3 Yli- ja alisovittamisen ehkäiseminen kapasiteettia kasvattamalla

Edellä kuvattujen keinojen lisäksi, yli- ja alisovittamista voidaan ehkäistä muuttamalla verkon kapasiteettia eli pätevyyttä (*capacity*). Käytännössä käytettävän mallin kapasiteetti tarkoitta sen kykyä käyttää erilaisia funktioita ratkaisun löytämisessä. Malleilla, joiden käyttämä funktioiden määrä on pieni, on vaikeuksia yleistää opetusjoukko. Kapasiteetin ollessa korkea, on olemassa riski, että malli ylioppii ja testausjoukon aikainen virhe kasvaa. (Goodfellow ja muut ,2016, s. 107–112)

Verkon kapasiteettiin vaikutetaan muuttamalla oppimiseen käytettävän algoritmin hypoteesiavaruutta (*hypothesis space)*. Hypoteesiavaruus on funktioiden joukko, josta algoritmi etsii ratkaisua. Esimerkkinä hypoteesiavaruuden laajentamisesta voidaan ajatella lineaarista algoritmia, joka etsii ratkaisua lineaaristen funktioiden joukosta. Lineaarinen algoritmi voidaan yleistää siten, että se kykenee käsittelemään

myös polynomeja. Näin hypoteesiavaruuteen lisätään myös polynomi muotoiset funktiot, jolloin algoritmin kapasiteetti kasvaa. *)* (Goodfellow ja muut, 2016, s. 110)

Goodfellow väittää kirjassaan Deep Learning Book (Goodfellow ja muut, 2016, s. 110), että tekoälyyn pohjautuvat algoritmit toimivat parhaiten kun niiden kapasiteetti vastaa tutkittavan ilmiön kompleksisuutta. Mallit, joiden kapasiteetti on pieni, eivät kykene yleistämään monimutkaisia ilmiöitä. Vastakohtana ovat suurikapasiteettiset mallit, jotka kykenevät ratkaisemaan monimutkaisia tehtäviä, epäonnistuvat taasen yksinkertaisten ongelmien ratkaisemisessa. Hypoteesiavaruuden ollessa tarpeettoman suuri, seurauksena on ylisovittaminen.

4 Aikaisempi tutkimus

Veden pinnan korkeuden muutoksen arviointi pohjautuu perinteisesti tilastotieteeseen. Ensimmäinen edelleen käytössä oleva ennustamisen metodi on 1900-luvun alusta. Tuolloin Gauss ja Legendre julkaisivat paperin, joka sisälsi ensimmäisen lineaarisen regression mallin. Lineaarista regressiota on myöhemmin käytetty tilastotieteeseen pohjautuvissa algoritmeissa kuin myös tekoälyyn pohjautuvissa malleissa.

Tietotekniikan kehittyminen on luonut mahdollisuuden datan automaattiseen keräämiseen, suurten datamassojen säilyttämiseen ja käsittelemiseen aiempaa kustannustehokkaammin. Tämän kehityksen johdosta nykyinen veden korkeuden ennustamiseen liittyvä tutkimus onkin keskittynyt datapohjaisten mallien suunnitteluun ja kehittämiseen.

4.1 Hydrodynaamiset- ja numeeriset mallit

Hydrodynaamiset ja numeeriset vedenkorkeuden ennustamisen mallit perustuvat fysiikan lakien mukaisiin prosesseihin sekä niiden numeeriseen esittämiseen. Niiden on tutkimuksissa havaittu edellyttävät lähes aina tarkkaa paikkasidonnaista dataa, eivätkä siis ole aina skaalautuvia datan keräyspisteen ulkopuoliselle alueelle. Kuitenkin numeeristen mallien tarvitsemaa dataa pitää usein kerätä vuosia, jotta luotettavien tulosten saaminen olisi mahdollista. (Han & Shi, 2008; Karimi ja muut, 2012). Numeeristen mallien tulosten tarkkuus on riippuvainen muun muassa käytettyjen mittauslaitteiden tarkkuudesta, mitattavien ilmiöiden parametrisoinnista ja reunaehdoista (Han & Shi, 2008).

Hydrodynaamiset mallit perustuvat veden kulkeutumisen arviointiin, sadantaan ja näiden muutoksiin. Sade ja erityisesti sademäärä ovat harvoin samoja koko tarkasteltavalla alueella. Usein on niin että sade on rankimmillaan vain yhdellä kapealla alueella ja muut alueet pysyvät lähes kuivina. Sadepilvillä on myös tapana liikkua tuulen mukana. Myös maaperän laatu, korkeus ja maankäyttö vaihtelee. Lisäksi hydrodynaamiset ja

numeeriset mallit vaativat aikaa (Devi ja muut, 2015). Näin ollen nämä mallit eivät kykene ennustamaan nopeaa muutosta ja tutkiminen vaatii aikaa. Tutkimus onkin siirtynyt hydrodynaamisista malleista kohti datapohjaisia malleja.

Yksi esimerkki on hydronumeerisesta mallista on niin kutsuttu korrelaatiomalli, jonka Karimi mainitsee tutkimuksessaan (Karimi ja muut, 2012). Tutkimuksessa todetaan, että mallia käytettäessä tarvitaan tunnittaisia mittauksia vähintään 29-päivän ajalta, jotta tuloksia voitaisiin pitää edes jotenkin luotettavina. Käytännössä tietoa täytyy kerätä pidemmältä aikaväliltä, jotta voidaan minimoidaan tuulen, myrskypuuskien ja ilmanpaineen satunnaisvaikutus mittaustuloksiin (Karimi ja muut, 2012). Korrelaatiomalli esitetään yleensä diagrammilla, josta nähdään kahden tai useamman muuttujan välinen korrelaatio. Tällainen diagrammi on usein yksinkertainen ja helposti ymmärrettävissä ja tutkimuksen mukaan se sopii käytettäväksi alueille, joissa ei ole varsinaista tulvan hallintaan ja tarkkailuun perehtynyttä henkilöstöä. Korrelaatiomallin haittana on ettei se varsinaisesti kerro muuta kuin joen veden pinnan mahdollisesta muutoksen suuntaa. Harvoin muuttujien välistä korrelaatiota on laskettu. Vaikka korrelaatiomalli onkin hyvin yksinkertainen ja karkea, se sopii ympäristöihin, jotka ovat syrjässä sää- ja mittausasemilta sekä aluille, joissa ei ole mahdollisuutta kalliiseen tekniikkaan.

Edellä kuvattua vastaavan kaltainen malli on yhtälömalli. Tässä mallissa on määriteltynä muuttujien välinen suhde regression avulla. Tämä helpottaa tarkasteltavien muuttujien valitsemista ja näin ollen nopeuttaa ja parantaa ennusteen tarkkuutta verrattuna korrelaatiomalliin, todetaan Singhin tutkimuksessa (Singh, V.P., 2018). Muita veden pinnan korkeuden muutoksia ennustavia numeerisia malleja ovat: Least Square metodi, jossa pinnan korkeus määritellään minimoimalla mitatun korkeuden ja sinusoidisen funktion välillä (Karimi ja muut, 2012); sekä lukuisat tietokonepohjaiset tulvan mallintamiseen käytettävät ohjelmat. Yksi tällainen ohjelma on WMS (water modeling system). Jotta esimerkiksi WMS:llä saadaan luotettavia tuloksia, ohjelmaan syötetään tiedot muun muassa alueen maaperästä, korkeuseroista sekä sateista. Tämän jälkeen ohjelma luo

simulaation, jonka pohjalta voidaan ennustaa tuleva muutos vesialueiden pinnan korkeuksissa. (Singh, V.P., 2018)

4.2 Datapohjaiset mallit

Datapohjaiset mallit sopivat epälineaaristen ilmiöiden mallintamiseen. Nämä mallit kykenevät yhdistelemään tietoa, joka sisältää lukuisia muuttujia ja luomaan siitä yleistyksen, jonka avulla voidaan ennustaa joen pinnan muutoksia. Ennustaminen tapahtuu tarkkailemalla muuttujissa tapahtuvia muutoksia ja yleistämällä nämä muutokset koskemaan joen pinnan muutoksia. (Han & Shi, 2008; Devi ja muut, 2015)

Tässä luvussa käsitellään veden pinnan korkeuden tai virtaaman ennustamiseen käytettäviä malleja, jotka eivät ole neuroverkkomalleja tai sisältävät muutakin kuin pelkästään neuroverkon.

4.2.1 Asiantuntijajärjestelmä ja sumea logiikka

Cluckie, Moghaddamnia sekä Han esittelevät kirjassa Practical Hydroinformatics (Cluckie ja muut, 2008, s. 201–213) prototyypin, joka hyödyntää sekä asiantuntijajärjestelmää (*expert system*) että sumeaa logiikkaa. Heidän tavoitteenaan oli luoda järjestelmä, joka asiantuntijajärjestelmän tavoin kykenisi sumean logiikan luomien päättelysääntöjen avulla jäljittelemään vastaavan henkilön tietämystä ja päätöksiä. He hyödynsivät sumeaa logiikkaa käsitellessään ylä- ja alajuoksuilta kerättyä tietoa. Kerätystä datasta muodostettiin "jos - niin sillon” -muotoisia ehtolauseita, jotka kerättiin tietokantaan. Nämä ehtolauseet edustavat syötteiden tilaa sekä niihin liitettäviä vasteita. Tätä tietokantaa käytettiin asiantuntijajärjestelmän pohjana. Kirjoittajien luoma prototyyppi kykeni toimimaan reaaliaikaisesti, mutta vaati tietokantaan käytettävän tiedon luotettavuuden varmistamista sekä päivittämistä. Kirjoittajat

toteavat järjestelmän luotettavuuden lisääntyvän kerätyn datamäärän mukana, mutta toisaalta tiedon käsittely mallia varten hidastuu.

4.2.2 ANFIS

Veden pinnan muutoksia on pyritty ennustamaan muun muassa sumeaa logiikkaa ja neuroverkkoja yhdistelevällä tekniikalla (Kaloop ja muut, 2017). Kaloopin, El-Diastyn ja Hun esittämällä ANFIS (*adaptive neuro-fuzzy inference system*) -mallilla ennustetaan veden pinnan muutoksia, silloin kun käytettävissä oleva data on lyhyeltä aikaväliltä. Tässä mallissa oppimiseen vaikuttavia parametrejä ohjataan sumean logiikan avulla.

ANFIS on epälineaarinen ja se yhdistää sekä neuroverkkoa että sumeaa logiikkaa, mikä tekee siitä eräänlaisen hybridimallin. Tutkimuksen mukaan yhdistäminen tapahtuu parhaiten käyttämällä neuroverkkojen opetusalgoritmia sumeiden JOS-NIIN SITTEN sääntöjen luomiseen. Tutkijoiden mallissa opettaminen tapahtui antamalla neuroverkolle harjoitusdatasta esimerkkejä, joiden ja haluttujen ulostulojen perusteella, verkko säätää painoarvot saavuttaakseen oikean vasteen. ANFIS eroaa tavallisesta neuroverkosta siinä, että se käyttää heuristista tietoa sääntöjen luomiseen ja säätää näitä sääntöjä datan perusteella. Vastaavasti tyypillinen neuroverkko luo sääntöjä vain datan pohjalta. (Kaloop ja muut, 2017)

4.2.3 SVR ja GA

Yksi esimerkki datapohjaisista malleista on SVR (support vector regression), jota käytetään ympäristön mallintamiseen (Karimi ja muut, 2012). SVR käyttää srm:ksi (structural risk minimization) kutsuttua tapaa, jolla pyritään minimoimaan sekä mallin monimutkaisuuden ja opetuksen aiheuttamat virheet. Voidaankin sanoa, että SVR yleistää

voimakkaasti tutkittavaa ilmiötä. SVR:ssä on vain muutamia vapaita parametrejä. SVR opettaminen vaatii kuitenkin aikaa, erityisesti isoilla datamäärillä.

Vuonna 2018 julkaistussa artikkelissa (Hu ja muut, 2018) käsiteltiin mahdollisuutta yhdistää SVR (*support vector regression*) sekä geneettiset algoritmin jokien ja järvien pinnan korkeuden ennustamiseen. SVR-metodia käytetään sisään-ulostulojen kartoittamiseen aineistosta ja sitä hyödynnetään ennustamiseen käytettävän mallin luomisessa. Kuten kaikissa dataan pohjautuvissa malleissa, kirjoittajat toteavat, että haasteeksi tulee nousemaan sellaisten sisääntulojen ja parametrien löytämien, jotka auttavat kuvaamaan tutkittavaa ilmiötä parhaiten. Tämän vuoksi he ehdottavat geneettistä algoritmia (GA) käytettävän parametrien etsintään. Geneettistä algoritmia käytetään hakualgoritmin lailla. Algoritmi etsii annetusta data-avaruudesta mahdollisimman optimaalisen sisääntulojen joukon. Raportissa mainitaan että geneettisellä algoritmilla haetut parametrit auttavat tehostamaan SVR:ää ja mahdollisesti taktiikkaa voitaisiin käyttää muihinkin ennustamismenetelmiin.

4.3 Veden korkeuden ennustaminen neuroverkoilla

Veden korkeuden ennustaminen neuroverkon avulla on yleistynyt samaa tahtia tekoälyn kehittymisen kanssa. Sopivalla aineistolla ja ratkaisuilla on kyetty rakentamaan ratkaisuja, joita voidaan hyödyntää kaupallisesti. Tulevaisuudessa tekniikan kehittymisen myötä ja datamäärän kasvaessa, veden pinnan muutoksia voidaan ennustaa myös alueilla, joissa nykyisen kaltainen ennustaminen ei toimi esimerkiksi alueen maaston vuoksi.

4.3.1 Meriveden korkeuden ennustaminen

Guoqi Han ja Yu Shi käsittelevät vuonna 2008 julkaistussa tutkimuksessaan (Han & Shi, 2008: *Development of an Atlantic Canadian Coastal Water Level Neural Network*

Model) neuroverkon käyttöä rannikkovesien pinnan korkeuden ennustamisessa. He toteavat, että koska perinteisten numeeristen mallien tarkkuus on riippuvainen huomioon otettavien tekijöiden määrästä. Tämä tarkoittaisi sitä, että hyvä numeerinen malli sisältäisi niin monta tekijää, että se olisi turhan hidas laskea sekä altis virheille. Tämän vuoksi Han ja Shi ehdottavat, että veden pinnan korkeuden arviointi tulisi perustua havaittujen pinnan korkeuksien muodostamaan säännönmukaisuuteen. Näiden säännönmukaisuuksien havaitsemiseen he ehdottavat ratkaisuksi neuroverkkoja. Neuroverkkojen käyttöä puoltaa tarkasteltavan ongelman luonne: meriveden korkeus riippuu muun muassa lämpötilasta, suolapituisuuksista, aallokosta sekä merivirroista.

Tutkimuksessaan Han ja Shi (Han & Shi, 2018) loivat neuroverkon, jonka opettamiseen he käyttivät backpropagation algoritmia. Syyksi mainitaan sillä opettamisen helppous. Algoritmi oppii syöte-vasteparien avulla halutunlaista käyttäytymismallia. Tutkijoiden mukaan esimerkkiparien avulla luotu malli kykenee ennustamaan ulostulon annetusta uudesta sisääntulosta ”kohtalaisen hyvin”. Parantaakseen tuloksia, Han ja Shi opettivat samanaikaisesti useita toisistaan eroavia neuroverkkoja samoilla syöte-vastepareilla. Näin saadut neuroverkot yhdistettiin yhdeksi kokonaisuudeksi, jonka kyky yleistää aineisto oli parempi kuin keskityttäessä vain yhden verkkorakenteen luomiseen. Käytettävä data oli saatu alueen tutkimusasemilta.

Valmiin mallin eduksi mainittiin sen kyky ennustaa suhteellisen tarkasti veden korkeutta, vaikka käytettävissä oleva data olisi vain lyhyeltä aikaväliltä. Tutkimuksessaan Han ja Shi (Han & Shi, 2018) mainitsevat, että malli sopisi käytettäväksi liikuteltavien tutkimusasemien kanssa ja näin ollen olisi sopiva myös muiden alueiden vesivarantojen pinnan korkeuden ennustamiseen.

4.3.2 Joen pinnan korkeuden ennustaminen meren läheisyydessä

Application of Artificial Neural Network into the Water Level Modeling and Forecast -nimisessä tutkimuksessaan Sztobryn kuvailee neuroverkon sovellusta, jolla voidaan

ennustaa tietyssä joen pisteessä tapahtuvaa veden pinnan korkeuden muutosta (Sztrobyn, 2013). Sztobryn tarkastelee tilannetta, jossa pieni satama sijaitsee 3 kilometriä joen suulta, jolloin meriveden pinnan korkeus täytyy myös huomioida yhdessä tulva-aallon kanssa. Tutkimuksessa käytettävä data oli kerätty satama-alueelta kolmen vuoden ajalta.

Tarkasteltaessa ja käsiteltäessä dataa neuroverkkoa varten, Sztobryn huomauttaa käsiteltävän aineiston datapisteiden osuvan lähes täysin tietylle pienelle välille (Sztrobyn, 2013). Hän jatkaa, että mikäli tutkimuksessa keskityttäisiin aineistoon sellaisenaan, vaarana olisi että luotu neuroverkkomalli ei kykenisi ennustamaan tarkasti tämän välin ulkopuolelta. Tämä seurauksena Sztobryn päätti tarkastella vain sitä osuutta aineistosta, joka sisältää keskimääräistä korkeampia veden pinnan korkeuksia. Perusteluna hän käyttää, että juuri tavanomaista korkeammat pinnan korkeudet muodostavat suurimman uhan satamalle. Vähentämällä käytettävän datan määrää, Sztobryn tarkoituksena oli myös löytää optimaalinen syötevektori neuroverkolle eli juuri ne syötteet, joilla olisi kaikkein eniten merkitystä.

Tutkimuksessaan Sztobryn testasi 500 erillistä neuroverkkoa, joista hän valitsi viisi parasta esiteltäväksi tarkemmin (Sztrobyn, 2013). Kaikille näille viidelle verkolla yhteistä oli että ne käyttivät aktivointifunktiona hyperbolista tangenttia piilokerroksissa ja logistista funktiota ulostulokerroksessa. Eritysesti Sztobryn nostaa esille verkon, joka kykenee käsittelemään viiden parametrin sisääntulovektorin. Tämä neuroverkko koostuu viidestä sisääntulo neuronista, yhdestä 34 neuronin piilokerroksesta sekä yhdestä ulostulokerroksessa olevasta neuronista. Kyseinen verkko kykeni ennustamaan normaalitilanteissa veden korkeuden parin senttimetrin erolla mitattuun tulokseen. Tilanteissa, joissa mallintaminen tapahtui voimakkaan myrskyn iskiessä rannikolla, neuroverkko ei kyennyt mallintamaan veden pinnan korkeutta. Sztobryn kuitenkin huomauttaa, että suurimmat erot mallinnetun ja mitatun tuloksen välillä syntyivät meriveden voimakkaasta virtaamisesta uomassa yhdistettynä voimakkaaseen meren

käyntiin, mitä ei voida mallintaa meteorologisten mallien avulla ja joista on vain vähän kerättyä tietoa niiden harvinaisuuden vuoksi.

4.3.3 Sademäärän käyttäminen joen pinnan korkeuden ennustamisessa

Biswas ja Jayawardena (Biswas & Jayawardena, 2014) tutkivat artikkelissaan voidaanko joen pinnan muutoksia ennustaa silloin kun tietoa ilmastosta sekä maantieteeteen vaikutuksista (esimerkiksi sade veden valumareiteistä) ei ole saatavilla tai saatava tieto on puutteellista. Tutkimuksessa tarkasteltavana oli Surma -joki Bangladeshissa. Kirjoittajat toteavat valinneensa neuroverkon olevan sopiva ennustamismenetelmä, sillä kyseessä on epälineaarinen ilmiö ja joen veden korkeuden muutos ei ole suoraan verrannollinen satavan veden määrään. Neuroverkoista käytätettiin MLP (*multilayer perceptron*) verkkoa. MLP valikoitui käytettäväksi, koska vastaavalla rakenteella oli saatu hyviä tuloksia mallinnettaessa kiinalaisia jokia ja niiden pinnan korkeuksia.

Tutkimuksessa (Biswas & Jayawardena, 2014) tarkasteltavaa jokea kuvataan maataloudelle tärkeäksi virraksi, jossa äkkinäiset tulvat ovat hyvin yleisiä monsuunikautta edeltävinä kuukausina. Alue on vuoristoista ja enin osa virtaavasta vedestä valuu Intian puolelta. Tämä aiheuttaa sen että jokeen valuvan veden määrä on vaikea yrittää ennustaa. Toinen haaste on alueen jokisuus. Jokisuus ilmenee toisiinsa yhdistyvien jokien määrällä. Tämä tarkoittaa, että mikäli virtaama joessa nousee, syöksyy vesi uomaa pitkin toiseen jokeen, jolloin tämänkin joen virtaama kasvaa. Biswas ja Jayawardena tarkastelivat jokea vain monsuunia edeltävinä kolmena kuukautena ja monsuunikauden aikana. Näiden kausien ulkopuolella, Surma -joki ei muodosta uhkaa tulvimisen suhteen.

Surma -joen pinnan korkeutta pyrittiin ennustamaan mahdollisimman yksinkertaisella verkolla. Verkon opettamiseen käytettiin backpropagation -algoritmia. Biswas ja Jayawardena (Biswas & Jayawardena, 2014) käyttivät useammasta mittauspisteestä saatuja sademäärän arvoja verkon syötteenä. Analysoidessaan dataa ja mahdollista verkon rakennetta, Biswas ja Jayawardena tulivat päätelmään, että käsiteltävän ilmiön

mallintamiseen sopisi verkko, jossa on yksi piilokerros kahdella neuronilla sekä yksi ulostulo. Sisääntulovektori koostuisi kahdesta sademäärää kuvaavasta arvoista, joista molemmat oli saatu eri mittauspisteistä.

Tutkimuksen (Biswas & Jayawardena, 2014) lopputulema on että joen pinnan korkeuden ennustaminen on mahdollista edellä kuvatulla verkolla. Luodun mallin haasteena on ettei se kyennyt ennustamaan matalia veden korkeuksia tarkasti, vaan pyrki ennustamaan ne todellista korkeammaksi. Tutkijoiden mukaan erityisesti monsuunikautta edeltävien kuukausien aikana tehdyissä ennustuksissa oli virheellisyyttä eli malli ei kykenisi ennustamaan nopeasti alkavia niin sanottuja syöksytulvia tarkasti. Saaduista tuloksista myös huomataan että käytettyjen mittauspisteiden sademäärien ja joen virran korkeuden välillä on lineaarinen yhteys. Tämä tarkoittaa ettei ennustamiseen tarvita tietoa tarkasteltavan joen ympäristöstä, esimerkiksi siihen valuvien vuoristojokien virtaamista, kunhan sademäärät mitataan sellaisilta alueilta, joista sadevettä siirtyy runsaasti tutkittavaan jokialueeseen.

5 Työn suunnittelu

Neuroverkko koostuu pienimmillään sisääntulokerroksesta, yhdestä piilokerroksesta sekä ulostulokerroksesta. Sisääntulokerros on tärkeässä osassa, sillä sen kautta syötetään parametrit, jotka erinäisten laskutoimitusten kautta vaikuttavat ulostuloon eli siihen kuinka hyvin verkko kykenee mallintamaan halutun ilmiön. Vaikka parametrien määrälle ei ole varsinaista ylärajaa, on usein järkevää valita mukaan vain ne muuttujat, jotka vaikuttavat eniten tutkittavan ilmiön mallintamiseen (mukaillen Sztobryn, 2013). Tämä vähentää tarvetta suurelle ja monimutkaiselle neuroverkolle, mikä taasen nopeuttaa parhaimman mahdollisen mallin löytymistä sekä vähentää tarvittavaa laskentatehoa. Muut kerrokset mallinnetaan sisääntulokerroksen jälkeen. Yksi haaste tulee olemaan valita sopiva piilokerrosten sekä niissä olevien neuronien määrä. Ulostulokerroksessa neuronien lukumäärä määrittyy ennustettavien parametrien määrän mukaan. Tässä tapauksessa ulostulokerroksessa on vain yksi neuroni, joka antaa veden pinnan korkeuden.

Sztobryn (Sztrobyn,2013) esittämän listauksen mukaisesti neuroverkon luominen voidaan jakaa neljään vaiheeseen. Näistä ensimmäinen vaihe käsittää saatavilla olevan aineiston käsittelyn. Tässä työssä tavoitteena on löytää kolmesta neljään tekijää, joiden avulla voidaan luoda sisääntulovektori, joka kuvaa parhaiten ennustettavaa ilmiötä eli tässä tapauksessa joen veden pinnan korkeuden muutoksia. Toinen suunnittelun vaihe keskittyy varsinaisen neuroverkon rakenteeseen: piilokerrosten lukumäärään ja neuroneiden määrään. Kolmannessa vaiheessa käsitellään piilokerrosten neuroneita, erityisesti funktioita, joilla sisääntulovektorista saadaan muutettua haluttu ulostulo. Viimeinen eli neljäs vaihe koostuu oppimisesta ja siihen vaikuttamisesta. Tässä tapauksessa halutun ulostulon ja saadun ulostulon vertailusta, sekä näiden välisen erotuksen pienentämisestä.

5.1 Aineisto

Jotta käytettävä aineisto olisi valmista käsiteltäväksi neuroverkossa, se on esikäsiteltävä. Se mitä esikäsittely sisältää riippuu käsiteltävästä aineistosta. Ensiksi on tunnistettava minkälaisia muuttujia aineisto sisältää (Välisuo, 2019). Mikäli muuttujat kuuluvat rajattuun joukkoon ja se voidaan ilmaista joko sanamuotoisena tai numeerisesti, kyseessä on diskreettimuuttuja. Toinen muuttuja tyyppi on jatkuva muuttuja. Ne ovat reaalilukuja, ja ne saavat arvonsa esimerkiksi mittauslaitteen tuottamana. Tämän työn tapauksessa aineiso voidaan esittää jatkuvana aikasarjana (*time series)*. Aikasarjoille tyypillisesti, veden korkeutta tilastoidaan mittaamalla kohteen pinnan korkeus tiettynä ajan hetkenä sekä tietyin aikavälein. Tutkittaessa joen pinnan muutoksia, voidaan olettaa, että ainakin joidenkin aineiston muuttujien välillä on lineaarinen yhteys.

5.1.1 Eri lähteistä oleva aineisto ja aineiston määrä

Mikäli työn aikana tulisi tarve käsitellä sellaisia aineistoja, jotka ovat eri lähteistä, tulee ne yhdistää ja muokata saman muotoisiksi. Muokkaaminen voidaan toteuttaa normalisoinnilla, jossa muuttujat skaalataan vaihteluvälille [0,1], tai standardoinnilla, jolloin keskihajonta on 1 ja keskiarvo 0. Tämän tutkimuksen tapauksessa vastaan voi tulla tilanne, jossa käytettävien aineistojen mittayksiköt eroavat toisistaan. Tyypillinen esimerkki olisi lämpötila: Celsius - Fahrenheit tai korkeus: senttimetri – tuuma.

Muita ongelmia eri lähteistä olevien aineistojen kohdalla on mittausten eri aikavälit sekä käytettävän laitteiston tarkkuus. Aikavälillä tässä tapauksessa olisi merkitystä vain mikäli aineiston A mittausväli olisi vuorokausi ja aineiston B vastaava väli poikkeaisi vuorokaudella tai useammalla aineiston A mittausvälistä. Mainittakoon kuitenkin, että mittausväli on yleisesti korkeintaan vuorokausi sekä mittausväli mainitaan lähes aina aineiston kohdalla, jolloin aineistojen vertailu mittausvälin suhteen on helppoa ja voidaan tehdä ennen kuin ainestoa otetaan käyttöön. Vaikeamman haasteen taasen muodostaa veden korkeuden mittaamiseen käytettävän laitteiston

tarkkuus. Eri laitteistoilla on oma tarkkuuteensa ja ympäristö asettaa myös omat rajoitteensa mittalaitteiden tarkkuudelle. Tässä työssä ei ole kuitenkaan anneta suurta paino arvoa mittalaitteiston tarkkuuden vaihteluille, sillä tarkoituksena ei ole luoda millien tarkkaa järjestelmää.

Aineistoa voi olla kerättynä liikaa suhteessa tutkittavaan ilmiöön (mukaillen Välisuo, 2019). Aineiston suuri määrä hidastaa analysointia ja voi tehdä siitä sekavaa. Mikäli aineiston vähentämiseen päädyttäisiin, tulisi se tehdä niin, että olennainen tieto säilyy. Vähentäminen voidaan tehdä valitsemalla tiettyjä ominaisuuksia datasta. Ominaisuudet, jotka karsitaan ovat sellaisia, joita toistetaan aineistossa tai ovat epärelevantteja analyysin kannalta. Ominaisuuden karsiminen ei kuitenkaan saa haitata aineiston ymmärrettävyyttä, eikä se saa haitata aineiston esittämistä. Ominaisuuksien karsimisen sijaan isolle aineistolle voidaan käyttää piirre-erotusta. Tässä menetelmässä luodaan kokonaan erillinen muuttujajoukko, johon on siirretty alkuperäisen joukon tärkeimmät ominaisuudet. Mikäli halutaan säilyttää aineiston muuttujat, mutta silti vähentää tutkittavan aineiston määrää, voidaan alkueräisestä aineistosta ottaa näytteitä eli osajoukkoja käsiteltäväksi.

Toisinaan aineisto sisältää virheellistä tai puutteellista tietoa. Voi myös olla että sama tietoa on syötettynä moneen kertaan. Näiden ongelmien poistamista kutsutaan siivouksesi (*data cleaning)*.

5.1.2 Muuttujat

Ennen kuin kerättyä dataa voidaan käyttää neuroverkon opettamiseen ja testaamiseen, on siihen tehtävä muutoksia. Tärkein muutos on erillisten datataulujen yhdistäminen yhdeksi tiedostoksi. Yhdistetyssä taulussa eli datakehyksessä tieto pitää järjestää siten että se on mielekästä käsitellä. Järjestämiseen käytettävä avainmuuttuja voi olla esimerkiksi päivämäärä, jolloin datakehyksen tiedot järjestetään tämän muuttujan perusteella nousevasti tai laskevasti. Sisään syötettävien muuttujien tulisi olla

oleellisia sekä tarpeellisia, jotta vältytään häiriön lisääntymiseltä ja vältytään turhan monimutkaisen mallin luomiselta. Mikäli jätetään oleellisia muuttujia pois vaarana on ettei luotu malli kykene kuvaamaan haluttua käytöstä tai on epätarkka eli tapahtuu alisovittamista. Toisaalta, jos valitaan liian monta muuttujaa voi tapahtua ylisovittamista. Tällöin malli epäonnistuisi yleistämään uuden datajoukon ja veden korkeuden ennustaminen ei onnistuisi. Vähäinen muuttujien määrä myös tekee suunniteltavasta verkosta yksinkertaisemman. (Hu ja muut, 2018)

Käsiteltävän aineiston muuttujat voidaan ryhmitellä niiden jatkuvuuden ja arvojoukon perusteella ryhmiin. Teoriassa diskreettejä muuttujia sekä rajallisia arvojoukkoja ei esiinny tässä työssä tutkittavassa ilmiössä, sillä sää ja siitä aiheutuvat ilmiöt voivat saada mitä tahansa arvoja. Kuitenkin esimerkiksi sade ja kosteus eivät saa negatiivia arvoja eli käsiteltävän aineiston muuttujilla olisi osittain rajattuja arvojoukkoja (arvojen alkaessa nollasta). Lisäksi muuttujien arvot ovat osittain järjestetty joukko.

Muuttujien suhdetta toisiinsa kutsutaan korrelaatioksi. Vähennettäessä aineiston muuttujien määrä, on hyvä varmistua siitä miten aineiston muuttuja vaikuttaa ulostuloon. Yksi käytetyimmistä korrelaatiomalleista, Pearsonin *r*, lähtee olettamuksesta että kahden muuttujan välillä on lineaarinen yhteys:

$$r=\frac{\sum(x_i-\bar{X})(yi-\bar{Y})}{\sqrt{\Sigma(x_i-\bar{X})^2(yi-\bar{Y})^2}} \quad (6)$$

,missä *x* ja *y* ovat vektoreita, joiden välistä korrelaatiota tutkitaan.
X ja *Y* ovat *x*:n ja *y*:n keskiarvot.

Mikäli *r* lähestyy arvoa -1, kyseisten funktioiden välillä on negatiivinen suhde. Käytännössä tämä havaitaan siten että toisen muuttujan kasvaessa toinen muuttuja pienenee. Jos *r* lähestyy arvoa 1, valitsee muuttujien välillä positiivinen korrelaatio. Tämä tarkoittaa että muuttujan kasvaessa, myös toisen muuttujan arvo lisääntyy. Korrelaation ollessa 0, muuttujien välillä ei ole suhdetta eli ne ovat toisistaan riippumattomat.

Tällöin muutos toisessa muuttujassa ei aiheuta muutosta toisessa muuttujassa. (Elmusrati, 2018)

Korrelaation tutkimisen lisäksi on olemassa muita tapoja arvioida muuttujien vaikutusta ulostuloon. Yksitällainen askelittainen regressio algoritmi (*stepwise regression algorithm*). Tässä algoritmissa seurataan ulostuloa lisäämällä ja poistamalla aineiston muuttujia, kunnes tietty lopetusehto täytetään. Algoritmin tarkoituksena on poistaa kaikki ne muuttujat, jotka eivät merkittävästi vaikuta vasteeseen. Askelittainen regressio helpottaa muuttujien valitsemista aineiston ollessa suuri. Haittapuolena on ettei kyseinen algoritmi välttämättä anna parasta mahdollista muuttujien joukkoa, jolloin muuttujien joukolla luotava neuroverkko taasen ei ole paras mahdollinen ennustamaan tutkittavaa ilmiötä. (Elmusrati, 2018)

Taulukon sarakkeiden tiedot ovat eri mittakaavassa, johtuen useasta lähteestä. Yksitapa puuttua tähän ongelmaan on standardointi. Standardoinnilla pyritään vähentämään tästä aiheutuvaa ongelmaa ja nopeuttamaan algoritmin oppimista. Standardoinnissa jokaisen sarakkeen yksittäisestä arvosta vähennetään sarakkeen keskiarvo, jonka jälkeen erotus jaetaan keskihajonnalla. Tällöin muuttujien arvot skaalautuvat 0 molemmille puolille siten että keskihajonta on 1 ja keskiarvo on 0.

$$Z_x = (X_i - X)/\sigma_x \tag{7}$$

,missä Z_x on muuttujan uusi arvo
X_i on yksittäinen arvo,
X sarakkeen keskiarvo,
ja σ_x on keskihajonta.

5.2 Neuroverkon suunnittelu

Luotavan neuroverkon tavoitteena on ennustaa veden pinnan korkeuden vaihteluita. Käytettävästä aineistosta valittujen muuttujien perusteella sisääntulossa on kolme neuronia; yksi jokaiselle muuttujalle.

Aktivointifunktion tulisi kuvastaa tutkittavaa ilmiötä. Toisin sanoen sen hypoteesi avaruuden tulisi sisältää sellaisia funktioita, jotka ovat hyödyllisiä ongelman ratkaisuksi. Tässä työssä käsiteltävä ongelma on riippuvainen tuntemattomasta määrästä muuttujia, jotka yhdessä tuottavat ulostulon. Varmasti ei voida sanoa mikä on muuttujien suhde toisiinsa, mutta koska kyseessä ei ole lajittelutehtävä, voidaan aktivointifunktiot sekä hypoteesiavaruuden funktiot, jotka tuottavat tulokseksi 0 tai 1 ohittaa.

Verkko voidaan rakentaa siten, että se suosii yhtä hypoteesi avaruuden ratkaisufunktiota toisen ratkaisun kustannuksella. Tällöin kahdesta tai useammasta ratkaisusta valitaan se, joka täyttää tietyn ennalta määritellyn kriteerin. Esimerkkinä tällaisesta kriteeristä Goodfellow (Goodfellow ja muut, 2016, s. 115–117) mainitsee verkon painojen pienentämisen (*weight decay).* Esimerkissä pyritään mahdollisimman pieneen summaan *J(w):*

$$J(w)=MSE + \lambda R(w) \quad (8)$$

,missä $J(w)$ on verkon painojen summa
MSE on virheen keskiarvo,
R on regulaattori,
w on verkon paino ja
λ tiettyjen ratkaisujen suosimisessa käytettävä apumuuttuja.

Arvo λ kontrolloi sitä kuinka paljon pieniä verkon arvoja suositaan. Kun $\lambda=0$, ei ratkaisuja suosita verkon painojen pienuuden mukaan. λ kasvaessa, ratkaisun painojen arvo pienenee. Summan *J(w)* minimoiminen käytännössä aiheuttaa sen että päädytään tulokseen, jossa verkon painot ovat niin pienet kuin mahdollista ja silti verkko kykenee yleistämään opetusjoukon. Painojen arvon pienentämisellä ja

kasvattamisella voidaan myös hallita verkon yli- ja alisovittamista. (Goodfellow ja muut, 2016, s. 115–117)

Edellä esitellyssä kaavassa oleva muuttuja *R*, on virhefunktiossa esiintyvä regulaattori (*regularizer*). Regulaattoreilla mallinnetaan funktioita, jotka aiheuttavat rangaistuksen. Hypoteesi avaruuden funktio, jota ei haluta käytettävän, kuvataan siis siten että sen R aiheuttaa summan *J(w)* arvon kasvun sellaiseksi, että se on vertailussa epäsuotuisan suuri. Tällöin tätä funktiota ei valita. (Goodfellow ja muut, 2016, s. 115–117).

5.3 Mallin opettaminen ja testaus

Mallin opettamista varten datakehys pitää voida jakaa testi- ja opetusjoukkoon. Opetusjoukko on noin 80 % koko datakehyksestä ja se jaetaan vielä kahteen osaan. Opetusjoukon jakamisen tarkoituksena on testata mallin toimivuutta lisäämällä ajoittain siitä dataa opetusjoukon mukaan. Testijoukko pidetään opetusjoukosta erillään, ja sitä käytetään vain valmiin mallin testaamiseen. Koska käsiteltävä tieto on lajiteltu päivämäärän eli ajan mukaan, kulloinkin käytettävää datajoukkoa ei sekoiteta. (Välisuo, 2019)

Verkko opetetaan datasta saatavilla syöte-vaste-pareilla. Nämä dataparit toimivat opetusesimerkkeinä. Opettamisen aikana seurataan syötteen antamaa tulosta sen hetkisellä virhefunktiolla ja sitä verrataan kyseisen syötteen vastepariin. Vertailulla yritetään vähentää virhettä muuttamalla virhefunktiota ja verkon parametrejä. Tavoitteena siis on saada kaksi verkossa esiintyvää virhettä pienennettyä. Nämä virheet ovat opettamisen aikana ilmenevä virhe sekä testauksen aikana ilmenevä virhe. Opettamisen aikana ilmenevä virhe kuvaa opetusjoukon käsittelyn aikana saatavaa erotusta syötteen ja vasteen välillä. Luonnollisesti tämän virheen halutaan olevan mahdollisimman alhainen, jotta verkko kykenee antamaan veden pinnan korkeuden mahdollisimman oikein. Toinen minimoitava virhe, testauksen aikana ilmenevä, kertoo kuinka hyvin verkko on oppinut yleistämään annetun ilmiön.

Alisovittaminen näkyy opetusjoukon aikaisen virheen suuruutena, kun taas ylisovittaminen ilmenee testauksen aikaisen ja opetuksen aikaisen virheiden erotuksen suuruutena. (Goodfellow ja muut, 2016, s.108–112)

Hyperparametreilla kyetään vaikuttamaan neuroverkon toimintaan. Näitä hyperparametreja ovat: aktivointifunktio, piilokerrosten määrä sekä opettamiseen käytettävien kierrosten (*epoch*) lukumäärä. Verkon hyperparametreja ja oppimisnopeutta voidaan säätää erillisellä vahvistusjoukolla. Vahvistusjoukko on erotettu alkuperäisestä datakehyksestä, eikä sitä ole välttämättä esitelty verkolle aiemmin. Vahvistusjoukkoa voidaan myös käyttää verkon toiminnan varmistamiseen. (Elmusrati, 2018)

5.4 Tulosten arviointi

Neuroverkon kykyä oppia voidaan mitata sen kyvyllä suoriutua annetusta tehtävästä. Käytettävät arviointi- ja mittaustavat riippuvat verkolle annetun tehtävän luonteesta. Useimmissa tehtävissä suorituskykyä voidaan arvioida tarkastelemalla verkon tarkkuutta. Tarkkuus on tässä tapauksessa määre, jolla voidaan ilmaista, kuinka usein verkko kykenee tuottamaan oikean ulostulon suhteessa sisääntuloihin. Vastaavaa arvoa kuvaa myös virheaste tai virhesuhde (*error rate).* Virhesuhde lajittelee tulokset kahteen luokkaan: 0-luokka kuvaa tilannetta, jossa verkko on kyennyt luokittelemaan sisääntulon oikein; 1-luokka taas kertoo että luokittelu on virheellinen. (Goodfellow ja muut, 2016, s. 108–111)

Tämän työn tapauksessa luodun verkon kykyä oppia, arvioidaan testauksessa käytettävän datan avulla. Koska testausvaiheessa käytettävä data on verkolle uutta, antaa se parhaimman kuvan luodun verkkomallin suoriutumisesta todellisessa tilanteessa. Tämän validoinnin tarkoituksena on selvittää, kuinka hyvin luotu malli toimii uuden aineiston kanssa. Validointi voidaan toteuttaa tutkittavasta aineistosta irrotetulla joukolla, joka esitetään mallille validointivaiheessa ensimmäisen kerran. Validointidata voidaan valita käsiteltävästä aineistosta jakamalla aineisto esimerkiksi kolmeen

osaan, jolloin yksi osa toimii validointiaineistona. Muita tapoja on esimerkiksi ristiinvalidointi k-kertaa. K-kertaisessa validoinnissa data jaetaan k määrään osia. Validointi suoritetaan k kertaa, siten että jokainen osa toimii vuorollaan validointiaineistona. Validointikerroista lasketaan keskiarvo, joka kuvaa mallin toimivuutta. (Nielsen, 2015; Välisuo, 2019)

Mallin kykyä ennustaa vaste voidaan arvioida myös *RMSE* (*root mean square error*) arvon avulla (kaava 9). Se kuvaa vasteen ennustamisen keskimääräistä hajontaa eli kuinka kaukana ennustetut arvot ovat todellisista arvoista.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (9)$$

, missä n on otoksen koko ja $(y_i - \hat{y}_i)^2$ on ennustettujen että todellisten arvojen erotus.

RMSE arvon ollessa suuri, kyseessä usein on, että tarkasteltava malli ei kykene yleistämään tutkittavaa aineistoa. Kyseisen arvon ollessa pieni, voidaan olettaa neuroverkon olevan kykenevä yleistämään tutkittava ilmiö hyvin. Koska *RMSE* kuvaa mallin virhettä, sen antama tulos on suhteellinen tutkittavaan ilmiöön. Tästä seuraa että *RMSE* antaa suhteellisen virheen. Esimerkkinä tästä voidaan ajatella kahta tilannetta, jossa *RMSE* voi antaa saman arvon: lumipeite kasvaa 10 millimetrillä tai veden pinta nousee 10 metriä. Molemmissa tilanteissa esiintyy sama luku 10, mutta tilanne ja erityisesti sen vakavuus on eri. Tästä johtuen, on oltava selvillä minkälaisista muuttujista ja kokoluokista on kyse analysoidessa sekä käyttäessä neuroverkkoa.

6 Toteutus ja tulokset

Neuroverkon tuottama ulostulo on riippuvainen sisään syötettävistä muuttujista. Tavoiteltavan mallin yksinkertaisuuden vuoksi valmiissa verkossa ei tule olemaan kuin korkeintaan muutama piilokerros ja vain joitain sisääntuloja. Tutkittavaksi aineistoksi valikoitui Suurten Järvien alue Yhdysvalloissa lähellä Kanadan rajaa. Alue on mielenkiintoinen siksi, että siitä löytyy runsaasti dataa aina 1900-luvun alusta lähtien. Toinen seikka on että alueella on myös jokia, jotka ovat yhteydessä toisiinsa ja järvialtaisiin. Alueen jokisysteemeistä valikoitui tarkasteltavaksi st. Clair, joka yhdistää Huronjärven st. Clair -järveen. Lisäksi mukaan otettiin Detroit -joen virtaama, sillä se vaikuttaa st. Clair -järveen. Neuroverkon toteutus on tehty Python -ohjelmointikielellä. Käytetty koodi löytyy kommentoituna kokonaisuudessaan liitteenä (Liite 1).

6.1 Aineisto

Käsiteltävä aineisto on koottu GLERL (*Great Lakes Environmental Research Laboratory*) sivuilta. Aineisto on julkisesti saatavilla. Aineistosta valikoitui 13 muuttujaa lähempään tarkasteluun, sen perusteella kuinka ne kuvasivat tutkittavaa aluetta ja st. Clair -jokea. Yksittäiset muuttujat kerättiin yhteen ja samaan data taulukkoon, jolloin niiden tarkastelu sekä käsittely helpottuu. Jotta valittua dataa voidaan käsitellä neuroverkossa, sen tulisi sisältää numeerista tietoa, sekä kaikkien muuttujien sisältää yhtä monta arvoa. Datataulukkoa voidaan tarkastella **info()** -komennolla, jolloin saadaan alla esitetty kuva (kuva 2) näkyviin. Kuvasta voidaan nähdä datataulukossa esiintyvät piirteet eli muuttujat, muuttujan arvojen määrän sekä minkälaisia arvoja ne sisältävät. Tässä tapauksessa kaikki muuttujat ovat yhtäsuuria (61) ja sisältävät numeeria arvoja (Dtype : float64). Mikäli data sisältäisi puuttuvia arvoja, ne voitaisiin korvata esimerkiksi rivin sarakkeen keskiarvolla tai niitä sisältävät rivit voitaisiin poistaa. Tämän aineiston kohdalla, sellaiseen ei ole tarvetta.

```
 5  #   Column                 Non-Null Count  Dtype
 6 ---  ------                 --------------  -----
 7  0   river_flow             61 non-null     float64
 8  1   clouds                 61 non-null     float64
 9  2   evaporation            61 non-null     float64
10  3   lake_airTemp           61 non-null     float64
11  4   lake_landTemp          61 non-null     float64
12  5   rain_lake              61 non-null     float64
13  6   rain_land              61 non-null     float64
14  7   runoff_toLake_fromLand 61 non-null     float64
15  8   water_temp             61 non-null     float64
16  9   Wind                   61 non-null     float64
17  10  lake_waterlevel        61 non-null     float64
18  11  overbasin_airTemp      61 non-null     float64
19  12  Detroit_river          61 non-null     float64
20  13  Clair_basinSupply      61 non-null     float64
21 dtypes: float64(14)
22 memory usage: 6.8 KB
```

Kuva 2. Käsiteltävän aineiston muuttujat.

Seuraavaksi voidaan tarkastella itse muuttujia lähemmin. **Data.head()** -komennolla saadaan tulostettua datataulukon alkuosa tarkastelua varten (kuva 3). Tässä tarkastelussa huomataan, että arvoja täytyy, jossain vaiheessa yhtenäistää desimaalien suhteen. Datan suurusluokka on jo aineistoa kerätessä huomioitu ja esimerkiksi sateeseen liittyvillä muuttujilla on kaikilla sama kokoluokka millimetri.

```
    river_flow  clouds  ...  Detroit_river  Clair_basinSupply
0  4815.833333    65.5  ...    5043.333333            6148.34
1  5532.500000    64.1  ...    5735.833333            4980.88
2  6124.166667    59.5  ...    6325.833333            4200.94
3  5987.500000    59.8  ...    6091.666667            2330.60
4  5680.833333    63.8  ...    5832.500000            4622.71
```

Kuva 3. Käsiteltävän datataulukon muuttujien 4 ensimmäistä arvoa.

Käytettävä data tarkistetaan seuraavaksi poikkeavien havaintojen varalta. Mikäli datassa on paljon muusta joukosta poikkeavia arvoja, ne tulisi kyetä käsittelemään. Mahdollisia käsittelytapoja on niiden poistaminen tai muokkaaminen lähemmäksi muita arvoja. Poikkeava havainto voi johtua virheestä mittauksesta tai poikkeavasta ilmiöstä, joka vaatii lähempää tarkastelua. Oli poikkeavuuden syy mikä tahansa, se vaikeuttaa verkon kykyä yleistää tutkittava ilmiö, varsinkin datan määrän ollessa pieni. **Describe ()** -metodi luo taulukon (kuva 4), joka sisältää tietoa, kuinka muuttujien (sarakkeet) saamat arvot ovat jakaantuneet. Count ilmaisee muuttujan arvojen lukumäärän jättäen tyhjät (NaN) arvot pois laskuista. Tästä huomataan että muuttujissa ei ole tyhjiä arvoja tai niitä on saman verran. Edellä kerrotusta kuitenkin tiedämme jo että kaikki arvot ovat tallella. Mean kuvaa keskiarvoa ja std sarakekohtaista keskihajontaa. Min ja max kertovat sarakkeen pienimmän ja suurimman arvon. Tästä on hyötyä silloin kun halutaan saada selville se, että sisältääkö muuttuja sellaisia arvoja, jotka poikkeavat merkittävästi sen ilmiön tyypillisistä arvoista, mitä muuttujalla yritetään kuvastaa. Esimerkkinä voidaan mainita tuulen nopeus. Suuri tuulen nopeus kertoo mahdollisesta myrskystä tai virheestä joko mitattaessa tai datan käsittelyvaiheessa. Mikäli kyseessä olisi kyseinen luonnonilmiö, olisi syytä selvittää että onko kyseessä vain yksittäinen poikkeava havainto ja kuinka se vaikuttaa aineiston käsittelyyn.

2	river_flow	clouds	...	Detroit_river	Clair_basinSupply
3 count	61.000000	61.000000	...	61.000000	61.000000
4 mean	5353.702186	65.224590	...	5520.546448	4509.548852
5 std	480.238958	2.859642	...	489.222926	1430.233285
6 min	4290.000000	58.400000	...	4382.500000	1458.510000
7 25%	4970.000000	63.700000	...	5123.333333	3565.110000
8 50%	5389.166667	65.300000	...	5558.333333	4536.600000
9 75%	5680.833333	67.300000	...	5847.500000	5348.420000
0 max	6281.666667	71.500000	...	6489.166667	7843.090000
1					

Kuva 4. Describe () -metodin luoma taulukko.

Poikkeavat havainnot voidaan määritellä kvartiilien eli vaihteluvälien avulla, jotka näkyvät kuvassa 4 prosenttiarvoina riveillä 8,9 ja 10. Kvartiiliväli on yläkvarttiilin (75 %) ja alakvartiilin (25 %) välinen erotus. Poikkeava havainto (tässä työssä) on

tämän kvartiiliväli kerrottuna kolmella verran, joka yläkvartiilin yläpuolella tai alakvartaalin alapuolella. Kuva 5 esittää poikkeavien havaintojen määrän aineistossa. Kuvasta nähdään etteivät muuttujat sisällä havaintoja, jotka poikkeavat merkittävästi.

```
 2 outliers: river_flow               False
 3 clouds                    False
 4 evaporation               False
 5 lake_airTemp              False
 6 lake_landTemp             False
 7 rain_lake                 False
 8 rain_land                 False
 9 runoff_toLake_fromLand    False
10 water_temp                False
11 Wind                      False
12 lake_waterlevel           False
13 overbasin_airTemp         False
14 Detroit_river             False
15 Clair_basinSupply         False
16 Name: outliers, dtype: bool
17
```

Kuva 5. Poikkeavien havaintojen määrä.

6.1.1 Muuttujat

Tavoiteltava neuroverkko on yksinkertainen, vain muutaman muuttujan sisältämä malli. Tämä tarkoittaa, että käsiteltävässä datataulukossa on muuttujia, joita ei välttämättä tarvita ennustamisessa. Koska ennustettavana on joen virtaama (virtaaman kasvu ennakoi joen tulvimista), luonnollista on valita ne muuttujat, jotka myös vaikuttavat siihen eniten. Tarkastelemalla muuttujien suhteita korrelaation avulla, saadaan käsitys kuinka yhden muuttujan kasvu vaikuttaa toiseen. Apuna tässä tarkastelussa käytettiin Pearsonin korrelaatiota. Alla olevassa taulukossa on esitetty korrelaatio st. Clair -joen virtaaman suhteessa.

```
riverflow_pearsons:
Wind                      -0.345431
evaporation               -0.280567
rain_lake                 -0.069466
rain_land                  0.028373
Basin_rain                 0.074390
lake_landTemp              0.091252
overbasin_airTemp          0.096252
lake_airTemp               0.161966
clouds                     0.199891
Clair_basinSupply          0.214229
runoff_toLake_fromLand     0.223895
water_temp                 0.257889
lake_waterlevel            0.850077
Detroit_river              0.993921
river_flow                 1.000000
```

Kuva 6. Muuttujien välinen korrelaatio.

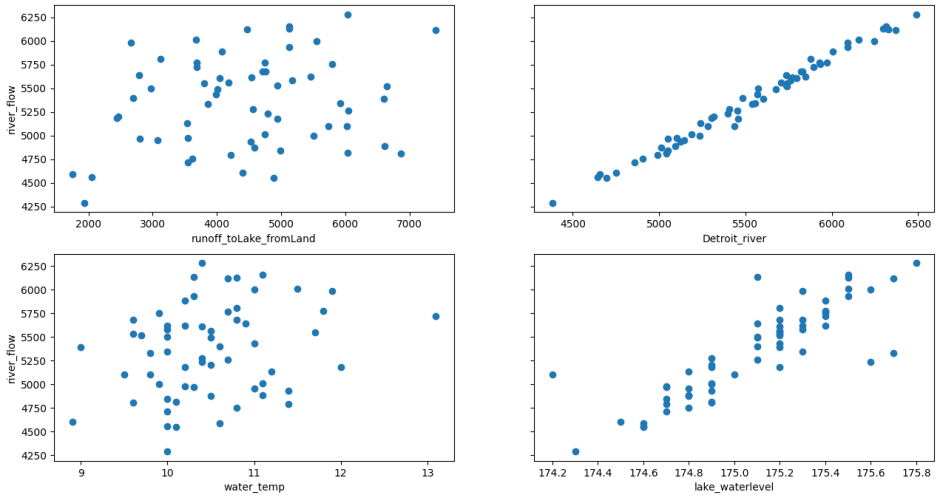
Joen virtaamaan näyttäisi taulukon perusteella vaikuttavan voimakkaasti kaksi muuttujaa: st. Clair -järvi sekä Detroit -joki. Kyseiset muuttujat ovat alttiita ympäristössä tapahtuviin muutoksiin, kuten sateen määrään sekä valumaan, jotka myös on kuvattu datataulukossa. Voidaankin ajatella, että muutos ympäristössä johtaa lopulta ketjureaktion lailla muutokseen st. Clair -joen virtaamassa. Seuraavaksi tarkastellaan st Clair -järveä että Detroit -jokea korrelaation avulla:

```
lake_waterlevel:
Wind                      -0.271856
evaporation               -0.239333
lake_landTemp              0.007695
overbasin_airTemp          0.011982
rain_lake                  0.032956
lake_airTemp               0.072846
rain_land                  0.118091
clouds                     0.149442
Basin_rain                 0.159872
water_temp                 0.189436
Clair_basinSupply          0.299854
runoff_toLake_fromLand     0.305790
river_flow                 0.850077
Detroit_river              0.866344
lake_waterlevel            1.000000
```

```
Detroit_river:
Wind                      -0.318643
evaporation               -0.310047
rain_lake                 -0.007991
lake_landTemp              0.085416
overbasin_airTemp          0.090190
rain_land                  0.098550
Basin_rain                 0.143152
lake_airTemp               0.152988
clouds                     0.233179
water_temp                 0.233467
Clair_basinSupply          0.295529
runoff_toLake_fromLand     0.306304
lake_waterlevel            0.866344
river_flow                 0.993921
Detroit_river              1.000000
```

Kuva 7. st Clair -järven ja Detroit -joen korrelaatio muuttujiin.

Tuloksista havaitaan että st. Clair -joki, Detroit -joki sekä st. Clair -järvi ovat vahvassa yhteydessä toisiinsa. Mikäli Suurten Järvien aluetta tarkasteltaisiin maantieteellisesti, huomattaisiin havainnon pitävän hyvin paikkaansa. Tarkastellaan vielä neljää merkittävintä muuttujaa visuaalisesti:



Kuva 8. Muuttujien välinen suhde esitettynä visuaalisesti.

Kuvaajan sekä taulukoiden pohjalta datataulukosta voidaan tiputtaa tarpeettomat muuttujat pois, ja valitaan Detroit_river, lake_waterlevel, sillä niiden muutos indikoi parhaiten st. Clair -joen virtaaman muutosta. Kuvasta 8 voidaan päätellä muuttujien ja joen virtaaman välisestä suhteesta, että tutkittava ilmiö on lineaarinen. Valitut muuttujat voivat periaatteessa saada mitä tahansa arvoja nollan ja äärettömän välillä eli muuttujien arvojoukko on osittain rajattu. Käytännössä arvot asettuvat jollekin mitattavalle välille ja muodostavat osittain järjestetyn joukon.

Lisäksi voitaisiin valita muuttuja runoff_toLake_fromLand, koska muutos valumassa aiheuttaa muutoksen (joskin ei suurta) Detroit-joen virtaamassa ja St. Clair -järven pinnan korkeudessa. Kuvaajassa näkyy myös veden lämpötila, jota ei huomioida myöskään tavoiteltavan mallin pienuuden vuoksi, sekä siksi että sen vaikutus on verrattain vähäinen tarkasteltaessa koko systeemiä (st. Clair -joki ja -järvi sekä Detroit -joki). Vaikka tarkasteltavan ilmiön muuttujilla ei välttämättä ole luonnossa lineaarista

yhteyttä toisiinsa, toimii korrelaatiokertoimen laskeminen apuna siinä mitä muuttujia tulisi tarkastella lähemmin.

Yleisesti voidaan sanoa että mikäli aineistossa on suurta vaihtelua muuttujien mittakaavassa, seurauksena verkon painojen kasvaminen tai muuttuminen rajusti opettamisen aikana Tästä taasen seuraa ettei neuroverkko kykene oppimaan tutkittavaa ilmiötä kyllin hyvin yleistääkseen sen. Usein myös käy niin että ne muuttujat, jotka saavat suuria arvoa, vaikuttavat enemmän verkon suoriutumiseen kuin muuttujat, joiden arvot ovat pieniä. Edellä kuvatussa on se ongelma, että suuri arvoiset muuttujat eivät aina ole merkitsevässä asemassa ennustamisessa eli toisin sanoen pieni arvoinen muuttuja voi kuvata paremmin ennustettavaa ilmiötä. Ongelmien välttämiseksi neuroverkolle syötettävä data tulisi skaalata samalle välille. Tässä tutkimuksessa käytettävän aineiston tapauksessa käytetään normalisoinniksi kutsuttua menetelmää sopivan skaalauksen saavuttamiseksi. Normalisointi väli on [0,1] valitun aktivointifunktion (ReLu) vuoksi. Normalisointia varten tarvitaan tieto suurimmasta ja pienimmästä arvosta. Normalisointi toteutettiin sklearn -kirjaston MinMaxScaler() avulla. Normalisointi vastaa lauseketta:

$$normalisoitu\_arvo = (normalisoitava\_arvo - min)/(max - min) \quad (10)$$

6.2 Neuroverkko

Verkkoa luodessa on huomioitava, että jokaisella neuronilla on oma painonsa. Lisäksi jokaiseen neuroniin on tarve tallentaa tietoa opettamisen aikana. Nämä tiedot ja paino voidaan tallentaa neuronikohtaiseen tauluun. Nämä taulut järjestetään, siten että muodostuu taulukko, joka kuvaa jokaista kerrosta. Näistä taulukoiduista kerroksista muodostuu neuroverkko. Taulukossa 1 on esitetty luodun verkon rakenne. Painojen alkuarvoksi asetetaan pieni satunnaisluku.

Taulukko 1. Verkon rakenne.

Topologia: MLP ,täysin yhdistetty Kerrokset: sisääntulokerros, piilokerros, ulostulokerros Iteraatioiden määrä: 10000 Virhefunktio: MSE				
	Neuronien lkm	aktivointifunktio	Muuttujat	
Sisääntulokerros	2		Detroit_river, lake_waterlevel	
Piilokerros	10	tanh		
Ulostulokerros	1	relu	st. Clair -joen pinnan korkeus	
Opetus: (koko datajoukon koko: 343)	Opetus-joukon koko: 282	Riippumattoman testijoukon koko: 61	Oppimisnopeus:0,5	Oppimistapa: (error)backpropagation

Alla on kuvattu Python-kielinen toteutus käytetystä koodista, jolla luodaan verkko:

```
def verkko(X, Y,piilokerros, oppimisnopeus=0.05, iteraa-
    tiot=10000):


    np.random.seed(3)
    sisaantulokerros = kerros_dims(X, Y)[0]
    ulostulokerros = kerros_dims(X, Y)[2]

    parametrit = alusta_parametrit(sisaantulokerros, piilokerros,
    ulostulokerros)
    Sisaantulo-piilokerros = parametrit['Sisaantulo-piilokerros']
    bias1 = parametrit['bias1']
    Piilokerros-ulostulo = parametrit['piilokerros-ulostulo']
    bias2 = parametrit['bias2']
```

```
    return parametrit
```
(mukaillen Ng, A. Neural Networks and Deep Learning)

Verkon luonnin jälkeen lähdetään tarkastelemaan, kuinka syöte liikkuu eri kerrosten välillä. Käytännössä tarvitsee tietää riittääkö syöte aiheuttamaan vasteen eli aktivoituuko neuroni. Syöte voi olla opetusjoukon datapiste tai edeltävän kerroksen neuronin vaste. Riippumatta syötteen laadusta, neuronin aktivointilauseke on muotoa:

*aktivointi= sum(paino_i * syöte_i)* (11)

missä *paino_i* on kyseisen neuronin paino, *syöte_i* on neuroniin sisään tuleva arvo ja bias on vakio. Mikäli syöte on riittävä aktivoimaan neuronin lasketaan vaste eli ulostulo. Ulostulo saadaan lisäämällä aktivointilauseke käytettävään aktivointifunktioon. Alla esiteltynä ReLu -aktivointifunktion toteutus:

```
def relu(syote):

    aktivointi = np.maximum(0, syote)
    assert (aktivointi.shape == syote.shape)

return aktivointi
```

Verkko käydään läpi kerroksittain ja jokaiselle neuronille lasketaan vaste. Jokainen vaste kerätään neuronin tauluun. Tauluista voidaan kerätä kyseisen kerroksen vasteet uudeksi taulukoksi, jota käytetään seuraavan kerroksen syötteinä. Esimerkiksi jos sisääntulokerroksessa on kolme neuronia ja ne tuottavat kukin vasteen, niin sen seurauksena seuraavaan kerrokseen on kolme syötettä. Alla on esitetty koodi, jolla tieto saadaan liikkumaan verkossa eteenpäin:

```
def aktivointi_eteenpain(X, parametrit)

    syote1 = np.dot(sisaantulo-piilokerros, X) + bias1
    vaste1 = np.tanh(syote1)
    syote2 = np.dot(piilokerros-ulostulo, vaste1) + bias2
    vaste2 = relu(syote2)

  muisti = {      "syote1": syote1,
                   "vaste1": vaste1,
```

```
                    "syote2": syote2,
                    "vaste2": vaste2}

    return syote2, muisti
```

(mukaillen Ng, A. Neural Networks and Deep Learning)

Kun kaikki neuronit ja kerrokset on käsitelty edellä kuvatulla tavalla, saadaan viimeisestä kerroksesta, ulostulokerroksesta, opetuspistettä vastaava vaste. Tämän vasteen ja opetuspisteen todellisen vastaavan (oletetun) arvon erotuksesta saadaan verkon virhe:

```
def laske_virhe(ennustetut, todelliset):
        m = ennustetut.shape[1]
  virhe = np.square(np.subtract(todelliset, ennustetut)).mean()
return virhe
```

Näin saatu virhe välitetään takaisin verkolle takaperin eli ensimmäisenä virhe välitetään ulostulokerrokselle ja sieltä verkon viimeiselle piilokerrokselle ja niin edelleen (niin sanotun backpropagation algoritmin tavoin). Lopulta virhe palaa sisääntulokerrokseen. Virheen siirtyessä takaisin päin verkon painoja päivitetään. Neuronin ulostulosta lasketaan sen jyrkkyys (gradientti), joka lisätään virhelausekkeeseen, jolloin se saa muodon:

virhe= (oletettu arvo – saatu vaste) aktivointifunktion derivaatta (saatu vaste)* (12)

Edellä kuvattua lauseketta käytetään kuitenkin vain ulostulokerroksessa. Piilokerrosten virhe lasketaan jokaisen ulostulokerroksen neuronin virheenä. Virhe siis siirretään verkossa takaisin päin verkkojen painoja avulla. Piilokerrosten virhelauseke on muotoa:

*virhe=(paino_k *virhe_j) * aktivointifunktion derivaatta(saatu vaste),* (13)

missä *paino_k* on se paino, joka on *k*:den neuronin ja käsiteltävän neuronin välillä, *virhe_j* on *j*:en ulostulokerroksen neuronin tuottama virhe ja saatu vaste on käsiteltävän neuronin tuottama vaste. Jokainen laskettu virhe tallennetaan neuroniin tunnuksella 'virhe'.

Kun virheet on laskettu, niitä voidaan käyttää apuna verkon painojen päivittämisessä:

*paino= paino+ oppimiskerroin * virhe * syöte* (14)

missä *paino* on annettu paino, oppimiskerroin on ennalta määritelty parametri ja virhe on se laskettu virhe, joka on saatu backpropagation -algoritmilla kyseiselle neuronille ja syöte on se sisääntulo, joka aiheutti virheen. Tässä työssä virhefunktiona käytettiin niin kutsutta MSE (*mean square error)* funktiota, joka laskee kahden pisteen välisen erotuksen keskiarvon. Tämän funktion käyttöä virhefunktiona puoltaa se että MSE aiheuttaa sen että verkkoa "rankaistaan" enemmän mikäli todellisten ja laskettujen vasteiden erotus on suuri eli painoka muutetaan enemmän kuin tilanteessa, jossa erotus olisi pieni. Funktio on myös yksinkertainen käyttää ja sen toiminta on helposti hahmotettavissa.

Verkon opettaminen tapahtuu (error) backpropagation ja gradientti menetelmällä. Menetelmässä verkkoa käydään toistuvasti läpi tietyn kierrosmäärän ajan. Jokaisella kierroksella verkkoon syötetään opetusjoukosta yksi datapiste (aineistossa kahden muuttujan muodostama rivi). Jokainen datapiste aiheuttaa verkon painojen päivittämisen, mikäli datapisteellä saatu vaste ja haluttu vaste poikkeavat toisistaan. Vasteiden välinen virhe siis siirtyy takaisin päin verkossa ja muuttaa painoja. Tätä jatketaan kunnes lopetusehto, tässä tapauksessa kierrosmäärä, saavutetaan.

6.3 Tulokset

Rakennettua neuroverkkoa testattiin aineistolla, joka koostui 343 kappaleesta syöte-vaste pareja. Kyseiset syöte-vaste parit on kerätty Suurten Järvien alueelta. Varhaisin

datapiste on vuodelta 1950 ja uusin vuoden 2010 joulukuulta. Syöte-vaste pareista erotettiin 61 kappaleen sarja verkon testausjoukoksi. Tätä osaa ei esitelty verkolle opettamisvaiheessa.

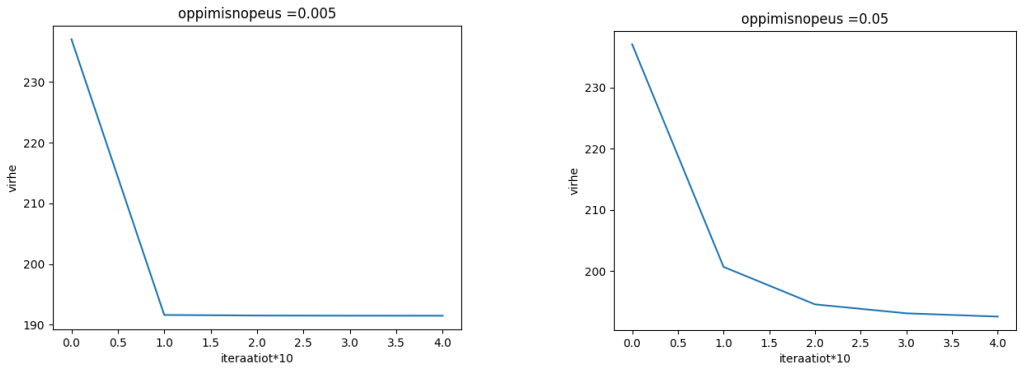
6.3.1 Opettaminen

Käytettäessä neuroverkkoa piilokerroksen kooksi valittiin 10 neuronia ja iteraatioiden määrä 10000 kertaan (Taulukko 2. s. 60). Hyperparametreistä oppimiskerroin eli oppimisnopeus kuvaa kuinka paljon painoa muutetaan virheen sattuessa. Pieniä oppimisnopeuksia suositaan yleensä kun opetus sisältää suuren määrän dataa sekä iterointeja. Näin lisätään todennäköisyyttä sille että verkko löytää sopivat painot koko verkkoon, eikä vain joihinkin neuroneihin. Tämä työn tapauksessa voidaan olettaa, että verkko toimii myös suurilla oppimisnopeuksia, sillä käsiteltävää aineistoa on vähän, neuroneita on verrattain vähän ja iterointien määrä on myös kohtuullinen. Iterointien määrän vähyydellä tavoitellaan niin sanottua aikaista pysähtymistä, joka estäisi verkon ylioppimista. Myös verkon pieni koko (yksi piilokerros, neuronien lukumäärä per kerros) vähentää riskiä ylioppimiseen. Toisaalta vaarana on ettei verkko kykene yleistämään annettua ilmiötä eli se alioppii. Taulukossa 2 on esitetty joitain tuloksia eri parametreilla:

Taulukko 2. Saatuja tuloksia eri parametreillä.

Saavutettu tarkkuus	oppimisnopeus	iteraatioiden määrä	piilokerroksen neuronien lkm
0 %	0.005	100	2
0 %	0.05	10000	2
20 %	0.05	100	5
32 %	0.5	100	5
59 %	0.5	100	10
81 %	0.5	10000	10
11 %	5	100	10
11 %	5	10000	10

Opettamisvaiheen aikana kokeiltiin kasvattaa iterointien määrää, mutta sen ei huomattu vaikuttavan oppimisnopeuden muutoksen tavoin. Pienillä oppimisnopeuksilla verkon neuronien määrän vaikuttaa tarkkuuteen: tarkkuus paranee neuronien määrän kasvaessa. Mikäli oppimisnopeus on verkon sekä aineiston kokoon nähden sopiva (tässä tapauksessa oppimisnopeus 0.5) iterointien määrä alkaa vaikuttamaan tarkkuuteen varsin nopeasti. Kuvassa 9 on esitetty kahden oppimisnopeuden vaikutus verkon virheeseen. Pienellä oppimisnopeudella verkon virhe pienenee nopeasti, mutta jää sen jälkeen jumiin paikalliseen arvoon. Seurauksena on että verkko niin sanotusti jumiutuu eli ei enää kykene oppimaan sille esitellystä datasta. Oppimisnopeuden ollessa suuri, virhe pienenee hitaampaa, mutta jumiutuminen epäsopivaan arvoon saatetaan välttää.



Kuva 9. Oppimisnopeuden muutoksen havainnollistaminen.

Neuroverkon tarkkuutta seurattiin myös oppimisnopeuden vaihtelun yhteydessä. Tarkkuuden kaavana käytettiin:

summa (ennustetut vasteet == todelliset vasteet) / muuttujien määrä (15)

Taulukosta 2 valituilla parametreillä, verkon tarkkuus saavutti 81 % eli verkko kykenee kutakuinkin oppimaan sille annetusta datasta. Tuloksia voitaisiin parantaa lisäämällä opettamiseen käytettävän aineiston määrä ja kiinnittämällä huomiota sen laatuun. Aineistosta tulisi kyetä paremmin tunnistamaan tärkeimmät muuttujat tutkittavan ilmiön kannalta. Tässä tapauksessa mukaan voisi tulevaisuudessa ottaa myös sadetta ja lämpötilaa kuvaavia muuttujia. Ongelmana on varmasti myös että datataulukossa saattaa esiintyä samoja arvoja useaan kertaan, mutta niitä vastaava ulostulo saattaa vaihdella eli aineiston keräämiseen voisi kiinnittää enemmän huomiota.

Tarkkuuteen vaikuttaa varmasti myös, että verkko on pieni, suhteessa yleistettävän ilmiön monimutkaisuuteen: kasvattamalla verkon kokoa (piilokerrosten ja neuronien määrää) voidaan saada aikaan parempia tuloksia, mutta tällöin verkon rakentaminen monimutkaistuu -mikä osaltaan lisää epäonnistumisen mahdollisuutta. Tärkeässä osassa ovat myös aktivointifunktiot. On mahdollista, että verkon kyky yleistää paranisi vaihtamalla käytettyjä aktivointifunktioita. On myös huomattava, että MLP-

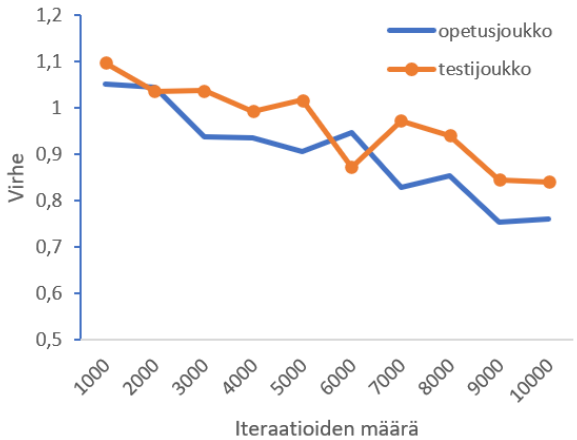
verkko ei ehkä ole paras mahdollinen käyttää tutkimuksen muuttujien kanssa. Tällöin harkittavaksi tulee malleja, jotka kykenevät käsittelemään muuttujien välisiä lineaarisia yhteyksiä paremmin.

6.3.2 Testaaminen

Testaaminen tapahtui verkolla, jonka oppimisnopeudeksi oli valittu 0.5, iteraatioiden määrä: 10000 ja neuroneita 10 kappaletta piilokerroksessa. Opettamisvaihe antoi syytä olettaa, että neuroverkko kykenisi ennustamaan testidatan vasteita joskin ei täydellisesti. Testaaminen suoritettiin aineistosta erotetulla 61 datapisteen joukolla. Vertailtaessa testijoukon muuttujien saamia arvoja opetusjoukon arvoihin havaittiin, että osa syöte-vaste pareista esiintyi molemmissa joukoissa. Tämä voi olla sattumaa ja johtua tutkittavan ilmiön syklisyydestä tai kyseessä voi olla myös että aineiston laatu on kärsinyt. Oli syy mikä tahansa, testijoukkoa ei lähdetty muuttamaan enää, sillä se olisi vaatinut myös opetusjoukon käsittelyä uudestaan. Tämä siksi, että kyseinen testijoukko on lohkaistu samasta aineistosta opetusjoukon kanssa, ja mikäli vain toista lähdettäisiin jälkikäteen muuttamaan vaarana olisi että toisesta joukosta häviää sellaista tietoa mikä toiseen jää. Tällä olisi myöhemmin vaikutusta neuroverkon kykyyn mallintaa tutkittavaa ilmiötä ja ennustaa sitä.

Taulukko 3. Testaamisen jälkeiset tulokset ja virheen suuruudessa tapahtuva muutos.

	Opetus-joukko	Testi-joukko
RMSE	0.88	0.98
Tarkkuus	81 %	75 %



Tuloksista nähdään tarkkuuden yltäneen 75 % eli verkko kykenee yleistämään tutkittavaa ilmiötä (taulukko 3). Tosin tuloksia tarkasteltaessa on huomioitava, että testauksessa käytettävä datassa esiintyy samoja datapisteitä kuin opettamisjoukossa. Taulukossa esiintyvä arvo RMSE (*root mean square error*) kertoo verkon tuottaman suhteellisen virheen datapisteille. Vasteen eli joen pinnan korkeus ilmoitetaan aineistossa millimetreinä, joten voidaan olettaa kyseisen RMSE-arvon kertovan että keskimääräisesti verkon antama vaste eroaa 0,88 millimetriä todellisesta vasteesta. Taulukosta nähdään myös että testijoukkoa käytettäessä keskimääräinen virhe kasvaa, mikä olikin oletettavissa. Taulukosta 3. havaitaan myös että iteraatioiden määrä voidaan jättää noin 9000, sillä tämän jälkeen iteraatioiden määrä ei vaikuta opetus- ja testijoukon virheeseen eli verkon oppiminen lähes lakkaa. Lisäksi voidaan sanoa, että verkolla on taipumusta ylioppimiseen sillä iteraatioiden 6000 ja 7000 välillä testijoukon virhe kasvaa huomattavasti.

Testaaminen ja opettamisvaihe toteutettiin uudestaan, mutta tulokset eivät poikkea mainittavasti edellä esitetystä. On mahdollista, että verkon tarkkuus paranisi käsiteltävien lukujen desimaaleja vähentämällä. Tällöin tarkkuus paranisi pyöristämällä aineiston luvut, jolloin väärin ennustetuksi ei luokiteltaisi arvoa, joka eroaa vain joitain millimetrin osia todellisesta arvosta. Liian suurta pyöristämistä tulisi kuitenkin välttää, sillä ilmiönä joen pinnan korkeus ja tulviminen ovat usein kiinni vain muutamista millimetreistä.

Neuroverkkoa olisi myös hyvä testata aineistolla, joka on kerätty toisesta samankaltaiselta alueelta. Tällaisia alueita on helppo löytää kartaston avulla, mutta haasteeksi nousee alueelta kerätty data tai lähinnä sen puuttuminen. Yleisesti voidaan sanoa, että teollisuusmaiden alueilta on tällaista tietoa hyvin saatavilla. Näillä alueilla on jo usein käytössä omat tulvavaroitusjärjestelmänsä, jotka kykenevät tässä työssä esitettyä verkkoa paljon tarkempaan ennustamiseen.

7 Päätelmät

Hydrologinen ennustaminen vaatii moninaisten muuttujien ja ilmiöiden huomioon ottamista. Pelkästään sade ja lämpötila eivät kuvaa koko ilmiötä. Mukaan on otettava koko tarkasteltavan alueen ekosysteemi, joko suoraa muuttujina tai välillisesti esimerkiksi alueen vesialtaita kuvaavien muuttujien kautta. Näin laajojen ilmiöiden mallintaminen on haastavaa, sillä ei ole vakiintunutta käytäntöä miten se olisi paras tehdä. Ajansaatossa on esitetty useita eri järjestelmiä ja malleja veden pinnan muutoksen ennustamisen tueksi. Toiset näistä ovat hyvinkin karkeita, kun taas toiset mallit kykenevät tarkkoihin ennustuksiin. Tässä työssä tutkittu neuroverkkoon pohjautuvalla mallilla voidaan teoreettisesti saavuttaa tarkkoja ennustuksia, vaikka mallinnettava ilmiö olisikin monimutkainen. Saavutetut tulokset tukivat osittain tätä väitettä.

Työn tulokset olivat odotusten suuntaisia. Tuloksista havaitaan että on mahdollista luoda pieni, yhden piilokerroksen ja vain muutaman sisään syötettävää muuttujaa tarvitseva neuroverkko. Opetusvaiheessa verkko osoitti kykyä yleistää ilmiö kohtalaisesti, ja kykeni ennustamaan testijoukkoa oikein. Neuroverkon luomisen kannalta voidaan siis ajatella, että onnistuttiin. Aineistoa käsiteltäessä havaittiin, että parhaiten tutkittavaa ilmiötä kuvaavat muuttujat olivat niitä, jotka kuvasivat tarkasteltavaan jokeen vaikuttavia muita vesisysteemejä. Vaikkakin aluksi olisikin mielekästä valita muuttujiksi sellaisia, jotka yleisesti mielletään vaikuttavan joen pinnan korkeuteen, kuten sade ja lämpötila, analysoimalla muuttujien lineaarista suhdetta kyetään valitsemaan aineistosta sopivimmat muuttujat kuvaamaan ilmiötä. Myös aineistossa olevien muuttujien tunteminen helpottaisi sopivimpien muuttujien valintaa. Nyt valituista kahdesta muuttujasta voidaan sanoa, että vesistöinä ne keräävät muiden ympäristössä esiintyvien muuttujien (sade, pilvisyys, tuuli ja niin edelleen) vaikutukset yhteen ja ilmentävät niitä, jolloin veden pinnan muutoksen ennustaminen helpottuu tarvittavien muuttujien määrän vähetessä. Ennustamiseen käytettävien muuttujien vähäisyys osittain myös mahdollistaa käytettävän neuroverkon yksinkertaisuuden. Jotta voitaisiin varmistua luodun neuroverkon todellisesta kyvystä oppia, tulisi neuroverkolle syöttää alkuperäisen aineiston lisäksi myös muilta alueilta kerättyä dataa.

Muilta alueilta kerätty data auttaisi kehittämään neuroverkkoa vielä eteenpäin. Universaalisti toimiva mallin voidaan ajatella olevan seuraava askel, kunhan vain ensin varmistutaan nykyisen mallin toimivuudesta.

Edellä esitetystä voidaan todetta, että MLP-neuroverkkoja voidaan käyttää luonnon ilmiöiden mallintamiseen sekä ennustamiseen. Ennustamisen kannalta on tärkeää valita aineisto ja muuttujat huolella sekä kiinnittää huomiota aktivointifunktioiden valintaan. Hyvin valittu ja laadukas aineisto edesauttaa neuroverkon oppimista ja näin osaltaan auttaa mallintamaan kuvattua ilmiötä paremmin. Tekniikan kehittyessä tulvien ennustaminenkin muuttuu. Tulevaisuuden tavoitteena voitaisiin pitää sellaisen järjestelmän luomista, joka on paitsi edullinen, luotettava myös yksinkertainen käyttää. Sen tulisi myös olla sellaisenaan sopiva vaihteleviin ympäristöihin. Haasteena on myös tulevaisuudessa löytää sellaista aineistoa, joka Tähän tulevaisuuden tavoitteeseen tässä työssä esitetyn neuroverkon kaltainen järjestelmä olisi sopiva.

Liite 1. Neuroverkon toteutus Python -ohjelmointikielellä

(Mukaillen Ng, A. Neural Networks and Deep Learning [kurssimateriaali]

```
def verkko(X, Y,piilokerros, oppimisnopeus=0.5, iteraatiot=10000):
#verkon rakenne ja toiminta. Luo tarvittavat kerrokset ja taulukot
joihin mm. painot tallennetaan

        np.random.seed(3)
        sisaantulokerros = kerros_dims(X, Y)[0]
        ulostulokerros = kerros_dims(X, Y)[2]

        parametrit = alusta_parametrit(sisaantulokerros, piilokerros,
        ulostulokerros)

        sisaantulo-piilokerros = parametrit['sisaantulo-piilokerros']
        bias1 = parametrit['bias1']
        piilokerros-ulostulo = parametrit['piilokerros-ulostulo']
        bias2 = parametrit['bias2']

       for i in range(0, iteraatiot):
        vaste2, muisti = aktivointi_eteenpain(X, parametrit)
       virhe = laske_virhe(vaste, Y)
       gradientit = backwardpropagation(parametrit, muisti, X, Y)
        parametrit = paivita_parametrit(parametrit, gradientit)

return parametrit


def relu(syote):
#käytettyn virhefunktion rakenne. Toinen virhefunktio tanh saadaan
     valmiina numpy -kirjastosta
         aktivointi = np.maximum(0, syote)
         assert (aktivointi.shape == syote.shape)

return aktivointi


def aktivointi_eteenpain(X, parametrit)
# datan liikkumien eteenpäin verkossa. Tarvitsee sisääntuloksi käsi-
        teltävän datapisteen sekä taulukon kerroksista ja painoista.
        Ulostulona on verkon ennustamat arvot, sekä kunkin kerroksen
        syötteet ja vasteet backpropagationia varten

         sisaantulo-piilokerros = parametrit['sisaantulo-piilokerros']
        bias1 = parametrit['bias1']
        piilokerros-ulostulo = parametrit['piilokerros-ulostulo']
        bias2 = parametrit['bias2']


         syote1 = np.dot(sisaantulo-piilokerros, X) + bias1
         vaste1 = np.tanh(syote1)
         syote2 = np.dot(piilokerros-ulostulo, vaste1) + bias2
         vaste2 = relu(syote2)
```

```
        muisti = {      "syote1": syote1,
                        "vaste1": vaste1,
                        "syote2": syote2,
                        "vaste2": vaste2}


    return syote2, muisti


def laske_virhe(vaste2, Y):
#verkon tuottaman virheen laskenta. Vaste2= verkon ennustamat vas-
        teet
            m = vaste2.shape[1]
            virhe = np.square(np.subtract(vaste2, Y)).mean()

return virhe


def ennusta(parametrit, X, y):
# Verkon tuottaman vasteen laskenta ja sen vertailu haluttuihin vas-
teisiin

    vaste2, muisti = aktivointi_eteenpain(X, parametrit)
    ennustetut = np.round(vaste2>0.05)
    y = np.round(y > 0.05)
    m = y.shape[0]

    print("ennustettu ulostulo ja todellinen ulostulo")
    print(ennustetut,y)

    print("tarkkuus: %s" % str(np.sum(ennustetut == y) / float(m)))

 return ennustetut


def kerros_dims(X, Y):
# neuroverkon kerrosten koon (neuronien määrän) määrittely
    sisaantulokerros = X.shape[0]
    piilokerros = 10
    ulostulokerros = Y.shape[0]

return (sisaantulokerros, piilokerros, ulostulokerros)


def alusta_parametrit (sisaantulokerros, piilokerros, ulostulokerros):
     # luo kerroskohtaiset painotaulukot ja asettaa niille alkuarvoiksi
     satunnaislukuja

         sisaantulo-piilokerros = np.random.randn(piilokerros, sisaan-
     tulokerros) * 0.01
         bias1 = np.zeros(shape=(piilokerros, 1))
         piilokerros-ulostulo = np.random.randn(ulostulokerros, piilo-
     kerros) * 0.01
         bias2 = np.zeros(shape=(ulostulokerros, 1))

         assert (sisaantulo-piilokerros.shape == (piilokerros, sisaan-
     tulokerros))
         assert (bias1.shape == (piilokerros, 1))
         assert (piilokerros-ulostulo.shape == (ulostulokerros,
```

```
    piilokerros))
        assert (bias2.shape == (ulostulokerros, 1))

        parametrit =
              {"sisaantulo-piilokerros": sisaantulo-piilokerros,
                     "bias1": bias1,
                     "piilokerros-ulostulo": piilokerros-ulostulo,
                     "bias2": bias2}

return parametrit


def paivita_parametrit(parametrit, gradientit, oppimisnopeus=0.5):
#Päivittää parametrit(painot) backpropagationista saatujen gradient-
tien ja oppimisnopeuden avulla

        sisaantulo-piilokerros = parametrit['sisaantulo-piilokerros']
        bias1 = parametrit['bias1']
        piilokerros-ulostulo = parametrit['piilokerros-ulostulo']
        bias2 = parametrit['bias2']

        G_sisaantulo-piilokerros = gradientit['sisaantulo-piilokerros']
        G_bias1 = gradientit['bias1']
        G_piilokerros-ulostulo = gradientit['piilokerros-ulostulo']
        G_bias2 = gradientit['bias2']


         sisaantulo-piilokerros = sisaantulo-piilokerros - oppimisno-
        peus * G_sisaantulo-piilokerros
         bias1 = bias1 - oppimisnopeus * G_bias1
          piilokerros-ulostulo = piilokerros-ulostulo - oppimisnopeus *
        G_piilokerros-ulostulo
          bias2 = bias2 - oppimisnopeus * G_bias2


            parametrit = {"sisaantulo-piilokerros ": sisaantulo-piilo-
        kerros,
                          "bias1": bias1,
                          "piilokerros-ulostulo": piilokerros-ulostulo,
                          "bias2": bias2}

return parametrit


def backwardpropagation(parametrit, muisti, X, Y):
#backpropagation. Verkko käydään takaperin läpi. Samalla lasketaan
kerroksittaiset gradientit, joita käytetään muokkaamaan/parantamaan
verkon painoja, ja sitä kautta vastetta. Gradientit lasketaan kerros-
kohtaisista vasteista

         m = X.shape[1]

        sisaantulo-piilokerros = parametrit['sisaantulo-piilokerros']
        piilokerros-ulostulo = parametrit['piilokerros-ulostulo']

                       vaste1 = muisti['vaste1']
                       vaste2 = muisti['vaste2']

                       G_syote2 = vaste2 - Y
```

```
                G_piilokerros-ulostulo = (1 / m) *
            np.dot(G_syote2, vaste1.T)
                        G_bias2 = (1 / m) * np.sum(G_syote2,
            axis=1, keepdims=True)

            G_syote1 = np.multiply(np.dot(piilokerros-ulostulo.T,
            G_syote2), 1 - np.power(vaste1, 2))

                   G_sisaantulo-piilokerros = (1 / m) *
            np.dot(G_syote1, X.T)
                G_bias1 = (1 / m) * np.sum(G_bias1, axis=1,
            keepdims=True)

            gradientit = {"G_sisaantulo-piilokerros ": G_sisaantulo-
            piilokerros,
                          "G_bias": G_bias1,
                          " G_piilokerros-ulostulo": G_piilokerros-
            ulostulo,
                          "G_bias2": G_bias2}


return gradientit
```

Lähteet

Biswas, R.K., Jayawardena, A.W, (2014) Water Level Prediction By Artificial Neural Network In A Flashy Transboundary River Of Bangladesh, *Global NEST Journal Vol 16, No 2, s. 432–444.*

Cluckie, I.D., Moghaddamnia, A., Han, D., (2008) *Practical Hydroinformatics: Using an Adaptive Neuro-Fuzzy Interference System in the Development of a Real-Time Expert System for Flood Forecasting.* Springer. https://doi.org/10.1007/978-3-540-79881-1

Devi, G. K., Ganasri, B. P., Dwarakish, G. S., A Review on Hydrological Models. *International Conference On Water Resources, Coastal And Ocean Engineering2015*. https:// dol.org/ 10.1016/j.aqpro.2015.02.126

Elmusrati, M. (2018). Vaasan Yliopisto : ICAT3120 Machine Learning [tulostettu kurssimateriaali].

GLERL (Great Lakes Environmental Research Laboratory) Great Lakes Dashboard https://www.glerl.noaa.gov/data/dashboard/data/

Goodfellow, I., Bengio, Y., & Courville, A. (2016) MIT Press. *Deep Learning*. https://www.deeplearningbook.org/

Graupe, D. (2013). *Principles of Artificial Neural Networks*. World Scientific Publishing Company. ISBN: 981-4522-73-2

Han, G., & Shi, Y. (2008). *Development of an Atlantic Canadian Coastal Water Level Neural Network Model*. https:// dol.org/10.1175/2008JTECHO569.1

Hua, T., Maoa, J., Panc, S., Daid, L., Zhanga, P., Xua, D. & Daia, H., (2018). Water level management of lakes connected to regulated rivers: An integrated modeling and analytical methodology https://doi.org/10.1016/j.jhydrol.2018.05.038

Kaloop, M., El-Diasty, M., & Hu, J. W., (2017). Real-time prediction of water level change using adaptive neuro-fuzzy inference system https://doi.org/10.1080/19475705.2017.1327464

Karimi, S., Kisi, O., Shiri. J., Makarynskyy, O., (2012). Neuro-fuzzy and neural network techniques for forecasting sea level in Darwin Harbor, Australia http://dx.doi.org/10.1016/j.cageo.2012.09.015

Kervinen, K. (2012, 23. Toukokuuta) Kevättulva pudottaa vesivoimalan tehoa. *YLE.* Noudettu 31.3.2021 osoitteesta https://yle.fi/uutiset/3-6107031

Kuusisalo, M. (2016, 2. Kesäkuuta). Euroopan tulvat eivät hellitä – Louvre sulkee ovensa. *YLE*. Noudettu 20.11.2010 osoitteesta https://yle.fi/uutiset/3-8927247

Ng, A. Neural Networks and Deep Learning [tulostettu kurssimateriaali] coursera [rajattu pääsy] Kurssin kotisivu: https://www.coursera.org/specializations/deep-learning

Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Noudettu 1.10.2020 osoitteesta http://neuralnetworksanddeeplearning.com/chap3.html

Pohjolan Voima (2021). Säätövoiman tarve kasvaa. Noudettu 1.6.2021 https://www.pohjolanvoima.fi/huomisen-energiaa/saatovoiman-tarve-kasvaa/

Singh, V.P. (2018). Hydrologic modeling: progress and future directions. *Geosci. Lett. 5, 15* https://doi.org/10.1186/s40562-018-0113-z

Sztobry, M. (2013). Application of Artificial Neural Network into the Water Level Modeling and Forecast. *Trans Nav.* https://doi.org/ 10.12716/1001.07.02.09

Välisuo, P. (2019). Vaasan Yliopisto : ICAT3190 Applied Machine Learning [kurssimateriaali]. Moodle [Rajattu pääsy]. Noudettu 1.2.2021 osoitteesta https://moodle.uwasa.fi/course/view.php?id=3648

Ympäristöhallinnon yhteinen verkkopalvelu: ympäristö.fi. (2013). Tulvasanasto. Noudettu 1.3.2021 osoitteesta https://www.ymparisto.fi/fi FI/Vesi/Tulviin_varautuminen/Tulvasanasto