SÄDEHOIDON ANNOSSUUNNITELMIEN POIKKEAVUUKSIEN HAVAITSEMINEN NEUROVERKOILLA

Joonas Hämäläinen

Pro gradu -tutkielma
Fysiikan laitos
Jyväskylän yliopisto
2.12.2013
Ohjaajat: Pertti Henttu
Jaana Kumpulainen

Tiivistelmä

Sädehoidossa potilaalle tehdään yksilöllinen annossuunnitelma, jonka mukaan hoito toteutetaan. Kaikilta annokseen vaikuttavilta tekijöiltä vaaditaan suurta tarkkuutta. Uusi lähestymistapa annossuunnitelmien laadunvarmistukseen on tiedonlouhintaan ja koneoppimiseen perustuvien menetelmien hyödyntäminen. Kyseisillä menetelmillä voidaan muodostaa hoidossa aiemmin toteutetuista annossuunnitelmista malli, jonka avulla voidaan havaita uusien annossuunnitelmien poikkeavuudet, ja näin lisätä sädehoidon turvallisuutta.

Tutkimuksen tavoitteena oli muodostaa SOM- ja PNN-neuroverkoilla malli, jolla voidaan havaita poikkeavuuksia annossuunnitelmista. Mallia varten haettiin rinnanpoiston jälkeisten kolmella kentällä toteutettujen sädehoitojen annossuunnitelmien parametreja Oulun yliopistollisen sairaalan sädehoidon yksikön tietokannasta. Malli muodostettiin erikseen fotoni- ja elektronikentille. Tutkimuksessa keskityttiin havaitsemaan kenttien monitoriyksiköiden ja elektronikenttien säteilyenergioiden poikkeavuuksia. Lisäksi työssä selvitettiin, kuinka Matlab-ohjelmistolla voidaan muodostaa tietokantayhteys OYS:n Aria-tietokantaan, mikä mahdollistaa poikkeavuuksien havaitsemismallin käyttöönoton annossuunnitelmien tarkastuksessa.

Tulosten perusteella tutkimuksessa muodostetulla mallilla onnistuttiin havaitsemaan keinotekoisesti tehtyjä virheitä hyvin. Mallilla todettiin olevan myös hyvä yleistyskyky. Lisäksi tietokantayhteys Matlab-ohjelmistolla onnistui, mikä mahdollistaa Matlab-pohjaisen annossuunnitelmien tarkastustyökalun kytkemisen annossuunnittelujärjestelmän rinnalle sädehoidon turvallisuutta lisäämään. Poikkeavuuksien havaitsemismallin soveltaminen muihin annossuunnitelmatyyppeihin ja -parametreihin vaatii vielä lisätutkimusta.

Sisällysluettelo

1 JOHDANTO

Sädehoidossa syöpäsolukkoon pyritään kohdistamaan riittävän suuri sädeannos sen tuhoamiseksi ja samalla minimoimaan tervekudosten saama sädeannos. Sädehoito on monimutkainen ja pitkä prosessi, jossa useat eri alojen ammattilaiset osallistuvat potilaan hoitoon. Säteilyn käyttö sädehoidossa vaatii erityistä huolellisuutta, koska käytettävät säteilyannokset ovat erittäin suuria. Virheellisesti annettu sädeannos voi heikentää syövästä parantumisen todennäköisyyttä tai lisätä hoidon sivuvaikutuksia. Pahimmillaan systemaattinen virhe voi johtaa useiden potilaiden menehtymisiin, mitä raportoidaan maailmassa tapahtuvan keskimäärin joka toinen vuosi [1].

Säteilyturvakeskuksen [2] mukaan sädehoidon laadunvarmistuksessa jokaisen potilaan yksilölliset annossuunnitelmat tulisi tarkastaa annossuunnittelujärjestelmästä mahdollisimman riippumattomalla menetelmällä. Uusi lähestymistapa annossuunnitelmien laadunvarmistuksessa poikkeavuuksien havaitsemiseen on tiedonlouhinnan ja koneoppimisen menetelmien hyödyntäminen. Sairaalaympäristössä tiedonlouhinnan ja koneoppimisen menetelmiä on sovellettu onnistuneesti muun muassa syövän diagnostiikkaan [3] ja sädehoidon annossuunnitteluun [4, 5].

Tutkimuksen lähtökohtana on Oulun yliopistollisen sairaalan sädehoidon yksikön tarve annossuunnitelmien tarkastustyökalulle, joka toimisi annossuunnitelmat tarkastavan fyysikon apuvälineenä. Tutkimuksen tavoitteena on muodostaa SOM- ja PNN-neuroverkoilla aiemmin toteutetuista annossuunnitelmista malli, jonka avulla havaitaan poikkeavuudet annossuunnitelmista. Mallia varten haetaan aiemmin hoidossa toteutettujen annossuunnitelmien parametreja OYS:n sädehoidon yksikön tietokannasta. Lisäksi työssä selvitetään, kuinka Matlab-ohjelmistolla saadaan muodostettua tietokantayhteys ja vastaavasti haettua annossuunnitelmien parametrit tietokannasta, mikä mahdollistaisi tarkastustyökalun kytkemisen annossuunnittelujärjestelmän rinnalle.

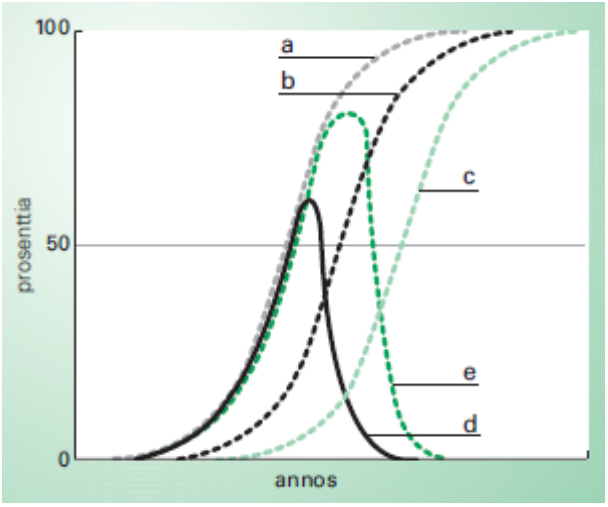
Tutkimus rajataan rinnanpoiston jälkeisiin kolmen kentän sädehoidon annossuunnitelmiin. Mallia sovelletaan kenttien monitoriyksiköiden ja elektronikentän säteilyenergioiden poikkeavuuksien havaitsemiseen. Mallin yleistyskyky määritetään ristiinvalidoinnilla, ja se testataan lopuksi keinotekoisesti tehdyillä virheillä.

Muodostettavan mallin tarkoituksena on havaita poikkeavuudet sädehoidon annos-suunnitelmista ennen sädehoidon toteutusta ja näin lisätä sädehoidon turvallisuutta entisestään.

2 SÄDEHOITO

2.1 Johdantoa sädehoitoon

Sädehoidon tavoitteena on kohdistaa syöpäkasvaimeen ja sen mahdollisiin leviämisalueisiin mahdollisimman suuri osa säteilyannoksesta. Lisäksi terveitä kudoksia ja elimiä pyritään suojaamaan säteilyannokselta mahdollisimman hyvin. Sädehoito on yksi tärkeimmistä syövän hoitomuodoista. Hoito yhdistetään usein muihin syövän hoitomuotoihin, ja noin puolet syöpäpotilaista saavat sairautensa aikana sädehoitoa. Sädehoitoa käytetään erityisesti silloin, kun syöpäsolukkoa ei pystytä poistamaan kokonaan kirurgisesti. [6, 1]



Kuva 2.1: Annosvaste. a) Paikallinen kasvaimen häviäminen, b) vakavat sivuvaikutukset normaaleilla kenttäjärjestelyillä, c) vakavat sivuvaikutukset optimoiduilla kenttäjärjestelyillä, d) paikallisen kasvaimen häviäminen ilman vakavia sivuvaikutuksia normaaleilla kenttäjärjestelyillä ja e) paikallisen kasvaimen häviäminen ilman vakavia sivuvaikutuksia optimoiduilla kenttäjärjestelyillä.

Sädehoidon annoksen tarkan määrittämisen tärkeyttä voidaan havainnollistaa annosvasteella. Kuvassa 2.1 [1] on tyypillinen sädehoidon annosvaste. Kasvattamalla sädeannos riittävän suureksi saadaan kaikki syöpäsolut kuolemaan (käyrä a), mutta samalla kasvaa myös vakavien sivuvaikutusten todennäköisyys. Näin ollen annoksella on olemassa optimiarvo. Käyrät d ja e kuvaavat todennäköisyyttä syöpäkasvaimen häviämiseen ilman vakavia sivuvaikutuksia. Muutamilla syöpätyypeillä käyrien a ja

b ero on niin pieni, että 3–4 %:n poikkeama oikeasta annoksesta pienentää paranemisen todennäköisyyttä yli 10 %. Kaikilta annokseen vaikuttavilta tekijöiltä vaaditaan suurta tarkkuutta, jotta suunnitellussa annoksessa olisi mahdollisimman vähän epävarmuutta. [1]

Säteilyn vaikutus kudokseen perustuu säteilykentän voimakkuuteen sekä säteilyn ja aineen välisiin vuorovaikutuksiin. Säteilyn energia siirtyy monivaiheisen prosessin jälkeen kudoksessa tapahtuviin atomi- ja molekyylitason muutoksiin, mistä edelleen saattaa seurata solutason muutoksia. [7] Absorboitunut annos on kudoksen massa-alkioon absorboitunut keskimääräinen energia massayksikköä kohti. Absorboituneen annoksen yksikkönä on 1 J/kg = 1 Gray = 1 Gy. [8]

Hoitoannos jaetaan usein pienempiin kerta-annoksiin eli fraktioihin. Fraktioinnissa määritetään kokonaisannos, kerta-annoksen suuruus, fraktioiden välinen aika ja kokonaishoitoaika. Tavanomaisessa fraktioinnissa kerta-annoksen suuruus on 2 Gy ja se annetaan viitenä päivänä viikossa haluttuun kokonaisannokseen asti. [8]

2.2 Säteily ja sen vuorovaikutukset aineen kanssa

Ionisoivan säteilyn ja aineen väliset fysikaaliset vuorovaikutukset muodostavat kudokseen kemiallisia muutoksia, jotka ilmenevät säteilyn biologisina haittavaikutuksina. Ionisoiva säteily voi olla sähkömagneettista säteilyä, jota on esimerkiksi röntgensäteily. Ionisoiva säteily voi olla myös hiukkassäteilyä, jota ovat esimerkiksi elektroni- ja neutronisäteily. [7]

2.2.1 Sähkömagneettinen säteily

Sähkömagneettinen säteily on aaltoliikettä, joka etenee valonnopeudella. Se muodostuu kvanteista tai fotoneista, jotka ovat energiaa sisältäviä paketteja. Kvantin energia voidaan laskea kaavalla

$$E = h f \,, \tag{2.1}$$

missä h on Planckin vakio ja f on aaltoliikkeen taajuus. Sähkömagneettinen säteily on ionisoivaa, kun sen energia riittää ioniparien muodostamiseen. Ionisoiva

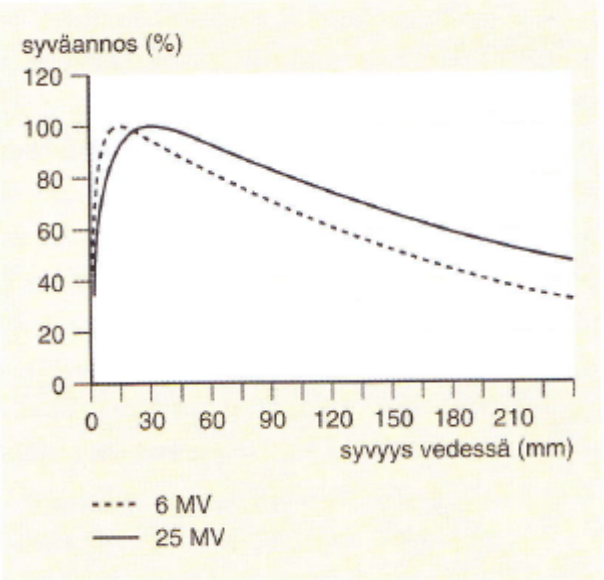
sähkömagneettinen säteily on välillisesti ionisoivaa säteilyä, koska sen vuorovaikutus aineen kanssa tuottaa ionisoivia sekundaarihiukkasia. [6, 7]

Ionisoivan sähkömagneettisen säteilyn energian absorboituminen aineeseen tapahtuu pääosin valosähköisessä absorptiossa, comptonabsorptiossa ja parinmuodostuksessa. valosähköisessä ilmiössä ja Comptonin ilmiössä vuorovaikutus tapahtuu atomin elektronien kanssa. Parinmuodostuksessa vuorovaikutus tapahtuu atomin ytimen sähkömagneettisessa kentässä. [6, 7]

Valosähköisessä ilmiössä atomin elektroni ottaa vastaan kaiken fotonin energian ja sinkoutuu ulos atomista. Elektroni saa liike-energiakseen fotonin energian ja elektronin sidosenergian erotuksen. Atomista sinkoutuneet elektronit ovat lähinnä atomin sisäkuorten (K, L, M) elektroneja. Atomista sinkoutuneen elektronin jättämä aukko täyttyy ylemmältä elektronikuorelta siirtyvällä elektronilla, jolloin vapautuu karakteristisen röntgensäteilyn fotoneja. Valosähköinen ilmiö on merkittävin vuorovaikutus pienillä energioilla. Esimerkiksi kudoksessa se on hallitseva ilmiö alle 30 KeV:n fotonin energioilla. [7]

Comptonin ilmiössä fotoni siroaa heikosti sitoutuneesta elektronista, jonka sidosenergia on huomattavasti pienempi kuin fotonin energia. Sironnassa fotonin energiaa siirtyy elektronille liike-energiaksi. Ilmiön tuloksena saadaan aina sironnut fotoni ja elektroni, jonka energiaa absorboituu nopeasti väliaineeseen. Comptonin ilmiö riippuu aineen elektronitiheydestä. Kudoksessa Comptonin ilmiö on merkittävin vuorovaikutus 0,06–20 MeV:n fotonin energioilla. [7]

Ytimen voimakenttään joutunut fotoni voi hävitä, jos sen energia ylittää kynnysarvon 1,022 MeV. Hävinneen fotonin energiasta syntyvät elektroni ja positroni, jotka saavat hävinneestä fotonista liike-energiaa. Positroni yhdistyy elektroniin sen menettäessä tarpeeksi liike-energiaansa, jolloin syntyy häviämissäteilyä. Parinmuodostuksessa lopputuloksena saadaan positronin ja elektronin liike-energian absorboituminen väliaineeseen ja kaksi vastakkaiseen suuntaan lähtevää 0,511 MeV:n fotonia. Parinmuodostus on tärkein vuorovaikutus kudoksessa yli 20 MeV:n fotonin energioilla. [7]



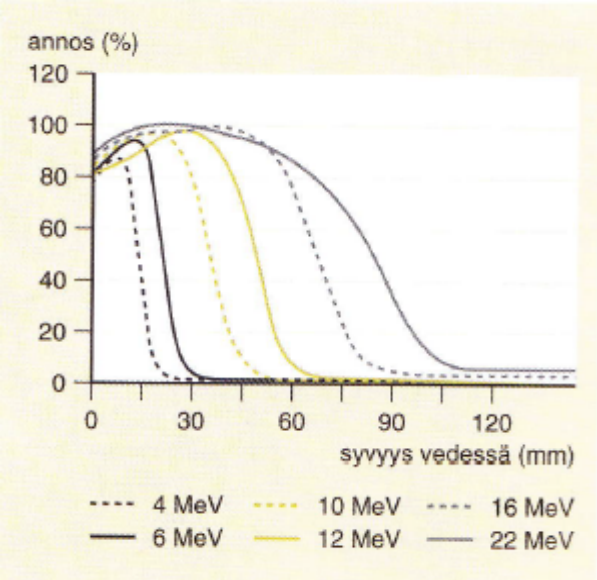
Kuva 2.2: 6 MV:n ja 25 MV:n kiihdytysjännitteillä tuotetun fotonisäteilyn syväannoskäyrät vedessä.

Syväannoskäyrä kuvaa annosta syvyyden funktiona vedessä tai kudoksessa. Syväannoskäyrä normitetaan yleensä annosmaksimiin. [8] Kuvassa 2.2 [6] ovat tyypilliset fotonisäteilyn syväannoskäyrät vedessä. Syväannoskäyrän alussa annos kasvaa, kunnes sekundaarielektronikertymä saavuttaa tasapainotilan. Tasapainotilassa saavutetaan annosmaksimi, jonka syvyys riippuu käytettävän säteilyn energiasta. Annosmaksimin jälkeen syväannoskäyrä laskee likimain eksponentiaalisesti primaarisäteilyn vaimenemisen takia. Fotonisäteilyn kiihdytysjännitteen kasvaessa pinta-annos pienenee. [6, 1]

2.2.2 Elektronisäteily

Elektronisäteily on elektronien muodostamaa hiukkassäteilyä. Elektronisäteilyn ja aineen vuorovaikutuksista hallitsevimpia ovat elektronien epäelastiset ja elastiset törmäykset atomin elektroniverhon elektronien kanssa. Törmäyksissä elektroniverhon elektronit saavat liike-energiaa elektroniverhoon törmänneeltä elektronilta, mikä saa aikaan molekyylien virityksiä ja ionisaatiota. Elektroniverhoon törmännyt elektroni menettää energiaa ja siroaa. Elektronisäteilyn elektronin energian menetys voi myös olla huomattava, koska positiivisesti varautunut ydin voi sähköisen vetovoiman vuoksi muuttaa elektronin suuntaa voimakkaasti. Tilanteessa voi mahdollisesti syntyä fotoni, jolle elektroni luovuttaa osan energiastaan. Syntynyttä fotonisäteilyä

kutsutaan jarrutussäteilyksi. Syntyvä jarrutussäteily riippuu elektroneiden energiasta ja väliaineen kemiallisesta järjestysluvusta. Jarrutussäteilyä syntyy enemmän, kun elektroneiden energia ja väliaineen järjestysluku on suurempi. [6,7]



Kuva 2.3: Elektronisäteilyn tyypillisiä syväannoskäyriä eri säteilyenergioilla.

Kuvassa 2.3 [6] on tyypillisiä elektronisäteilyn syväannoskäyriä vedessä. Elektronisäteily absorboituu heti pinnasta alkaen, minkä vuoksi pinta-annos on suhteellisen suuri. Annosmaksimin jälkeen annos pienenee nopeasti, koska elektronit ovat menettäneet energiansa. Elektroneilla on äärellinen kantama. Syvemmälle ulottuva elektronisäteilyn häntä koostuu elektronien synnyttämästä jarrutussäteilystä. Toisin kuin fotonisäteilyllä, elektronisäteilyllä energian kasvaessa pinta-annoskin kasvaa. [6]

2.3 Lineaarikiihdytin

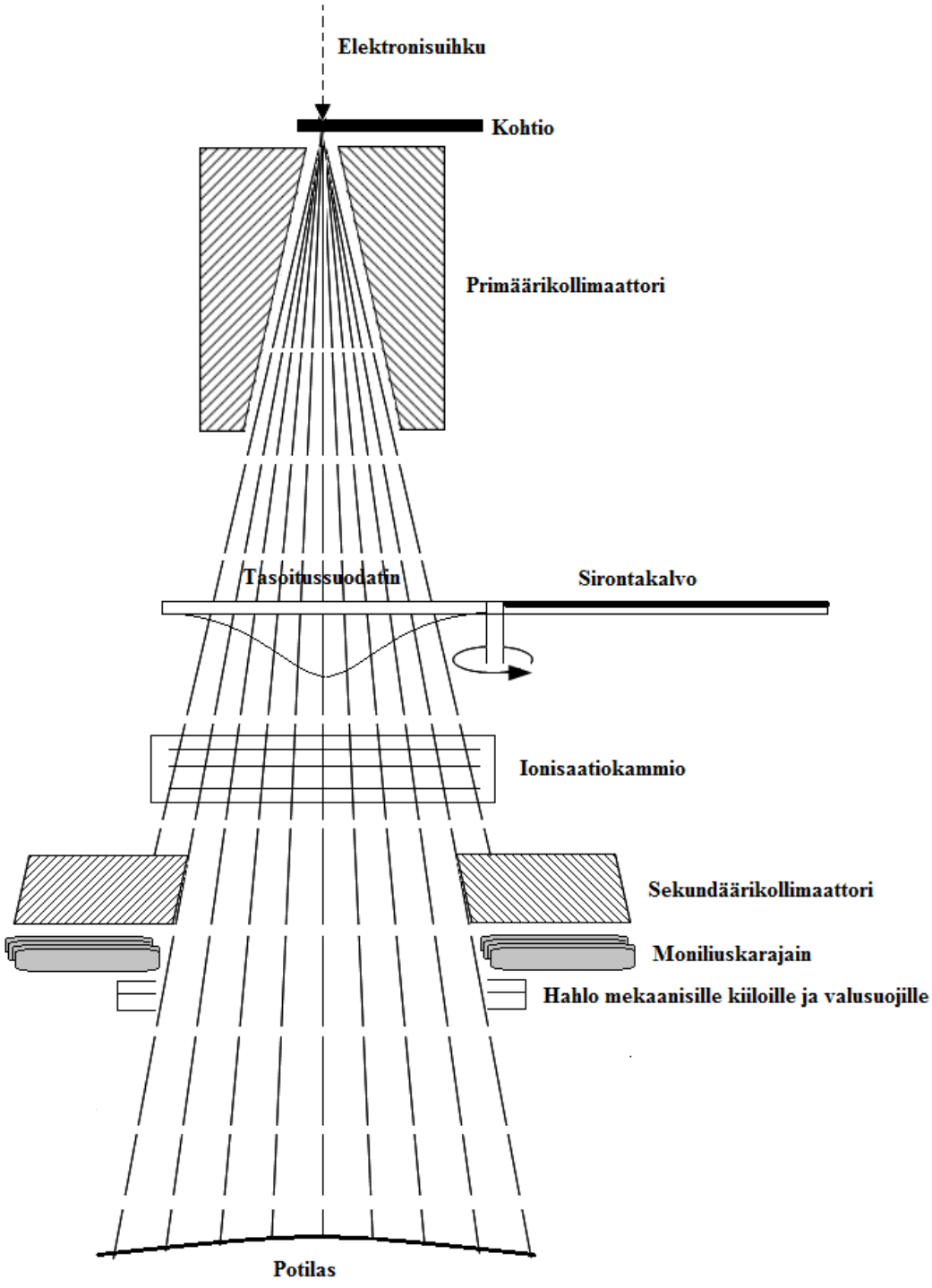
Sädehoito toteutetaan tyypillisesti lineaarikiihdyttimellä (kuva 2.4 [9]), jolla voidaan tuottaa fotonisäteilyä ja elektronisäteilyä. Säteilyn tuottaminen lineaarikiihdyttimellä perustuu elektronien kiihdyttämiseen vaadittuun energiaan suorassa aaltoputkessa suuritaajuisilla sähkökentillä. Fotonisäteilyn tuottamiseksi suurienergiset elektronit ohjataan voimakkailla kääntömagneeteilla törmäämään kohtioon. Elektronit joutuvat voimakkaaseen hidastuvaan liikkeeseen, jolloin syntyy jarrutussäteilyä. Syntyvä jarrutussäteily muodostaa pisaran muotoisen säteilyn intensiteettijakauman ja jatkuvan

säteilyspektrin, joka riippuu käytettävästä elektronien kiihdytysjännitteestä. Elektronisäteilyä käytettäessä kohtion tilalle siirretään sirontakalvo, joka muokkaa ohuen elektronisuihkun tasaiseksi ja leveäksi. Sirontakalvon paksuus ja materiaali valitaan siten, että suurin osa elektroneista siroaa kalvosta ilman, että elektronit menettävät kalvomateriaalin atomin ytimien sähkökentän vuorovaikutuksesta energiaa. Pieni osuus elektronien energiasta kuitenkin häviää, minkä vuoksi elektronisäteilyllä on myös fotonisäteilykomponentti. [10, 6]

Kuva 2.4: Varian Clinac iX -lineaarikiihdytin.

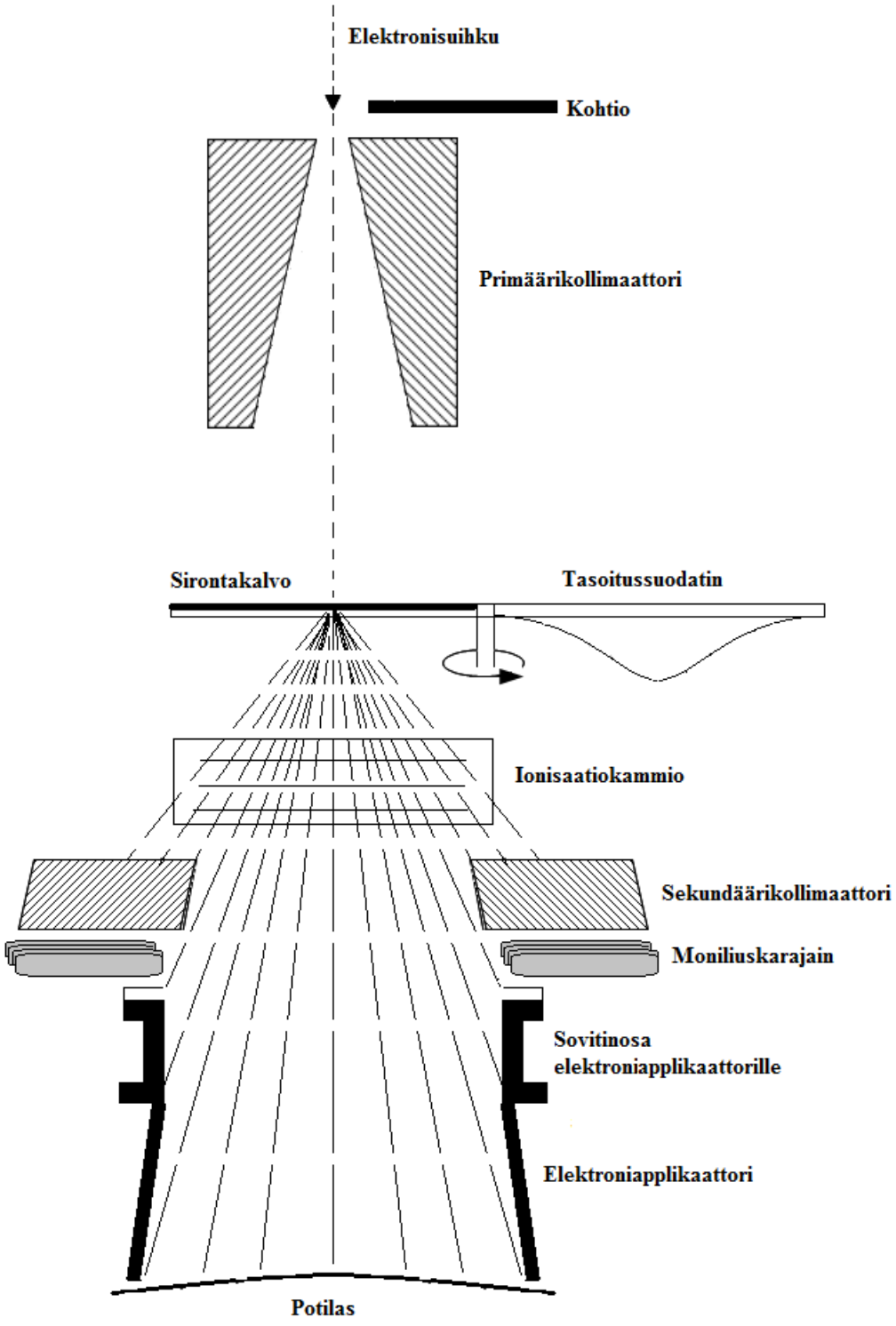
Elektronisäteilyllä hoidetaan yleensä lähellä ihon pintaa sijaitsevia hoitokohteita, koska elektronisäteilyn energia absorboituu nopeasti väliaineeseen pinnasta alkaen. Fotonisäteilyllä hoidetaan yleensä syvemmällä sijaitsevia hoitokohteita, koska fotonisäteilyn energia absorboituu hitaammin väliaineessa kuin elektronisäteilyn energia. [6]

Kuvassa 2.5 [10] kohtiosta tuleva fotonisäteilykeila muokataan tasaiseksi tasoitussuodattimen avulla, minkä jälkeen säteilykeila kulkeutuu ionisaatiokammioon. Ionisaatiokammiossa syntyvän ionisaatiovirran avulla tarkkaillaan annosnopeutta, säteilykeilan symmetriaa ja kokonaisannosta. Lopuksi säteilykeilaa muotoillaan sopivan muotoiseksi sekundaarisen kollimaattorin ja moniliuskakollimaattorin (multileaf collimator, MLC) avulla. [10] Usein joudutaan myös muokkaamaan säteilyn intensiteettijakaumaa, mikä saadaan aikaiseksi esimerkiksi dynaamista kiilaa hyödyntäen eli liikuttamalla toista sekundaarikollimaattorin leukaa säteilytyksen aikana. Intensiteettiä voidaan myös muokata liikuttamalla MLC:itä säteilytyksen aikana, mitä kutsutaan intensiteettimoduloiduksi sädehoidoksi (intensity modulated radiotherapy, IMRT). [8] IMRT:tä käytetään usein haastavissa sädehoidoissa, joissa vaaditaan suurta tarkkuutta.



Kuva 2.5: Fotonisäteilykeila.

Elektronit siroavat ilmassakin voimakkaasti, minkä vuoksi elektronisäteilykeilan rajaaminen on tehtävä lähellä potilaan ihoa hoitokoneen päähän kiinnitettävällä elektroniapplikaattorilla (kuva 2.6 [10]). [10] Elektronisäteilykeila rajataan hoitokohteen muotoiseksi elektroniapplikaattorin päähän kiinnitettävällä metallisuojalla. Metallisuoja valetaan sopivan muotoiseksi yleensä annossuunnitelmakohtaisesti.



Kuva 2.6: Elektronisäteilykeila.

Lineaarikiihdyttimen tuottama säteilymäärä ilmoitetaan monitoriyksiköinä (monitor unit, MU), mikä kuvaa ionisaatiokammion herkkyyttä. Säteilymäärän ja absorboituneen annoksen välinen suhde [MU/Gy] annosjakauman normalisointipisteessä

saadaan annostaulukosta. Annos lasketaan monitoriyksiköinä annossuunnittelujärjestelmällä kullekin lineaarikiihdyttimelle ominaisesta datasta. Lasketut monitoriyksiköt ovat siis hoitokoneesta riippuvaisia. Suhteeseen vaikuttavat myös muun muassa säteilylaji, kenttäkoko, elektroniapplikaattorin koko ja kiilat. [6]

2.4 Ulkoisen sädehoidon suunnittelu ja toteuttaminen

Ulkoisen sädehoito suunnitellaan ja toteutetaan lääkäreiden, sairaalafyysikoiden ja röntgenhoitajien yhteistyönä. Lääkärit tekevät potilaan hoitoon vaikuttavat päätökset, kuten päätökset annettavasta sädeannoksesta ja hoidon fraktioinnista eli hoitokerroista. Sairaalafyysikot vastaavat dosimetriasta, annossuunnittelusta, sädehoitomenetelmistä ja laitetekniikasta sekä varmistavat, että potilaalle annetaan lääkärin määrittelemä annos. Röntgenhoitajilla on vastuu potilaiden kuvantamisesta sekä hoidon käytännön toteutuksesta annossuunnittelun eri työvaiheissa ja hoitokoneilla. [6]

Sädehoito suunnitellaan ja toteutetaan seuraavien vaiheiden mukaisesti: [6]

1) hoitoasennon päättäminen
2) kohteen määrittäminen
3) annoksen ja fraktioinnin päättäminen
4) annossuunnittelu
5) kenttien kohdistus potilaaseen
6) hoidon toteuttaminen lineaarikiihdyttimellä.

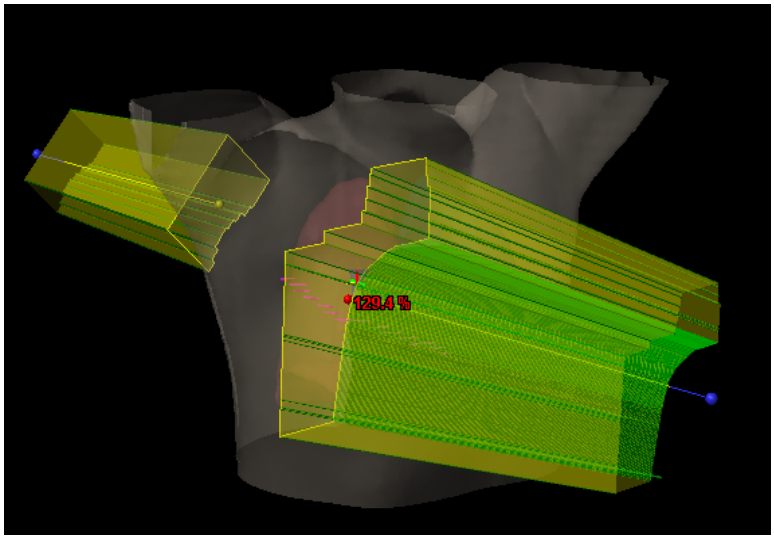
Annoksen ja fraktioinnin päättämisen jälkeen annossuunnittelussa sairaalafyysikot tai annossuunnittelusta vastaavat röntgenhoitajat laskevat yleensä tietokonetomografia-leikekuvien avulla muodostettuun kolmiulotteiseen malliin mahdollisimman hyvän annosjakauman annoslaskentajärjestelmällä. Tavoitteena on, että hoitokohde saa määrätyn annoksen ja terveet kudokset säästyvät säteilyltä mahdollisimman hyvin eivätkä riskielimille asetetut annosrajat eivät ylity. Annossuunnittelun tuloksena saadaan annossuunnitelma, joka sisältää kaikki tarvittavat parametrit potilaan hoidon toteuttamiseen hoitokoneella. Annossuunnitelma sisältää muun muassa tiedot potilaan

paikasta, säteilyenergioista, kenttien suunnasta, kenttien muodosta ja kenttien monitoriyksiköistä. [6]

2.5 Rinnanpoiston jälkeinen sädehoito kolmella kentällä

Rinnanpoiston jälkeen annettavan sädehoidon toteutus poikkeaa Suomen sädehoitoyksiköiden välillä hieman toisistaan. Oulun yliopistollisessa sairaalassa hoidot toteutetaan tyypillisesti kolmella kentällä.

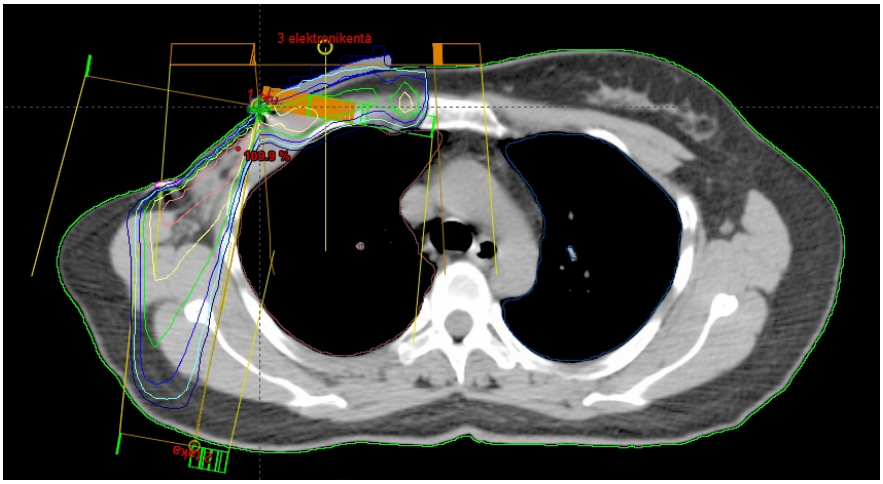
Kolmen kentän hoidoissa arpialue hoidetaan elektronikentällä ja solis-kainaloalueen imusolmukkeet hoidetaan vastakkaisilla fotonikentillä. Elektronikenttä ja etufotonikenttä hoidetaan potilaan etupuolelta siten, että kenttien reunat yhtyvät ihon pinnalla. Elektronikentän säteilykeila on käännettynä potilaan pituussuunnassa 10 astetta potilaan päähän päin. Vasemmanpuoleisessa hoidossa etufotonikenttää käännetään 10 astetta potilaan oikealle puolelle. Vastaavasti oikeanpuoleisessa hoidossa etufotonikenttää käännetään 10 astetta potilaan vasemmalle puolelle. Takafotonikenttä hoidetaan potilaan takapuolelta vastakkaisesta suunnasta kuin etufotonikenttä. Kuvassa 2.7 on erään annossuunnitelman etu- ja takafotonikenttien säteilykeilat Eclipse-annossuunnitteluohjelmassa. Takafotonikentän (vas.) säteilykeilan alareuna on muotoiltu MLC:illä elektronikentän ja fotonikentän sauman annosmaksimien pienentämiseksi. Etufotonikentän säteilykeilan (oik.) vasenta yläkulmaa on muotoiltu MLC:illä olkavarren pään suojaamiseksi ja oikeaa yläreunaa kilpirauhasen suojaamiseksi. Oikea alareuna on muotoiltu metallisella valusuojalla. Oikean alareunan aukko hoidetaan elektronikentällä, joka on valittuna pois kuvan näkymästä. Elektronikentän säteilykeila rajataan kokonaan metallisella valusuojalla.



Kuva 2.7: Erään oikean rinnanpoiston jälkeisen sädehoidon annossuunnitelman taka- ja etufotonikenttien säteilykeilat ja tietokonetomografialla muodostettu potilasmalli.

Tyypillisesti etufotonikenttä hoidetaan 6 MV:n fotonisäteilyllä ja takafotonikenttä 10 MV:n, 15 MV:n tai 18 MV:n fotonisäteilyllä. Elektronikentässä voidaan käyttää säteilyenergioina 6 MeV, 9 MeV, 12 MeV, 15 MeV, 16 MeV tai 20 MeV, mutta yleensä 6 MeV tai 9 MeV on riittävä säteilyenergia. Kohdealueen annos normalisoidaan takakentän keskipisteen tasossa takakentän keskiakselin sisäänmenopisteen ja etukentän keskiakselin tason sisäänmenopisteen välisen janan 1/3 pituuteen etukentän sisäänmenopisteestä laskien. Ennen normitusta etukentän painokertoimeksi asetetaan 1,3 ja takakentän painokertoimeksi 0,7. Painokertoimilla voidaan säätää kentän osuutta säteilyannoksesta. Tyypillisesti fotonikenttien kohdealueelle suunnitellusta annoksesta noin 65 % saadaan etukentästä ja 35 % takakentästä. Suunniteltu 50 Gy:n kokonaisannos annetaan 2 Gy:n fraktioissa viitenä päivänä viikossa fotonikenttien kohdealueelle ja elektronikentän kohdealueelle.

Kuvassa 2.8 esiintyvässä TT-leikekuvassa on annossuunnitelman isodoosit eli tasa-annoskäyrät. Annossuunnitelma on sama kuvissa 2.7 ja 2.8. Leikekuva on kuvan 2.7 etukentän keskellä näkyvän monivärisen ristikon kohdalta. Elektronikenttään on lisätty bolus (kuvassa sinisellä), joka on potilaan iholle asetettava kudosta jäljittelevä kappale, jolla voidaan muokata annosjakaumaa.



Kuva 2.8: Kuvan 2.7 annossuunnitelman tietokonetomografialeikekuva, jossa on näkyvissä kolme kenttää ja isodoosit.

3 TIEDONLOUHINTA JA TIETÄMYKSEN HANKINTA TIETOKANNOISTA

3.1 Tietämyksen hankinta tietokannoista -prosessi

Tiedonlouhinta (data mining) on tietämyksen hankinta tietokannoista (knowledge discovery in databases, KDD) -prosessin yksi askel, jolla pyritään saamaan suuresta määrästä dataa hyödyllinen informaatio. Tietokannoista tapahtuvassa tiedonlouhinnassa pyritään soveltamaan erilaisia menetelmiä ja algoritmeja, joilla voidaan havaita ja tuottaa erilaisia malleja datasta. Tiedonlouhinnan sovellusalueita ovat muun muassa lääketiede, Web-sovellukset, tietoverkkojen tietoturva, taloustiede, ilmaston mallintaminen ja tähtitiede. Viimeisen parin vuosikymmenen aikana tiedonlouhinta ja KDD-sovellukset ovat saaneet paljon huomiota ja niistä on tullut keskeinen osa erilaisia organisaatioita. [11, 12]

KDD-prosessi [11] sisältää yhdeksän askelta:

1) sovelluksen toiminta-alueen oppiminen

   – keskeisen sisällön ymmärtäminen ja sovelluksen tavoitteen asettaminen

2) datan valinta

   – käytettävän datan valinta tietokannan datasta

3) datan siistiminen ja esikäsittely

   – kohinan tai poikkeavuuksien poistaminen

   – tyhjien datakenttien käsittely

   – tuntemattomien arvojen käsittely

4) datan redusointi ja projisointi

   – datan dimension alentaminen tai muuntaminen

5) tiedonlouhintafunktion valinta

   – tiedonlouhinta-algoritmin tuottaman mallin tarkoitus (esim. luokittelu, klusterointi ja regressio)

6) tiedonlouhinta-algoritmin valinta

– sopiva menetelmä mallien löytämiseen datasta

7) tiedonlouhinta

– luokittelu

– klusterointi

– sekvenssien mallintaminen

– regressio

8) tulkinta

– mallien tulkinta

– mallien visualisointi

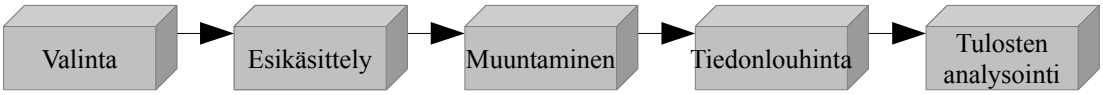
– mahdollinen palaaminen edellisiin vaiheisiin

9) hankitun tiedon käyttäminen

– uuden datan luokittelu

– poikkeavuuksien löytäminen uudesta datasta

Kuvassa 3.1 [11] on KDD-prosessin päävaiheet.



Kuva 3.1: KDD-prosessi.

3.2 Poikkeavuuksien havaitseminen

3.2.1 Johdantoa poikkeavuuksien havaitsemiseen

Poikkeavan havainnon havaitseminen (outlier detection) viittaa ongelmaan, jossa pyritään löytämään tapauksia datasta, jotka eivät noudata odotettua normaalia käyttäytymistä. Näitä epätavallisia havaintoja nimitetään kirjallisuudessa muun muassa poikkeaviksi havainnoiksi, anomalioiksi (anomaly), uutuuksiksi (novelty), poikkeuksiksi (exception), vioiksi (fault), häiriöiksi (noise) ja virheiksi (error). [13]

Poikkeavuudet välittävät tärkeää ja usein keskeistä tietoa datasta monilla eri sovellusaloilla. Esimerkiksi terveydenhoidossa potilaan poikkeava sairauskertomus voi olla merkki sairaudesta. Tietoverkon normaalista poikkeava tietoliikenne voi tarkoittaa tietomurtoa. Vastaavasti epätavalliseen kellonaikaan tapahtuva suuri luottokorttimaksu voi olla merkki varastetusta luottokortista. [13]

Poikkeavien havaintojen havaitsemiseen on olemassa kirjava joukko erilaisia menetelmiä. Menetelmän valinta riippuu datasta ja sovelluskohteesta. Poikkeavuuksien havaitsemisen ja siihen liittyvien käsitteiden terminologia voidaan erotella itsenäisiksi kokonaisuuksia sovellusalasta riippuen. [14, 13]

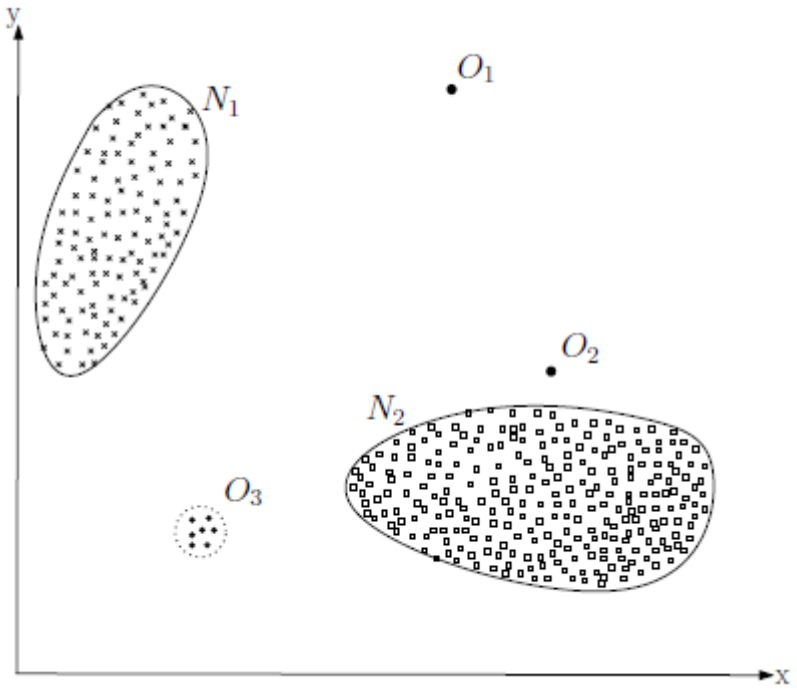
Havainto määritellään poikkeavuudeksi Grubbsin [15] mukaan seuraavasti.

> Poikkeava havainto on se, joka poikkeaa merkittävästi muista jäsenistä havaintoaineistossa.

Barnet ja Lewis [16] määrittelevät havainnon poikkeavuudeksi seuraavasti.

> Havainto (tai havaintojen osajoukko) on poikkeavuus, kun se on epäjohdonmukainen verrattuna muuhun datajoukkoon.

Poikkeavalla havainnolla on pieni todennäköisyys, että se kuuluu saman havaintoaineiston tilastolliseen jakaumaan. Ääriarvo on havainto, jonka todennäköisyys havaintoaineistossa esiintymiseen on pieni, mutta jota ei pystytä osoittamaan kuulumaan mihinkään muuhun tilastolliseen jakaumaan. Poikkeavan havainnon löytämisen jälkeen on syytä selvittää, voidaanko poikkeavuudelle löytää selitys. [16]



Kuva 3.2: 2-dimensioisen pistejoukon poikkeavuudet.

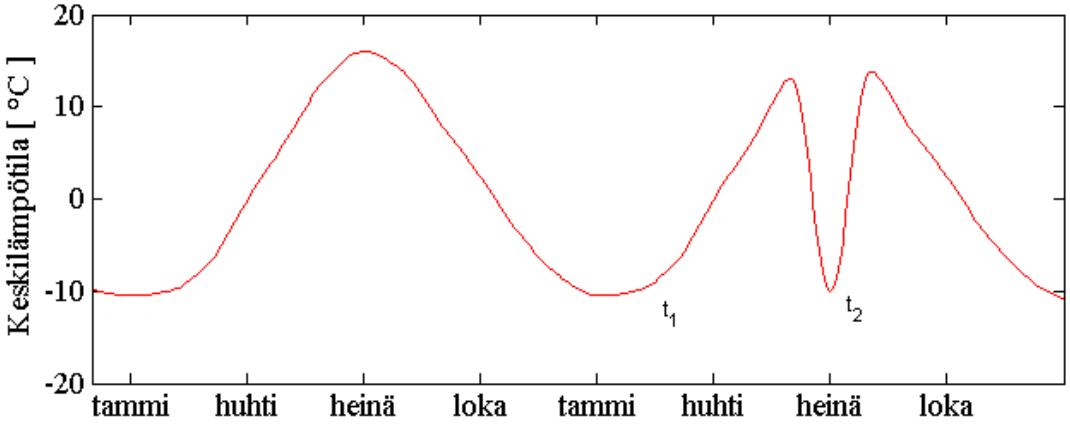
Kuvassa 3.2 [13] on yksinkertainen 2-dimensioinen pistejoukko. Kuvassa on kaksi normaalia joukkoa N_1 ja N_2. Normaalijoukoista eroavat pisteet O_1 ja O_2 ovat poikkeavia tapauksia, kuten myös alueella O_3 sijaitsevat pisteet. Poikkeavien havaintojen esiintyminen johtuu ihmisten virheistä, mittalaitteiden virheistä, luonnollisesta jakaumien deviaatiosta, vilpillisestä toiminnasta tai systeemien käyttäytymisen muuttumisesta. [13]

3.2.2 Poikkeavuustyypit

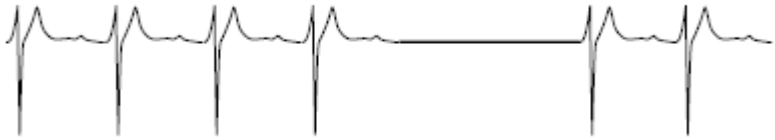
Olennainen osa poikkeavuuksien havainnoinnissa on määritellä, minkä tyypin poikkeavuuksia pyritään löytämään. Poikkeavuudet voidaan luokitella kolmeen kategoriaan perustuen niiden rakenteeseen ja relaatioon muuhun dataan verrattuna. [13]

Tyypin I poikkeavuudet ovat annetussa datajoukossa yksittäisiä poikkeavia tapauksia. Ne ovat tyypillisimpiä kohteita poikkeavuuksien havaitsemisessa. Tyypin I poikkeavuuksien havainnointiin käytettävät menetelmät analysoivat yksittäisten tapausten relaatioita muihin tapauksiin [13]. Esimerkiksi kuvassa 3.2 olevia tyypin I poikkeavuuksia ovat tapaukset O_1 ja O_2.

Tyypin II poikkeavuudet ovat yksittäisiä poikkeavuuksia vastaavasti, kuten tyypin I poikkeavuudet, mutta ne havaitaan tietyssä viitekehyksessä. Tyypin II poikkeavuudet määritellään viitekehykseen verraten. Datan rakenne määrittää viitekehyksen, jonka vuoksi datan rakenne tulee määritellä osana ongelman muodostamista [13]. Oletetaan, että kuvan 3.3 [13] kuvaavan esimerkiksi Oulun keskilämpötilaa ajanfunktiona. Kuvan perusteella lämpötila helmikuussa t_1 ja t_2 heinäkuussa olisivat molemmat noin $-10\,°C$. Nyt kuitenkin lämpötila t_2 esiintyy eri viitekehyksessä kuin lämpötila t_1, jonka vuoksi se luokitellaan tyypin II poikkeavaksi havainnoksi.



Kuva 3.3: Tyypin II poikkeava havainto t_2.



Kuva 3.4: Tyypin III poikkeava havainto.

Tyypin III poikkeavissa havainnoissa tapauksien osajoukko poikkeaa muuhun dataan verrattuna. Tyypin III poikkeavuudet eivät ole yksinään poikkeavuuksia, mutta niiden luoma alirakenne poikkeaa muusta datasta. Nämä poikkeavuudet voidaan tunnistaa silloin, kun datalla on toistuva tai avaruudellinen rakenne. [13] Kuvassa 3.4 [13] esiintyvä tasainen osa signaalia on tyypin III poikkeavuus, koska sama amplitudi esiintyy tavallista pidemmän ajan. Kuvan III signaali esittää ihmisen sydänsähkökäyrää. Kuvassa 3.2 ryhmän O_3 pisteet ovat myös tyypin III poikkeavuuksia, koska ryhmän rakenne poikkeaa muista ryhmistä. Ryhmässä O_3 on huomattavasti vähemmän alkioita kuin normaaleissa joukoissa N_1 ja N_2.

3.2.3 Yleistä poikkeavuuksien havaitsemismenetelmistä

Poikkeavuuksien havaitsemista voidaan erotella kolmeen keskeiseen lähestymistapaan. Tyypin I lähestymistavassa poikkeavuudet määritetään ilman etukäteistietoa datasta. Käytännössä poikkeavuudet havaitaan datasta ohjaamattomien klusterointimenetelmien avulla. [14] Klusteroinnissa data luokitellaan ryhmiin eli klustereihin, jotka ovat sopivia ratkaistavalle ongelmalle [17]. Klustereista etäisimmät pisteet merkitään poikkeaviksi tapauksiksi. Tyypin II menetelmissä datasta on olemassa malli normaaleille tapauksille ja epänormaaleille tapauksille. Tämä lähestymistapa on analoginen ohjattuun luokitteluun. Tyypin II lähestymistapa vaatii etukäteen normaaliksi ja epänormaaliksi merkittyä dataa. Tyypin III menetelmissä on olemassa malli vain normaaleille tapauksille. Tyypin III menetelmiä voidaan pitää puoliohjattuna opettamisena, koska vain normaali data opetetaan ohjatusti, mutta algoritmi oppii ohjaamattomasti tunnistamaan epänormaalin datan. [14]

Poikkeavuuden havaitsemismenetelmä oppii opetusdatan avulla, ja tämän jälkeen menetelmä testataan validointidatan avulla. Validoinnissa testataan poikkeavuuden havaitsemismenetelmän kyky havaita poikkeavuuksia validointidatasta, joka koostuu merkityistä normaaleista ja epänormaaleista tapauksista. Poikkeavuuden havaitsemismenetelmien suorituskykyä voidaan arvioida niiden kyvykkyydellä tunnistaa poikkeavuuksia annetusta datasta, jonka tapaukset on merkitty. [13] Tärkeimpänä kriteerinä on maksimoida oikein havaittujen poikkeavuuksien lukumäärä ja minimoida normaaliksi luokiteltujen poikkeavuuksien lukumäärä. [18] Poikkeavuuden havaitsemisalgoritmin tulokset validointidatalle voidaan esittää konvoluutiomatriisin avulla (taulukko 3.1). [13]

Taulukko 3.1: Konvoluutiomatriisi.

		Todelliset	
		Poikkeavuudet	Normaalit tapaukset
Ennustetut	Poikkeavuudet	*TP*	*FP*
	Normaalit tapaukset	*FN*	*TN*

Konvoluutiomatriisissa (taulukko 3.1) parametri *TP* (true positive) on oikein havaittujen poikkeavuuksien lukumäärä, *FP* (false positive) on poikkeavuuksiksi luokiteltujen normaalien tapausten lukumäärä, *FN* (false negative) on normaaleiksi luokiteltujen poikkeavuuksien lukumäärä ja *TN* (true negative) on oikein luokiteltujen normaalien tapausten lukumäärä. Konvoluutiomatriisin parametrien avulla voidaan laskea tyypillisesti käytetyt poikkeavuuden havaitsemismenetelmien suorituskykyä kuvaavat lukuarvot: oikein havaittujen poikkeavuuksien suhde *TPR* (true positive rate), poikkeavuuksiksi luokiteltujen normaalien tapausten suhde *FPR* (false positive rate) ja tarkkuus. [19]

$$TPR = \frac{TP}{TP+FN} \tag{3.1}$$

$$FPR = \frac{FP}{FP+TN} \tag{3.2}$$

$$Tarkkuus = \frac{TP+TN}{TP+FN+FP+TN} \tag{3.3}$$

Ideaalitapauksessa $TPR = 1$, $FPR = 0$ ja $Tarkkuus = 1$. Tarkkuus on siis oikein luokiteltujen tapausten suhde kaikkiin tapauksiin.

3.2.4 Ristiinvalidointi

Poikkeavuuksien havaitsemismenetelmän testaukseen ei usein ole saatavissa kattavaa testidataa. Lisäksi koneoppimiseen ja tekoälyyn perustuvat poikkeavuuksien havaitsemismenetelmän yleistyskyky tulisi testata ylioppimisen välttämiseksi. Ristiinvalidoinnilla voidaan menetelmän yleistyskyky ja tarkkuus arvioida ilman, että osaa datasta tarvitsisi varata menetelmän testaukselle.

Ristiinvalidoinnin perusidea voidaan jäsennellä seuraavasti. Olkoon D datajoukko, jonka tapauksien luokat ovat tiedossa. Seuraavaksi data jaetaan satunnaisesti ja approksimatiivisesti k:n yhtä suureen osajoukkoon $D_1, D_2, ..., D_k$ siten, että jokainen osajoukko koostuu tapauksista, jotka eivät kuulu muihin osajoukkoihin. Menetelmä opetetaan k kertaa siten, että jokaisella kerralla $t \in \{1, 2, ..., k\}$ se opetetaan datalla $D \setminus D_t$ ja testataan datalla D_t. Tämä tarkoittaa sitä, että jokainen tapaus on

kerran testijoukossa ja $k-1$ kertaa opetusjoukossa. Menetelmän tarkkuusestimaatti ristiinvalidoinnilla saadaan laskemalla kunkin testijoukon tarkkuus ja laskemalla keskiarvo näistä tarkkuuksista. [20]

Kaksi todennäköisesti käytetyintä ristiinvalidointimenetelmää ovat yksi pois -ristiinvalidointi (leave-one-out cross-validation) ja 10-kertainen ristiinvalidointi (10-fold cross-validation). Yksi pois -ristiinvalidoinnissa arvoksi *k* asetetaan tapauksien lukumäärä N eli jokaiseen osajoukkoon D_k kuuluu yksi tapaus. Menetelmä opetetaan ja testataan näin ollen *N* kertaa siten, että yksi tapaus jätetään aina testitapaukseksi. 10-kertaisessa ristiinvalidoinnissa arvoksi *k* valitaan 10 eli data jaetaan 10 osajoukkoon. Menetelmä opetetaan ja testataan näin ollen 10 kertaa siten, että 10 % tapauksista jätetään aina testitapauksiksi. [20]

Ristiinvalidoinnilla saatava estimaatti menetelmän toimivuudesta riippuu siitä, kuinka moneen osajoukkoon data jaetaan. 10-kertainen ristiinvalidointi saattaa olla tutkimuksien mukaan yksi pois -ristiinvalidointia parempi menetelmä, kun valitaan paras luokittelija muiden luokittelijoiden joukosta asetettuun ongelmaan. Lisäksi 10-kertainen ristiinvalidointi on yksi pois -ristiinvalidointia laskennallisesti nopeampi toteuttaa. Kohavi [20] suosittelee menetelmän valitsemiseen ositettua 10-kertaista ristiinvalidointia (stratified 10-fold cross-validation), joka on eroaa hieman 10-kertaisesta ristiinvalidoinnista. Menetelmät eroavat osajoukkoihin jakamisessa siten, että ositetussa 10-kertaisessa ristiinvalidoinnissa osajoukoissa esiintyvien tapausten luokat ovat approksimatiivisesti samassa suhteessa kuin koko datajoukossa esiintyvät luokat. [20]

3.2.5 Sädehoidon annossuunnitelmien poikkeavuuksien havaitseminen

Poikkeavuuksien havaitsemista sädehoidon annossuunnitelmista on tehty vain vähän tutkimusta. Azmandian ym. [21] ovat julkaisseet kokonaisen tutkimuksen aiheesta. Naqa on julkaisut tutkimuksesta ainoastaan posterin [22], ja projektista [23] on saatavilla ainoastaan lyhyt yleisen tason kuvaus.

Azmandian ym. [21] soveltavat eturauhasen neljän kentän sädehoidon virheellisten annossuunnitelmien havaitsemiseen tehdyssä tutkimuksessa sovellettiin koneoppimisen ja tiedonlouhinnan menetelmiä. Tutkimuksessa käytettiin *k*:n keskiarvon

ryvästysmentelmää poikkeavuuksien havaitsemiseen. Annossuunnitelmat haettiin itse kirjoitetulla ohjelmointikoodilla Bostonin Massachusettsin keskussairaalan IMPAC-rekisteri- ja verifiointijärjestelmästä.

Klusterien muodostamista varten valittiin 1 000 annossuunnitelmaa ja menetelmän testausta varten 650 annossuunnitelmaa. Tutkimuksen tavoitteena oli löytää annossuunnitelmista merkittäviä virheitä. Annossuunnitelmista valittiin 8 muuttujaa, jokaisesta hoitokentästä monitoriyksiköt ja säteilyenergiat. Klusteroinnin tuloksena saatiin kahdeksan klusteria. Klusterien oletettiin noudattavan Gaussin jakaumaa.

Annossuunnitelman datavektori luokiteltiin poikkeavuudeksi mikäli, jokin sen parametreista poikkeaa klusterin parametrien keskiarvosta enemmän kuin kahden keskihajonnan verran. Menetelmää testattiin keinotekoisesti tuotetuille virheille. Menetelmällä onnistuttiin havaitsemaan 100 %:n suhteelliset virheet monitoriyksiköissä 100 %:n tarkkuudella ja 50 %:n suhteelliset virheet 80 %:n tarkkuudella. FPR:ksi ilmoitettiin 10 %. Säteilyenergioiden virheet tunnistettiin etukentistä 77 %:n tarkkuudella ja muista kentistä 100 %:n tarkkuudella.

Naqa [22] soveltaa poikkeavien annossuunnitelmien havaitsemisessa tukivektorikonetta (support vector machine, SVM). Tukivektorikonetta käytetään tyypillisesti yhden luokan luokitteluongelmien ratkaisussa, ja se on myös hyvin suosittu menetelmä poikkeavuuksien havaitsemiseen. Tukivektorikone pystyy oppimaan epälineaarisen luokkarajan. Menetelmää testattiin stereotaktisella tekniikalla hoidettujen keuhkosyöpien annossuunnitelmilla. Opetusdatan muuttujat liittyivät monitoriyksiköihin, säteilykeilojen energioihin, säteilykeilojen lukumäärään, fraktioiden lukumäärään ja keuhkoannokseen. Kernelifunktiona käytettiin radiaalia kantafunktiota. Opetuksen tarkkuudeksi ristiinvalidoinnilla saatiin 84 % ja testauksessa tarkkuudeksi saatiin 80 %, jossa TPR oli 100 % ja TNR oli 80 %. Testauksessa käytettiin turvallisiksi luokiteltuja tapauksia ja simuloituja riskeiksi luokiteltuja tapauksia.

SciberQuest-organisaatio [23] käyttää projektissaan koneoppimisen menetelmiä pohjana SmartTool-nimiselle poikkeavien annossuunnitelmien havaitsemistyökalulle. Työkalulla on tarkoitus tarkistaa annossuunnitelmat reaaliaikaisesti ennen niiden toteuttamista ja hälyttää käyttäjälle mahdollisista virheistä. Poikkeavuuksien havaitseminen perustuu annossuunnitelmien klusteroimiseen, jonka avulla muodostetaan

malli turvallisista annossuunnitelmista. Kuvauksessa ei mainita tarkemmin menetelmästä, miten poikkeavuudet tunnistetaan.

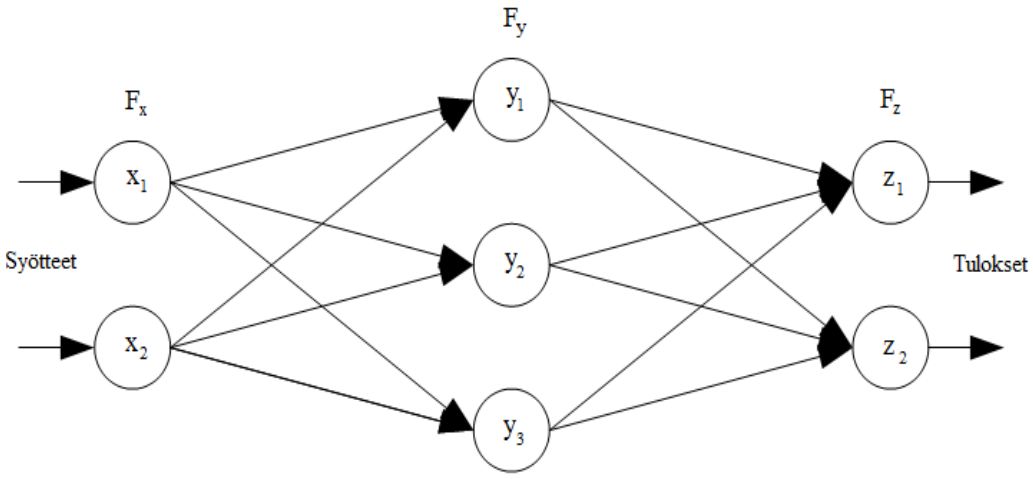
4 NEUROVERKOT

4.1 Johdantoa neuroverkkoihin

Keinotekoiset neuroverkot (artificial neural networks) tai yleisimmin kirjallisuudessa käytettynä neuroverkot pohjautuvat aivojen tapaan prosessoida tietoa. Aivot prosessoivat tietoa täysin eri tavalla kuin tavanomainen tietokone: monimutkaisesti ja epälineaarisesti rinnakkaisilla laskentayksiköillä. Neuronit eli hermosolut ovat aivojen rakenneosasia, joita ihmisen aivokuorella arvioidaan olevan 10^{13} kappaletta. Neuronit suorittavat aivojen operaatioita hyödyntäen valtavaa synapsien eli neuronien välisten yhteyksien määrää. Ihmisen aivokuorella neuronien välisiä yhteyksiä arvioidaan olevan 6×10^{19} kappaletta. [24]

Neuroverkot ovat tietojen käsittelyjärjestelmiä, jotka pyrkivät hyödyntämään ihmisaivojen tapaa prosessoida tietoa. Neuroverkot muistuttavat aivoja kahdella tavalla: verkko oppii tiedon oppimisprosessin avulla ja neuronien välisten yhteyksien voimakkuudet eli synaptiset painot tallentavat tiedon verkkoon. [24]

Neuroverkot voidaan yleisesti mieltää ”mustaksi laatikoksi”, joka vastaanottaa syötteitä (inputs) ja tuottaa tuloksia (outputs). Neuroverkot voivat suorittaa sellaisia tehtäviä kuten luokittelua, optimointia, hallintaa, hahmontunnistusta, mallin täydentämistä ja häiriön poistamista. Esimerkiksi luokittelussa neuroverkolle syötetään hahmo, ja neuroverkko tuottaa tuloksena luokan, johon hahmo kuuluu. Neuroverkolle syötettävä hahmo voi esimerkiksi olla kuvadataa, jonka perusteella kuva luokitellaan. [25]



Kuva 4.1: Tyypillinen 3-kerroksinen perceptron-neuroverkko.

Neuroverkot koostuvat prosessointielementeistä eli neuroneista ja painotetuista yhteyksistä eli synapseista. Jokainen neuroverkon neuroni vastaanottaa kaikista sen synapseista syötteet, minkä jälkeen se suorittaa etukäteen määrätyn matemaattisen operaation ja tuottaa yhden vastearvon. [25]

Kuvan 4.1 [25] monikerroksisella perceptron-neuroverkolla (multilayer perceptron, MLP) neuronit ovat kolmessa kerroksessa. Kerros F_x sisältää neuronit x_1 ja x_2, kerros F_y sisältää neuronit y_1, y_2 ja y_3 sekä kerros F_z sisältää neuronit z_1 ja z_2. Neuroverkoissa kaikki neuronienväliset yhteydet ovat painotettuja. Esimerkiksi kuvan 4.1 neuroverkon neuronien x_1 ja y_2 synapsilla $x_1 \rightarrow y_2$ on painokerroin w_{12}. Synapsien painokertoimet säilyttävät tietoa neuroverkosta. Painokertoimet saavat arvonsa neuroverkon opetusprosessin aikana. Neuroverkon opetusprosessin jälkeen se voi soveltaa opittua tietoa samantyyppisiin tapauksiin. Kaikille neuroverkoille tyypillistä on se, että jokainen neuroni toimii riippumattomasti muista neuroneista. Neuronin tuottama vaste riippuu ainoastaan siihen tulevien yhteyksien tuomista syötteistä. [25]

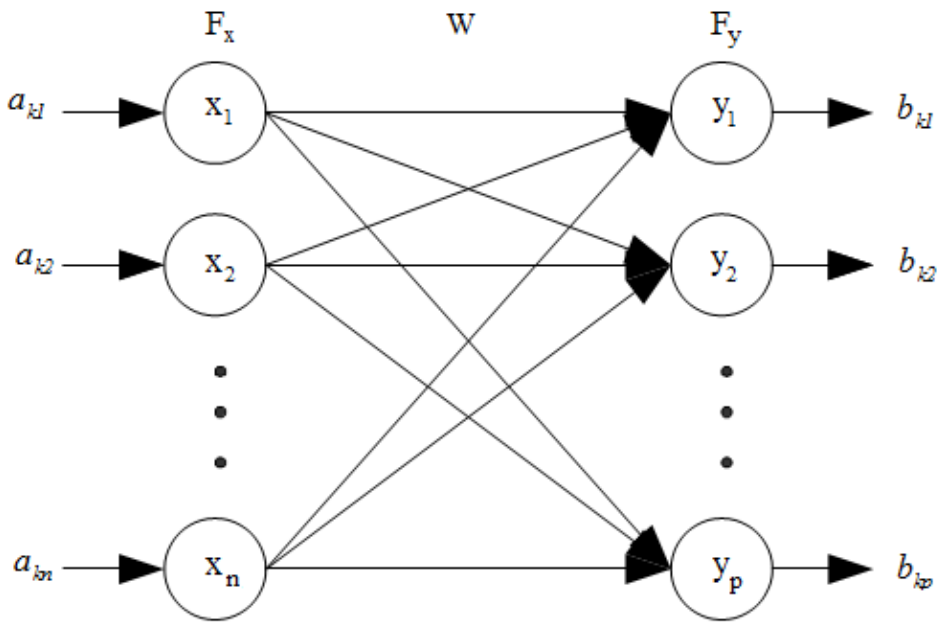
Kuvan 4.2 [25] tilanteessa on esiteltynä kaksikerroksinen neuroverkko. Kerroksen F_X neuroneille syötevektori

$$A_k = (a_{k1}, a_{k2}, \cdots, a_{kn}), \tag{4.1}$$

kun $k, n \in \mathbb{N}$. Vastaavasti kerroksen F_Y neuroneille vastevektori

$$B_k=(b_{k1},b_{k2},\cdots,b_{kp}), \tag{4.2}$$

kun k , $p\in\mathbb{N}$.



Kuva 4.2: Neuroverkon kerrokset F_X ja F_Y kytkettynä painomatriisilla W.

Neuronit muodostavat neuroverkkoihin eritasoisia kerroksia, joita voidaan kuvata vektoreilla. Kuvassa 4.2 syötekerroksen neuronit muodostavat vektorin

$$F_X=(x_1,x_2,\cdots,x_n), \tag{4.3}$$

kun $n\in\mathbb{N}$. Kerroksen F_X jokainen neuroni x_i vastaanottaa syötteen jokaiselta syötevektorin A_k komponentilta a_{ki}. Vastekerroksen neuronit muodostavat kerroksen

$$F_Y=(y_1,y_2,\cdots,y_p), \tag{4.4}$$

kun $p\in\mathbb{N}$. Kerroksen F_Y jokainen y_j vastaa vastevektorin B_k elementtiä b_{kj}. Vastekerroksen neuronien lukumäärän tulee vastata vasteiden lukumäärää.

Painokertoimet tallennetaan painokerroinmatriiseihin. Kuvan 4.2 kaksikerroksiselle neuroverkolle voidaan kerroksen F_X neuronit yhdistää kerroksen F_Y neuroneihin yhdellä painokerroinmatriisilla

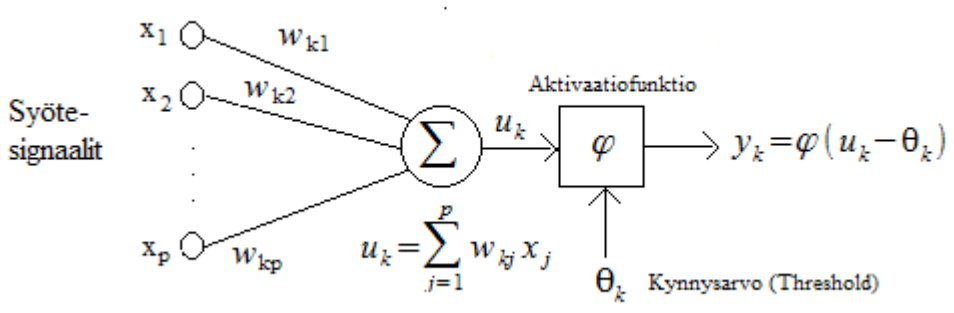
$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1p} \\ w_{21} & w_{22} & \cdots & w_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{np} \end{bmatrix}, \tag{4.5}$$

missä jokainen synapsin painokerroin w_{ij} on yhteys neuronien x_i ja y_j välillä. Neuroverkko on ekvivalentti suunnatun graafin kanssa. Suunnatulla graafilla on solmujen välillä kaaret, jotka päästävät tiedon virtaamaan ainoastaan yhteen suuntaan. Tieto siirtyy kaarten välityksillä solmuille, jotka keräävät tiedon. [25]

Synapsien painokertoimiin on tallennettuna opittu aineisto. Synapsien painokertoimet ovat aktivoivia painokertoimia, jos ne ovat positiivisia. Negatiiviset painokertoimet ovat vaimentavia painokertoimia. Kun painokertoimen arvo on 0, on painokerrointa vastaavan synapsin yhteys poissa käytöstä. [25]

4.2 Neuroni

Neuroni on tiedonkäsittely-yksikkö, joka on neuroverkon keskeinen osa. Kuvassa 4.3 on esiteltynä yksittäisen neuronin malli. Neuronin mallista voidaan luokitella kolme peruselementtiä. Jokainen syöte x_j tulee syötteenä synapsille j, joka on yhdistettynä neuroniin k. Syöte x_j kerrotaan neuronin k synapsin j painokertoimella w_{kj}.



Kuva 4.3: Neuronin malli.

Neuronin k painokertoimilla kerrottujen syötteiden summaksi u_k saadaan [24]

$$u_k = \sum_{j=1}^{p} w_{kj} x_j . \tag{4.6}$$

Summa 4.6 syötetään seuraavaksi neuronin k aktivaatiofunktiolle φ, joka rajoittaa neuronin k vasteen amplitudia. Aktivaatiofunktio φ rajoittaa vasteen amplitudin arvon äärelliseksi. Tyypillisesti normalisoitu neuronin amplitudi on suljetulla välillä $[0,1]$ tai $[-1,1]$. Kuvassa 4.3 esitelty neuronin malli sisältää neuronin k ulkoisen kynnysarvon θ_k, joka pienentää aktivaatiofunktion kuvaamaa vastetta y_k.

Neuronin vaste voidaan kirjoittaa aktivaatiofunktion avulla muodossa [24]

$$y_k = \varphi(u_k - \theta_k). \tag{4.7}$$

Negatiivista kynnysarvoa kutsutaan biakseksi, joka vaikuttaa aktivaatiofunktioon kasvattamalla vastetta.

4.3 Neuroverkon oppiminen

Neuroverkolle tulee näyttää esimerkkitapauksia, joista se vähitellen oppii toimimaan ympäristönsä mukaisesti. Oppimisessa neuroverkko vahvistaa oikeaa ja heikentää väärää toimintaa. Neuroverkon opettaminen on iteratiivinen prosessi, jossa verkko oppii reagoimaan syötteisiin tietyllä tavalla säätämällä sen painokertoimia. [24]

Oppimismenetelmät voidaan luokitella pääosin kahteen kategoriaan: ohjattuun oppimiseen (supervised learning) ja ohjaamattomaan oppimiseen (unsupervised learning). Ohjatussa oppimisessa neuroverkko hyödyntää ulkoista opettajaa tai globaalia tietoa. Ohjatussa oppimisessa on etukäteen tiedossa, kuinka neuroverkon tulisi reagoida kuhunkin syötteeseen. Ohjaamattomassa oppimisessa neuroverkko järjestää datan ja oppii datan rakenteesta ominaisuuksia. [25]

Neuroverkon opetusdata on kätevä käsitellä matriisimuodossa siten, että matriisin rivillä on yhden hahmon data eli datavektori. Datavektorin komponentteja kutsutaan piirteiksi. Datavektorista muodostetaan neuroverkon opetuksessa käytettävä syötevektori, jonka komponenteista kukin muodostaa neuroverkolle syötteen. Yksittäisen syötevektorin esittäminen ja kaikkien painovektoreiden päivitys neuroverkolle on

yksi opetusiteraatio. Opetusiteraatioiden suorittamista kaikille opetusdatan syötevektoreille kerran kutsutaan epookiksi.

4.4 Monikerroksinen perceptron-neuroverkko ja takaisinlevitysalgoritmi

Yksi tunnetuimmista oppimisalgoritmeista on ohjatussa oppimisessa käytetty takaisinlevitysalgoritmi (back-propagation algorithm). Takaisinlevitysalgoritmia on käytetty lähinnä MLP-neuroverkon opettamisessa, ja sitä voidaan pitää yleistyksenä yhtä tunnetulle pienimmän neliösumman (least mean square, LMS) algoritmille. Takaisinlevitysalgoritmin kehitys on tarjonnut laskennallisesti tehokkaan keinon MLP-neuroverkkojen opetukseen. [24]

Takaisinlevitysalgoritmi perustuu laskevan gradientin menetelmään, jolla pyritään löytämään minimi pinnalle, joka kuvaa mallin sopivuutta sovitettavaan dataan [26]. Laskevan gradientin menetelmässä minimi pyritään löytämään negatiivisen gradientin suunnasta. Takaisinlevitysalgoritmin tapauksessa minimipintana on neuroverkon antamien tulosten ero halutuista tuloksista. Tavoitteena on saada toivotut tulokset mahdollisimman lähelle haluttuja tuloksia. [24]

Takaisinlevitysalgoritmilla tapahtuva oppimisprosessi voidaan jakaa pääpiirteittäin kahteen osaan: eteenpäin- ja takaisinpäinsyöttöön. Eteenpäinsyötössä neuroverkkoon syötetään syötevektori, joka tuottaa neuroverkon läpi kulkiessaan tulokset. Eteenpäinsyötössä synapsien painokertoimet pysyvät vakioina. Takaisinpäinsyötössä kaikki painokertoimet säädetään virheen korjaussäännön mukaisesti. Neuroverkon tuottama tulossignaali vähennetään toivotusta tulossignaalista, jolloin saadaan neuroverkon virhesignaali. Virhesignaali johdetaan tämän jälkeen neuroverkon läpi synaptisia painokertoimia vastaan takaisinpäin. Synaptiset painokertoimet säädetään tämän takaisinpäin etenevän virhesignaalin avulla. [24]

Virhesignaali $e_j(n)$ tuloskerroksen neuronin j tuloksena iteraatiolla n (neuroverkon syötevektorille n) määritellään seuraavasti [24]

$$e_j(n) = d_j(n) - y_j(n), \tag{4.8}$$

missä $d_j(n)$ on toivottu tulosignaali ja $y_j(n)$ on tulosignaali neuronille j. Hetkellinen virheiden neliöiden summa saadaan laskettua kaavalla [24]

$$E(n)=\frac{1}{2}\sum_{j\in C} e_j^2(n), \tag{4.9}$$

missä joukkoon C kuuluvat kaikki tuloskerroksen neuronit. Olkoon neuroverkon opetusdatan syötevektoreiden lukumäärä N. Minimoitava virhefunktio määritellään funktion $E(n)$ keskiarvona [24]

$$E_{av}=\frac{1}{N}\sum_{n=1}^{N} E(n). \tag{4.10}$$

Yhtälön 4.10 virhefunktio on kaikkien vapaiden parametrien, kuten synapsien painojen ja kynnysarvojen, funktio. Neuroverkon opetusprosessin tarkoituksena on säätää nämä vapaat parametrit siten, että virhefunktion globaali minimi löytyy.

Neuronin i ja j välisen synaptisen painojen korjaus $\Delta w_{ji}(n)$ saadaan yhtälöstä

$$\Delta w_{ji}(n)=\eta\,\delta_j(n)\,y_i(n), \tag{4.11}$$

missä η on oppimiskerroin, $y_i(n)$ neuronille j tuleva syötesignaali ja $\delta_j(n)$ on lokaali gradientti neuronille j. Lokaalin gradientin laskeminen tuloskerroksen neuronille ja piilokerroksen neuronille poikkeavat toisistaan. Tuloskerroksen neuronille toivottu signaali tiedetään, minkä vuoksi sen lokaalingradientin laskeminen on suoraviivaista. Tulosneuronin j lokaali gradientti $\delta_j(n)$ voidaan laskea yhtälöstä [24]

$$\delta_j(n)=e_j(n)\frac{\partial\varphi(v_j(n))}{\partial v_j(n)}, \tag{4.12}$$

missä $(v_j(n))$ on neuronin j syötevektori iteraatiolle n vastaavasti, kuten kaavassa 4.7. Piilokerroksen neuronin lokaali gradientti tulee laskea rekursiivisesti, koska virhesignaalin määrittämisessä tulee ottaa huomioon piiloneuronin suorat yhteydet muihin neuroneihin. Piilokerroksen neuronin j lokaaligradientti $\delta_j(n)$ lasketaan relaatiosta [24]

$$\delta_j(n)=\frac{\partial\varphi(v_j(n))}{\partial v_j(n)}\sum_k \delta_k(n)w_{kj}(n), \tag{4.13}$$

missä lokaali gradientti $\delta_k(n)$ viittaa piiloneuronia j yhtä kerrosta ylempään neuroniin k sekä synapsin painokerroin $w_{kj}(n)$ neuronien j ja k välille. Lokaalien gradienttien kaavat 4.12 ja 4.13 sisältävät aktivaatiofunktion $\varphi(v_j(n))$ osittaisderivaattoja, jonka vuoksi takaisinlevitysalgoritmia käyttävän neuroverkon kaikkien aktivaatiofunktioiden tulee olla derivoituvia. [24]

Takaisinlevitysalgoritmi iteroi kunnes virhefunktio (kaava 4.10) saavuttaa asetetun minimiarvon tai epookkien määrä saavuttaa maksimiarvon. Jokaiselle epookille tulisi opetusdatan syötevektoreiden järjestys arpoa. Takaisinlevitysalgoritmin yksi ongelma on globaalin minimin sijasta lokaalin minimin löytyminen annetulle virhefunktiolle. [24]

4.5 Tilastollinen neuroverkko

Tilastollinen neuroverkko eli PNN-neuroverkko (probabilistic neural network, PNN) on neurolaskennan malli, joka perustuu tiheysfunktion (probability density function, PDF) approksimoimiseen Parzen ikkunoilla. Tilastollisen neuroverkon esitteli Specht vuonna 1990. PNN:ää käytetään pääosin luokitteluongelmien ratkaisemiseen. PNN:n luokittelu perustuu Bayesin päätösteoriaan. PNN:llä voidaan lisäksi estimoida ehdollisia todennäköisyyksiä. [27]

PNN:n päävahvuuksia ovat nopea oppiminen ja luonnostaan rinnakkainen rakenne sekä konvergoiminen optimaaliseksi luokittelijaksi opetusnäytteiden lukumäärän kasvaessa. Lisäksi opetusnäytteitä voidaan lisätä tai poistaa ilman kattavaa uudelleen opetusta. PNN on onnistunut monissa sovelluksissa vahvuuksiensa ansiosta. [28]

4.5.1 Parzen tiheysestimaatti

Olkoon B d-dimensioinen hyperkuutio, jonka sivun pituus on h. Hyperkuution tilavuus on näin ollen $V = h^d$. Määritellään ikkunafunktioksi [29]

$$\phi(\vec{u}) = \begin{cases} 1, & \text{kun } |u_j| \leq 1/2 \text{ ja } j = 1, \ldots, d \\ 0, & \text{muulloin.} \end{cases} \tag{4.14}$$

Kaavassa 4.14 määritellyn origoon asetetun hyperkuution sisällä sijaitsevat pisteet saavat arvon 1 ja muulloin arvon 0. Tällöin hyperkuutiolle B, jonka keskipiste on $\vec{x}$, saadaan ikkunafunktiosta $\phi((\vec{x}-\vec{x}_i)/h) = 1$, kun opetusnäyte $\vec{x}_i$ sijaitsee hyperkuutiossa. Muulloin ikkunafunktio saa arvon 0. Opetusnäytteiden lukumäärä hyperkuution sisällä on näin ollen

$$k = \sum_{i=1}^{N} \phi\left(\frac{(\vec{x}-\vec{x}_i)}{h}\right), \tag{4.15}$$

missä N on opetusnäytteiden lukumäärä. Tiheysestimaatti voidaan laskea nyt ikkunafunktioiden avulla kaavalla

$$p(\vec{x}) = \frac{k}{NV} = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{V}\phi\left(\frac{(\vec{x}-\vec{x}_i)}{h}\right). \tag{4.16}$$

Kaavalla 4.16 saadaan Parzenin tiheysestimaatti, kun funktioiksi ϕ (kaava 4.14) sallitaan muunkinlaiset ikkunafunktiot. Kirjallisuudessa ikkunafunktioita kutsutaan näissä muissa tilanteissa useimmin kernelifunktioiksi (kernel function), jonka vuoksi tästä eteenpäin käytetään tätä suositumpaa nimitystä funktioille ϕ. Kirjallisuudessa tyypillisimmin puhutaankin myös tämän vuoksi kernelitiheysestimaatista (kernel density estimation, KDE).

Tyypillisesti kernelifunktiot ovat symmetrisiä funktioita, joiden tiheysmaksimi on origossa. Tavallisin valinta kernelifunktioksi on normaalijakauman tiheysfunktio

$$\phi(\vec{x}) = \frac{1}{(2\pi)^{d/2}}\exp(-\vec{x}^T\vec{x}/2). \tag{4.17}$$

Normaalijakaumalle Parzen tiheysestimaatti saadaan muotoon [30]

$$p(\vec{x}) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{(2\pi h^2)^{d/2}}\exp\left\{\frac{\|\vec{x}-\vec{x}_i\|^2}{2\mathrm{h}^2}\right\}. \tag{4.18}$$

Yleisesti kernelifunktioille tulisi olla voimassa ehdot [30]

$$\phi(\vec{x}) \geq 0 \tag{4.19}$$

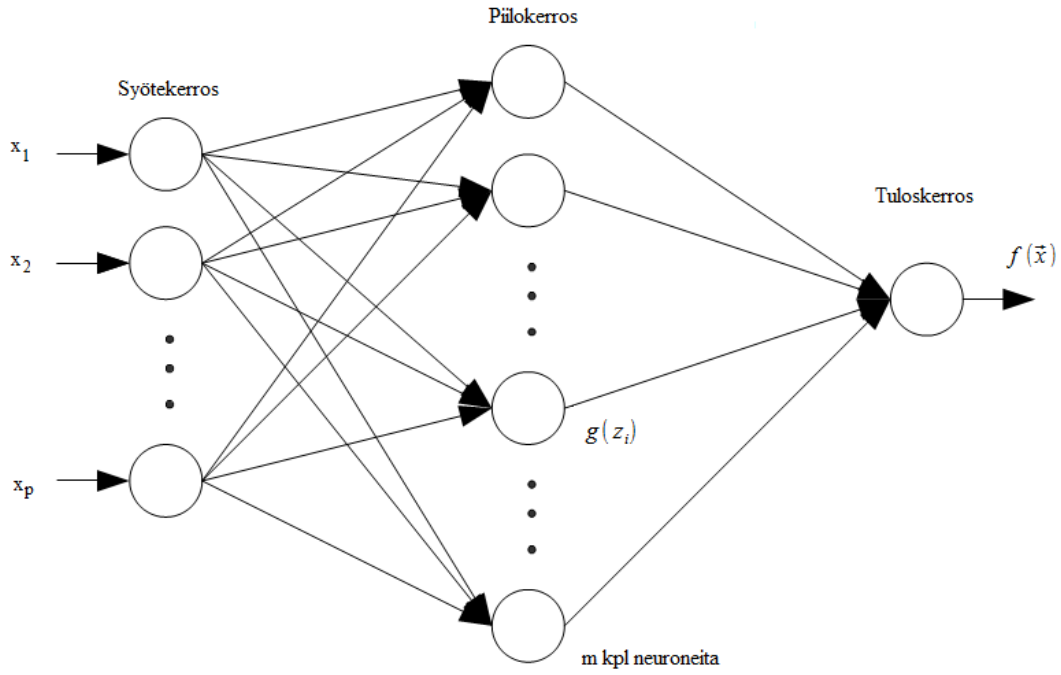
ja

$$\int \phi(\vec{x}) d\vec{x} = 1, \tag{4.20}$$

jolloin tiheysestimaatti kaavassa 4.16 toteuttaa tiheysfunktion perusehdot $p(\vec{x}) \geq 0$ ja $\int p(\vec{x}) d\vec{x} = 1$.

4.5.2 Tilastollisen neuroverkon rakenne

PNN on eteenpäin syöttävä monikerroksinen neuroverkko ja se tyypillisesti koostuu neljästä kerroksesta. Tämän tutkielman osalta PNN:ää rajoitutaan tarkastelemaan kolmikerroksisena versiona (kuva 4.4). PNN:n neljäs kerros valitsee syötevektorille todennäköisimmän luokan hyödyntäen Bayesin päätösteoriaa. Kolmikerroksisella PNN:llä voidaan estimoida opetusdatan tiheysfunktio ilman oletuksia datan jakaumasta.



Kuva 4.4: Tilastollinen neuroverkko.

Syötekerroksessa sijaitsevat neuronit tuottavat kukin samanlaisen syötteen kaikille piilokerroksen neuroneille. Piilokerroksen neuroni i laskee sisätulon [27]

$$z_i = \vec{x} \cdot \vec{w}_i, \tag{4.21}$$

missä $\vec{x}$ on syötevektori ja $\vec{w}_i$ on painovektori. Tämän jälkeen piilokerroksen neuroni laskee vasteen epälineaarisella funktiolla [27]

$$g(z_i) = \exp((z_i - 1)/\sigma^2), \tag{4.22}$$

missä σ on tasoitusparametri (smoothing parameter), joka määrää gaussisen funktion leveyden. Syötevektori $\vec{x}$ ja painovektorin $\vec{w}_i$ ollessa normitettuja on ekvivalenttia käyttää funktiota [27]

$$g(z_i) = \exp[(\vec{w}_i - \vec{x})^T(\vec{w}_i - \vec{x})/2\sigma^2]. \tag{4.23}$$

Tuloskerroksessa on jokaiselle luokalle neuroni, joka laskee summan $f(\vec{x})$ piilokerroksen neuronien vasteista. Tuloskerroksen neuroni laskee vasteen kaavalla [27, 31]

$$f(\vec{x}) = \frac{1}{N}\sum_{i=1}^{N} \exp[(\vec{w}_i - \vec{x})^T(\vec{w}_i - \vec{x})/2\sigma^2], \tag{4.24}$$

missä N on opetusvektoreiden lukumäärä. Kaava 4.24 on samaa muotoa kuin kaava 4.18. Kaavassa 4.18 esiintyvää kerrointa ei yleensä käytetä PNN:n yhteydessä, koska se supistuu pois neuroverkon luokittelupäätöstä laskettaessa.

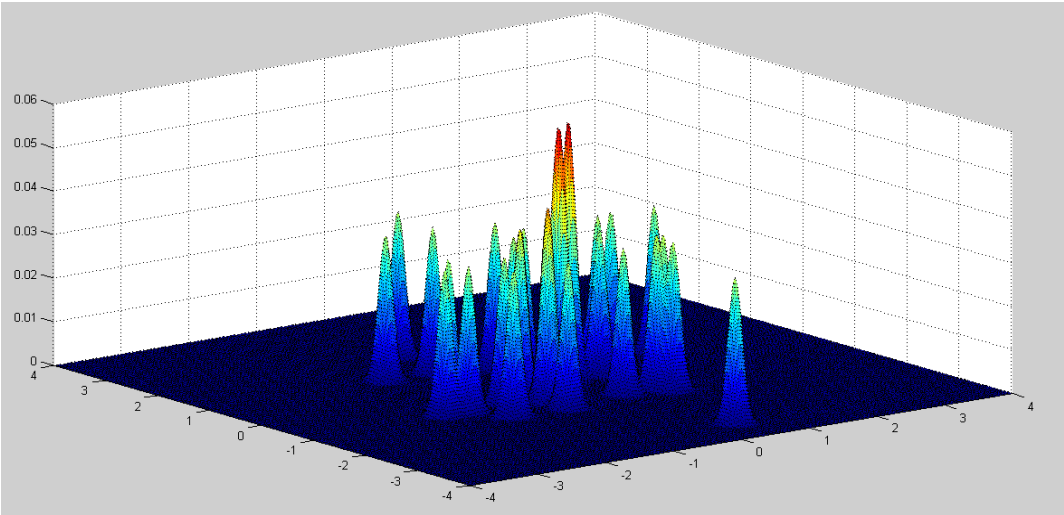
4.5.3 Tilastollisen neuroverkon opettaminen

Tilastollisen neuroverkon opettaminen on hyvin yksinkertainen ja nopea prosessi verrattuna muiden neuroverkkojen opettamiseen. Syötekerroksen ja piilokerroksen väliset painovektorit asetetaan vastaaman kukin yhtä opetusnäytettä. Tämän jälkeen piilokerroksen neuronit kytketään syöttämään vasteet tuloskerroksen neuronille. [27]

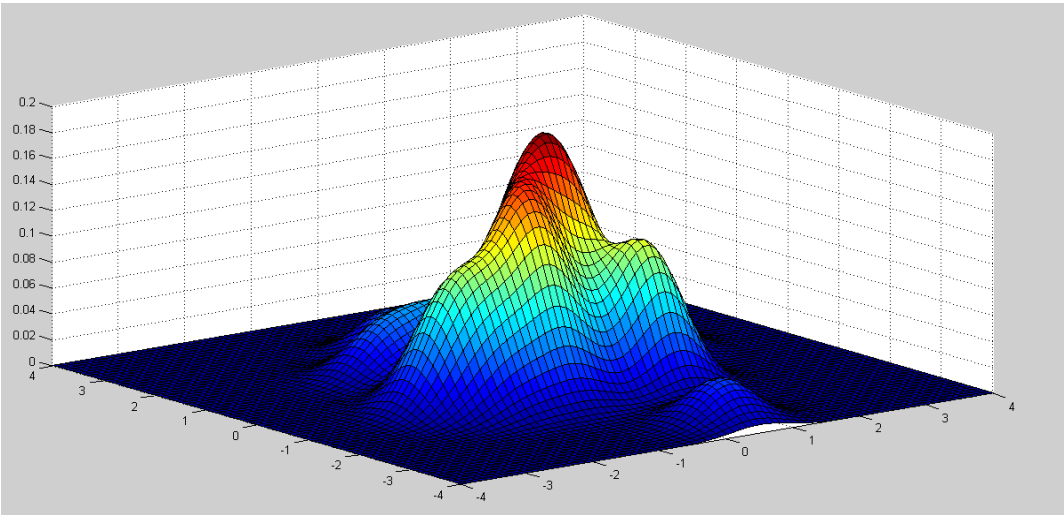
PNN:n käyttöön ottaminen vaatii oikeastaan vain sopivan tasoitusparametrin σ valinnan. Parametrin σ arvo haetaan yleensä kokeellisesti. Parametrille voidaan valita arvo, joka toteuttaa verkolle asetetun tehtävän parhaiten. Esimerkiksi luokitteluongelmassa parametrille valitaan arvo, joka antaa parhaan luokittelutarkkuuden ristiinvalidoinnilla. Verkon muut parametrit määräytyvät suoraan käytettävästä opetusdatasta. Syötekerroksen neuronien lukumäärä on sama kuin syötevektoreiden dimensio,

piilokerroksen neuronien lukumäärä on sama kuin opetusnäytteiden lukumäärä ja tuloskerroksen neuronien lukumäärä sama kuin opetusluokkien lukumäärä.[27]

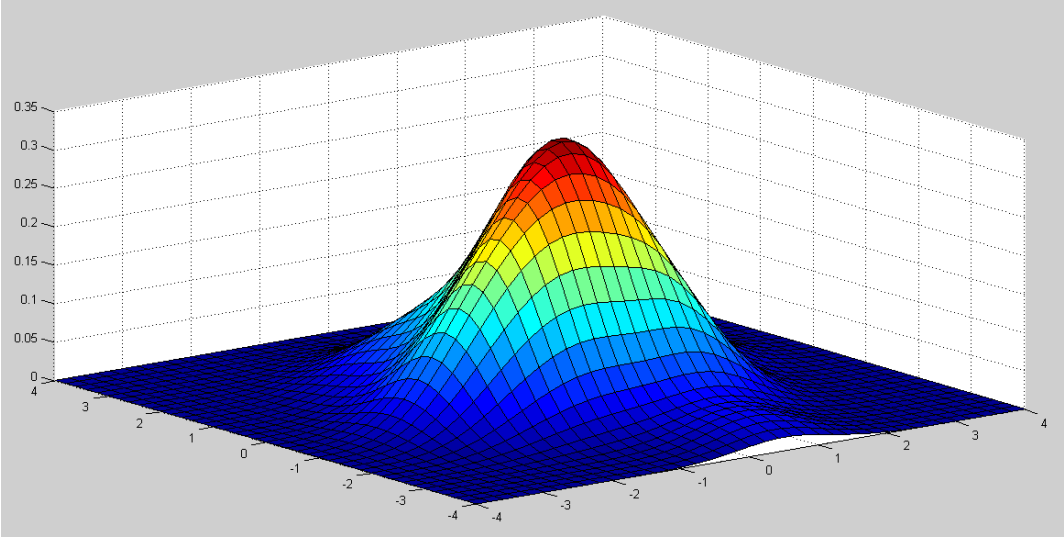
Kuvissa 4.5–4.7 on nähtävissä tasoitusparametrin vaikutus PNN-verkolla saatuun vastepintaan. Opetusdataksi generoitiin 2-ulotteisesta Gaussin jakaumasta 30 pistettä. Pienellä tasoitusparametrin saadaan hyvin piikikäs vastepinta, jossa lähes jokaista pistettä edustaa yksi Gaussin jakauman tiheysfunktio (kuva 4.5). Kasvattamalla tasoitusparametria riittävän suureksi saadaan vastepinta vastaamaan melko hyvin Gaussin jakaumaa, josta opetusdata generoitiin (kuva 4.7).



Kuva 4.5: PNN-neuroverkon vastepinta, kun $\sigma = 0{,}1$.



Kuva 4.6: PNN-neuroverkon vastepinta, kun $\sigma = 0{,}5$.



Kuva 4.7: PNN-neuroverkon vastepinta, kun $\sigma = 0{,}8$.

4.6 Itseorganisoituva kartta

Itseorganisoituva kartta eli SOM-neuroverkko (self-organizing map, SOM) on Teuvo Kohosen [32] kehittämä ohjaamattomaan oppimiseen perustuva neuroverkko-menetelmä, joka on tehokas menetelmä datan visualisoimiseen ja ominaisuuksien tutkimiseen. Kohosen menetelmää on sovellettu laaja-alaisesti datan klusterointiin ja tarkasteluun teollisuudessa, taloustieteissä, luonnontieteissä ja kielitieteissä [33].

Itseorganisoituva kartta kuvaa moniulotteisen jakauman alempiulotteiseen verkkoon, minkä vuoksi se pystyy esittämään moniulotteisten syötevektorien väliset monimutkaiset epälineaariset riippuvuussuhteet yksinkertaisena geometrisena esityksenä [33]. Verkko tiivistää datasta saatavan tiedon tulosneuroneihin säilyttäen datan tärkeimmät topologiset ja metriset relaatiot [32].

Tyypillisesti itseorganisoituva kartta koostuu kaksidimensioisesta tulosneuronikerroksesta, johon syötevektorit kuvataan. Itseorganisoituvan kartan tulosneuroneita nimitetään kirjallisuudessa myös solmuiksi (nodes). [32]

4.6.1 Kartan rakenne

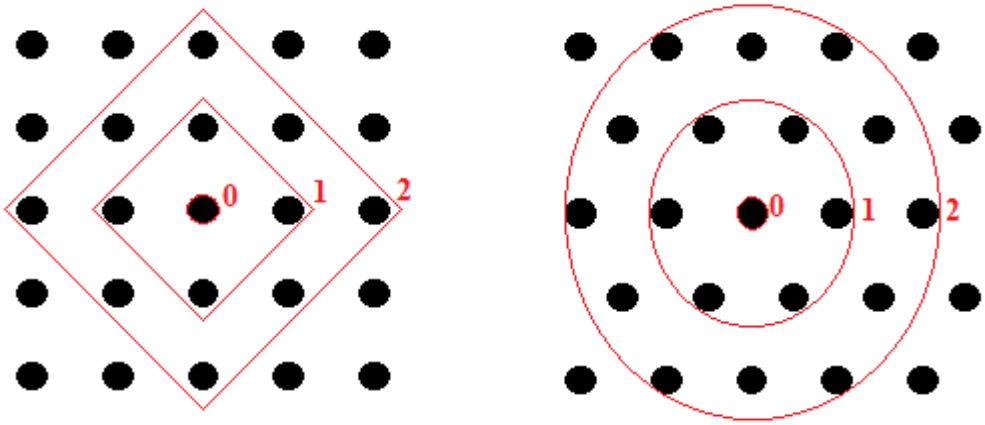
Itseorganisoituva kartta koostuu tyypillisesti neuronikerroksesta, jossa neuronit sijaitsevat 2-dimensioisessa säännöllisessä hilarakenteessa. Jokaiseen neuroniin i liittyy

yksi painovektori $\vec{w}_i = [w_{i1}, w_{i2}, ..., w_{in}]^T \in \Re^n$, missä n on syötevektorin dimensio. Indeksi i viittaa neuronin sijaintiin hilassa. [34]

Hilarakenteena käytetään tyypillisesti suorakulmaista tai heksagonaalista hilaa. Heksagonaalinen hilarakenne on suositellumpi, koska se ei suosi vertikaalisia ja horisontaalisia suuntia niin paljon kuin suorakulmainen hilarakenne. Hilarakenteen naapuruustyypistä riippuen jokaisella neuronilla on määritelty tietyt naapurineuronit. [34] Neuronin j naapurineuronien joukko N_j voidaan määritellä seuraavasti: [35]

$$N_j = \{ j \mid \|\vec{r}_i - \vec{r}_j\| \leq \sigma \}, \tag{4.25}$$

missä $\vec{r}_i, \vec{r}_j \in \Re^2$ ovat neuronien i ja j paikkavektorit hilassa ja σ on naapuruussäde. Kuvassa 4.8 [35] heksagonaalisen ja suorakulmaisen hilan naapuruusjoukot naapuruussäteille $\sigma = 0$, 1 ja 2.



a) Suorakulmainen hila **b)** Heksagonaalinen hila

Kuva 4.8: Itseorganisoituvan kartan naapurustojoukot suorakulmaiselle ja heksagonaaliselle hilalle. Erikokoiset naapuruusjoukot on rajattu punaisen viivan sisäpuolelle.

4.6.1 Inkrementaalinen algoritmi

Alkuperäisen SOM–algoritmin formulointi jäljittelee laskevan gradientin menetelmää. Se on laskennallisesti yksinkertainen ja kevyt algoritmi sekä se on myös käyttökelpoinen korkeadimensioisille syötevektoreille. [33] Inkrementaalisessa SOM-algoritmissa kartan painovektoreiden päivitys tapahtuu iteratiivisesti siten, että painovektoreita päivitetään kullekin syötevektorille kerrallaan.

Olkoon $\vec{x}(t)$ n-dimensioinen syötevektori iteraatiolla t. Tuloskerroksen solmun painovektori $\vec{w}_i(t)$ on myös n-dimensioinen vektori. Algoritmin painovektorien päivitys määritellään seuraavasti [32, 33]

$$\vec{w}_i(t+1) = \vec{w}_i(t) + h_{ci}(t)[\vec{x}(t) - \vec{w}_i(t)], \tag{4.26}$$

missä $h_{ci}(t)$ on naapuruusfunktio (neighborhood function) ja alaindeksi c viittaa parhaaseen vastinsolmuun (winner unit). Paras vastinsolmu määritellään siten, että painovektorille $\vec{w}_i(t)$ kaikilla i pätee ehto

$$\|\vec{x}(t) - \vec{w}_c(t)\| \leq \|\vec{x}(t) - \vec{w}_i(t)\|. \tag{4.27}$$

Tyypillisesti kaavan 4.27 paras vastinsolmu määritetään euklidisen etäisyyden avulla. Diskreetille datalle on mahdollista, että parhaita vastinsolmuja on useita. Paras vastinsolmu tulisi tällöin arpoa ehdon 4.27 toteuttavasti solmuista. Naapuruusfunktioksi valitaan usein gaussinen funktio

$$h_{ci}(t) = \alpha(t)\exp\left(- \frac{\|\vec{r}_i - \vec{r}_c\|}{2\sigma^2(t)} \right), \tag{4.28}$$

missä $0 < \alpha(t) < 1$ on monotonisesti vähenevä funktio, $\vec{r}_i \in \Re^2$ on solmun i paikkavektori verkossa, $\vec{r}_c \in \Re^2$ on parhaan vastinsolmun c paikkavektori verkossa ja $\sigma(t)$ on toinen monotonisesti vähenevä funktio. Parhaan vastinsolmun ja sen määräämän naapuruston painovektorit siirtyvät painojen päivityksessä 4.26 kohti käsiteltävää syötevektoria. Gaussista naapuruusfunktiota käytettäessä parhaasta vastinsolmusta kauempana verkkohilassa sijaitsevien solmujen painovektoreita siirretään syötevektoria kohti vähemmän kuin parhaan vastinsolmun lähempänä sijaitsevia naapureita. Toinen käytetty vaihtoehto naapuruusfunktioksi on kuplanaapuruusfunktio,

jolloin painovektoreiden päivitys tapahtuu siten, että parhaan vastinsolmun tietyn naapuruussäteen sisällä olevia painovektoreita siirretään kaikkia saman verran kohti syötevektoria.

Gaussisesta naapuruusfunktiosta on myös olemassa katkaistu versio, jossa gaussisen funktion hännät on leikattu pois. Tämä kyseinen katkaistu gaussinen naapuruusfunktio on muotoa [36]

$$h_{ci}(t) = \alpha(t) H(\sigma(t) - \|\vec{r}_i - \vec{r}_c\|) \exp\left(- \frac{\|\vec{r}_i - \vec{r}_c\|}{2\sigma^2(t)} \right), \tag{4.29}$$

missä $H(x)$ on askelfunktio

$$H(x) = \begin{cases} 1, & \text{kun } x \geq 0 \\ 0, & \text{kun } x < 0. \end{cases} \tag{4.30}$$

4.6.2 Eräalgoritmi

Eräalgoritmilla (batch map) [32] SOM-verkon painovektori päivitetään kerralla kaikille syötevektorille. Eräalgoritmi on nopeampi konvergoimaan kuin inkrementaalinen SOM-algoritmi, eikä sille tarvitse määrittää kaavassa 4.28 esiintyvää parametria $\sigma(t)$. Vesanto ym. [37] vertailivat erikokoisille syötedatajoukoille eräalgoritmin ja inkrementaalisen algoritmin laskentanopeutta. Eräalgoritmi osoittautui parhaimmillaan jopa 20 kertaa nopeammaksi kuin inkrementaalinen algoritmi.

Aluksi jokaisella syötevektorille $\vec{x}_j$ haetaan parhaat vastinsolmut c kaavalla 4.27. Seuraavaksi painovektoreiden päivitys lasketaan kaavalla

$$\vec{w}_i = \frac{\sum_{j=1}^{N} h_{ci} \vec{x}_j}{\sum_{j=1}^{N} h_{ci}}, \tag{4.31}$$

missä N on syötevektoreiden lukumäärä. Lasketut painovektorit ovat syötevektoreiden painotettuja keskiarvoja. [35]

4.6.3 Painovektoreiden alustus

Ennen SOM-neuroverkon opettamista tulee painovektorit alustaa sopivilla arvoilla. Painovektoreiden hyvällä alustuksella voidaan opetuksen laskenta toteuttaa useita kertaluokkia nopeammin. [32]

Alun perin SOM-neuroverkon vahvaa oppimiskykyä havainnollistettiin alustamalla painovektorit satunnaisesti. Tutkimuksissa on osoitettu, että SOM-neuroverkon satunnaiset painovektorit pystyvät järjestäytymään riittävälle määrälle opetusiteraatioita. Tätä ominaisuutta ei kuitenkaan tarvitse havainnollistaa jokaisessa tutkimuksessa. Satunnainen painovektoreiden alustus on tehoton alustusmenetelmä verrattuna muihin alustusmenetelmiin. [32, 34]

Tehokkaaksi alustusmenetelmäksi on osoittautunut lineaarinen alustus (linear initialization), jossa painovektorit alustetaan syötevektoreiden autokorrelaatiomatriisista laskettavien kahden suurimman ominaisarvon omaavien ominaisvektorien virittämän tason pisteistä, joiden keskipisteenä on syötevektoreiden keskiarvo. Alustuksessa painovektoreiden välit saadaan approksimatiivisesti tasavälisiksi valitsemalla itseorganisoituvan kartan 2-ulotteisen hilan horisontaalisesti ja vertikaalisesti olevien neuroneiden lukumäärät samassa suhteessa kuin suurimmat ominaisarvot. [34]

4.6.4 Oppimisvaiheet

Itseorganisoituvan kartan oppiminen voidaan jakaa järjestäytymisvaiheeseen ja konvergointivaiheeseen. Järjestäytymisvaiheessa painovektorit asettuvat oikeaan järjestykseen. Konvergointivaiheessa painovektorit hienosäädetään suurella määrällä iteraatioita riittävän tilastollisen tarkkuuden saavuttamiseksi. [34]

Eräalgoritmilla ja inkrementaalisella algoritmilla on edellä mainitut oppimisvaiheet, mutta jatkossa keskitytään inkrementaalisen algoritmin oppimisvaiheisiin ja niiden parametrien valintaan. Tässä tutkimuksessa itseorganisoituvan kartan opetus toteutetaan inkrementaalisella algoritmilla, koska eräalgoritmi ei anna merkittävää ajallista hyötyä, kun syötevektoreiden dimensio ja syötevektoreiden lukumäärä on pieni [37]. Lisäksi inkrementaalisen algoritmin opetusparametrien valintaan löytyy

kirjallisuudesta hyvä teoreettinen pohja toisin kuin eräalgoritmille. Opetusdatan syötevektorien dimension ja lukumäärän kasvaessa suureksi on eräalgoritmin käyttöä syytä kuitenkin harkita.

Järjestäytymisvaiheessa opetus tehdään pienellä määrällä iteraatioita, tyypillisesti noin 1 000 iteraatiolla. Naapuruusfunktiolla on keskeisin osuus kartan järjestäytymisessä. Naapuruusfunktion leveys $\sigma(t)$ asetetaan järjestäytymisvaiheen alussa suureksi, jonka jälkeen se laskee lineaarisesti opetusiteraatioiden t funktiona pieneksi. Naapuruusfunktion leveyden on oltava järjestäytymisvaiheen alussa riittävän suuri, muuten kartta ei järjestäydy globaalisti. Naapuruusfunktion leveydeksi suositellaan asetettavaksi noin puolet kartan suurimmasta dimensiosta. Oppimiskertoimen funktion $\alpha(t)$ suositellaan vaiheen alussa olevan 0,9 ja laskevaan opetuksen aikana monotonisesti arvoon 0. Oppimiskertoimen funktion tarkempi valinta ei ole tärkeää. Funktio voi olla esimerkiksi lineaarinen. [34, 33]

Hienosäätövaiheessa opetusiteraatioita tehdään huomattavasti enemmän kuin järjestäytymisvaiheessa. Iteraatioiden lukumäärän hyvänä sääntönä riittävällä tilastolliselle tarkkuudelle voidaan pitää sitä, että iteraatioiden lukumäärä on vähintään 500-kertainen verkon neuronien lukumäärään nähden. Naapuruusfunktion leveys ja oppimiskerroin asetetaan pieniksi. Oppimiskertoimen suositellaan vähenevän arvosta 0,02 arvoon 0 vaiheen aikana. [34]

4.6.5 Laadukkuuden mittaaminen

Opetetun SOM-neuroverkon laatuun vaikuttavat kartan rakenteeseen liittyvät parametrit ja oppimisalgoritmin parametrit. Jotta SOM-algoritmin toimintaa voidaan ymmärtää, tulee SOM-verkoille olla jokin mittari, jolla voidaan arvioida minkälainen SOM-verkko on hyvä [35]. Tyypillisesti SOM-verkkojen laadukkuutta mitataan kahdella tapaa: erottelukyvyllä (resolution) ja topologian säilymisellä (topology preservation) [38].

Erottelukykyä voidaan mitata keskimääräisellä kvantisointivirheellä [38]

$$q_e = \frac{1}{N}\sum_{i=1}^{N} \|\vec{x}_i - \vec{w}_c\|. \tag{4.32}$$

Keskimääräinen kvantisointivirhe q_e on syötevektoreiden keskimääräinen etäisyys parhaisiin vastinsolmuihin c. Topologian säilymistä voidaan mitata topografisella virheellä [38]

$$t_e = \frac{1}{N}\sum_{i=1}^{N} u(\vec{x}_i), \tag{4.33}$$

missä $u(\vec{x}_i) = 1$, jos syötevektorin $\vec{x}_i$ paras ja toiseksi paras vastinsolmu ovat rinnakkaisia solmuja verkkohilassa. Muulloin $u(\vec{x}_i) = 0$. Kaava 4.33 ei ota huomioon diagonaalisia naapureita, minkä vuoksi SOM:lla saadaan heksagonaaliselle hilalle aina pienempi topografinen virhe kuin suorakulmaiselle hilalle. [38]

4.7 Poikkeavuuksien havaitseminen neuroverkoilla

Neuroverkkoja on käytetty laaja-alaisesti poikkeavuuksien havaitsemiseen. Neuroverkkojen vahvuuksia poikkeavuuksien löytämisessä on se, että optimoitavien parametrien määrä on pieni ja datan jakaumasta ei tarvitse tehdä etukäteen mitään oletuksia. [18] Neuroverkoilla on myös hyvä datan yleistämiskyky ei-nähdyille tapauksille, ja niillä on kyky oppia monimutkaisia luokkarajoja [18, 39, 14]. Monet neuroverkot ovat herkkiä sietämään dimension kasvamista (curse of dimensionality), mutta ne ovat kuitenkin parempia sietämään sitä kuin tilastotieteelliset menetelmät. [14]

Vertailevaa tutkimusta eri neuroverkkoarkkitehtuurien tehokkuudesta poikkeavuuksien havaitsemiseen samoille datajoukoille on tehty vähän [18]. Neuroverkkojen käyttöönotto poikkeavuuksien havaitsemiseen vaatii sekä opetuksen että testauksen. Testauksessa neuroverkkoa hienosäädetään ja määritetään kynnysarvot poikkeavuuksien erottelemiseksi. [14]

4.7.1 Poikkeavuuksien havaitseminen MLP-neuroverkoilla

Poikkeavuuksien havaitseminen MLP-neuroverkoilla perustuu ohjattuun luokitteluun. MLP-neuroverkkoilla poikkeavuuksien havaitseminen on haastavaa, koska ne eivät muodosta suljettuja luokkarajoja. [18] Kyseiset neuroverkot interpolaroivat

dataa hyvin mutta ekstrapolaroivat sitä huonosti, minkä vuoksi ne eivät luokittele hyvin opetusdatan ulkopuolisia tapauksia. [14]

Bishop [40] soveltaa MLP-neuroverkkoa poikkeuksien havaitsemiseen putkistovirtauksissa. Kyseinen neuroverkko on opetettu tunnistamaan öljyn osuus putkistovirtauksessa 12 parametrin avulla. Opetusdata koostuu 1 000 mittapisteestä. Bishop käyttää neuroverkossa yhtä piilokerrosta 8 neuronilla, jotka antavat vasteita 2 lineaariselle tuloskerroksen neuronille. Toinen tuloskerroksen neuroneista kuvaa veden ja toinen öljyn osuutta virtauksesta. Piilokerroksen neuroneille käytetään logistista aktivaatiofunktiota. Poikkeavuudet tunnistettiin neuroverkon tuloksista asettamalla sopiva kynnysarvo tulossignaalin todennäköisyystiheydelle. Kynnysarvon määritystä varten tutkimuksessa määritetään opetusdatan tuloksille ehdollinen todennäköisyystiheysfunktio. Funktion approksimointiin käytettiin Parzenin ikkunoita, jotka hyödynsivät gaussisia kernelifunktioita. Poikkeavuuksien havaitseminen testattiin keinotekoisten poikkeavuuksien avulla. Sopivan kynnysarvon avulla kaikki merkittävät keinotekoiset poikkeamat pystyttiin erottelemaan neuroverkon tuloksista.

4.7.2 Poikkeavuuksien havaitseminen tilastollisella neuroverkolla

Tilastollisella neuroverkolla on paremmat edellytykset poikkeavuuksien havaitsemiseen kuin MLP-neuroverkoilla [41], koska se pystyy muodostamaan suljettuja luokkarajoja. Useimmat tilastollisella neuroverkolla tehdyt poikkeavuuksien havaitsemismallit ovat muunnelmia sen alkuperäisestä versiosta.

Shaffer ym. [42] käyttävät paranneltua tilastollista neuroverkkoa (improved probabilistic neural network) vaarallisten kaasujen reaaliaikaiseen tunnistamiseen kolmen kemiallisen anturin avulla. Shafferilla on patentti [43] kyseisestä sovelluksesta, jonka sisältö on pitkälti sama kuin kappaleen alussa viitatussa tutkimuksessa. Paranneltu PNN eroaa alkuperäisestä versiosta lähinnä vain opetuksessa käytettävien syötevektoreiden osalta. Syötevektoreina käytetään oppivan vektorikvantisoinnin (learning vector quantization, LVQ) referenssivektoreita, jotka kuvaavat datavektoreiden virittämän avaruuden pienemmällä vektoreiden lukumäärällä.

Ohjattuun oppimiseen perustuva LVQ liittyy hyvin läheisesti ohjaamattomattomaan oppimiseen perustuvien oppiva vektorikvantisointi- ja SOM-algoritmeihin. LVQ:n

opetusprosessissa referenssivektoreille ei ole määritelty naapureita, toisin kuin SOM:n referenssivektoreille (painovektoreille). [34]

PNN-sovelluksella voidaan kaasujen tunnistamisen lisäksi pystyä automaattisesti havaitsemaan poikkeavuudet, mikä vähentää väärien tunnistusten määrään. Shaffer pitää poikkeavuuksien tunnistamista oleellisena osana sovelluksen toimintaa. Shafferin käyttämä tilastollinen neuroverkko opetetaan tunnistamaan kaasut ja hylkäämään poikkeavat verkon syötteet kolmiulotteisten opetusvektorien avulla. [43]

Shaffer hyödyntää LVQ:ta PNN:n piilokerroksessa vähentämään kerroksen neuronien lukumäärää. PNN:n piilokerroksen neuronien lukumäärä on yhtä suuri kuin verkon opetuksessa käytettyjen syötevektoreiden lukumäärä, minkä vuoksi tilastollisen neuroverkon piilokerros on usein suurempi kuin muiden neuroverkkojen piilokerrokset. Oppivalla vektorikvantisoinnilla saadaan datan rakenne esitettyä pienemmällä määrällä piilokerroksen neuroneita tilastollisessa neuroverkossa kuin tavanomaisen tilastollisen neuroverkkon tapauksessa. [43]

Shafferin käyttämän tilastollisen neuroverkon optimaalinen tasoitusparametri σ_{opt} määritetään yhtälöstä

$$\sigma_{opt} = K\sqrt{\frac{1}{DN}\sum_{j=1}^{D}\|\epsilon_j^* - \epsilon\|^2}, \tag{4.34}$$

missä ϵ_j^* on syötevektorin ϵ lähin naapuri, D on syötevektorin dimensio, K korjauskerroin ja N on syötevektoreiden lukumäärä. Sovelluksen optimaaliseksi korjauskertoimeksi saatiin 1,44. [43]

PNN:n poikkeavuuden havaitseminen voidaan tehdä todennäköisyyksien tai PNN:n kolmannen kerroksen vasteiden summan perusteella. Shaffer ym. käyttävät edellä mainittua lähestymistapaa. Syötevektori luokitellaan poikkeavuudeksi, mikäli neuroverkon vasteiden summa on pienempi kuin asetettu kynnysarvo. Kynnysarvo poikkeavuuksien hylkäämiseen määritettiin Monte Carlo -simuloinnin avulla. Monte Carlo -simuloinnilla generoidaan joukko satunnaisia syötevektoreita, jonka jälkeen neuroverkko asetetaan hylkäämään tietty osuus satunnaisista syötevektoreista. [42]

Toisin kuin edellä esitellyssä tutkimuksessa, Oliveiran ym. [41] poikkeavuuksien havaitseminen perustuu PNN:n antamiin ehdollisiin todennäköisyyksiin. Tutkimuksessa käytetään poikkeavuuksien havaitsemiseen tilastollista neuroverkkoa, jonka opetuksessa käytetään PNN-DDA-algoritmia (dynamic decay adjustment algorithm). Kyseistä opetusalgoritmia käytetään optimoimaan PNN-neuroverkon koko mahdollisimman pieneksi. PNN-neuroverkon toiseen kerrokseen lisätään neuroneita opetuksen syötevektoreille valikoidusti. Poikkeavuudet havaitaan laskemalla ehdollinen todennäköisyys syötevektorille ehdolla, että se kuuluu poikkeavuusluokkaan. Tämä todennäköisyys määritellään siten, että se on suurempi kuin ehdollinen todennäköisyys syötevektorin kuulumiselle johonkin muuhun luokkaan. Neuroverkon opetuksessa käytetään ainoastaan normaalia dataa. Menetelmää testattiin neljään testidatajoukkoon. Kunkin testidatajoukon opetusluokista yksi valittiin poikkeavaksi luokaksi, jota ei käytetty tilastollisen neuroverkon opetuksessa. Menetelmää verrattiin NNDD-menetelmään (nearest neighbor data description) eri kynnysarvoilla poikkeavuuksien havaitsemisessa. Tilastollinen neuroverkko ja DDA-algoritmi osoittautui kolmessa testidatajoukossa paremmaksi kuin NNDD-menetelmä.

Dominguez ym. [44] käyttävät visuaaliseen poikkeavuuden havaitsemiseen sumeaa logiikkaa, tilastollista neuroverkkoa ja geneettisiä algoritmeja. Sumeaa logiikkaa käytetään opetusdatan esikäsittelyvaiheessa ja potentiaalisten poikkeavuusalueiden havaitsemisessa. Varsinaiset poikkeavuudet tunnistetaan lopuksi tilastollisen neuroverkon avulla. Geneettisiä algoritmeja käytetään opetusvaiheessa tilastollisen neuroverkon tasoitusparametrin optimoimiseen.

Bastke ym. [31] soveltavat tilastollista neuroverkkoa poikkeavan tietoliikenteen havaitsemiseen tarkkailemalla tietokoneen graafisten laskentayksiköiden käyttöä. Tilastollista neuroverkkoa käytetään approksimoimaan opetusdatan tiheysfunktio, jonka perusteella alle kynnysarvon jäävät tiheydet luokitellaan poikkeavuuksiksi. Kynnysarvo neuroverkolle määritetään automaattisesti käyttämällä laatikkokuvauksen sääntöä poikkeavuuksien havaitsemiseen. Menetelmällä onnistuttiin havaitsemaan hyökkäyksiä, jotka ovat yleisiä internetissä.

4.7.3 Poikkeavuuksien havaitseminen itseorganisoituvalla kartalla

Itseorganisoituvaa karttaa on käytetty useissa tutkimuksissa poikkeavuuksien havaitsemiseen. Tunkeutujan havaitsemisjärjestelmiä (intrusion detection system, IDS) koskevissa tutkimuksissa on erityisesti käytetty itseorganisoituvaa karttaa poikkeavuuksien havaitsemiseen [45, 46, 47, 48]. Lisäksi itseorganisoituvaa karttaa on sovellettu muun muassa tähtitieteessä [49]. Itseorganisoituvan kartan etuja muihin neuroverkkoihin verrattuna on poikkeavuuksien havaitsemisessa se, ettei se tarvitse datasta etukäteen tietoa toisin kuin ohjattuun oppimiseen perustuvat neuroverkot. Lisäksi itseorganisoituvalla kartalla voidaan opetusdatasta saada uutta tietoa kuten esimerkiksi syötevektoreiden muodostaman avaruuden tiheyden vaihtelut. Itseorganisoituvalla kartalla tyypillisin tapa havaita poikkeavuuksia on laskea aluksi syötevektorin euklidinen etäisyys parhaan vastinsolmun painovektoriin, jonka jälkeen syötevektori tunnistetaan poikkeavaksi, mikäli etäisyys on pienempi kuin kynnysarvo [45, 46, 47, 50, 51].

Munõz ja Muruzábal [50] käyttävät itseorganisoituvaa karttaa poikkeavuuksien havaitsemiseen kahdella eri menetelmällä. Muista neuroneista kaukana olevat neuronit havaitaan poikkeavaksi projisoimalla painovektorit Sammonin kuvauksella (Sammon's mapping) ja tarkastelemalla MID-matriisia (median interneuron distance matrix). MID-matriisi antaa tietoa neuronien etäisyyksistä naapureihin. Syötevektorit, joiden paras vastinsolmu on poikkeava, havaitaan poikkeavuuksiksi eli poikkeavan neuronin edustaman klusterin kaikki syötevektorit havaitaan poikkeaviksi. Toinen menetelmä on euklidiseen etäisyyteen perustuva (ks. ed. kappale). Kyseisellä menetelmällä laskettua mittaa kutsutaan kvantisointivirheeksi. Kynnysarvo poikkeavuudelle määritellään laatikkokuvauksella, jonka avulla saadaan tietoa kvantisointivirheen jakaumasta. Menetelmiä testataan tutkimuksessa useisiin erityyppisiin datajoukkoihin. Menetelmät täydentävät hyvin toisiaan, ja niillä pystytään ainakin havaitsemaan tyypin I ja III poikkeavuuksia.

Höglund ym. [45] käyttävät itseorganisoituvaa karttaa poikkeavuuksien havaitsemiseen UNIX-käyttöjärjestelmässä. Verkolle opetetaan järjestelmän normaali profiili yli 900 käyttäjältä kerätystä lokidatasta 1 100 päivän ajanjaksolta. Neuroneiden lukumäärä valitaan huomattavasti pienemmäksi kuin syötevektoreiden lukumäärä.

Esimerkkivalintana mainitaan neuroneiden määräksi 10 % syötevektoreiden lukumäärästä. Poikkeavuudet havaitaan laskemalla kvantisointivirheet, kuten edellä mainitussa tutkimuksessa. Kynnysarvoksi kvantisointivirheelle asetetaan yksinkertaistettuna syötevektorin kvantisointivirhe, joka jakaa syötevektoreiden kvantisointivirheiden lukuarvot 5 %:n ja 95 %:n osuuksiin. Höglundin ym. mukaan menetelmällä pystytään havaitsemaan yksittäisten ja useampien muuttujien poikkeavuuksia ilman oletuksia jakaumasta. Menetelmällä voidaan myös havaita eniten poikkeavuuteen vaikuttava muuttuja.

Cao ym. [48] yhdistävät kernelin tiheysestimoinnin ja itseorganisoituvan kartan poikkeavuuksien havaitsemiseen tietoliikenteestä. Kernelin tiheysestimaatti muodostetaan käyttämällä SOM:n painovektoreita, koska SOM:n painovektorit pystyvät approksimoimaan tiheysavaruutta. Tiheysfunktion avulla muuttujille asetetaan todennäköisyyksiin perustuen ala- ja ylärajat poikkeavuuksien havaitsemiseksi. Menetelmällä pystytään havaitsemaan yksittäisten muuttujien poikkeamia ilman oletuksia jakaumasta. Tiheysfunktio muodostetaan lähes samalla menetelmällä kuin Shaffer ym. [42] tekivät PNN:llä ja LVQ:n referenssivektoreilla.

Ramadas ym. [46] käyttävät itseorganisoituvaa karttaa poikkeavan tietoliikenneyhteyden tunnistamiseen. Myös tässä tutkimuksessa käytetään aikaisemmin esiintynyttä kvantisointivirhettä poikkeavuuksien havaitsemisessa. Kynnysarvo määritetään syötevektoreiden keskihajonnan perusteella. Syötevektori tunnistetaan poikkeavaksi, jos se sijaitsee 6-ulotteisen hyperpallon, jonka keskipisteenä on parhaan vastinneuronin painovektori ja säteenä kaksi kertaa keskihajonta, ulkopuolella. Klusterien oletetaan noudattavan Gaussin jakaumaa, minkä vuoksi kynnysarvoksi valitaan kaksi kertaa keskihajonta. Menetelmää testattiin muutamalla tunnetulla hyökkäyksellä, jotka se onnistui tunnistamaan poikkeavuuksiksi.

González ym. [51] generoivat keinotekoisia poikkeavuuksia RNS-algoritmilla (real-valued negative selection algorithm) käyttäen normaalia dataa. Poikkeavuuksien havaitsemiseen käytetään MLP-neuroverkkoa, joka opetetaan tunnistamaan normaaliluokka ja RNS-algoritmilla generoitu poikkeavuusluokka. González ym. tutkivat myös samassa tutkimuksessa SOM-neuroverkon kykyä poikkeavuuksien havaitsemiseen. Poikkeavuudet määritettiin kvantisointivirheillä, mutta etäisyysmetriikoina

kokeiltiin kolmea eri vaihtoehtoa. Poikkeavuuksien havaitsemiskykyä vertailtiin tarkastelemalla ROC-käyriä euklidisella, normalisoidulla ja Minkowskyn etäisyysmetriikalla. Parhaaksi etäisyysmetriikaksi osoittautui Minkowskyn etäisyysmetriikka, jonka poikkeavuuksien havaitsemiskykyä tutkittiin tarkemmin eri kokoisille SOM-neuroverkoille. Neuronien lukumäärä paransi poikkeavuuksien havaitsemiskykyä, mutta ROC-käyrien ero ei ollut suuri. RNS-algoritmin ja MLP-neuroverkon hybridimenetelmää ja SOM-neuroverkkoa vertailtiin eri testidatajoukoille. Menetelmien välillä ei havaittu tilastollisesti merkitsevää eroa.

5 KOKEELLINEN OSUUS

5.1 Lähtökohdat poikkeavuuksien havaitsemismallin muodostamiseen sädehoidon annossuunnitelmista

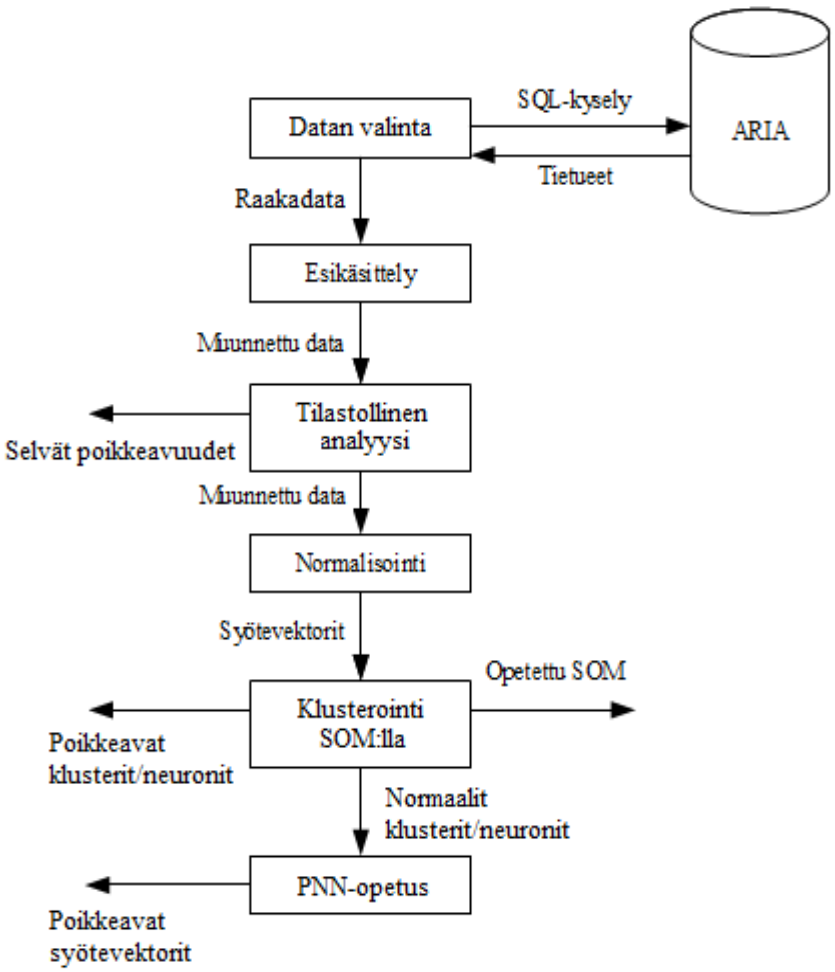
Tutkimuksen tavoitteena on muodostaa poikkeavuuksien havaitsemismalli sädehoidon annossuunnitelmista käyttäen SOM- ja PNN-neuroverkkoja. Mallin muodostamiseen valittiin kolmella hoitokentällä toteutetut rinnanpoiston jälkeisten sädehoitojen annossuunnitelmat. Mallilla pyritään havaitsemaan poikkeavuuksia kenttien monitoriyksiköistä ja elektronikenttien säteilyenergioista. Poikkeavuuksien havaitsemismallin muodostamiseen käytetään OYS:n sädehoidon yksikön Aria 8.9 -potilastietokannasta saatavaa dataa aiemmin toteutetuista sädehoidon annossuunnitelmista. Tietokannassa on tallennettuna kaikki tarvittava tieto sädehoidon toteuttamiseen, kuten hoitokoneet, kuvaukset, annossuunnitelmat, aikataulutus ja laskutus [52].

Kuvaan 5.1 on jäsennetty mallinmuodostusprosessi Aria-tietokannasta. Aluksi data valitaan tietokannasta SQL-kyselyllä hakemalla tietueita tietyntyyppisistä annossuunnitelmista. Saatuja tietueita rajataan vielä tämän jälkeen Matlab-ohjelmiston tekstinkäsittelyfunktioilla. Esikäsittelyvaiheessa data muokataan neuroverkkoja varten numeeriseen muotoon. Seuraavaksi data analysoidaan tilastollisesti ja visuaalisesti, minkä jälkeen datasta poistetaan selvät poikkeavuudet. Ennen klusterointia data normalisoidaan 0-keskiarvoiseksi ja 1-keskihajontaiseksi. Normalisoitu data opetetaan SOM-neuroverkolle sekä poikkeavat klusterit ja neuronit merkitään.

Poikkeavuuksien havaitsemismalli muodostetaan kahdella tavalla. Fotonikenttien osalta jokaisen normaalin klusterin syötevektoreille opetetaan PNN-neuroverkko käyttämällä klusterin syötevektoreista laskettavaa tasoituskerrointa (kaava 4.34). Tällä tavoin voidaan hyödyntää SOM:n tallentamaa lokaalia tietoa ottamalla huomioon tiheyden vaihtelut datassa. Bishopin [30] mukaan jakamalla ongelma osajoukkoihin, joista kukin ratkaistaan siihen opetetulla neuroverkolla, voidaan saavuttaa merkittäviä hyötyjä verrattuna yksittäisen neuroverkon käyttöön. Kaikkien fotonikenttien PNN-verkot opetetaan samalla korjauskertoimella K, jolloin ratkaistavaksi

ongelmaksi saadaan optimaalisen korjauskertoimen hakeminen PNN-verkkojen joukolle. Korjauskertoimen optimaalinen arvo haetaan kokeellisesti ristiinvalidoinnilla.

Elektronikenttien osalta poikkeavuuksien havaitsemismalli muodostetaan pelkästään yhdestä PNN-neuroverkosta siten, että se opetetaan käyttämällä ainoastaan normaaleiden neuroneiden painovektoreita. Erisuuruisten tasoituskertoimien käytölle ei nähty hyötyä diskreetissä tapauksessa. Lisäksi kaava 4.34 ei ole käyttökelpoinen diskreetissä tapauksessa, koska tasoituskertoimen arvoksi voi tulla 0.



Kuva 5.1: Poikkeavuuksien havaitsemismallin muodostusprosessi sädehoidon annossuunnitelmille ARIA-potilastietokannasta.

5.2 Poikkeavuuksien havaitseminen annossuunnitelmista

Fotonikenttien osalta syötevektori merkitään poikkeavuudeksi, mikäli PNN-neuroverkkojen antamien normitettujen vasteiden summa on pienempi kuin asetettu kynnysarvo t_{fot}. PNN-neuroverkon k antama vaste normitetaan suhteelliseksi tiheydeksi [53]

$$f_{suht,k}(\vec{x}) = \frac{f(\vec{x})}{\sum_{\vec{y} \in C} f(\vec{y}) / N}, \tag{5.1}$$

missä C on verkon opetuksessa käytettyjen syötevektoreiden joukko ja N joukon syötevektoreiden lukumäärä. PNN-neuroverkon antama normitettu vaste on syötevektorin vaste jaettuna opetuksessa käytettyjen syötevektoreiden vasteiden keskiarvolla. Syötevektori luokitellaan poikkeavuudeksi ehdolla

$$t_{fot} \geq \sum_{k \in D} f_{suht,k}(\vec{x}), \tag{5.2}$$

missä D on opetettujen PNN-neuroverkkojen joukko. Käyttämällä PNN-neuroverkkojen vasteiden summaa poikkeavuuksien havaitsemisessa saadaan vältettyä tilanteet, joissa poikkeavuudeksi havaitaan lähekkäisten klustereiden välissä sijaitsevat pisteet.

Elektronikenttien poikkeavuudet havaitaan yhdellä PNN-neuroverkolla. PNN opetetaan SOM:n normaaleiden klustereiden painovektoreilla. Syötevektori luokitellaan poikkeavaksi ehdolla

$$t_{ele} \geq f(\vec{x}). \tag{5.3}$$

5.3 Käytetyt ohjelmistot

Annossuunnitelma-aineisto haettiin Sybasen tietokantahakuihin tarkoitetulla Infomaker-ohjelmistolla. Neuroverkkoihin liittyvä laskenta toteutettiin käyttämällä Matlab R2012b -ohjelmiston Neural Network Toolbox, Statistics Toolbox ja SOM Toolbox 2.0 -lisäosien funktioita. Tietokantayhteyden muodostamisessa käytettiin Matlab-ohjelmiston Database Toolbox -lisäosaa. SOM Toolbox -lisäosaa käytettiin

SOM:n opetuksessa, koska se mahdollisti helpommin kaikkien opetusparametrien hallinnan verrattuna Matlabin Neural Network Toolbox -lisäosan funktioihin.

5.4 Potilasdatan valinta

Aria-tietokanta on suuri ja kompleksinen relationaalinen tietokanta, joka rakentuu yli 150 taulusta [52]. Liitteessä A [54] on Aria 8.9 -tietokannan tauluja, joita käytetään usein tietokantahauissa. Taulujen tarkat tiedot ja niiden väliset relaatiot ovat Varianin internet-sivuilla [52, 54]. Tietokantahaku voidaan tehdä SQL-kielellä (structured query language), joka on käytetty ohjelmointikieli relationaalisten tietokantojen hallintajärjestelmissä. SQL-komennoilla voidaan hakea dataa tietokannasta ja luoda uusia tauluja sekä lisätä uutta dataa tai muokata olemassa olevaa dataa. Tietokanta-haku tehtiin reaaliaikaiseen tietokantaan kytketyllä Arian tukemalla InfoMaker-ohjelmistolla, jolla voidaan ainoastaan hakea dataa tietokannasta. InfoMakerilla SQL-kyselyn koodi voidaan generoida graafisella käyttöliittymällä. [52]

Tavoitteena oli hakea tietokannasta neuroverkkojen opetusdataksi annossuunnitelmien hoitokenttien monitoriyksiköt, säteilyenergiat, hoitokone ja elektroniapplikaattori. Rinnanpoiston jälkeisten sädehoitojen annossuunnitelmien hoitomuodon tunnistetieto HA013 on Course-taulun Comment-kentässä. Annossuunnitelmat rajattiin aikavälille 1.7.2011–19.3.2013. Potilaan asettelua varten suunnitellut kuvauskentät saatiin rajattua pois valitsemalla ne tietueet, joiden Field-taulun SetupField-kentässä oli arvo 0. Potilaalle tehdään usein monta annossuunnitelmaa, joista paras valitaan toteutettavaksi. Haetut annossuunnitelmat tulee näin ollen rajata hoitoon hyväksyttyihin annossuunnitelmiin. Hoitoon hyväksyttyjen annossuunnitelmien rajaus tehtiin valitsemalla tietueet, joissa PlanSetup-taulun Status-kentässä on TreatApproval-merkintä.

Infomakerin generoimaa SQL-koodia käytettiin myöhemmin hyödyksi Matlabilla tehdyssä tietokantahaussa, jonka käyttö on tarkemmin kuvattu kappaleessa 5.11. Liitteessä B on Matlabilla tehdyssä haussa käytettty SQL-koodi, joka on sama kuin InfoMakerilla tehdyssä SQL-kyselyssä lukuun ottamatta potilaiden henkilötunnuksia Patient-taulun PatientId-kentästä. InfoMakerilla Patient-taulun ja Course-taulun

yhdistäminen tapahtui automaattisesti PatientSer-kentällä. Matlabissa taulut yhdistettiin kahdella erillisellä SQL-kyselyllä, minkä jälkeen hakujen tulokset yhdistettiin. Matlabilla ja InfoMakerilla tehdyillä SQL-kyselyillä saatiin samat tietueet.

InfoMakerilla tehdyllä SQL-kyselyllä saatiin tuloksena raakadatatiedosto, jossa on tietueita kolmella kentällä ja pelkällä elektronikentällä sekä useammalla fotonikentällä toteutetuista annossuunnitelmista. Tiedoston kussakin tietueessa oli dataa yhdestä hoitokentästä. Tiedostossa oli duplikaatteja kentistä, joissa käytettiin hoitopäähän kytkettävää elektroniapplikaattoria ja metallista valusuojaa. Duplikaatti-ongelma ratkaistiin poistamalla tiedoston kaikki tietueet, jotka sisälsivät arvon ”Tray” AddOn-taulun AddOnType-kentässä.

Raakadatatiedosto sisälsi myös tietueita annossuunnitelmista, jotka oli tehty uudella L2-lineaarikiihdyttimellä. Uuden L2-lineaarikiihdyttimen annossuunnitelmia oli tehty vasta vain yksittäisiä kertoja, minkä vuoksi ne karsittiin raakadatatiedosta pois. Tiedostoa muokattiin vielä tämän jälkeen siten, että annossuunnitelman hoitokenttien lukumäärä rajattiin kolmeen kappaleeseen ja elektronikenttien lukumäärä yhteen kappaleeseen.

5.5 Datan esikäsittely

Datan käsittely Infomaker-ohjelmiston työtilassa ei ollut mahdollista. Lisäksi monimutkaisempien SQL-hakujen tekeminen oli hankalaa, koska SQL-koodia ei onnistuttu muokkaamaan käsin. Infomaker-ohjelmistolla saatiin SQL-haussa hoitokoneiden tunnistenumerot kentästä ResourceSer ja hoitokenttien säteilyenergioiden tunnistenumerot kentästä EnergyModeSer. Tunnistenumeroiden avulla voidaan hakea hoitokoneiden nimet ja säteilyenergiat tietokannan konekohtaisesta osuudesta (vaaleanpunaiset taulut). Infomaker-ohjelmistolla tämä ei ollut mahdollista, joten tunnistenumeroiden esikäsittely tehtiin Matlab-ohjelmistolla.

AddOn-taulun kentästä AddOnId saadaan lineaarikiihdyttimen hoitopäähän lisättävän laitteen tunnistenimi, joka esimerkiksi 15×15 cm:n elektroniapplikaattorille on A15. Neuroverkot pystyvät käsittelemään dataa ainoastaan lukuina, minkä vuoksi AddOnId-kentän tunnistenimet numeroitiin positiivisilla kokonaisluvuilla

tasavälisesti. Hoitokoneisiin ja energioihin liittyvät tunnisteluvut numeroitiin myös samalla periaatteella, koska epätasaiset välit voivat johtaa epätoivottuihin painotuksiin neuroverkon opetuksessa.

Annossuunnitelman hoitokentän monitoriyksiköt saadaan tietokannasta laskettua tietokannan kenttien MUpGy, FieldDose ja DoseSpecificationFlag tiedoista, jotka löytyvät tauluista Field ja FieldRefPoint. MUpGy-kenttä ilmaisee hoitokoneen monitoriyksiköiden ja absorboituneen annoksen välisen suhteen [MU/Gy], jota käytetään hoitokentän annoslaskennassa. FieldDose-kentästä saadaan hoitokentän kenttäannos annossuunnitelman jokaisella fraktiolla. DoseSpecificationFlag-kenttä viittaa siihen kenttäannokseen, jota hoidossa käytetään. Monitoriyksiköt määritettiin kaavalla

$$MU = MUpGy * FieldDose\,, \quad \text{kun} \quad DoseSpecificationFlag = 1. \tag{5.4}$$

Esikäsitelty data oli Matlabissa tallennettuna matriisiin, jossa kullakin rivillä oli hoitokentän muuttujat sarakkeittain. Monitoriyksiköiden määrittämisen jälkeen matriisin M rivit saatiin muotoon

$$M_{rivi} = [\,PlanSetupSer\,,\, MU\,,\, L\,,\, E\,,\, A\,], \tag{5.5}$$

missä $PlanSetupSer$ on annossuunnitelman identifioiva tunnusluku, MU on kaavan 5.4 mukaan määritetyt monitoriyksiköt, L on lineaarikiihdyttimen tunnusluku, E on hoitoenergian tunnusluku ja A on elektroniapplikaattorin tunnusluku. Muuttujan A arvon ollessa 4 hoitokentällä ei ole elektroniapplikaattoria.

Datankäsittelyn helpottamiseksi matriisin muokattiin vielä siten, että kullekin riville asetettiin yhden annossuunnitelman data. Fotonikentistä muuttuja A jätettiin pois, koska fotonikentissä ei käytetä elektroniapplikaattoria. Muuttujat $PlanSetupSer$ ja L ovat yhteisiä kaikille kentille, jonka vuoksi kullekin riville jätettiin vain yksi kappale kumpiakin muuttujia. Matriisin rivit saatiin lopulta siis muotoon

$$M_{rivi} = [\,PlanSetupSer\,,\, L\,,\, MU_{taka}\,,\, E_{taka}\,,\, MU_{etu}\,,\, E_{etu}\,,\, MU_{ele}\,,\, E_{ele}\,,\, A\,], \tag{5.6}$$

missä MU_{taka}, MU_{etu} ja MU_{ele} ovat taka-, etu- ja elektronikentän monitoriyksiköt. Vastaavasti E_{taka}, E_{etu} ja E_{ele} ovat taka-, etu- ja elektronikentän säteilyenergiat.

5.6 Datan tilastollinen analyysi

Tilastollisessa analyysissä keskitytään lähinnä kenttien MU_{taka}, MU_{etu} ja MU_{ele} monitoriyksiköihin. Annossuunnitelmia oli mallin muodostusta varten 254 kpl. Kenttien monitoriyksiköt vaihtelivat välillä 71–204 MU. Etukenttään oli kaikille annossuunnitelmille valittu 6 MV:n fotonisäteily. Takakentässä oli käytetty 6 MV:n, 10 MV:n (L1), 15 MV:n (L3 ja L4) tai 18 MV:n (L2) fotonisäteilyä. Elektronikentissä esiintyivät elektronienergiat 6 MeV, 9 MeV, 12 MeV ja 16 MeV. Elektronikentässä oli käytetty 15×15 cm:n, 20×20 cm:n tai 25×25 cm:n elektroniapplikaattoria. Annossuunnitelmat oli tehty hoitokoneille L1, vanha L2, L3 ja L4.

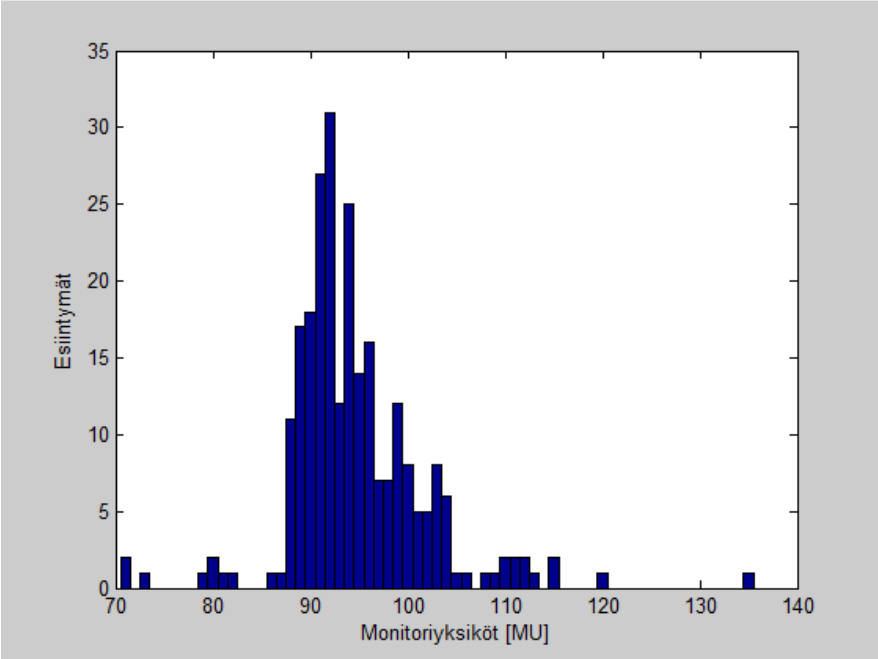
Taulukossa 5.1 on laskettuna kullekin kentälle monitoriyksiköiden minimi, maksimi keskiarvo ja keskihajonta. Taulukossa 5.2 on laskettuna monitoriyksiköiden samat tilastolliset suureet eri hoitokoneille. Taulukko 5.2 on käyttökelpoinen apuvälinen annossuunnitelmien monitoriyksiköiden järkevyyden arviointiin annossuunnitelman tarkastusvaiheessa.

Taulukko 5.1: Monitoriyksiköiden tilastollisia suureita.

Kenttä	Keskiarvo	Keskihajonta	Minimi	Maksimi
MU_{taka}	94,7	7,1	71,1	135,3
MU_{etu}	166,1	5,4	157,0	183,6
MU_{ele}	195,7	5,0	182,2	203,8

Taulukko 5.2: Monitoriyksiköiden tilastollisia suureita eri erihoitokoneille.

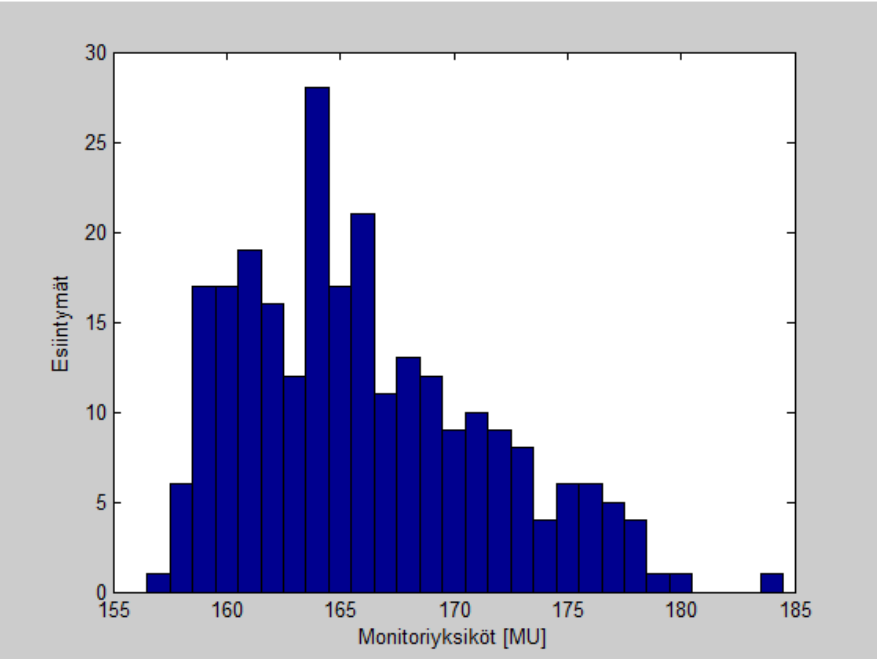
Kenttä, hoitokone	Keskiarvo	Keskihajonta	Minimi	Maksimi
MU_{taka}, *L1*	93,2	7,9	71,1	119,8
MU_{taka}, *L2*	92,8	7,0	71,3	111,7
MU_{taka}, *L3*	94,8	7,4	81,0	135,3
MU_{taka}, *L4*	96,7	5,9	88,1	115,1
MU_{etu}, *L1*	164,5	4,3	158,5	176,5
MU_{etu}, *L2*	165,2	5,2	158,3	178,4
MU_{etu}, *L3*	169,0	5,5	161,0	183,6
MU_{etu}, *L4*	165,1	5,1	157,0	177,7
MU_{ele}, *L1*	187,3	2,0	183,2	194,2
MU_{ele}, *L2*	193,6	3,2	182,2	196,8
MU_{ele}, *L3*	198,9	1,9	193,6	202,3
MU_{ele}, *L4*	199,0	2,2	191,0	203,8



Kuva 5.2: Takakentän monitoriyksiköt [MU].

Takakentän monitoriyksiköiden histogrammista (kuva 5.2) on nähtävissä, kuinka jakauma muistuttaa jonkin verran Gaussin jakaumaa mutta ei ole symmetrinen. Jakauman oikea puoli ulottuu jakauman huippukohdasta huomattavasti etäämmälle

kuin jakauman vasen puoli. Jakauman molemmin puolin on muutamia monitoriyksikkölukemia, jotka poikkeavat selvästi jakaumasta.

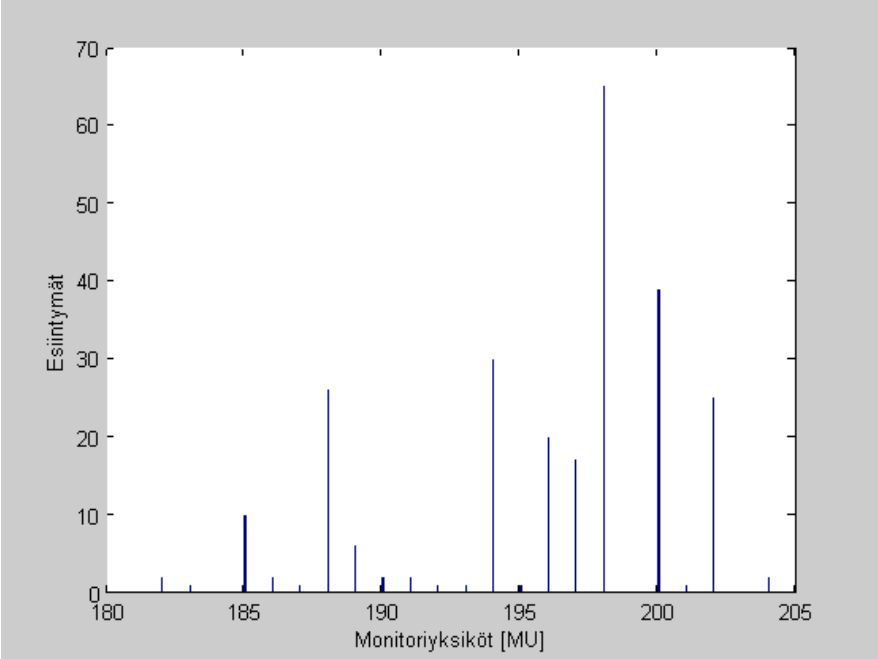


Kuva 5.3: Etukentän monitoriyksiköt [MU].

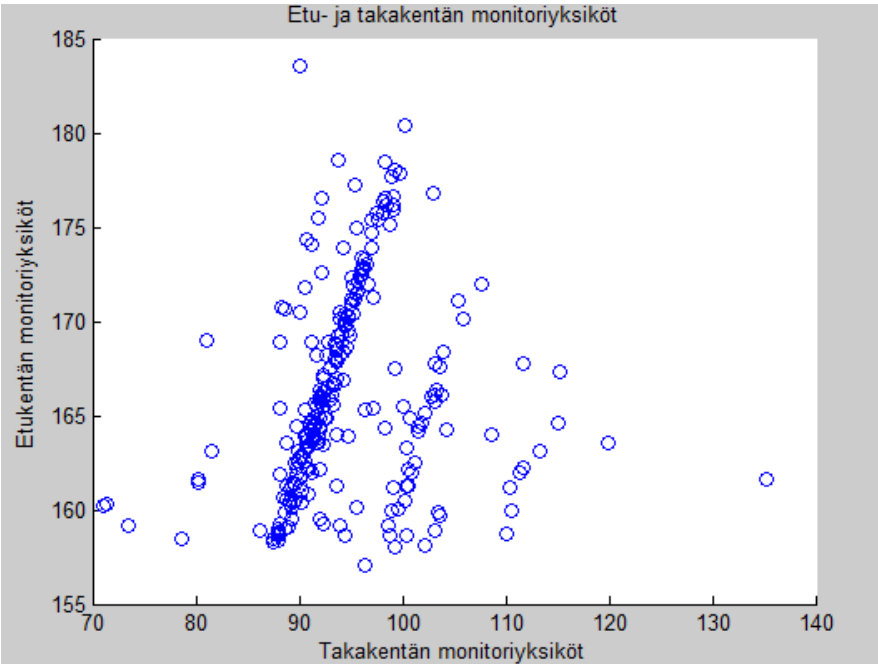
Etukentän monitoriyksiköiden jakauma (kuva 5.3) on yhtenäisempi kuin takakentän jakauma. Ainoastaan yksi monitoriyksikkölukema poikkeaa selvästi jakaumasta. Etukentän jakaumassa on huomattavissa sama muoto kuin takakentän jakaumassakin.

Elektronikentän monitoriyksikköjakauma (kuva 5.4) on diskreetti, mikä johtuu siitä, että elektronikentät suunnitellaan 2 Grayn kenttäannoksena. Annossuunnittelun tuloksena saatu monitoriyksikkölukema on yleensä aina sama kullekin lineaarikiihdytin-, elektroniapplikaattori- ja elektronienergiayhdistelmälle.

Etu- ja takakentän monitoriyksikköpisteparien kuvaaja on esitettynä kuvassa 5.5. Etu- ja takakentän monitoriyksikkölukemien välillä on selvästi korrelaatiota. Elektronikentän monitoriyksikkölukemat eivät korreloi fotonikenttien monitoriyksikkölukemien kanssa, minkä vuoksi elektronikenttä on johdonmukaista käsitellä erillisenä osana poikkeavuuksien havaitsemisessa. Fotonikenttien osalta annossuunnitelmien poikkeavuuksia pyritään näin ollen havaitsemaan muuttujien $L, MU_{taka}, E_{taka}, MU_{etu}$ ja E_{etu} avulla. Elektronikenttien osalta poikkeavuuksia pyritään havaitsemaan muuttujien L, MU_{ele}, E_{ele} ja A avulla.



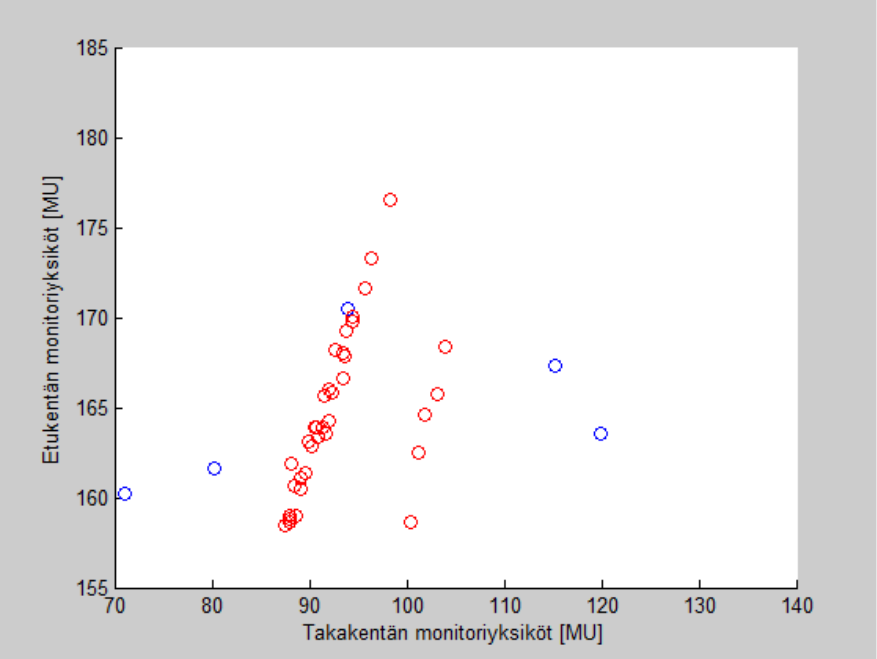
Kuva 5.4: Elektronikentän monitoriyksiköt [MU].



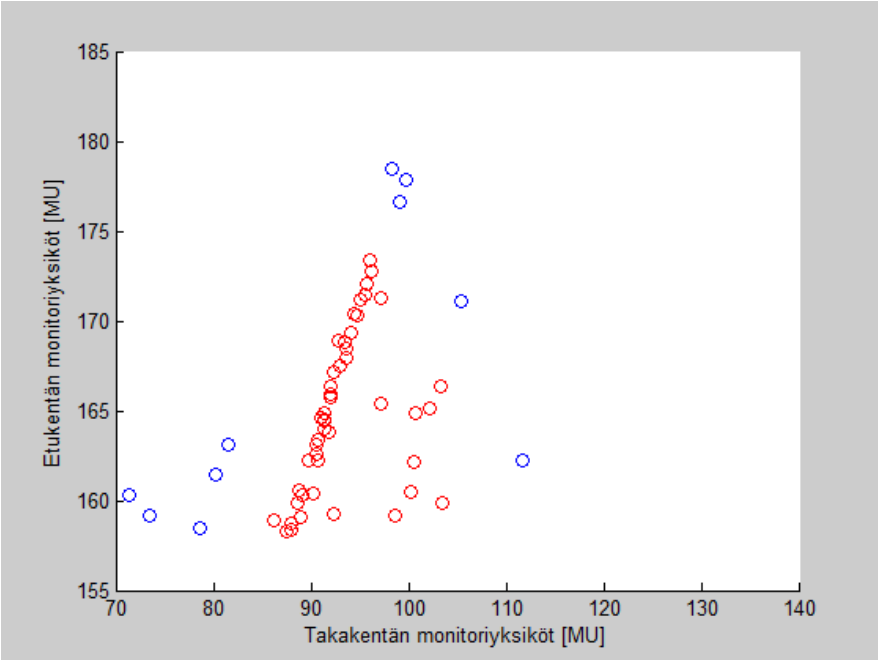
Kuva 5.5: Etu- ja takakentän monitoriyksiköiden korrelaatio.

Fotonikenttien opetuksessa käytettävistä datavektoreista poistettiin selviä poikkeavuuksia 28 kpl, jotta poikkeavuuksien havaitsemismallin muodostus onnistuisi paremmin. Poikkeavuudet poistettiin hoitokonekohtaisesti tarkastelemalla etukentän ja takakentän muodostamia pistejoukkoja. Kuvissa 5.6–5.9 selvät poikkeavuudet on

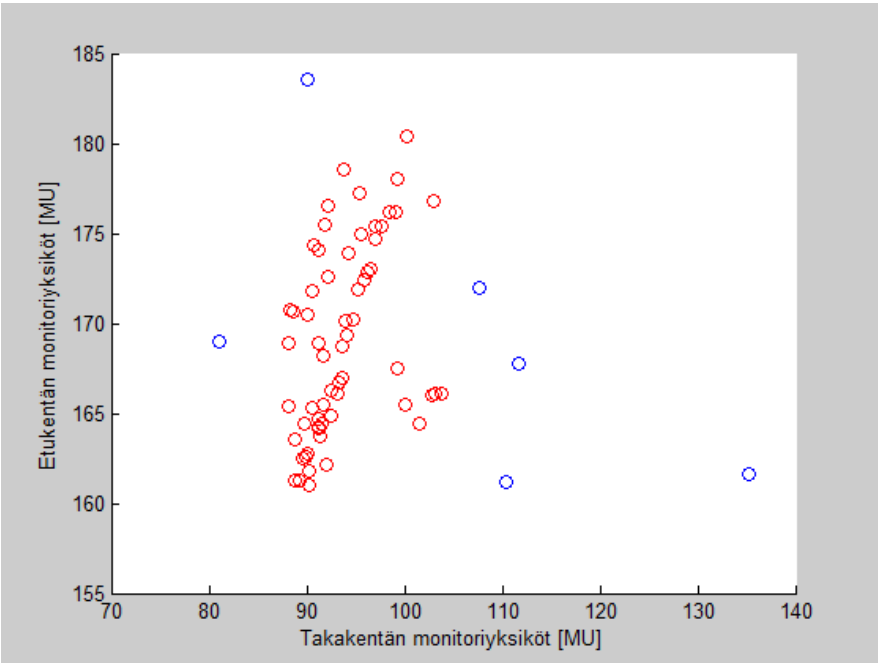
merkitty sinisellä ja muut pisteet punaisella. Kuvassa 5.6 pistejoukon keskellä sijaitsevan selvän poikkeavuuden syy on se, että takakentässä on poikkeuksellisesti käytetty 6 MV:n fotonisäteilyä. Potilaan sädehoidon suunnitelma tarkistettiin Eclipse–annossuunnitteluohjelmasta, ja pienen säteilyenergian käyttö kyseisessä annossuunnitelmassa selittyi potilaan pienellä koolla. Muitakin selvästi poikkeavia annossuunnitelmia tarkastettiin. Pääasiallinen selitys poikkeavuuksille oli potilaan suuri koko, josta johtuen etu- ja takakentän painotuksia oli jouduttu muuttamaan. Opetusdatasta poistetut datavektorit eivät sisältäneet virheellisiä annossuunnitelmia, mutta annossuunnitelmat olivat kuitenkin sellaisia, joihin on syytä kiinnittää tarkempaa huomiota annossuunnitelman tarkastusvaiheessa.



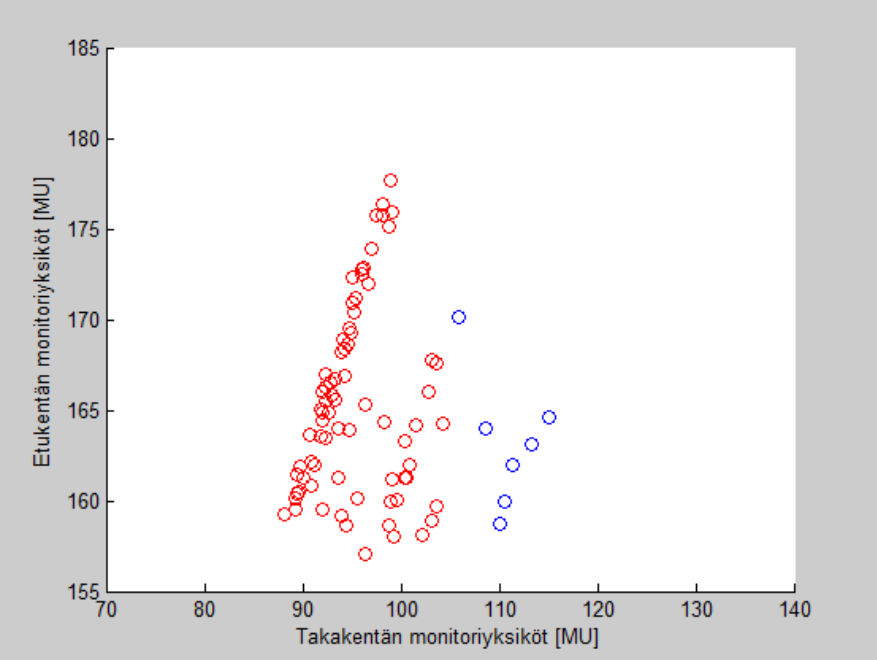
Kuva 5.6: Hoitokoneen L1 etu- ja takakentän monitoriyksikköpisteparien poistetut selvät poikkeavuudet.



Kuva 5.7: Hoitokoneen L2 etu- ja takakentän monitoriyksikköpisteparien poistetut selvät poikkeavuudet.



Kuva 5.8: Hoitokoneen L3 etu- ja takakentän monitoriyksikköpisteparien poistetut selvät poikkeavuudet.



Kuva 5.9: Hoitokoneen L4 etu- ja takakentän monitoriyksikköpisteparien poistetut selvät poikkeavuudet.

Fotonikenttien datavektoreissa oli 4 duplikaattia, minkä ei pitäisi olla mahdollista jatkuvalle muuttujalle. Asia tarkistettiin 8 suunnitelman osalta Eclipsestä. Duplikaatit selittyivät sillä, että suunnitelmasta oli tehty toinen suunnitelma samalle hoitokoneelle samoilla monitoriyksikkölukemilla ja säteilyenergioilla, mutta jotain muuta parametria tai muita parametreja oli muutettu. Yhdelle potilaalle oli esimerkiksi päätetty poikkeavasti antaa yksi lisäfraktio, minkä vuoksi alkuperäisestä suunnitelmasta oli tehty kopio ja fraktioiden lukumääräksi oli asetettu 1. Tämä on todennäköisesti tehty kompensoimaan hoitojakson aikana ollutta taukoa. Tauon jälkeen tapahtuva kompensointi on normaali käytäntö sädehoidossa. Opetusdatasta poistettiin 4 duplikaattia vääristämästä jakaumaa.

Selvien poikkeavuuksien ja duplikaattien poistamisen jälkeen fotonikenttien opetusdata koostui 222 datavektorista. Elektronikenttien osalta opetusdatan muokkaukselle ei ollut tarvetta, minkä vuoksi esikäsittelyvaiheen jälkeen elektronikenttien datavektoreita oli edelleen 254 kappaletta.

5.7 Datan normalisointi

SOM-neuroverkon syötevektorin muuttujat tulee normalisoida, jotta laajalla skaalalla vaihtelevat muuttujat eivät hallitse klusterien muodostumisessa. Opetusdatan normalisointikäytäntö on yleinen muillekin neuroverkoille ja muille klusterointi-algoritmeille. Opetusdatan kaikki muuttujat normalisoitiin siten, että niiden keskiarvo on 0 ja keskihajonta 1. Opetusdatan jokaisen datavektorin i muuttuja j normalisoitiin kaavalla [17]

$$p_{ij}^{*} = \frac{m_j - p_{ij}}{\sigma_j}, \tag{5.7}$$

missä on p_{ij}^{*} on normalisoitu datavektorin muuttuja, p_{ij} normalisoitava datavektorin muuttuja, m_j on opetusdatan muuttujan j keskiarvo ja σ_j on opetusdatan muuttujan j keskihajonta. Mikäli muuttuja on vakio normalisoiduksi muuttujaksi asetetaan 0 ja keskihajonnaksi 0. Opetetun neuroverkon uudet syötteet normalisoitiin käyttämällä kaavaa 5.7 opetuksessa käytetyille normalisointiparametreille m_j ja σ_j.

Datavektorin normalisoitu muuttuja voidaan palauttaa takaisin normittamattomaksi muuttujaksi kaavalla

$$p_{ij} = p_{ij}^{*}\sigma_j + m_j \ . \tag{5.8}$$

5.8 Itseorganisoituvan kartan opetus

5.8.1 Itseorganisoituvan kartan opetus fotonikenttien datalla

Ennen SOM-neuroverkon opetusta tulee tehdä useita valintoja sen parametrien suhteen. SOM-neuroverkon etäisyysmetriikaksi valittiin euklidinen, koska se on tyypillisesti käytetty SOM:n etäisyysmetriikka. Verkkohilan rakenteeksi valittiin heksagonaalinen hilarakenne. Neuroneiden lukumäärää M voidaan arvioida kaavalla [35]

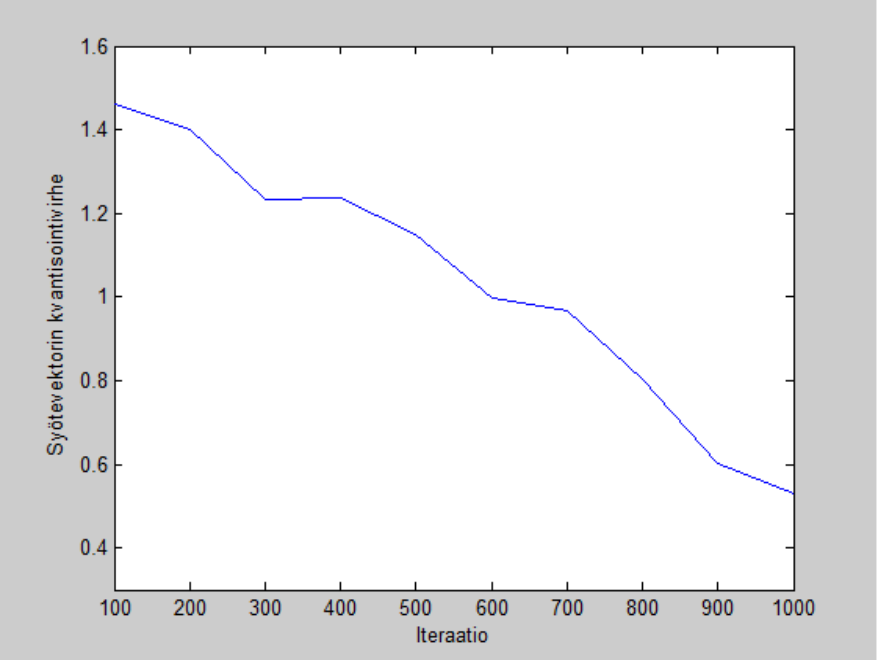
$$M = 5\sqrt{N}. \tag{5.9}$$

Kaavan 5.9 määrittämillä neuroneiden lukumäärillä saadut klusterit olivat kuitenkin kooltaan turhan pieniä PNN-neuroverkkojen käyttöön, minkä vuoksi sopiva neuroneiden lukumäärä haettiin kokeilemalla pienemmillä verkkohiloilla. Hilan dimensiot d_1 ja d_2 laskettiin siten, että syötevektoreiden kahden suurimman ominaisarvon suhdeluku s on approksimativiisesti sama kuin hilandimensioiden suhdeluku d_1/d_2. Kaksi suurinta ominaisarvoa saadaan laskettua syötevektoreiden autokorrelaatiomatriisista [34].

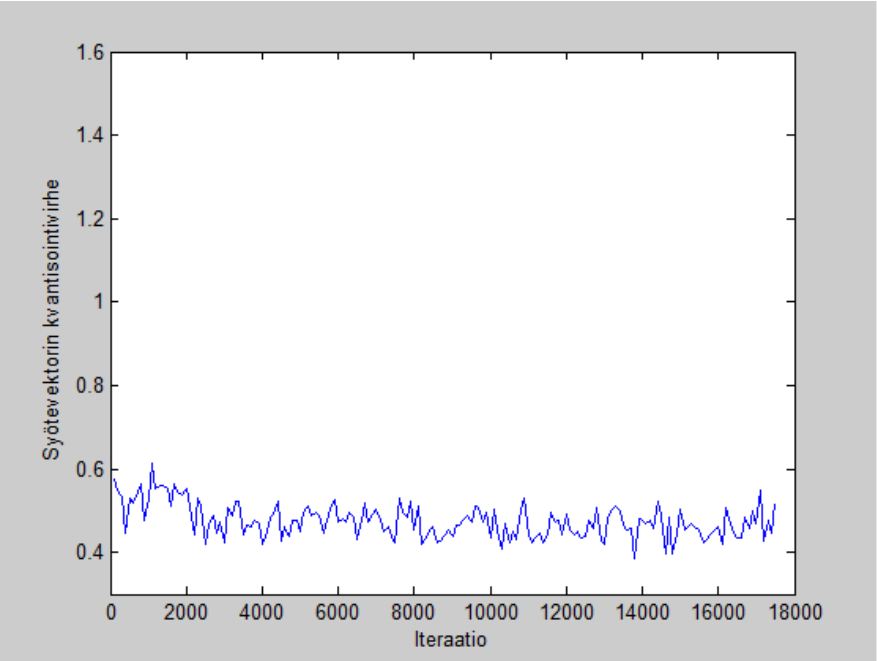
Autokorrelaatiomatriisi laskettiin Matlabilla käyttäen *cov*-funktiota ja ominaisarvot saatiin autokorrelaatiomatriisista laskettua *eig*-funktiolla. Kahden suurimman ominaisarvon suhdeluvuksi fotonikenttien syötevektoreille saatiin 1,29. Hilandimensioiksi valittiin $d_1 = 7$ ja $d_2 = 5$, koska näillä dimensiolla saatujen klustereiden PNN-neuroverkoilla havaitut poikkeavuudet vaikuttivat järkeviltä ja suhdeluku erosi ominaisarvojen suhdeluvusta vain 0,11.

Naapuruusfunktio valittiin myös kokeellisesti vertailemalla keskimääräistä kvantisointivirhettä ja topografista virhettä eri naapuruusfunktioille. Edellä mainitut suureet laskettiin *som_quality*-funktiolla. Klusterointia testattiin kuplanaapuruusfunktiolle, gaussiselle naapuruusfunktiolle ja katkaistulle gaussiselle naapuruusfunktiolle. Katkaistu gaussinen naapuruusfunktio antoi molemmille laatukriteereille pienimmät virheet. Topografinen virhe saatiin katkaistulla gaussisella naapuruusfunktiolla useissa tapauksissa nollaksi. Painovektorit alustettiin lineaarisella alustuksella käyttämällä *som_lininit*-funktiota, jolloin painovektorit ovat lähtötilanteessa tasaisesti levittäytyneet painoavaruuteen syötevektoreiden varianssia mukaillen.

Opetusalgoritmina käytettiin inkrementaalista algoritmia (Matlabissa *som_seqtrain*) ja syötevektorit esitettiin verkolle satunnaisessa järjestyksessä. Oppimiskertoimen $\alpha(t)$ ja naapuruussäteen $\sigma(t)$ funktioksi valittiin lineaarisesti laskeva funktio. Järjestäytymisvaiheessa naapuruussäde asetettiin alkutilanteessa arvoon 4 ja lopputilanteessa arvoon 1. Oppimiskerroin asetettiin alkutilanteessa arvoon 0,9 ja lopputilanteessa arvoon 0. Hienosäätövaiheessa oppimiskerroin oli alkutilanteessa 0,02 ja lopputilanteessa 0. Naapuruussäde pidettiin hienosäätövaiheen ajan vakioarvossa 1. Järjestysvaiheessa opetusiteraatiota tehtiin 1 000 kpl. Hienosäätövaiheessa opetusiteraatioita tehtiin $7 \times 5 \times 500 = 17\,500$ kpl.



Kuva 5.10: Erään SOM-opetuksen syötevektorin kvantisointivirhe iteraatioiden funktiona järjestäytymisvaiheessa.

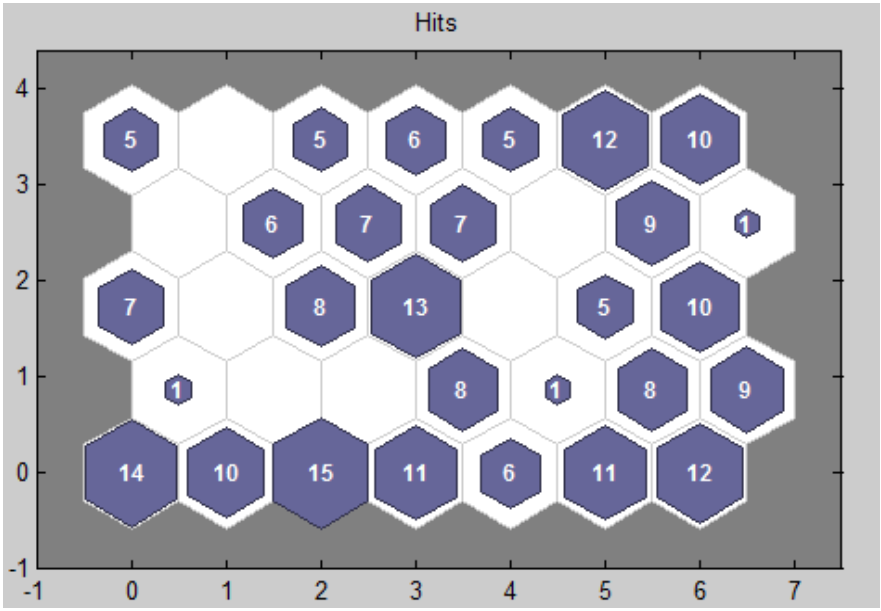


Kuva 5.11: Erään SOM-opetuksen syötevektorin kvantisointivirhe iteraatioiden funktiona hienosäätövaiheessa.

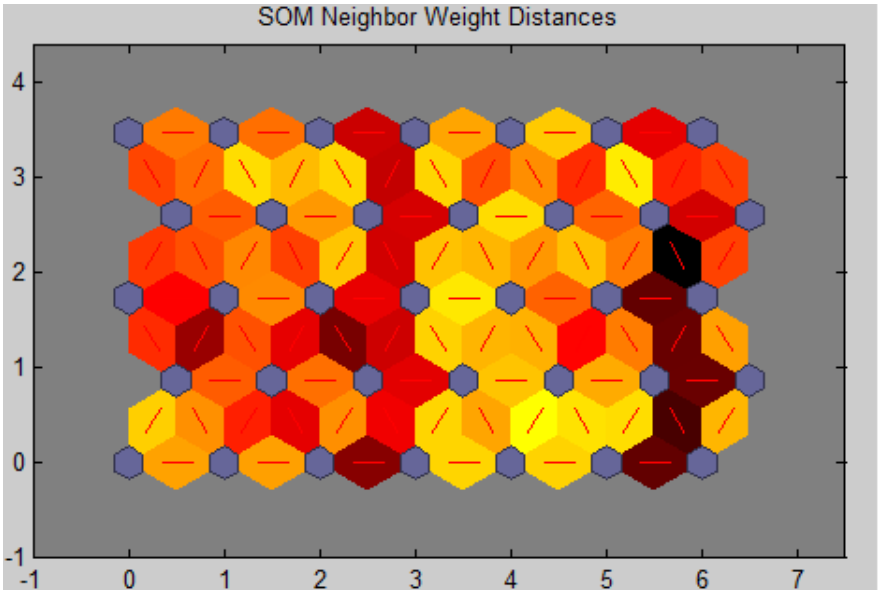
SOM-opetuksen aikaisia syötevektorien kvantisointivirheitä seuraamalla voidaan saada tietoa oppimisparametrien järkevyydestä opetuksen aikana. Kuvassa 5.10 on järjestäytymisvaiheen syötevektorin kvantisointivirhe iteraatioiden funktiona. Vastaa-

vasti kuvassa 5.11 on hienosäätövaiheen kvantisointivirhe. SOM-opetuksen parametreina käytettiin tässä kappaleessa esitetyttyjä parametrivalintoja. Kvantisointivirhe laskee voimakkaasti järjestäytymisvaiheessa, kun oppimiskerroin ja naapuruussäde on suuri. Järjestäytymisvaiheessa oppimista ei juuri enää tapahdu ja kvantisointivirhe saturoituu tietylle tasolle. Järjestäytymisvaiheen iteraatioiden lukumäärä on näin ollen riittävä.

SOM-neuroverkon opetuksessa saadaan satunnaisuuden vuoksi joka kerta eri tulos. SOM-opetus tehtiin 10 kertaa samoilla parametreilla ja pienimmän keskimääräisen kvantisointivirheen antanut SOM valittiin käytettäväksi PNN-verkkojen opetuksessa. Keskimääräiseksi kvantisointivirheeksi q_e valitulle SOM-verkolle saatiin 0,4312 ja topografiseksi virheeksi t_e saatiin 0,0541.



Kuva 5.12: Fotonikenttien klustereiden koot SOM:n hilassa.



Kuva 5.13: Fotonikenttien SOM:n painovektoreiden väliset etäisyydet.

SOM:n opetuksen jälkeen kaikki syötevektorit syötetään uudestaan SOM-verkolle, jolloin jokaiseen syötevektoriin voidaan liittää luku, joka ilmaisee, mihin klusteriin se kuuluu. Kuvassa 5.12 on SOM-verkon klusterit verkkohilassa. Luku kuusikulmion sisällä ilmaisee klusterin alkioiden lukumäärän. Klustereita saatiin yhteensä 28 kappaletta. Kaikkiaan 7 neuronilla ei ole yhtään alkiota, minkä vuoksi niistä yksikään ei ole yhdenkään syötevektorin paras vastinsolmu. Poikkeaviksi klustereiksi määritetään ne, joihin kuuluu vain yksi alkio. Normaaleja klustereita on näin ollen 25 kappaletta.

Kuvassa 5.13 on esitettynä naapurineuronien painovektoreiden väliset etäisyydet. Neuroneiden paikat hilassa on merkitty sinisillä kuusikulmioilla. Eriväriset suunnikkaat kuvaavat painovektoreiden välisiä etäisyyksiä. Suuremmat etäisyydet on merkitty tummemmilla väreillä. Poikkeavista klustereista oikeassa yläkulmassa ja vasemmassa alakulmassa sijaitsevat neuronit ovat muita neuroneista etäämmällä. Sen sijaan oikeassa alakulmassa sijaitseva neuroni on lähempänä muita neuroneita. Liitteessä C on kuvattuna klusterit etu- ja takakentän monitoriyksiköille hoitokoneittain. Eri klustereihin kuuluvat monitoriyksikköparit on merkitty eri väreillä kuhunkin kuvaan. Yksi klusteri sisälsi hoitokoneiden 3 ja 4 datavektoreita, mikä johtuu todennäköisesti siitä, että hoitokoneilla 3 ja 4 on samat takakentän säteilyenergiat.

5.8.2 Itseorganisoituvan kartan opetus elektronikenttien datalla

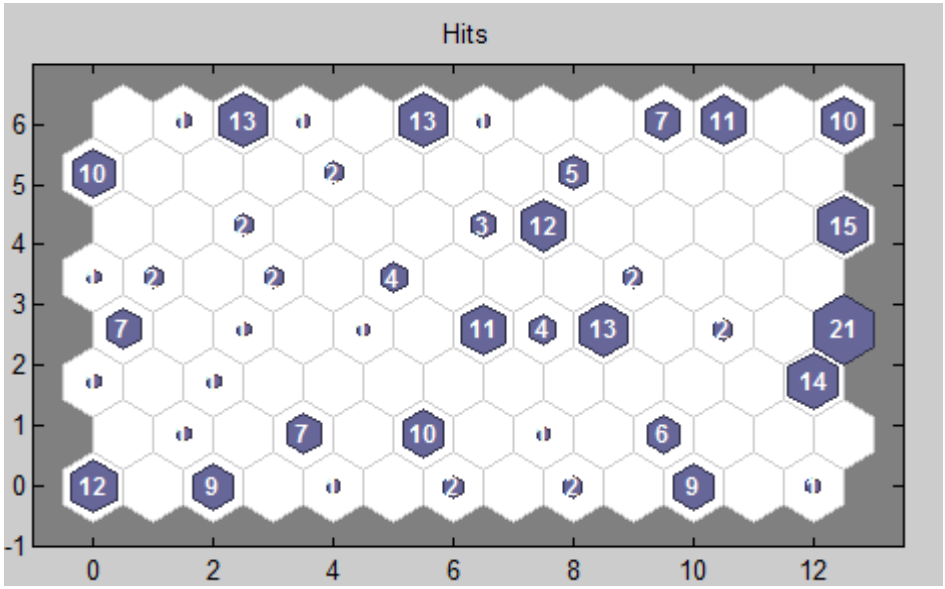
Etäisyysmetriikkana elektronikenttien SOM-opetuksessa käytettiin myös euklidista etäisyysmetriikkaa. Elektronikenttien syötevektoreiden autokorrelaatiomatriisista laskettujen kahden suurimman ominaisarvon suhteeksi saatiin 1,46. SOM-verkon hilan dimensioiksi valittiin 13 ja 8, jolloin hilan dimensioiden suhde on 1,63. Kaava 5.9 antaa neuroneiden lukumääräksi elektronikenttien syötevektoreille 24 neuronia pienemmän verkon kuin valitulle verkolle. Kokoa 13×8 pienimmille verkoille muutamat painovektorit saivat verkon reunoilla tämän edustaman klusterin syötevektoreista poikkeavan arvon. Riittävän suurella verkolla saatiin kyseinen reunaefekti häviämään.

Opetusalgoritmina, naapuruusfunktiona, oppimiskertoimen funktiona ja alustusmenetelmänä käytettiin samoja valintoja kuin fotonikentillekin. Elektronikenttien syötevektoreille opetusiteraatoita tehtiin 2 kertaa normaalia enemmän ja oppimiskerroin pidettiin hieman suurempana opetuksen aikana, jolloin verkon reunoilla olevat painovektorit saatiin paremmin vastaamaan klusterin syötevektoreita. Järjestäytymisvaiheessa naapuruussäde asetettiin alkutilanteessa arvoon 7 ja lopputilanteessa arvoon 1. Oppimiskerroin asetettiin alkutilanteessa arvoon 0,9 ja lopputilanteessa arvoon 0. Hienosäätövaiheessa oppimiskerroin oli alkutilanteessa 0,05 ja lopputilanteessa 0. Naapuruussäde pidettiin hienosäätövaiheen ajan vakioarvossa 1. Järjestäytymisvaiheessa opetusiteraatioita tehtiin 1 000 kpl. Hienosäätövaiheessa iteraatioita tehtiin $13\times8\times1000 = 104\,000$ kpl.

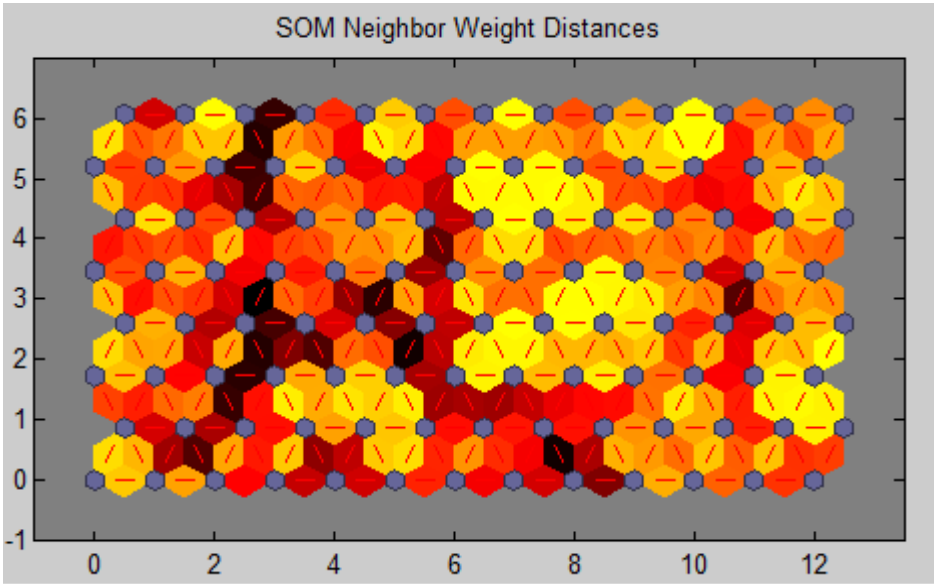
Elektronikenttien osalta SOM-opetus tehtiin myös 10 kertaa ja pienimmän keskimääräisen kvantisointivirheen antanut SOM valittiin. SOM:n keskimääräiseksi kvantisointivirheeksi saatiin 0,0300 ja topografiseksi virheeksi saatiin 0,1220.

Kuvassa 5.14 niitä SOM:n hilan neuroneita, jotka eivät ole syötevektoreiden parhaita vastinsolmuja, on suhteellisen iso osuus verkon neuroneista. Poikkeaviksi neuroneiksi määriteltiin ne neuronit, joiden edustamassa klusterissa alkioiden lukumäärä on 1 ja ne neuronit, jotka eivät ole parhaita vastinsolmuja. Normaaleja neuroneita on näin ollen 31 kappaletta. Kuvassa 5.15 on painovektoreiden väliset etäisyydet. Huolimatta verkon isosta koosta useat isojen klustereiden painovektorit ovat hyvin lähellä toisiaan. Lähellä olevien isojen klustereiden painovektorit osoittautuivat samoiksi,

minkä vuoksi PNN-opetukseen valittiin vain normaalien neuroneiden painovektorit, jotka olivat erilaisia. Samat painovektorit eivät olisi antaneet lisäinformaatiota datasta, vaan ne olisivat vääristäneet PNN-opetusta.



Kuva 5.14: Elektronikenttien klustereiden koot SOM:n hilassa.



Kuva 5.15: Elektronikenttien SOM:n painovektoreiden väliset etäisyydet.

5.9 Mallien ristiinvalidointi

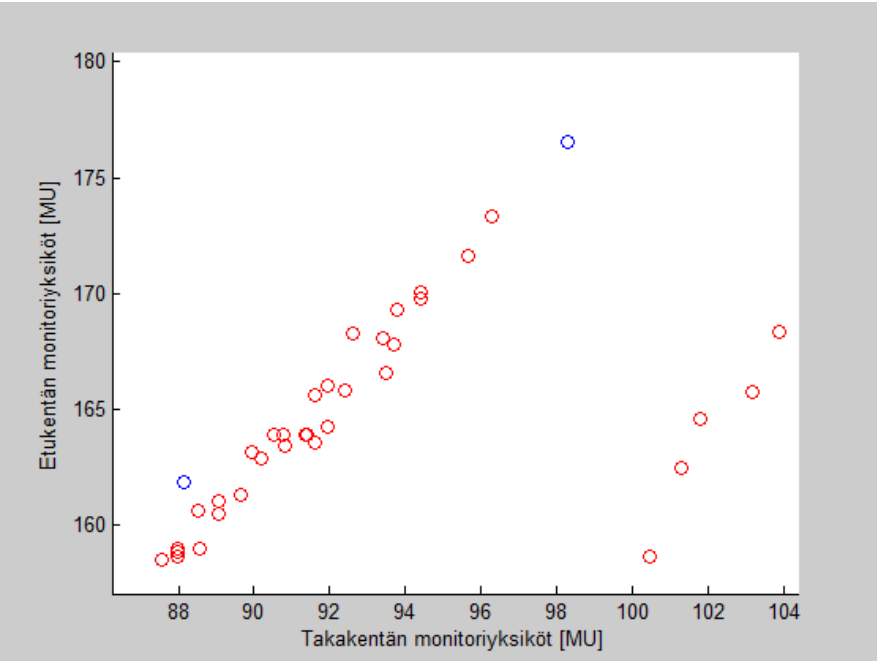
5.9.1 Fotonikenttien mallin ristiinvalidointi

Fotonikenttien PNN-mallin muodostusta varten kaavassa 4.34 esiintyvän korjauskertoimen *K* optimaalinen arvo haettiin 10-kertaisella ositetulla ristiinvalidoinnilla. Ristiinvalidoinnilla määritettiin myös mallin tarkkuus, jotta havaittiin, ettei mallia oltu yliopetettu. SOM:n muodostamista normaaleista klustereista muodostettiin PNN-mallit tutkittavalla korjauskertoimen arvolla. PNN-verkkojen muodostus tehtiin *newpnn*–funktiolla. SOM-opetuksessa käytetyt normalisoimattomat datavektorit syötettiin tämän jälkeen PNN-verkoille normalisoimalla kunkin PNN-verkon syötevektori sen opetuksessa käytetyillä normalisointiparametreilla. SOM-verkon poikkeavien klustereiden syötevektorit syötettiin siis myös PNN-verkoille. Pienimpiä vasteita antaneet 5 % syötevektoreista merkittiin poikkeaviksi ja loput 95 % syötevektoreista merkittiin normaaleiksi.

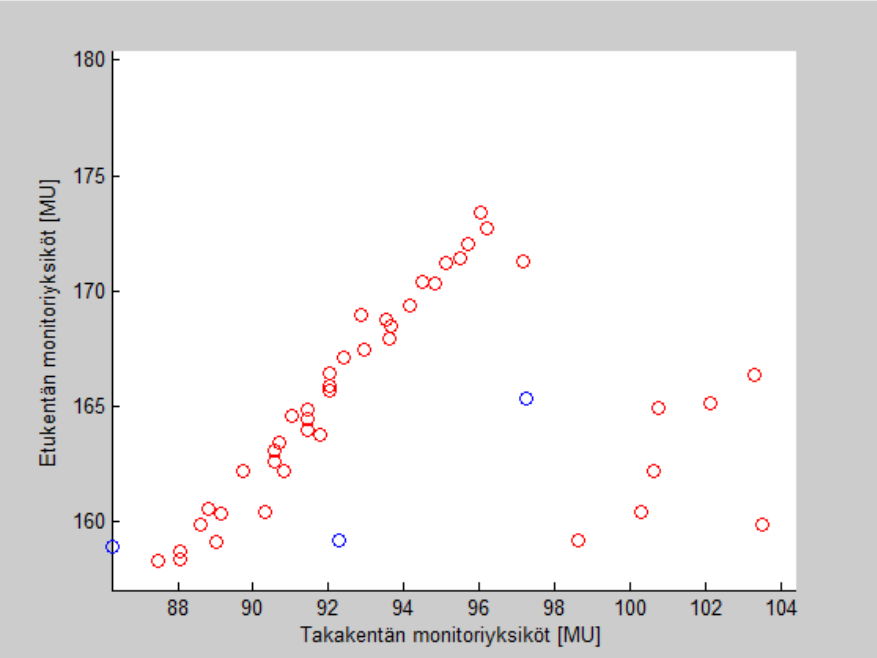
Merkityt syötevektorit jaettiin satunnaisesti kymmeneen ryhmään *cvpartition*-funktiolla. Opetuksessa käytettävä ryhmä klusteroitiin SOM:lla käyttäen samoja SOM:n valinta kriteerejä ja parametrivalintoja kuin käytettiin kaikkien fotonikenttien syötevektoreiden klusteroinnissa. SOM:n klustereista muodostettiin tämän jälkeen PNN-neuroverkot käyttäen samaa PNN-mallien korjauskerrointa, jolla syötevektorit oli merkitty. Kynnysarvoksi t_{fot} (kaava 5.2) asetettiin opetuksessa käytetyn syötevektorin vaste, joka on suurin 5 %:n pienimmän vasteen joukosta. PNN-malleille syötettiin tämän jälkeen testiryhmä ja poistetut selvät poikkeavuudet. Aiemmin poistetut selvät poikkeavuudet syötettiin testiryhmän lisäksi verkoille, koska tällöin saatiin parempi arvio mallin kyvystä havaita poikkeavuuksia. Edellä kuvaillut vaiheet tehtiin 10 kertaa kullekin opetusryhmälle. Testiryhmien ja poistettujen selvien poikkeavuuksien lukuarvot TP, FP, TP ja TN summattiin. Summatuista suureista saatiin laskettua ristiinvalidoinnin arvio käytetyn korjauskertoimen antaman mallin tarkkuudelle.

Sopivan korjauskertoimen arvoa haettiin kokeilemalla väliltä 1,2–2,2. Parhaaksi tarkkuudeksi saatiin 0,9622 korjauskertoimella 1,95. Ristiinvalidointi toistettiin kyseiselle korjauskertoimelle vielä 9 kertaa. Tarkkuuden keskiarvoksi saatiin 0,9378 ja

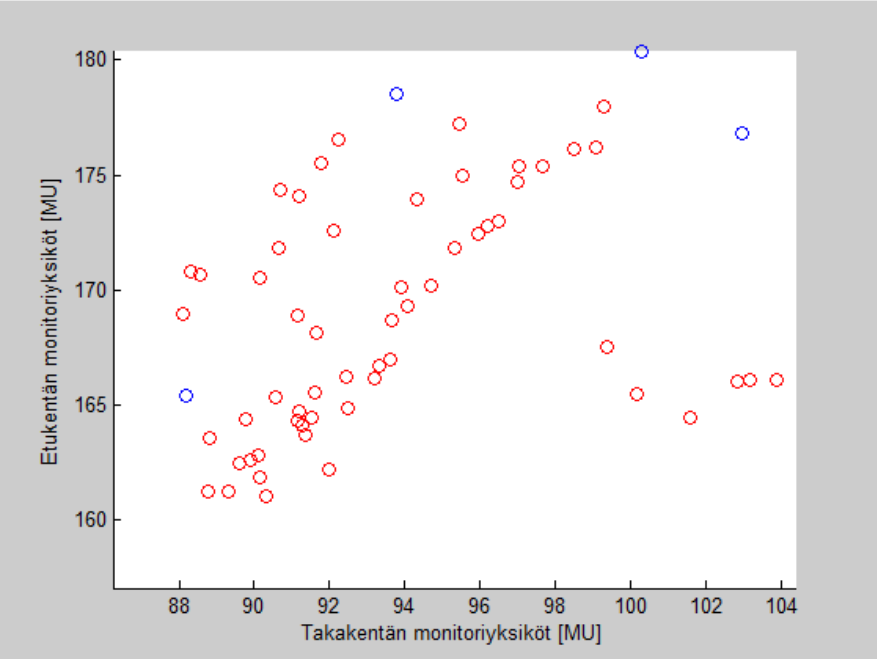
keskiarvon keskivirheeksi saatiin 0,0044, joten tarkkuudeksi virheineen saadaan 0,938 ± 0,005. TPR:n keskiarvoksi saatiin 0,9653 ja keskiarvon keskivirheeksi 0,003, joten $TPR = 0{,}965 \pm 0{,}003$. FPR: n keskiarvoksi saatiin 0,1000 ja keskiarvon keskivirheeksi saatiin 0,0074, joten $FPR = 0{,}100 \pm 0{,}008$. Kuvissa 5.16–5.19 on kuvattu korjauskertoimella 1,95 pienimmät 11 vastetta (5 %:n joukko) antaneet etu- ja takakentän monitoriyksikköpisteparit hoitokoneittain.



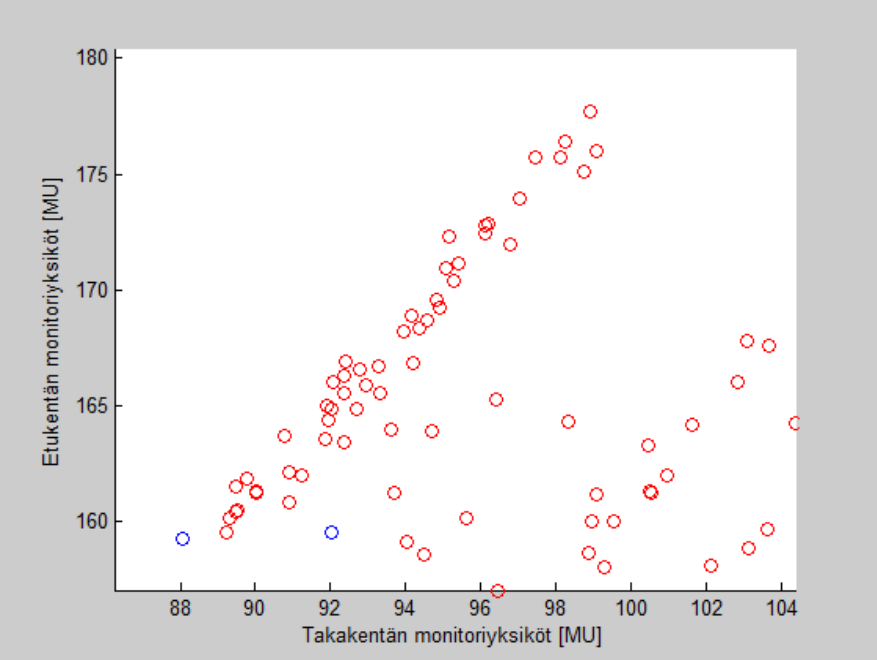
Kuva 5.16: Pienimmän 5 %:n vasteiden joukkoon kuuluvat hoitokoneen L1 etu- ja takakentän monitoriyksikköpisteparit korjauskertoimella 1,95.



Kuva 5.17: Pienimmän 5 %:n vasteiden joukkoon kuuluvat hoitokoneen L2 etu- ja takakentän monitoriyksikköpisteparit korjauskertoimella 1,95.



Kuva 5.18: Pienimmän 5 %:n vasteiden joukkoon kuuluvat hoitokoneen L3 etu- ja takakentän monitoriyksikköpisteparit korjauskertoimella 1,95.



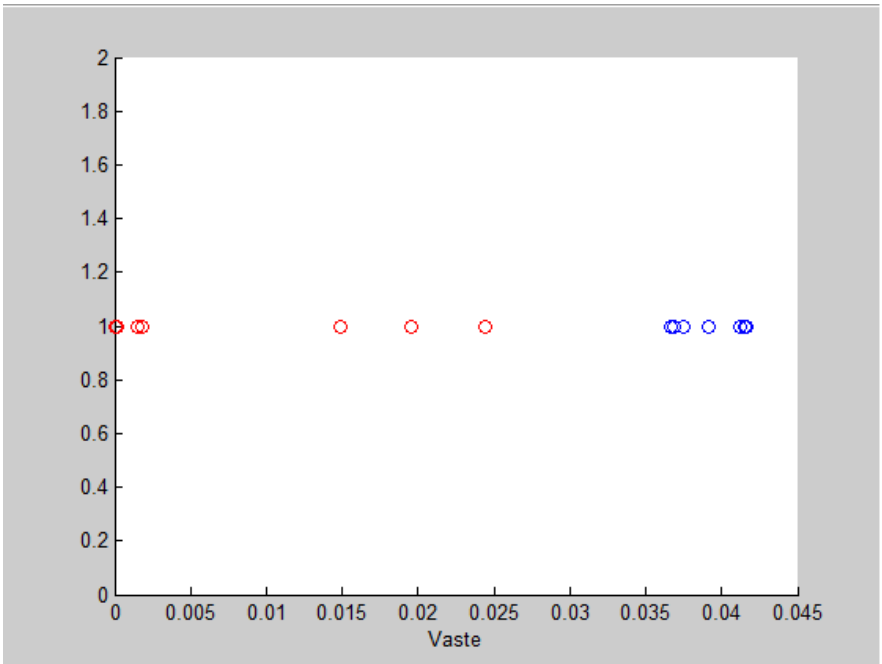
Kuva 5.19: Pienimmän 5 %:n vasteiden joukkoon kuuluvat hoitokoneen L4 etu- ja takakentän monitoriyksikköpisteparit korjauskertoimella 1,95.

5.9.2 Elektronikenttien mallin ristiinvalidointi

Elektronikenttien ristiinvalidoinnilla tarkoituksena oli hakea sopiva PNN:n tasoituskertoimen arvo ja määrittää muodostetun mallin tarkkuus riistiinvalidoinnilla, jotta havaittaisiin, ettei mallia oltu yliopetettu. SOM:n normaaleiden neuroneiden normalisoidut painovektoreiden palautettiin normalisoimattomaksi kaavalla 5.8 käyttämällä SOM:n opetuksen normalisointiparametreja. Näin saadaan poikkeavien neuroneiden klusteriin kuuluvien syötevektoreiden vaikutus poistettua. Seuraavaksi SOM:n painovektoreista muodostetuille datavektoreille laskettiin normalisointiparametrit ja normitus tehtiin kaavalla 5.7 käyttämällä laskettuja parametreja. Normalisoiduilla painovektoreilla opetettiin PNN-verkko valitulla tasoituskertoimella. Seuraavaksi SOM:n opetuksessa käytetyt datavektorit normalisoitiin painovektoreiden normalisointiparametreilla ja syötettiin opetetulle PNN-verkolle. PNN:n syötevektoreista merkittiin valitun kynnysarvon mukaisesti osa poikkeavuuksiksi ja loput normaaleiksi syötevektoreiksi.

Merkityille syötevektoreille 10-kertainen ositettu ristiinvalidointi toteutettiin vastaavasti kuten fotonikenttien ristiinvalidointikin käyttämällä elektronikenttien

SOM-opetuksen ja PNN-opetuksen samoja parametrivalintoja sekä SOM:n valintakriteerei-tä.



Kuva 5.20: Elektronikenttien PNN:n opetuksessa käytettyjen syötevektoreiden vasteet tasoituskertoimella 0,05. Pienimmän 13 %:n vasteiden joukko on merkitty punaisella.

Tasoituskertoimen ja kynnysarvon suuruus haettiin kokeilemalla. Tasoituskertoimen valinta ei vaikuttanut herkästi ristiinvalidoinnista saatavaan tarkkuuteen, mutta se vaikutti testauksessa saataviin tuloksiin. Parhaaksi tasoituskertoimeksi saatiin 0,05 ja kynnysarvoksi asetettiin 0,031. Kynnysarvon valinta tehtiin kuvan 5.20 perusteella, jolloin 13 % syötevektoreista merkittiin poikkeaviksi ja 87 % normaaleiksi. Ristiinvalidoinnilla tarkkuudeksi saatiin 0,9764. Ristiinvalidointi toistettiin vielä 9 kertaa samalla tasoitusparametrilla 0,05 ja kynnysarvolla 0,031. Tarkkuuden keskiarvoksi saatiin 0,9524 ja tarkkuuden keskiarvon keskivirheeksi saatiin 0,0056, joten tarkkuudeksi virheineen saadaan $0{,}952 \pm 0{,}006$. TPR:n keskiarvoksi saatiin 0,9088 keskiarvon keskivirheen ollessa 0,0188, jolloin TPR:ksi virheineen saadaan $0{,}91 \pm 0{,}02$. FPR:n keskiarvoksi saatiin 0,0409 ja keskiarvon keskivirheeksi 0,0051. FPR:ksi virheineen saadaan $0{,}041 \pm 0{,}006$.

5.10 Mallien testaus keinotekoisilla virheillä

5.10.1 Mallin testaus fotonikenttien keinotekoisilla virheillä

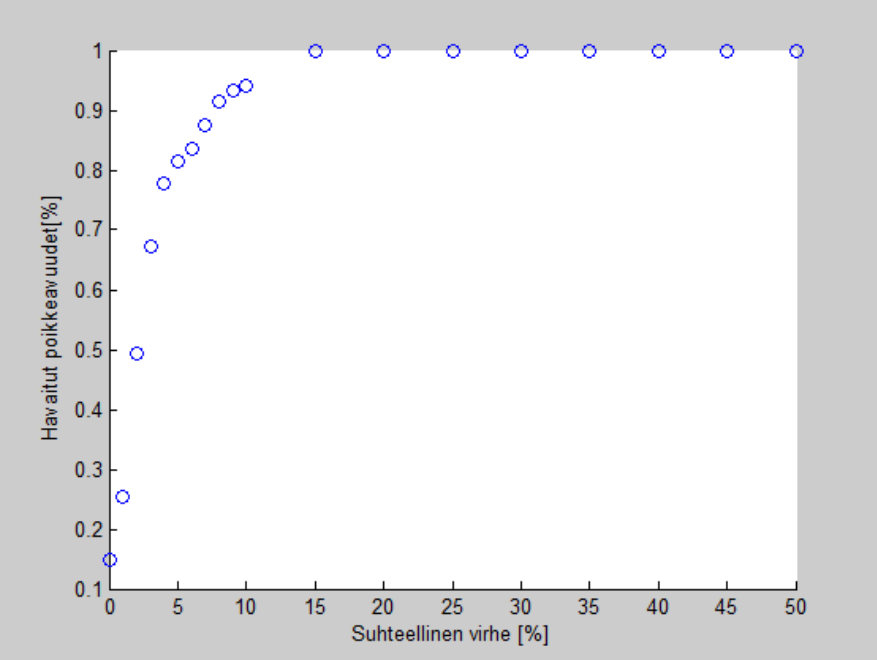
Molemmat mallit testattiin keinotekoisesti tuotetuilla virheillä. Keinotekoisia virheitä tehtiin poikkeuttamalla monitoriyksiköitä ja säteilyenergioita normaaleista arvoistaan. Virheitä tehtiin pitkälti samalla menetelmällä kuin Azmandian ym. [21] tutkimuksessaan. Tässä tutkimuksessa virheet generoitiin esikäsittelyvaiheen jälkeisestä datasta fotonikenttien ja elektronikenttien datavektoreille erikseen. Monitoriyksiköille virheitä generoitiin poikkeuttamalla jokaisen datavektorin monitoriyksikkölukemaa MU_{alk} kaavalla

$$MU_{virhe} = MU_{alk} \pm MU_{alk}\frac{\delta}{100\,\%}, \tag{5.10}$$

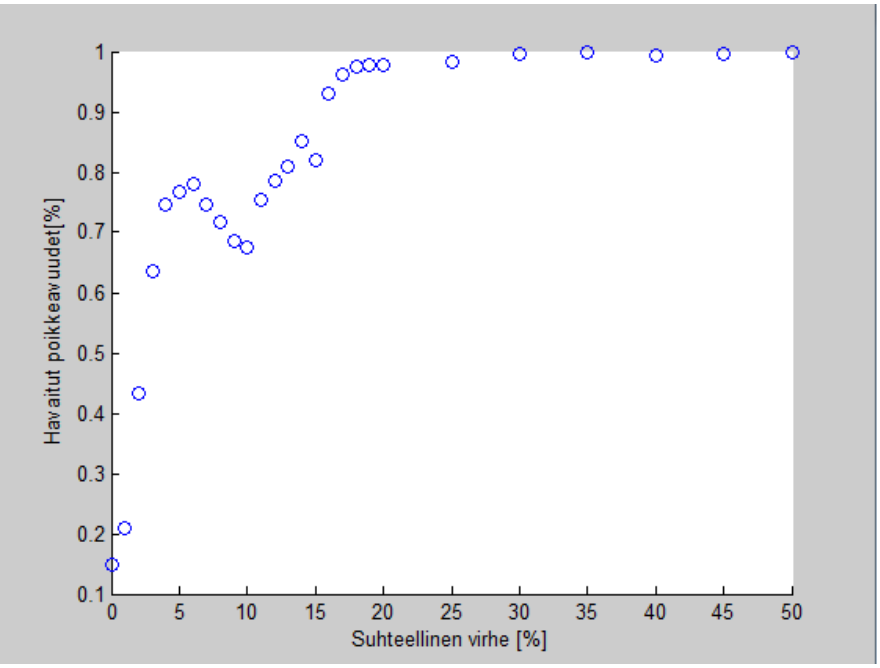
missä δ on virhe suhteessa alkuperäiseen arvoon. Datavektorin muiden muuttujien arvot säilytettiin ennallaan. Elektronikentille säteilyenergioiden virheet tehtiin muuttamalla datavektorin säteilyenergiaksi arvoksi jokin muu säteilyenergian arvo. Jokaiselle datavektorille tehtiin virheet kaikille mahdollisille energian arvoille. Esimerkiksi elektronikentän datavektorille, jonka säteilyenergiana on 9 MeV tehtiin virheiksi datavektorit, joiden säteilyenergioina olivat 6 MeV, 12 MeV, 15 MeV, 16 MeV ja 20 MeV. Muiden muuttujien arvot pidettiin alkuperäisissä arvoissaan.

Fotonikenttien ja elektronikenttien datavektoreille muodostetut mallit testattiin monitoriyksiköiden ja säteilyenergioiden virheillä. Testattavat mallit ovat samat, joita käytettiin merkitsemään ristiinvalidoinnin syötevektorit. Kuvassa 5.21 on havaitut poikkeavuudet etukentän monitoriyksiköiden suhteellisille virheille. Monitoriyksiköiden 15 %:n suhteellisista virheistä tai sitä suuremmista havaittiin kaikki, 10 %:n virheistä havaittiin 94 % ja 5 %:n virheistä havaittiin 82 %.

Kuvassa 5.22 on havaitut poikkeavuudet takakentän monitoriyksiköiden suhteellisille virheille. Lähes kaikki 20 %:n virheet ja sitä suuremmat havaittiin, 15 %:n virheistä havaittiin 82 % ja 5 %:n virheistä 77 %. Poikkeuksellisesti etukenttään verrattuna 10 %:n suuruisista virheistä havaittiin vain 67 %, mikä johtuu todennäköisesti tehtyjen virheiden osumisesta toiseen klusteriin.



Kuva 5.21: Etukentän monitoriyksiköiden suhteellisten virheiden havaitseminen.



Kuva 5.22: Takakentän monitoriyksiköiden suhteellisten virheiden havaitseminen.

Säteilyturvakeskus (STUK) [2] luokittelee 5–25 % poikkeaman suunnitellusta kokonaisannoksesta lieväksi haittavaikutukseksi ja yli 25 %:n poikkeaman vakavaksi haittavaikutukseksi. Tehdään seuraavaksi arvio näiden poikkeavuuksien havaitsemisesta fotonikentille.

Suunnitellusta 50 Gy:n kokonaisannoksesta etukentän osuus on noin 65 %, jolloin noin 8 %:n suhteellinen virhe etukentän monitoriyksiköissä aiheuttaa 5 %:n poikkeaman suunnitellusta kokonaisannoksesta. Etukentän 8 %:n suhteellisista virheistä pystytään havaitsemaan noin 92 %, joten kaikista lievän haittavaikutuksen aiheuttavista etukentän monitoriyksiköiden poikkeamista pystytään havaitsemaan vähintään 92 %. Vastaavasti etukentän monitoriyksiköiden noin 38 %:n suhteellinen virhe aiheuttaa 25 %:n poikkeaman annoksessa, joten kaikki etukentän monitoriyksiköistä johtuvat vakavaksi haittavaikutukseksi luokitellut poikkeamat pystytään havaitsemaan.

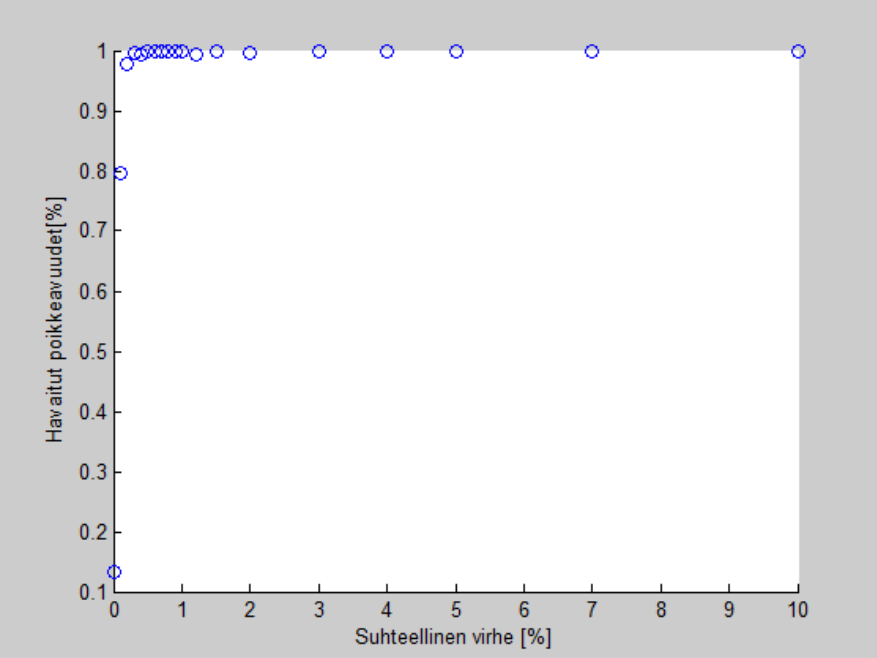
Takakentän osuus suunnitellusta kokonaisannoksesta on noin 35 %. Noin 14 %:n suhteellinen virhe takakentän monitoriyksiköissä vastaa 5 %:n poikkeamaa suunnitellusta annoksesta. Takakentän monitoriyksiköiden 14 %:n suhteellisista virheistä pystytään havaitsemaan 85 %, joten lievistä haittavaikutuksista aiheutuvista poikkeamista havaitaan vähintään 85 %. Takakentän monitoriyksiköiden noin 71 %:n suhteellinen virhe aiheuttaa 25 %:n poikkeaman annoksessa, joten kaikki takakentän monitoriyksiköistä johtuvat vakavaksi haittavaikutukseksi luokitellut poikkeamat pystytään havaitsemaan.

5.10.2 Mallin testaus elektronikenttien keinotekoisilla virheillä

Kuvassa 5.23 on esitettynä elektronikentille tehdyn mallin havaitsemat poikkeavuudet monitoriyksiköiden suhteellisille virheille. Lähes kaikki 0,3 %:n virhettä suuremmat poikkeamat havaittiin. Datan diskreettisyydestä johtuen elektronikenttien monitoriyksiköiden virheet pystytään havaitsemaan tarkasti. Elektronikenttien monitoriyksiköiden virheistä pystytään havaitsemaan kaikki lieviksi ja vakaviksi haittavaikutuksiksi luokiteltavat virheet.

Elektronikentän säteilyenergioille tehdyistä virheistä havaittiin 95 %. Kaikki 15 MeV:n, 16 MeV:n ja 20 MeV:n energiavirheet pystyttiin havaitsemaan. Hoitokoneen L2 kaikki energiavirheet pystyttiin havaitsemaan. Havaitsemattomista virheistä 65 % selittyi sillä, että hoitokoneella L4 hoidetuissa 6 MeV:n, 9 MeV:n ja 12 MeV:n elektronikentissä oli ollut 20×20 cm:n elektroniapplikaattorille sama 200 MU:n monitoriyksikkölukema. Vastaavasti 13 % selittyi sillä, että hoitokoneen L3 6 MeV:n

ja 9 MeV:n hoidoissa oli ollut sama 200 MU:n monitoriyksikkölukema 20×20 cm:n elektroniapplikaattorilla.



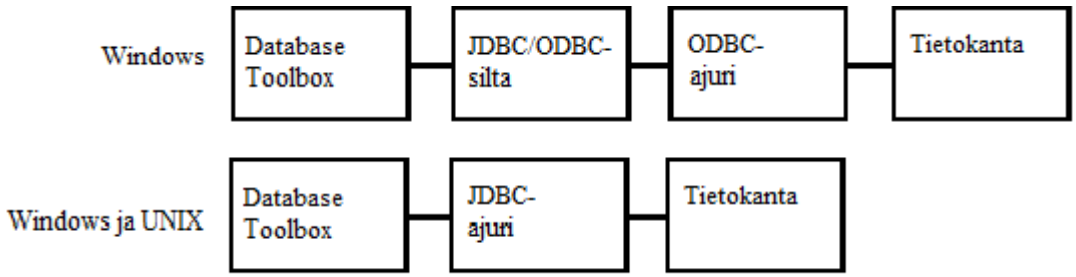
Kuva 5.23: Havaitut poikkeavuudet elektronikenttien monitoriyksiköiden suhteellisille virheille.

5.11 Tietokantayhteyden muodostaminen ja SQL-kyselyn ajaminen Ariaan Matlabilla

Poikkeavuuksien havaitsemistyökalun kytkeminen annossuunnittelujärjestelmän rinnalle vaatii aluksi Matlab-ohjelmiston yhteyden tietokantaan, jotta se voisi tarkistaa uuden annossuunnitelman ja ilmoittaa mahdollisista virheistä käyttäjälle. Lisäksi poikkeavuuksien havaitsemismalli vaatii ylläpitoa. Uudet annossuunnitelmat tulisi lisätä malliin, jotta se pysyisi ajan tasalla. Aria-tietokantapäivityksen myötä Aria 8.9:n tietokanta vapautui syksyllä 2013 tutkimuskäyttöön, jolloin Matlabin kytkeminen tietokantaan pystyttiin testaamaan.

Matlab Database Toolbox [55] on Java-pohjainen ohjelmisto, jonka avulla voidaan muodostaa tietokantayhteys käyttämällä JDBC-ajureita (Java database connectivity) tai ODBC-ajureita (open database connectivity). ODBC-ajuri on Microsoft

Windowsin standardirajapinta, jonka avulla tietokantapalvelin voi kommunikoida SQL-pohjaisen ohjelmiston kanssa. Database Toolboxin ODBC-ajurilla muodostettu tietokantayhteys käyttää JDBC/ODBC-siltaa, joka on automaattisesti asennettuna osana Matlab JVM™ -pakettia. JDBC-ajuri on standardirajapinta, joka mahdollistaa Java-pohjaisen ohjelmiston ja tietokantapalvelimen välisen kommunikoinnin. Kuvassa 5.24 on esitettynä, kuinka ajurit kommunikoivat Database Toolbox -ohjelman kanssa.



Kuva 5.24: JDBC- ja ODBC-ajurit Database Toolbox -ohjelmiston käytössä.

Aria 8.9 on Sybasen tietokanta, johon voidaan muodostaa tietokantayhteys käyttämällä Sybasen JDBC-ajuria. Sybasen tietokannoille JDBC-ajurina voidaan käyttää joko jTDS-ajuria [56] tai Sybasen toteuttamaa jConnect-ajuria [57]. JDBC-ajurina käytettiin jConnect-ajuria nopeampaa jTDS-ajuria, joka on JDBC 3.0 tyypin ajuri.

Matlabilla JDBC-ajureille tietokantayhteys voidaan muodostaa *database*-funktiolla, jolle parametreina syötetään tietokantapalvelimen porttinumero, tietokantayhteden URL, JDBC-ajurin tiedot, käyttäjänimi ja salasana. Aria 8.9 -tietokannan hallintajärjestelmänä toimii Sybase Adaptive Server Enterprise 12 [58], jolle tietokantayhteyden URL on muotoa jdbc:sybase:Tds:*palvelimenIPosoite.* JDBC-ajurin tiedot jTDS:llä on com.sybase.jdbc3.SybDriver.

Arian ASE 12 -palvelin on jaettu kuvadataan liittyvään VISIONBOX8-palvelimeen ja VARISBOX8-palvelimeen. Annossuunnitelmien parametrit löytyvät VARISBOX8-palvelimelta. Tietokantayhteyden URL:iin näin ollen syötetään VARISBOX8-palvelimen IP-osoite. JDBC-ajurin jar-tiedoston polku tulee lisätä Matlabiin kommennolla *javaaddpath* 'polku'.

Kuvassa 5.25 on koodia tietokantayhteyden muodostamisesta ja SQL-kyselyn tekemisestä tietokantaan. *Database*-funktio palauttaa dbConn-tietokantayhteys-objektin,

joka säilyttää tietoa tietokantayhteydestä. Liitteessä C on ajettu SQL-kysely, jolla saatiin haettua sama data kuin InfoMakerilla tehdyssä kyselyssä lukuunottamatta potilaiden henkilötunnuksia. Täysin sama hakutulos saatiin tekemellä erillinen SQL-kysely henkilötunnuksille ja yhdistämällä kyselyiden tulokset PatientSer-kentän tunnusluvulla.

```
% polku käytettävälle jTDS-ajurille
javaaddpath 'polku'

kayttajanimi = '';
salasana = '';

% tietokantayhteyden URL VARISBOX8-palvelimelle
db_URL =  'jdbc:sybase:Tds:VARISBOX8IPosoite';
% palvelimen porttinumero
porttinumero = ':porttinumero';
% jdbc-ajurin tiedot jTDS-ajurille
jdbc_ajuri = 'com.sybase.jdbc3.SybDriver';

% Tietokantayhteyden muodostus
dbConn = database(porttinumero,kayttajanimi,salasana,jdbc_ajuri,db_URL);

% Tarkistetaan onko tietokantayhteys muodostettu
if isconnected(dbConn)

    % SQL-kyselyn luominen
    SQL_query = query.sql;

    % SQL-kyselyn ajaminen
    data = runsqlscript(dbConn,SQL_query);

    % tulostetaan haetta data
    disp(data)

else
    % tulostetaan virheviesti, jos tietokantayhteys ei onnistunut
    dbConn.Message
end

% suljetaan tietokantayhteys
close(dbConn);
```

Kuva 5.24: Tietokantayhteyden muodostaminen ja SQL-kyselyn ajaminen Aria-tietokantaan Matlabilla.

6 JOHTOPÄÄTÖKSET

Tutkimuksen lähtökohtana oli Oulun yliopistollisen sairaalan tarve annossuunnitelmien tarkastustyökalulle, jolla pystyttäisiin tarkistamaan annossuunnitelmien parametrit annossuunnittelujärjestelmästä riippumattomalla menetelmällä. Ohjelman tarkoituksena on toimia annossuunnitelmia tarkastavan fyysikon apuvälineenä, joka hälyttää poikkeavista hoitoparametreista.

Työssä tutkittiin neuroverkkojen soveltuvuutta sädehoidon annossuunnitelmien poikkeavuuksien havaitsemiseen laajasta sädehoidon Aria-tietokannasta. Tavoitteena oli hakea tietokannasta aikaisemmin toteutettujen rinnanpoiston jälkeisten sädehoitojen annossuunnitelmien parametrit ja muodostaa haetuista parametreista neuroverkoilla poikkeavuuksien havaitsemismalli. Tarkoituksena oli muodostaa malli monitoriyksiköiden ja elektronikentän säteilyenergioiden poikkeavuuksien havaitsemiseen. Lisäksi työssä muodostettiin tietokantayhteys Matlabilla Aria-tietokantaan.

Mallin muodostaminen neuroverkoilla onnistui hyvin. Mallilla pystyttiin havaitsemaan kaikki vakavaa haittavaikutusta aiheuttavat poikkeamat yksittäisten kenttien monitoriyksiköissä. Lisäksi lievää haittavaikutusta aiheuttavista poikkeamista yksittäisten kenttien monitoriyksiköissä havaittiin vähintään 85 %. Elektronikenttien säteilyenergioiden virheistä onnistuttiin havaitsemaan 95 %. Lisäksi tietokantayhteyden muodostaminen Matlab-ohjelmistolla onnistui, ja sillä saatiin myös haettua samat annossuunnitelmien parametrit kuin InfoMakerilla tehdyssä tietokantahaussa. Tutkimuksen perusteella näyttää mahdolliselta, että Matlab-pohjaisen annossuunnitelmien tarkastustyökalun kytkeminen annossuunnittelujärjestelmän rinnalle onnistuu ja se pystyy lisäämään sädehoidon turvallisuutta entisestään.

SOM- ja PNN-neuroverkot soveltuivat hyvin poikkeavuuksien havaitsemiseen. Ristiinvalidoinnissa määritettiin mallien tarkkuus, *TPR* ja *FPR*. Fotonikentille muodostetussa mallissa tarkkuudeksi saatiin $0{,}938 \pm 0{,}005$ ja elektronikentille $0{,}952 \pm 0{,}006$. Fotonikentille muodostetun mallin $TPR = 0{,}965 \pm 0{,}003$ ja $FPR = 0{,}100 \pm 0{,}008$. Elektronikentille muodostetun mallin $TPR = 0{,}91 \pm 0{,}02$ ja $FPR = 0{,}041 \pm 0{,}006$.

Naqan [22] tutkimuksessa ristiinvalidoinnilla tukivektorikoneen opetuksessa tarkkuudeksi saatiin 84 %, kun 10 % opetuksen datavektoreista merkittiin poikkeaviksi. Tässä tutkimuksessa saadut tarkkuudet ovat tämän perusteella olevan paremmat. Tutkimukset eivät kuitenkaan ole täysin rinnastettavissa, koska tulokset riippuvat käytettävästä aineistosta. Voidaan kuitenkin todeta, että käytetylle aineistolle neuroverkoilla saatiin hyvä yleistyskyky.

Sädehoidon annos annetaan potilaalle monitoriyksiköissä, minkä vuoksi monitoriyksiköiden tarkastaminen ennen potilaan annossuunnitelman toteuttamista on tärkeää. Poikkeavuuksien havaitsemismallit testattiin erisuuruisille monitoriyksiköiden keinotekoisille virheille, koska sädehoidon kannalta on myös hyödyllistä tietää, kuinka suuria virheitä pystytään havaitsemaan. Etukentän monitoriyksiköiden osalta vähintään 15 %:n virheistä havaittiin 100 % ja 5 %:n virheistä 82 %. Takakentän monitoriyksiköiden osalta 20 %:n virheistä havaittiin 98 % ja 5 %:n virheistä 77 %. Fotonikenttien mallilla havaittiin hyvin pienempiäkin virheitä, ja mallin yleistyskyky oli hyvä. Elektronikentän monitoriyksiköiden 0,3 %:n ja sitä suuremmat virheet havaittiin lähes kaikki. Elektronikenttien monitoriyksiköistä pystytään havaitsemaan pienetkin virheet, mikä johtuu datan diskreettisyydestä. Elektronikenttien säteilyenergioiden virheistä onnistuttiin havaitsemaan 95 %, joten myös säteilyenergioiden virheet pystyttiin havaitsemaan hyvin.

Azmandian ym. [21] havaitsivat monitoriyksiköiden 50 %:n suhteellisista virheistä 80 %. Lisäksi lähes kaikki 100 %:n suuruiset virheet havaittiin. *FPR*:ksi saatiin 10 %. Tässä tutkimuksessa neuroverkoilla toteutettu malli antaa Azmandian ym. tutkimukseen nähden paremmat tulokset, mutta heidän käyttämänsä aineisto on haastavampi opettaa.

Aiheesta on mahdollista tehdä useita jatkotutkimuksia ja menetelmää voidaan edelleen kehittää. PNN:n antamaa vastetta voidaan käyttää liittämään havaitulle poikkeavuudelle poikkeavuuden suuruusluokka, mikä olisi ohjelmaa käyttävälle annossuunnitelmia tarkastavalle fyysikolle hyödyllinen tieto. Tämän voi toteuttaa esimerkiksi moniarvologiikkaan perustuvalla sumealla logiikalla. Lisäksi SOM:lla on mahdollista havaita poikkeavuudeksi tunnistetun syötevektorin eniten poikkeavuutta aiheuttava muuttuja laskemalla syötevektorin muuttujien etäisyydet painovektoriin ja

luokittelemalla suurimman etäisyyden antanut muuttuja eniten poikkeavuutta aiheuttavaksi [45]. Muodostettujen mallien päivittämistä uusien annossuunnitelmien parametreilla olisi myös hyödyllistä tutkia.

Poikkeavuuksien havaitsemismallin muodostamista muille perinteisille sädehoidoille ja IMRT-hoidoille olisi mielenkiintoista testata. Menetelmää voidaan laajentaa myös muihin annossuunnitelmaparametreihin, kuten riskielinten saamiin annoksiin, jotka ovat monitoriyksiköiden rinnalla toinen tärkeä annossuunnitelmista tarkastettava parametri. Annossuunnitelmatyyppien luokitteluun voi muodostaa luokittelumallin neuroverkoilla, jolloin annossuunnitelmaan ei tarvitse liittää tyyppitietoa. Tällöin annossuunnitelmia luokittelevan neuroverkon luokittelutulos osoittaisi sen, mitä poikkeavuuksien havaitsemismallia käytettäisiin tarkastettavalle annossuunnitelmalle. Lisäksi annossuunnitelmien tarkastustyökalu olisi hyödyllinen apuväline annossuunnitelmia tarkastavan fyysikon lisäksi muille sädehoitoon osallistuville ammattilaiselle jo annossuunnitelmien tekovaiheessa. Annossuunnitelmista saatava arvio sen parametrien järkevyydestä voisi hyödyttää esimerkiksi annossuunnittelijoita ja lääkäreitä.

KIRJALLISUUS

[1] O. Pukkila (toim.), *Säteily- ja ydinturvallisuus 3: Säteilyn käyttö*, Säteilyturvakeskus, Karisto Oy:n kirjapaino, 2004.

[2] STUK, *Sädehoidon turvallisuus*, Ohje ST 2.1, 18.4.2011.

[3] A. T. Azar ja S. A. El-Said, *Probabilistic neural network for breast cancer classification*, Neural Computing and Applications, Vol. 23, Iss. 6, pp. 1737–1751, 2013.

[4] D. P. Nazareth, S. Brunner, M. D. Jones, H. K. Malhotra ja M. Bakhtiari. *Optimization of beam angles for intensity modulated radiation therapy treatment planning using genetic algorithm on a distributed computing platform*, Journal of Medical Physics, Vol. 34, No. 3, pp. 129–132, 2009.

[5] P. W. J. Voet, M. L. P. Dirkx, S. Breedveld ja B. J. M. Heijmen, *Automated generation of IMRT treatment plans for prostate cancer patients with metal hip prostheses: Comparison of different planning strategies*, Medical Physics, Vol. 40, 2013.

[6] H. Joensuu, M. Kouri, A. Ojala, M. Tenhunen ja L. Teppo, *Kliininen sädehoito*, Duodecim, 2002.

[7] T. K. Ikäheimonen (toim.), *Säteily- ja ydinturvallisuus 1: Säteily ja sen havaitseminen*, Säteilyturvakeskus, Karisto Oy:n kirjapaino, 2002.

[8] M. Tenhunen, *Sädehoidon fysiikka ja tekniikka*, luentomoniste, Helsinki, 2007.

[9] Varian Medical Systems, http://www.varian.com/us/oncology/radiation_oncology/clinac/clinac_ix.html#.UW0CPrWeN7I, viitattu elokuussa 2013.

[10] F. M. Khan, *The physics of radiation therapy*, 3. painos, Lippincott willams & wilkins, 2003.

[11] G. Mariscal, Ó. Marbán ja C. Fernández, *A survey of data mining and knowledge discovery process models and methodologies*, The Knowledge Engineering Review, Vol. 25:2, pp. 137–166, 2010.

[12] M. Venkatadri ja L. C. Reddy, *A Review on Data mining from Past to the Future*, Vol 15, No. 7, pp. 19–22, 2011.

[13] V. Chandola, A. Banerjee ja V. Kumar, *Anomaly Detection: A Survey*, ACM Computing Surveys, Vol. 41, Iss. 3, July 2009.

[14] V. J. Hodge ja J. Austin, *A Survey of Outlier Detection Methodologies*, Artificial Intelligence Review, 22, pp. 85–126, 2004.

[15] F. E. Grubbs, *Procedures for Detecting Outlying Observations in Samples*, Technometrics, Vol. 11, No. 1, pp. 1–21,1969.

[16] V. Barnett ja T. Lewis, *Outliers in Statistical Data, Second Edition*, John Wiley & Sons, 1984.

[17] A. K. Jain ja R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1988.

[18] M. Markou ja S. Singh, *Novelty detection: a review – part 2: neural network based approaches*, Signal Prosessing, Vol. 83, pp. 2499–2521, 2003.

[19] R. A. Maxion ja R. R. Roberts, *Proper Use of ROC Curves in Intrusion/Anomaly Detection*, Technical Report Series CS-TR-871, 2004.

[20] R. Kohavi, *A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection*, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1995.

[21] F. Azmandian, D. Kaeli, J. G. Dy, E. Hutchinson, M. Ancukiewicz, A. Niemierko and S. B. Jiang, *Towards the development of an error checker for radiotherapy treatment plans: a preliminary study*, Physics in Medicine and Biology, Vol. 52, pp. 6511–6524, 2007.

[22] I. E. Naqa, SU-E-J-69: *An Anomaly Detector for Radiotherapy Quality Assurance Using Machine Learning*, Medical Physics, Vol. 38, pp. 3458, 2011.

[23] http://projectreporter.nih.gov/project_info_description.cfm?icde=0&aid=8250930, viitattu maaliskuussa 2013.

[24] S. Haykin, *Neural Networks a comprehensive foundation*, Macmillan College Publishing Company, 1994.

[25] E. Sánchez-Sinencio ja C. Lau, *Artificial Neural Networks: Paradigms, Applications and hardware implementations*, Institute of Electrical and Electronics Engineers, 1992.

[26] J. W. Kay ja D. M. Titterington, *Statistics and Neural Networks, Advances at the Interface*, Oxford University Press, 1999.

[27] D. F. Specht, *Probabilistic Neural Networks*, Neural Networks, Vol. 3, pp.109-118, 1990.

[28] I. M. M. El Emary and S. Ramakrishnan, *On the Application of Various Probabilistic Neural Networks in Solving Different Pattern Classification Problems*, World Applied Sciences Journal, Vol. 4 (6), pp. 772–780, 2008.

[29] J. Tohka, *SGN-2500: Johdatus hahmontunnistukseen*, luentomoniste, Tampeereen teknillinen yliopisto, Signaalinkäsittelyn laitos, 2012.

[30] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[31] S. Bastke, M. Deml ja S. Schmidt, *Combining statistical network data, probabilistic neural networks and the computational power of GPUs for anomaly detection in computer networks*, 1st Workshop on Intelligent Security ,Security and Artificial Intelligence, 2009.

[32] T. Kohonen, *The self-organizing map*, Neurocomputing, Vol. 21, pp. 1–6, 1998.

[33] T. Kohonen, *Essentials of the self-organizing map*, Neural Networks, Vol. 37, pp. 52–65, 2013.

[34] T. Kohonen, *Self-Organising Maps*, 2. Painos, Springer, 1997.

[35] J. Vesanto, *Data Exploration Process Based on the Self-Organizing Map*, Väitöskirja, Helsingin yliopisto, 2002.

[36] J. Vesanto, J. Himberg, E. Alhoniemi ja J. Parhankangas, *SOM Toolbox for Matlab 5*, Report A57, Libella Oy, 2000, saatavana osoitteessa: http://www.cis.hut.fi/somtoolbox/package/papers/techrep.pdf, viitattu 2013 syyskuu.

[37] J. Vesanto, J. Himberg, E. Alhoniemi ja J. Parhankangas, *Self-organizing map in Matlab: the SOM Toolbox*, Proceedings of the Matlab DSP Conference 1999, pp. 35–40, 1999.

[38] M. Peña, W. Barbakh ja C. Fyfe, *Topology-Preserving Mappings for Data Visualisation*, Lecture Notes in Computational Science and Engineering, Vol. 58, pp. 131–150, 2008.

[39] A. Patcha ja J. M. Park, *An overview of anomaly detection techniques: Existing solutions and latest technological trends*, Computer Networks, Vol. 51 , pp. 3448–3470, 2007.

[40] C. M. Bishop, *Novelty detection & Neural Network validation*, IEE Proceedings – Vision, Image and Signal Processing, Vol. 141, No. 4, 1994.

[41] A. L. I. Oliveira, F. R. G. Costa ja C. O. S. Filho, *Novelty detection with constructive probabilistic neural networks*, Neurocomputing, Vol. 71 , pp. 1046–1053, 2008.

[42] R. E. Shaffer and S. L. Rose-Pehrsson, *Improved Probabilistic Neural Network Algorithm for Chemical Sensor Array Pattern Recognition*, Analytical Chemistry, Vol. 71, pp. 4263–4271, 1999.

[43] R. E. Shaffer, *Chemical Sensor Pattern Recognition System and Method Using a Self-Training Neural Network Classifier with Automated Outlier Detection*, Patentti, Patent No.: US 6,289,328 B2, 2001.

[44] J. A. Dominguez ja S. Klinko, *Visual Anomaly Detection via Soft Computing:A Prototype Application at NASA*, Mathware & Soft Computing, Vol. 10, pp. 131–139, 2003.

[45] A. J. Höglund, K. Hätönen ja A. S. Sorvari, *A computer host-based user anomaly detection system using the self-organizing map*, Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, Vol. 5, pp. 411–416, 2000.

[46] M. Ramadas, S. Ostermann ja B. Tjaden, *Detecting anomalous network traffic with self-organizing maps*, Recent Advances in Intrusion Detection, pp. 36–54, 2003.

[47] P. Kumpulainen ja K. Hätönen, *Anomaly Detection Algorithm Test Bench for Mobile Network Management*, Tampereen teknillinen yliopisto, 2008.

[48] Y. Cao, H. He, H. Man, X. Shen, *Integration of Self-organizing Map (SOM) and Kernel Density Estimation (KDE) for Network Intrusion Detection*, Proceedings of the SPIE, Vol. 7480, pp. 12 , 2009.

[49] D. Fustes, C. Dafonte, B. Arcay, M. Manteiga, K. Smith, A. Vallenari ja X. Luri, *SOM ensemble for unsupervised outlier analysis. Application to outlierv identification in the Gaia astronomical survey*, Expert Systems with Applications, Vol. 40, pp. 1530–1541, 2013.

[50] A. Muñoz and J. Muruzábal. *Self-organising maps for outlier detection*, Neurocomputing, Vol. 18, pp. 33–60, 1998.

[51] F. A. González ja D. Dasgupta, *Anomaly Detection Using Real-Valued Negative Selection*, Genetic Programming Evolvable Machines, Vol. 4, pp. 383–403, 2003.

[52] Varian Medical Systems, *The Database Reference Guide* (P/N 100034991-

02), https://myvarian.com, viitattu lokakuussa 2012.

[53] J. J. P. Tsai ja P. S. Yu, *Machine Learning in Cyber Trust Security, Privacy, and Reliability*, Springer, 2009.

[54] Varian Medical Systems, *ARIA Reports Schema* (P/N 100052309-01), https://myvarian.com, viitattu lokakuussa 2012.

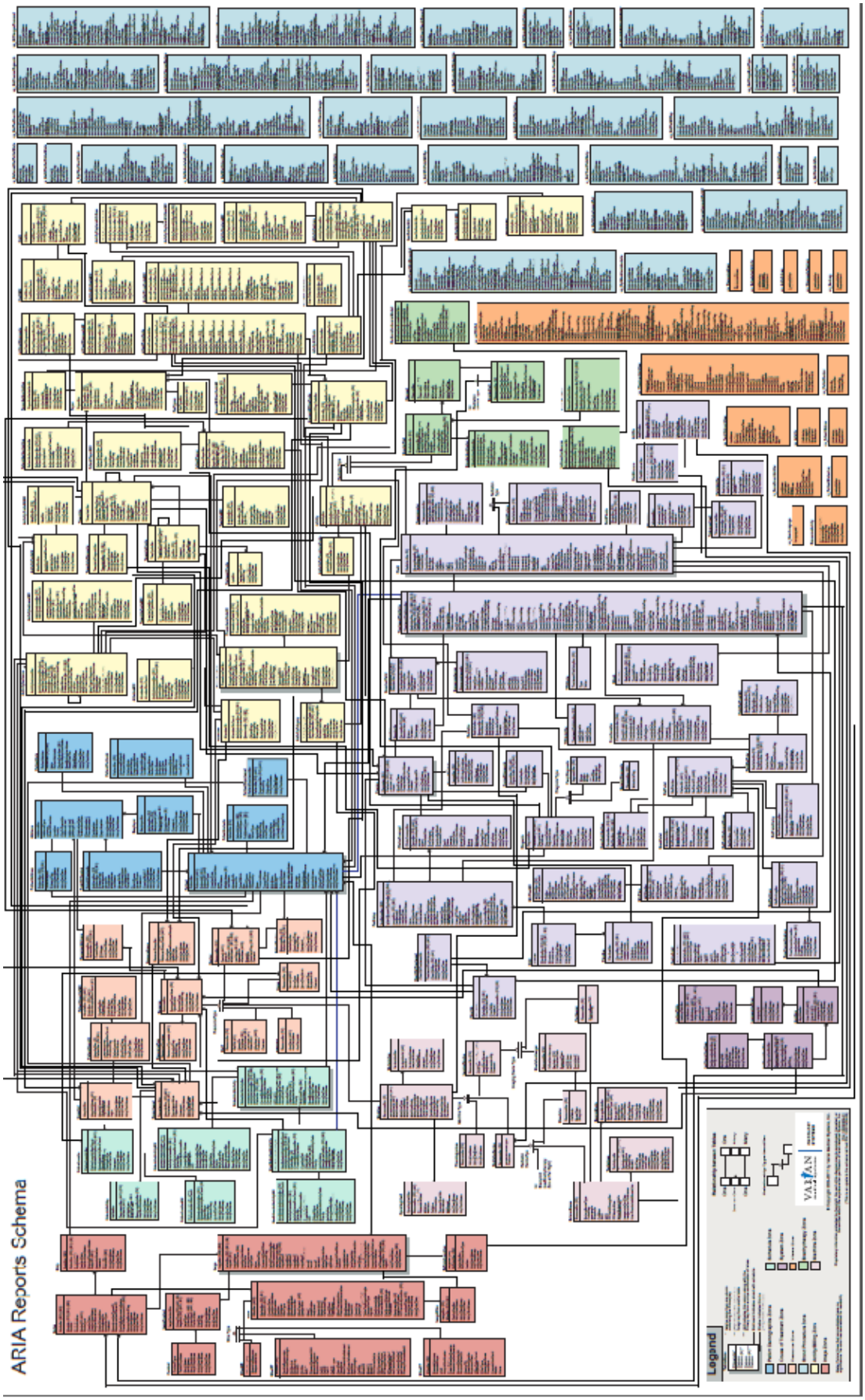
[55] The MathWorks Inc, Matlab-ohjelmiston dokumentaatio, http://www.mathworks.se/help, 1994–2013, viitattu marraskuussa 2013.

[56] The jTDS project, http://jtds.sourceforge.net/, viitattu marraskuussa 2013.

[57] Sybase, http://www.sybase.com/products/allproductsa-z/softwaredeveloperkit/jconnect, viitattu marraskuussa 2013.

[58] Sybase, http://www.sybase.com/products/databasemanagement/adaptiveserverenterprise, viitattu marraskuussa 2013.

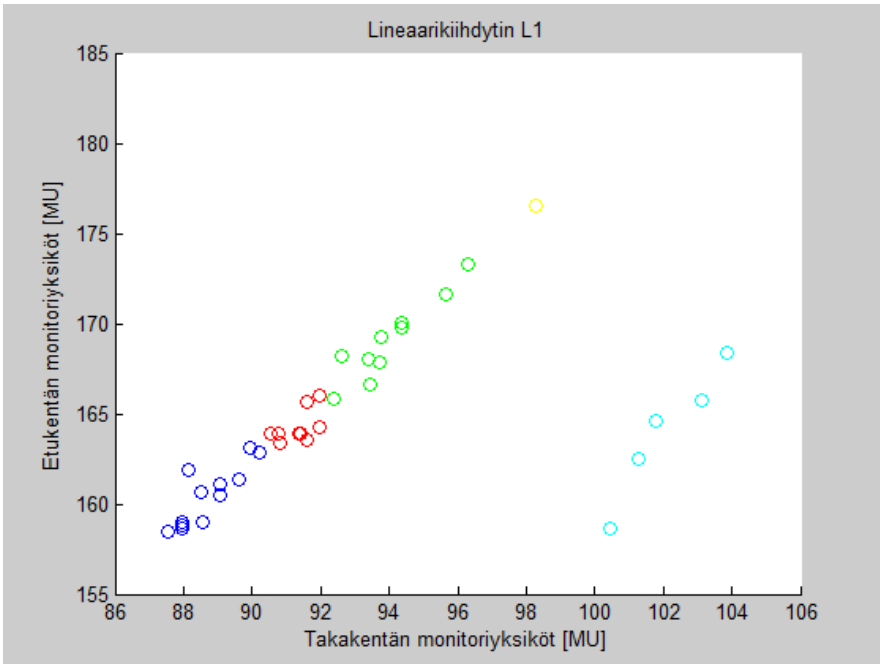
LIITE A: Aria 8.9 -potilastietokannan taulut

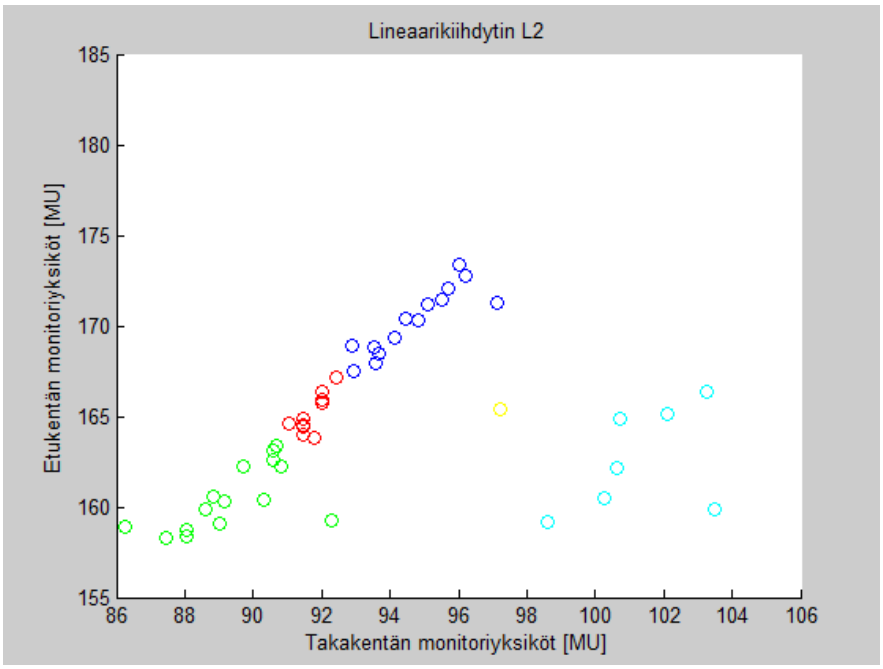


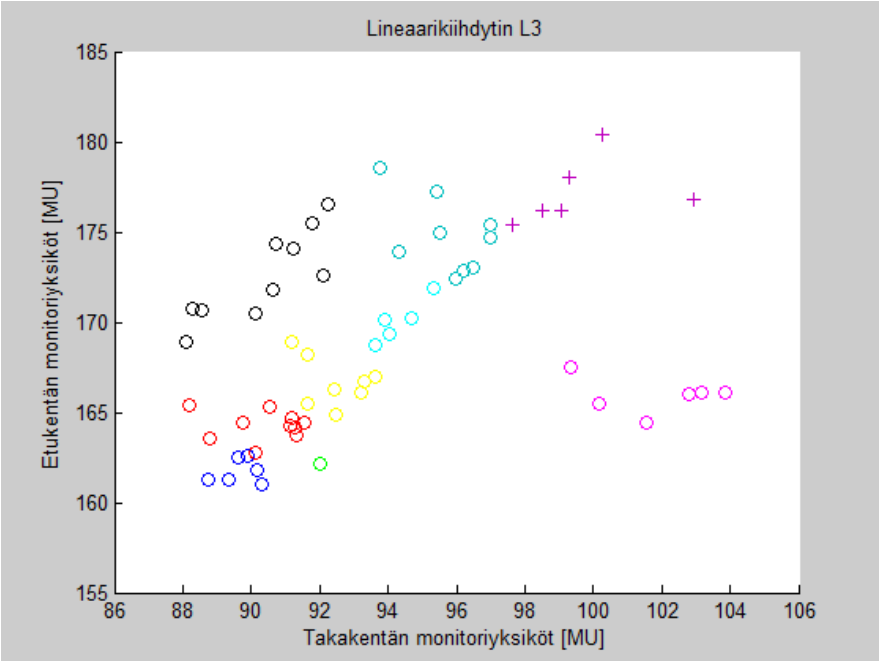
LIITE B: Matlabilla tehty SQL-kysely

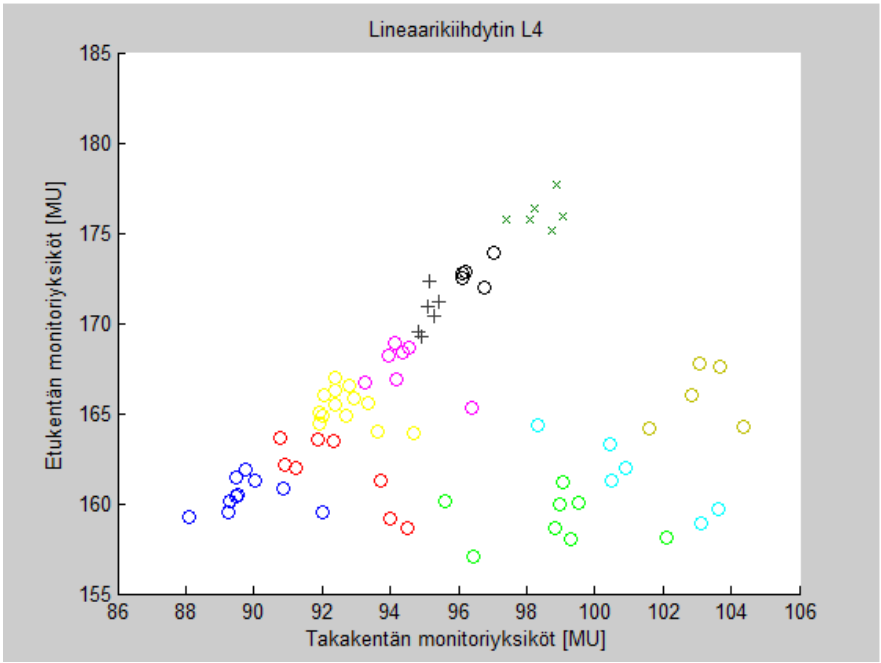
```
SELECT DISTINCT   Course.PatientSer,
      PlanSetup.PlanSetupSer,
      FieldRefPoint.FieldDose,
      Field.MUpGy,
      Radiation.ResourceSer,
      Field.EnergyModeSer,
      AddOn.AddOnType,
      AddOn.AddOnId,
      Course.CourseSer,
      PlanSetup.PlanSetupId,
      Field.GantryRtn,
      Radiation.RadiationId
  FROM Course,
      PlanSetup,
      Radiation,
      FieldRefPoint,
      Field,
      FieldAddOn,
      AddOn
 WHERE ( Course.CourseSer = PlanSetup.CourseSer ) and
      ( PlanSetup.PlanSetupSer = Radiation.PlanSetupSer ) and
      ( Radiation.RadiationSer = FieldRefPoint.RadiationSer ) and
      ( FieldRefPoint.RadiationSer = Field.RadiationSer ) and
      ( Field.RadiationSer = FieldAddOn.RadiationSer ) and
      ( FieldAddOn.AddOnSer = AddOn.AddOnSer ) and
      ( ( Course.StartDateTime > '7/1/2011 00:00:00' ) AND
      ( Course.StartDateTime <= '3/19/2013 23:59:59' ) AND
      ( Course.Comment like '%HA013%' ) AND
      ( PlanSetup.Status like 'TreatApproval' ) AND
      ( Field.SetupField = 0 ) AND
      ( FieldRefPoint.DoseSpecificationFlag = 1 ) )
ORDER BY PlanSetup.PlanSetupSer ASC
```

LIITE C: Fotonikenttien klusterit






Lineaarikiihdytin L3
Etukentän monitoriyksiköt [MU]
Takakentän monitoriyksiköt [MU]
185
180
175
170
165
160
155
86
88
90
92
94
96
98
100
102
104
106


Lineaarikiihdytin L4
Etukentän monitoriyksiköt [MU]
Takakentän monitoriyksiköt [MU]
185
180
175
170
165
160
155
86
88
90
92
94
96
98
100
102
104
106