Koneoppiminen röntgenramansirontakokeen data-analyysissä

Pro gradu –tutkielma
Turun yliopisto
Fysiikan ja tähtitieteen laitos
Fysiikka
2020
Kimmo Pyyhtiä
Tarkastajat:
Johannes Niskanen
Edwin Kukk

niille jotka ovat kärsineet COVID-19:sta

TURUN YLIOPISTO
Fysiikan ja tähtitieteen laitos
PYYHTIÄ, KIMMO TAPIO Koneoppiminen röntgenramansirontakokeen data-analyysissä
Pro gradu –tutkielma, 71 sivua, 25 liitetiedostoa
Fysiikka
Kesäkuu 2020

Röntgenramansirontaspektroskopia (XRS) on energisiin fotoneihen perustuva spektroskopian laji, jossa röntgensäteet siroavat näytteestä epäelastisesti. Siroava fotoni menettää energiaa virittäen atomin tai molekyylin kuorielektronin korkeampaan energiatilaan. Energiahäviöitä mittaamalla muodostetaan röntgenabsorptiospektriä vastaava spektri. Osa fotoneista siroaa kuitenkin näyteympäristöstä eikä näytteestä muodostaen taustasäteilyä. Perinteisesti tutkijan pitää luokitella näytteestä peräisin olevat spektrit käsin.

Tässä tutkielmassa olen käyttänyt koneoppimismenetelmiä jotta tutkijan tarvitsisi luokitella vain murto-osa kaikista spektreistä käsin antaen koneoppimismallin luokitella loput. Käytetty röntgenspektrometridata on peräisin European Synchrotron Radiation Facilitystä (ESRF), jonka varastorenkaan ja röntgenramanspektrometrin toimintaa käsittelen teoreetisesti. Olen käyttänyt kahta koneoppimismallia, tukivektorikonetta ja neuroverkkoa, näytteestä sironneiden ja taustasta sironneiden spektrien luokittelemiseksi. Kumpikin malli opetetaan luokittelemaan spektrejä koulutusjoukolla, joka koostuu käsin luokitelluista spektreistä. Lisäksi käsittelen molempien mallien teoreettista toimintaperiaatetta.

Teen päätelmän kokeellisiin tuloksiin perustuen, että sekä tukivektorikone että neuroverkko kykenevät tehokkaaseen spektrien luokitteluun vetäen vertoja ihmisen luokittelukyvylle. Molempien suorituskyky on hyvin samankaltainen mutta erityisesti tukivektorikone erottuu neuroverkosta sen helpon käytettävyyden, nopean laskennan ja yksinkertaisen toimintaperiaatteen vuoksi. Neuroverkko on tässä tapauksessa paljon raskaampi menetelmä, jonka suorituskyky ei ole tukivektorikonetta parempi.

Asiasanat: röntgensironta, röntgenramanspektroskopia, neuroverkko, tukivektorikone, ESRF, XRS

Sisältö

Johdanto

Atomit ja molekyylit koostuvat protoneista ja neutoroneista, jotka muodostavat atomiytimet, ja niitä kiertävistä elektroneista. Elektronit asettuvat orbitaaleille atomiytimien ympärille. Jokaisella orbitaalilla on sille ominainen energia, ja elektroni voidaan virittää korkeammalle orbitaalille antamalla sille energiaa. Näitä elektronien siirtymiä orbitaaleilta toisille voidaan tutkia spektroskopian keinoin.

Röntgenramansirontaspektroskopia perustuu energisten röntgensäteiden epäelastiseen sirontaan sisäkuoren elektroneista. Fotonit luovuttavat osan energiastaan elektroneille virittäen niitä korkeampaan energiatilaan. Fotonin energiahäviö vastaa elektronin saamaa energiaa eli kahden orbitaalin välisen siirtymän energiaa. Tätä energiahäviötä havainnoimalla saadaan spektri siirtymien todennäköisyyksistä niiden energian funktiona. Kaikki fotonit eivät kuitenkaan siroa näyteaineesta vaan näytealue, kuten ilma, tuottavat haitallista taustaa, joka pitää suodattaa pois. Tavallisesti tutkija valikoi käsin taustan spektrit pois, mutta tämä on aikaavievää ja puuduttavaa koska luokiteltavia spektrejä voi olla tuhansia. Tässä tutkielmassa spektrien luokittelu on pyritty automatisoimaan koneoppimismenetelmin, siten että tutkijan täytyisi luokitella vain murto-osa kaikista spektreistä itse ja antaa koneoppimismallin luokitella loput.

Työssä käytän tukivektorikonetta ja neuroverkkoa. Tukivektorikone perustuu avaruudellisesti kahden eri luokan erottelemiseen hypertason avulla. Neuroverkko pohjautuu aivojen biologisten hermoverkkojen toiminnalle. Kuvaan molempien mallien toimintaa teoreettisesti. Lisäksi avaan käytetyn datan esikäsittelyä.

1 Aineen rakenne

Hiukkasfysiikan standardimalli kuvaa suurinta osaa aineen tunnetuista alkeishiukkasista sekä kolmea neljästä maailmankaikkeuden perusvuorovaikutuksesta, heikkoa, vahvaa ja sähkömagneettista vuorovaikutusta. Alkeishiukkaset jaetaan kahteen luokkaan niiden spinien perusteella [1]; spin on hiukkasen sisäinen ominaisuus jota verrataan usein pyörimismäärään, vaikka tämä ei todellisuudessa pidä kokonaan paikkaansa. Hiukkaset joiden spinit ovat kokonaislukuja ovat bosoneita [1, 2], jotka ovat vuorovaikutusten välittäjähiukkasia [1, 2]. Hiukkaset joiden spinit ovat puolilukuja, ovat fermioneja, jotka muodostavat aineen rakennusosat [1, 2].

Fermionit jakautuvat kahteen luokkaan, kvarkkeihin ja leptoneihin. Kvarkit vuorovaikuttavat vahvan vuorovaikutuksen kautta keskenään sillä kvarkeilla on värivarausta, toisin kuin leptoneilla jotka eivät ole värivarautuneita eivätkä siten vuorovaikuta vahvan vuorovaikutuksen kautta. Leptonit vuorovaikuttavat heikon vuorovaikuksen välityksellä. Tärkeimmät kvarkit ovat ylös- ja alas-kvarkki (*engl. up, down, u, d*), jotka muodostavat protonit ja neutronit. Ylös-kvarkkien varaus on $q = \frac{2}{3}e$, jossa e on alkeisvaraus ($1.60217662 \cdot 10^{-19}$ C) ja alas-kvarkkien varaus on $q = -\frac{1}{3}e$. Kaksi ylös-kvarkkia ja yksi alas-kvarkki muodostavat protonin, jonka varaus on $\frac{2}{3}e + \frac{2}{3}e - \frac{1}{3}e = 1e$. Vastaavasti kaksi alas-kvarkkia ja yksi ylös-kvarkki muodostavat varauksettoman neutronin $-\frac{1}{3}e - \frac{1}{3}e + \frac{2}{3}e = 0e$. Muut kvarkit ja niiden yhdistelmät voivat muodostaa omia hiukkasiaan, mutta ne ovat hyvin lyhytikäisiä [2].

Leptoneista vain elektronit ja neutriinot ovat vakaita. Elektronit, joiden varaus on $-1e$, ovat yksi atomien rakentamiseen tarvittavista hiukkasista atomiytimien protonien ja neutronien lisäksi. Kaikista atomeista yksinkertaisin, vetyatomi, koostuu positiivisesti sähkövarautuneesta protonista, jota ”kiertää” negatiivisesti sähkövarautunut elektroni. Atomin sähkövaraus onkin nolla [1]. Ihmisten toimintaympäristössä atomien tutkimus on tärkeää sillä niitä tarkastelemalla voidaan selittää niin

kemiaa, materiaalien ominaisuuksia kuin jopa elävien olentojen toimintaa. Aineen atomiskaalan kuvaukseen sopiva teoria on kvanttimekaniikka.

Neutraalilla atomilla on yhtä monta elektronia kuin sillä on protoneita. Richard M. Martin on kirjassaan "Electronic Structure: Basic Theory and Practical Methods" [3] maininnut, että kvanttimekaniikan yksi merkittävimmistä löydöistä on Paulin kieltosääntö, jonka mukaan kaksi elektronia (tai muuta fermioinia) eivät voi olla täysin samassa kvanttitilassa. Elektronien kvanttitilaa kuvataan eri kvanttilukujen avulla.

Kvanttiluvut saavat alkunsa Schrödingerin yhtälöstä [4].

$$i\hbar\frac{\partial\Psi}{\partial t} = \hat{H}\Psi, \tag{1}$$

missä $\hbar$ on redusoitu Planckin vakio, Ψ on aaltofunktio, t on aika ja $\hat{H}$ on Hamiltonin operaattori

$$\hat{H} = T + V = -\frac{\hbar^2}{2m}\nabla^2 + V, \tag{2}$$

missä m on elektronin massa. Aaltofunktio Ψ ja potentiaalienergia V ovat molemmat paikan $\mathbf{r} = (x, y, z)$ ja ajan t funktioita. Mikäli potentiaali ei muutu ajan myötä, systeemin stationääriset tilat voidaan ratkaista ajasta riippumattomasta Schrödingerin yhtälöstä

$$-\frac{\hbar^2}{2m}\nabla^2\psi + V\psi = E\psi, \tag{3}$$

jossa ψ on avaruusaaltofunktio, joka riippuu vain paikasta $\mathbf{r}$ [4]. Tämän yhtälön ratkaisut määrittävät kolme eri kvanttilukua n, l, m_l ja asettavat niiden arvoille rajoitteita. Kvanttiluvuista vain n ja l vaikuttavat tilan energiaan [4].

Atomeissa elektronien kvanttitiloja kuvaamaan tarvitaan yhteensä neljä kvanttilukua, edelliset kolme kvanttilukua ja spinkvanttiluku [5]. Pääkvanttiluku n määrää

kyseisen energiakuoren energian. Tällöin yhdellä pääkvanttiluvun määräämällä orbitaalilla voi olla monta elektronia, joilla on sama energia mutta eroavat muut kvanttiluvut. Sivukvanttiluku l edustaa elektronin kulmaliikemäärä ($L^2 = l(l+1)\hbar^2$). Magneettinen kvanttiluku m_l kuvaa eletronien energiaa magneettikentässä ja määrää kuinka monta elektronia voi olla yhdellä n ja l kvanttilukujen muodostamalla alikuorella.

Taulukko 1: Elektronin kvanttiluvut atomissa [5].

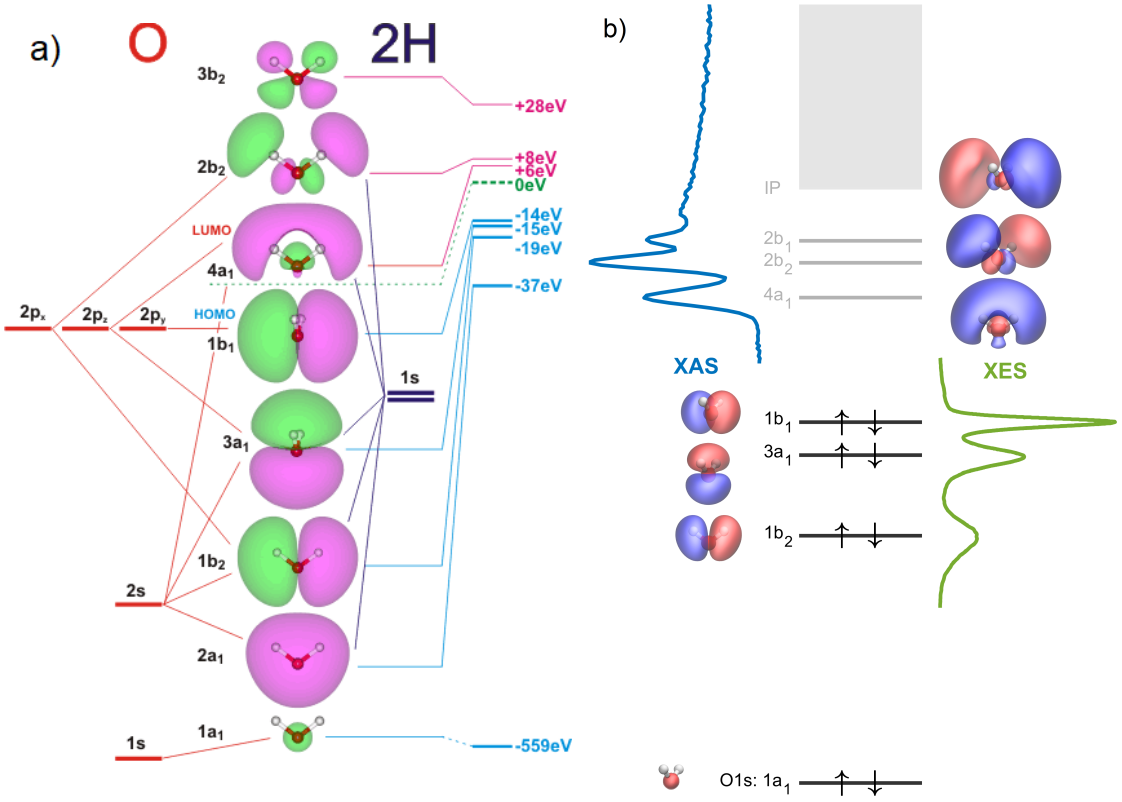
Nimi	**Tunnus**	**Kuvaus**	**Arvot**
Pääkvanttiluku	n	Orbitaalin koko ja energia	n=1,2,3...
Sivukvanttiluku	l	Orbitaalin muoto	$0 \leq l \leq n-1$
Magneettinen kvanttiluku	m_l	Elektronien energia magneettikentässä	$-l \leq m \leq l$
Spinkvanttiluku	s	Spin	$-\frac{1}{2}, \frac{1}{2}$

D. Griffiths ja D. Schroeter kertovat kirjassaan "Introduction to Quantum Mechanics" neljännen kvanttiluvun, spinin s, löytyneen Stern-Gerlachin kokeessa. Kokeessa havaittiin sähköisesti neutraalien hopea-atomien lentoradan taipuvan magneettikentässä kahdeksi erilliseksi partikkelisäteeksi. Tämä johtui siitä, että hopea-atomin uloimmalla elektronikuorella on vain yksi elektroni. Havaittiin, että tällä elektronilla on sisäinen magneettimomentti, jonka ansiosta hopea-atomien lentorata taipui magneettikentässä. Atomin muiden elektronien vuorovaikutuksen magneettikentän kanssa kumoaa niiden omalla orbitaalillaan oleva pari, jonka spin on vastakkaissuuntainen. Yhdellä kvanttilukujen n, l, m määrämällä orbitaalilla voikin olla kaksi elektronia joilla on vastakkaisuuntaiset spinit.

Griffithsin ja Schroeterin mukaan Bohrin yhtälö on kvanttimekaniikan historian yksi merkittävimmistä kaavoista. Se kuvailee vetyatomin elektronien sallittuja energiatiloja.

$$E_n = -\left[\frac{m}{2\hbar^2}\left(\frac{e^2}{4\pi\varepsilon_0}\right)^2\right]\frac{1}{n^2} = \frac{E_1}{n^2}, \quad n = 1, 2, 3, ... \tag{4}$$

Elektroni on perustilassaan kun $n = 1$, jolloin sen energia $E_1 = -13,6$ eV. Elektroniin pitäisi tehdä vähintään tämän verran työtä, jotta se voitaisiin irroittaa atomista ionisaatiorajan yläpuolella oleville miehittämättömille tiloille. Bohrin yhtälö ei kuitenkaa kuvaa tätä käyttäytymistä. Elektroni voidaan myös nostaa perustilastaan viritettyyn tilaan, esimerkiksi $n = 2$, jolloin sen energia E_2 on $-3,40$ eV.



Kuva 1: a) Vesimolekyylin orbitaalien muodostuminen hapen ja vedyn orbitaaleista. HOMO (*Highest Occupied Molecular Orbital*) -orbitaali on korkein miehitetty molekyyliorbitaali ja LUMO (*Lowest Unoccupied Molecular Orbital*) -orbitaali on alin miehittämätön molekyyliorbitaali. Kuva on lähteestä [6] ja se on julkaistu Creative Commons -lisenssin alla `https://creativecommons.org/licenses/by-nc-nd/2.0/uk/`. b) Siirtymän energiasta ja intensiteetistä syntyy spektri (sininen viiva) kun hapen 1s-orbitaalilta viritetään elektroni miehittämättömille tiloille. Mikäli elektronille annetaan energiaa ylittää ionisaatioraja, se irtoaa molekyylistä ja molekyyli ionisoituu. Jos molekyyli on ionisoitunut, aukkotila voi täyttyä niin että elektroni tippuu miehitetyltä orbitaalilta aukon orbitaalille, jolloin syntyy emissiospektri (vihreä viiva). Kuva on saatu Johannes Niskaselta.

Molekyylit koostuvat useista atomeista. Peter Atkins ja Ronald Friedman kertovat kirjassaan ”Molecular Quantum Mechanics” [7] molekyylien olevan hyvin hanka-

lia kappaleita kuvata matemaattisesti. Niiden kohdalla täytyy tehdä useita olettamuksia, kuten Born-Oppenheimer -approksimaatio, jossa elektronien oletetaan seuraavan ytimien liikettä välittömästi. Toinen laskennan kannalta käytännöllinen approksimaatio on esittää molekyylien orbitaalit atomiorbitaalien lineaarikombinaationa (*engl. linear combinations of atomic orbitals, LCAO*). Tällöin atomiorbitaalien aaltofunktiot muodostavat kantajoukon jotka määrittävät yhdessä molekylaariset orbitaalit

$$\psi = \sum_i c_i \chi_i, \tag{5}$$

jossa ψ on molekylaarinen orbitaali, χ_i on atomiorbitaali ja c_i on kyseistä atomiorbitaalia vastaava kerroin [7]. Mallissa molekyylin orbitaalit muodostuvat siis atomaaristen yksi-elektroniorbitaalien lineaarikombinaatiosta, jonka ratkaiseminen on kuitenkin vaikeaa. Atomiorbitaali on tässä yhteydessä ajateltava atomiin keskittyneenä funktiona, nk. kantafunktiona. Yleensä teoreettisissa tarkasteluissa ei kuitenkaan käytetä ”oikeita” atomiorbitaaleja, vaan gaussin funktioon pohjautuvia funktioita joita on lukumäärältään enemmän kuin atomissa orbitaaleja.

Elektronit ovat Richard Martinin [3] mukaan keskenään korreloituneita ja muodostavat siten keskenään vuorovaikuttavan monen kappaleen systeemin. Elektronien keskenäistä vuorovaikutusta voidaan kuitenkin approksimoida erinäisillä keinoilla, joista yhtenä Martin esittää Hartree–Fock (HF) -menetelmän, jossa tehdään oletus, että elektronit eivät ole korreloituneita paitsi siten että niiden täytyy noudattaa Paulin kieltosääntöä. HF-menetelmässä elektronien välinen Coulombin vuorovaikutus otetaan huomioon niiden energian laskemisessa. Mallissa elektronien liikettä kuvataan toisten elektronien muodostamassa keskimääräisessä sähköstaattisessa kentässä. Martinin mukaan Hartree–Fock-approksimaatiossa monielektroniaaltofunktio Φ kirjoitetaan Slaterin determinantin avulla. Slaterin determinantilla on useita toivottuja ominaisuuksia.

Fermionit ovat antisymmetrisiä hiukkasia, eli kahden fermionin koordinaattien vaihtaminen muuttaa aaltofunktion merkkiä. Slaterin determinantin merkki muuttuu mikäli kaksi pystyriviä tai vaakariviä vaihtavat paikkaa keskenään. Lisäksi kahden vaakarivin ollessa samat determinantti on nolla; Slaterin determinantti toteuttaa Paulin kieltosäännön. Slaterin determinantti N:lle hiukkaselle on

$$\Phi = \frac{1}{\sqrt{N!}} \begin{vmatrix} \phi_1(\mathbf{r}_1, \sigma_1) & \phi_1(\mathbf{r}_2, \sigma_2) & \cdots & \phi_1(\mathbf{r}_N, \sigma_N) \\ \phi_2(\mathbf{r}_1, \sigma_1) & \phi_2(\mathbf{r}_2, \sigma_2) & \cdots & \phi_2(\mathbf{r}_N, \sigma_N) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_N(\mathbf{r}_1, \sigma_1) & \phi_N(\mathbf{r}_2, \sigma_2) & \cdots & \phi_N(\mathbf{r}_N, \sigma_N) \end{vmatrix}. \tag{6}$$

Spin-orbitaaliaaltofunktiot $\phi_i(\mathbf{r}_j, \sigma_j)$ koostuvat elektronin sijaintiosasta $\psi_i^{\sigma}(\mathbf{r}_j)$ ja spinosasta $\alpha_i(\sigma_j)$. Martinin mukaan spin-orbitaalien tulee olla lineaarisesti riippumattomia ja mielellään ortonormaaleja, jolloin tästä seuraavat HF-yhtälöt yksinkertaistuvat huomattavasti. Martinin mukaan HF-yhtälöt pyrkivät minimoimaan kokonaisenergian kaikkien aaltofunktion vapausasteiden suhteen rajoituksella, että yhtälön (1) täytyy edelleen toteutua. Martin kertoo HF-yhtälöiden olevan analyyttisesti ratkaistavissa vain erityistapauksille kuten pallosymmetrisille atomeille tai homogeeniselle elektronikaasulle. Muissa tapauksissa pitää käyttää kantajoukkoja approksimaation etsimiseen.

Edellä kuvailtu Hartree–Fock-teoria ennustaa elektronien olevan ”hyllyillä”, joiden energiat riippuvat atomien ympäristöstä ja sidoksista siihen. Näin ollen atomien välitön ympäristö vaikuttaa elektronitasojen välisiin siirtymiin; niiden energioihin ja todennäköisyyksiin, ks. kuva 1 b). Pommittamalla näytettä fotoneilla ja havaitsemalla siirtymien spektri voidaan saada tietoa aineen rakenteesta. Siirtymiä voidaan tutkia esimerkiksi virittyneiden elektronien viritystilan purkautuessa vapautuvaa säteilyä mittaamalla tai kuten röntgenramanspektroskopiassa, elektroneja virittäneiden fotonien energian ja liikemäärän muutoksia havainnoimalla.

2 Synkrotroni ja spektroskopia

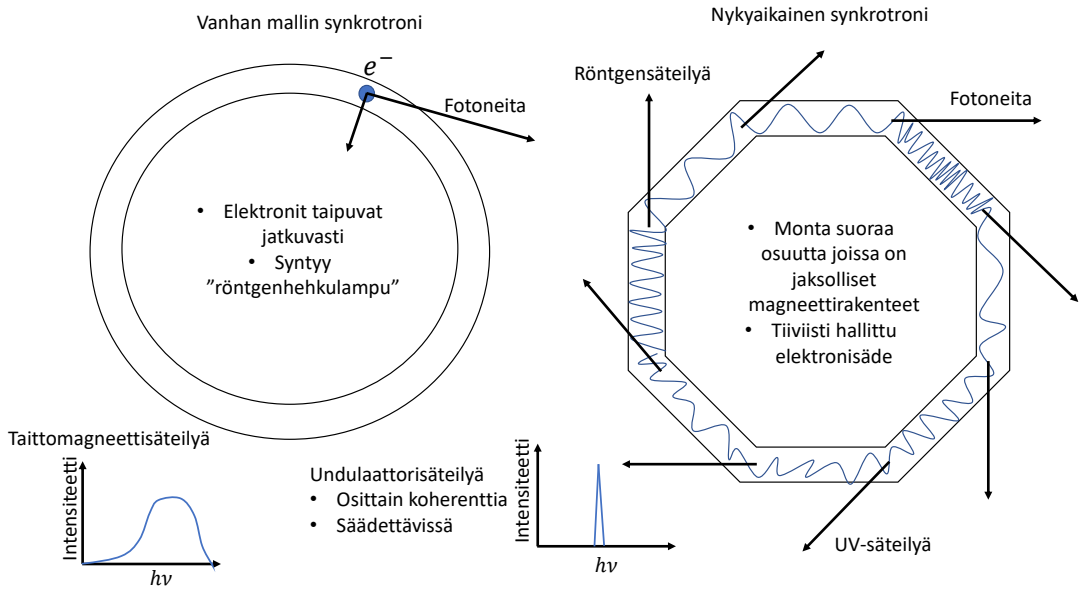
Sisäkuoren elektronien virittäminen korkeammille energiatiloille vaatii korkeaenergisiä fotoneja. Monessa sovelluskohteessa on tarpeellista, että käytetty säteily on monokromaattista ja koherenttia. Lisäksi tällaisen säteilyn intensiteetin tulisi olla varsin suuri. Synkrotronit ovat laitteita, joiden avulla voidaan tuottaa säteilyä jolla on nämä ominaisuudet.

2.1 Synkrotroni säteilylähteenä

Synkrotronin toiminta perustuu kiihtyvän varatun hiukkasen emittoimaan säteilyyn. David Attwood kertoo kirjassaan "Soft X-Rays and Extreme Ultraviolet Radiation: Principles and Applications" [8] synkrotronisäteilyä syntyvän relativistisien elektronien (tai positronien) kiihtyessä, eli niiden muuttaessa lentosuuntaansa magneettikentässä.

Hänen mukaansa synkrotronisäteilyä tuotetaan pääasiassa kolmen erilaisen magneettikonfiguraation avulla. Ensimmäinen näistä on taittomagneetti, jossa elektronit lentävät yhtä kaarevaa rataa tuottaen laaja-alaisen säteilykartion. *Wiggler* on toinen magneettikonfiguraatio, jossa elektronit lentävät periodisten magneettikenttien lävitse, jolloin ne alkavat oskilloimaan voimakkaasti menosuuntaansa nähden kohtisuorassa. Tämä tuottaa hyvin intensiivistä ja energistä säteilyä, mutta sen luoma spektri kattaa varsin laajan aallonpituusalueen. Taittomagneetteihin verrattuna syntynyt spektri on samankaltainen, mutta wigglerin fotonivuo on voimakkaampi ja sen aallonpituus lähestyy kovien röntgensäteiden aallonpituutta.

Undulaattori sen sijaan kykenee luomaan hyvinkin kapean aallonpituusalueen säteilyä. Undulaattorissa magneettikenttä on heikompi, jolloin syntynyt oskillointi on vähäisempää mutta säteilykartio kapenee. Lisäksi undulaattorissa tapahtuu interferenssiä, joka osaltaan edesauttaa kapean säteilykartion syntymistä ja tekee säteilystä osittain koherenttia.

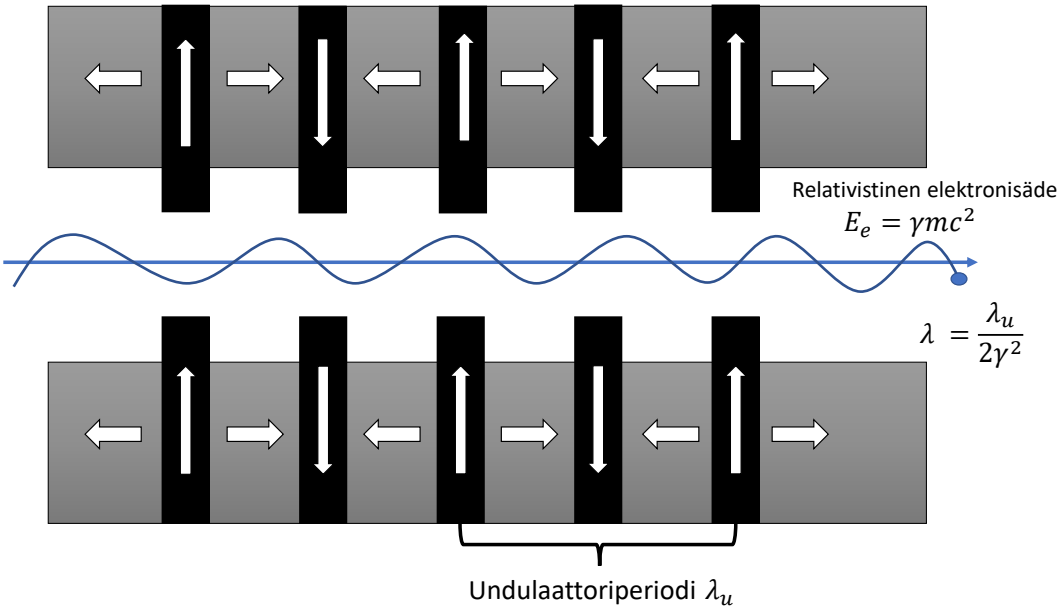


Kuva 2: Vanhemmat synkrotronit tuottivat säteilyä taittomagneettien avulla, jolloin syntynyt spektri oli niin sanottu röntgenhehkulamppu (fotoneja laajalta energia-alueelta). Nykyaikainen synktroni koostuu monesta suorasta osasta, joiden varrella on periodisia magneettirakenteita. Kullakin suoralla osalla voidaan tuottaa halutun aallonpituuden säteilyä. Nykyaikaiset synkrotronit tuottavat undulaattori- ja wiggler-säteilyä joka on varsin monokromaattista, osittain koherenttia sekä säädettävää polarisaatioltaan ja aallonpituudeltaan. Kuva pohjautuu lähteeseen [8].

Tässä tutkielmasssa keskityn tarkastelemaan undulaattorisäteilyä sillä ESRF:n synktrotroni käyttää undulaattoreita röntgensäteilyn tuottamiseen [9]. Undulaattorissa elektronisäteen keskiakselin koordinaatistossa elektronit kohtaavat jaksollisen magneettirakenteen, joka on Lorentz-kontraktoitunut. Jaksollinen magneettijärjestelmä saa elektronit oskilloimaan ja tuottamaan dipolisäteilyä. Elektronisäteen keskiakselin koordinaatistossa tämä vastaa klassista oskilloivaa dipolia, jossa pistevaraus oskilloi ylös alas amplitudilla, joka on paljon pienempi kuin sen säteilemä aallonpituus. Syntyneelle säteilylle tapahtuu relativistinen Doppler-siirtymä, jolloin säteilyn aallonpituus lyhenee. Attwoodin mukaan yhdessä Lorentz-kontraktio ja Doppler-siirtymä saavat aikaan sen, että laboratorion koordinaatistossa elektronisäteen akselin suunnassa syntyvän säteilyn aallonpituus on

$$\lambda = \frac{\lambda_u}{2\gamma^2}, \tag{7}$$

missä γ on Lorentz-tekijä $\gamma = 1/\sqrt{1 - v^2/c^2}$, jossa v on elektronisäteen keskimääräinen nopeus ja c on valonnopeus tyhjiössä.



Kuva 3: Undulaattori koostuu useasta periodisesta magneetista, joiden periodi on λ_u. Elektronien lentäessä magneettien välistä ne alkavat oskilloimaan ja säteilemään synktronisäteilyä. Kuva pohjautuu lähteeseen [8].

Attwood esittelee *undulaattoriyhtälön* joka kuvaa undulaattorissa syntyvän säteilyn aallonpituutta kun otetaan huomioon keskiakselista poikkeava kulma ja säteilyn interferenssin vaikutus aallonpituuteen

$$\lambda = \frac{\lambda_u}{2\gamma^2}\left(1 + \frac{K^2}{2} + \gamma^2\theta^2\right). \tag{8}$$

Yhtälössä λ on undulaattorisäteilyn aallonpituus, λ_u on undulaattorin magneettiperioidin pituus (ks. kuva 3), θ on emissiokulma ja K on nk. undulaattoriparametri.

Termi $\gamma^2\theta^2$ ottaa huomioon keskiakselilta poikkeavan säteilyn aallonpituuden kulmassa θ. Säteilyn intensiteetti on suurimmillaan kun $\theta = 0$. Kapean energia-alueen aallonpituuksien keräämiseksi säteilyä on kerättävä vain kapealta alueelta keskiakselin ympäristöstä sopivaa rakoa käyttäen. Rakokoon kasvaessa säteilyn energiakaista levenee.

Undulaattoriparametri K on magneettikentästä ja magneettien periodista riippuva termi, joka ottaa huomioon säteilyn interferenssin vaikutuksen lopulliseen aallonpituuteen. Attwoodin mukaan jos $K \leq 1$ syntynyt säteily on undolaattorisäteilyä ja mikäli magneettikenttä on vahva ($K >> 1$), syntynyt säteily on wiggler-säteilyä. Termi K riippuu magneettivuon tiheydestä B_0 keskiakselilla ja undulaattorin magneettien periodista λ_u:

$$K = \frac{eB_0\lambda_u}{2\pi m_e c} = 0,9337\ B_0(\mathrm{T})\lambda_u(\mathrm{cm}), \tag{9}$$

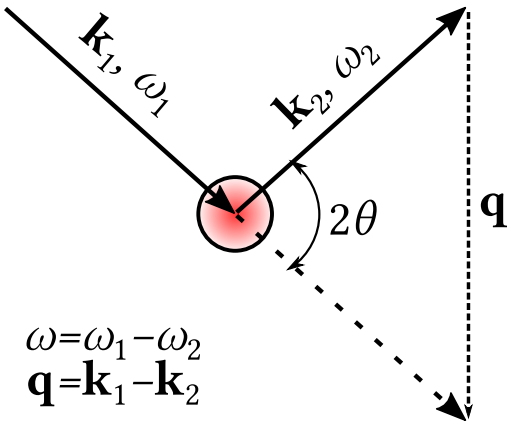
missä e on alkeisvaraus ja m_e on elektronin massa. Undulaattoriparametrin K suuruutta voidaan muuttaa magneettikentän voimakkuutta säätämällä, mikä tapahtuu muuttamalla magneettien etäisyyttä elektronisäteen keskilinjasta. Undulaattorilla syntyvän säteilyn aallonpituutta onkin mahdollista muuttaa varsin helposti.

2.2 Röntgenramansirontaspektroskopia

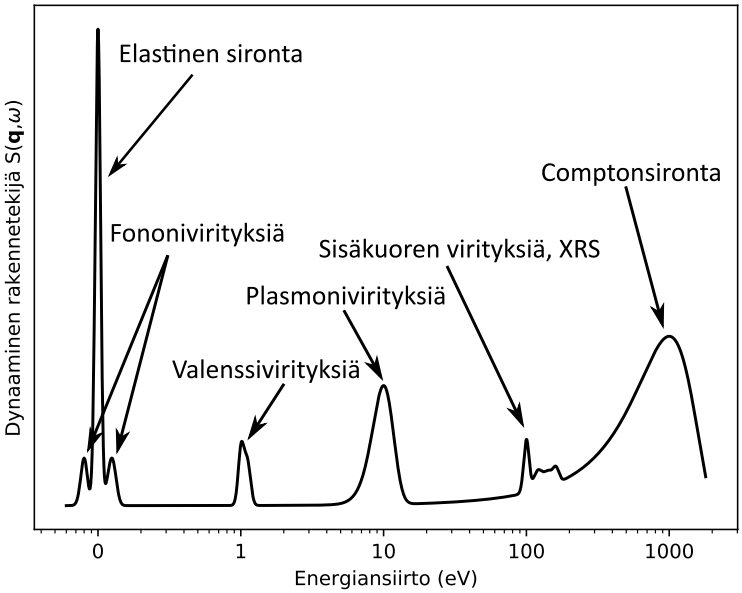
Röntgenramanspektroskopia (*engl. X-ray Raman scattering spectroscopy, XRS*) on kovilla röntgensäteillä tehtävää spektroskopiaa. Ch. J. Sahle ja muut ovat artikkelissaan ”Planning, performing and analyzing X-ray Raman scattering experiments” [10] kertoneet atomiyksiköitä käyttäen, (huom. $\hbar = 1$) epäelastisen röngensirontakokeen toiminnan perustuvan fotonin energialla ω_1 vuorovaikuttavan näytteen elektronien kanssa nostaen elektronin korkeampaan tilaan. Tällöin fotoni siroaa näytteestä pienemmällä energialla ω_2. Tällöin näytteeseen siirtyy energiaa $\omega = \omega_1 - \omega_2$ ja liikemäärää $\mathbf{q} = \mathbf{k}_1 - \mathbf{k}_2$, missä $\mathbf{k_1}$ ja $\mathbf{k_2}$ ovat fotonin liikemäärän aaltovektorit ennen ja jälkeen sironnan. Energiasiirron määrittämiseksi fotonin energia pitää tietää tarkasti myös ennen sen osumista näytteeseen. Tällöin on erittäin tärkeää, että fotonit ovat hyvin kapealta aallonpituusalueelta. Tämän takia röntgenramanspektroskopiassa käytetään synkrotronisäteilyä. Liikemäärän muutoksen suuruus on heidän mukaansa

$$q^2 = |\mathbf{q}|^2 = \frac{1}{c^2}\left[\omega_1^2 + \omega_2^2 - \omega_1\omega_2\cos(2\theta)\right], \tag{10}$$

jossa sirontakulma on 2θ. Liikemäärän muutos voidaan siis laskea kun tiedetään fotonin energiat ω_1 ja ω_2 sekä sirontakulma. Artikkelin kirjoittajien mukaan röntgenramanspektroskopiassa virittävän fotonin energia ω_1 valitaan siten, että energian muutos ω vastaa pehmeillä röntgensäteillä (*engl. soft X-rays*), n. 0,25-4 keV [8], saatavaa röntgenabsorptiospektriä ja tällöin havainnoidaan ω_2-energisiä sironneita fotoneita jossain avaruuskulmaelementissä $d\Omega$. Kirjoittajat mainitsevat, että tavallisesti fotonien energiat ovat n. 10 keV luokkaa ja ovat siten kovia röntgensäteitä (*engl. hard X-rays*). Kovien röntgensäteiden tyypillinen energia-alue on noin 5 keV ylöspäin [8]. Esimerkiksi kuvan 1 vesimolekyylin sisäkuoren elektronien sidosenergia on 559 eV, joka on huomattavasti pienempi kuin kovien röntgensäteiden energia.



Kuva 4: Röntgenramansironnan toimintaperiaate. Fotoni siroaa virittäessään sisäkuoren elektronin menettäen energiaa ja liikemäärää, joiden muutoksia voidaan mitata. Kuva pohjautuu lähteeseen [10].



Kuva 5: Röntgenramanspektroskopiassa energiansiirto näytteeseen on sisäkuoren elektronien sidosenergioiden suuruusluokkaa. Kuva pohjautuu lähteeseen [11].

Kovat röntgensäteet tuottavat myös taustasäteilyä Comptonin ilmiön kautta kun fotonin sirotessa valenssielektroneista. Röntgenramanspektroskopiassa tutkitaan sisäkuoren elektroneista sironneiden fotonien spektrireunaa. Comptonin sironnasta syntyneen taustasäteilyn vaikutusta voidaan vähentää valitsemalla sellainen liikemääränsiirto $\mathbf{q}$ jolla ydinelektronien virittäytymisen muodostama spektri ei ole Compton-taustan huipun lähistöllä [10].

Winfried Schülke kertoo kirjassaan "Electron Dynamics by Inelastic X-ray Scattering" [12], että tavallisessa epäelastisessa röntgensirontakokeessa käytetään hyvin monokromaattista ja kollimoitua fotonisädettä. Sironneesta säteilystä valitaan tietty avaruuskulmaelementti $\mathrm{d}\Omega$, joka hänen mukaansa määrittää liikemäärän siirron $\mathbf{q}$ systeemiin, ja tähän avaruuskulmaan sironnutta säteilyä havainnoidaan energiaresoluutiolla $\mathrm{d}\omega_2$. Schülken mukaan tällöin voidaan mitata sironnan kaksoisdifferentiaalivaikutusala (*engl. double differential scattering cross section, DDSCS*) $\mathbf{q}$:n ja

ω:n funktiona

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\mathrm{d}\omega_2} = \frac{\begin{array}{c}\text{avaruuskulmaelementtiin } [\Omega,\ \mathrm{d}\Omega] \text{ ja energia-alueelle} \\ [\omega,\ \mathrm{d}\omega_2] \text{ sironnut fotonivuo}\end{array}}{\text{näytteeseen osuvien fotonien vuontiheys} \times \mathrm{d}\Omega \times \mathrm{d}\omega_2} \tag{11}$$

Sahle ja muut [10] esittävät sironnan kaksoisdifferentiaalivaikutusalalle muodon

$$\frac{\mathrm{d}^2\sigma}{\mathrm{d}\Omega\mathrm{d}\omega_2} = \left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}\right)_{\mathrm{Th}} S(\mathbf{q},\omega), \tag{12}$$

missä termi $(\mathrm{d}\sigma/\mathrm{d}\Omega)_{\mathrm{Th}}$ on Thomsonin sirontavaikutusala, joka on

$$\left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}\right)_{\mathrm{Th}} = r_e^2 \left(\frac{\omega_2}{\omega_1}\right) (\hat{\mathbf{e}}_1 \cdot \hat{\mathbf{e}}_2), \tag{13}$$

missä r_e on elektronien klassinen säde, ja $\hat{\mathbf{e}}_1$ ja $\hat{\mathbf{e}}_2$ ovat näytteeseen osuvien ja siitä siroavien fotonien polarisaatiovektorit [9]. Termi $S(\mathbf{q},\omega)$ on dynaaminen rakennetekijä, joka sisältää kaiken informaation mitä systeemistä voidaan havaita röntgenramanspektroskopialla. Sahle ja kollegat tiivistävät, että monen partikkelin perustilasta $|\mathrm{i}\rangle$ virittäytymiset lopullisiin tiloihin $|\mathrm{f}\rangle$ voidaan kirjoittaa fyysikoille yleisesti tutun Fermin kultaisen säännön avulla

$$S(\mathbf{q},\omega) = \sum_{\mathrm{i,f}} p_{\mathrm{i}} \left| \langle \mathrm{f} | \sum_j \exp(i\mathbf{q}\cdot\mathbf{r}_j) | \mathrm{i} \rangle \right|^2 \delta(E_{\mathrm{i}} - E_{\mathrm{f}} + \omega), \tag{14}$$

jossa p_{i} on todennäköisyys löytää systeemi sitä vastaavasta alkutilasta $|\mathrm{i}\rangle$ ja deltafunktio δ huolehtii siitä, että energia säilyy. Jos Hartree–Fock-approksimaatio olisi tarkka, eikä systeemi relaksoituisi ollenkaan kuoriaukkotilassa, tämä summa edustaisi yksielektronitransitioita orbitaaleilta toisille.

Simo Huotari ja muut ovat artikkelissaan ”Direct tomography with chemical-bond contrast” [11] kertoneet kovilla röntgensäteillä tehtävän XRS-spektroskopian paikkaavan monia pehmeillä röntgensäteillä tehtävän spektroskopian ongelmakohtia. Röntgenramanspektroskopiassa voidaan tutkia näytettä pintaa syvemmältä, sillä kovat röntgensäteet pureutuvat jopa useita millimetrejä näytteen pinnan alle – toisin kuin pehmeät röntgensäteet, jotka vaikuttavat vain näytteen pinnan läheisiin

atomikerroksiin. Näytteen ei myöskään tarvitse olla tyhjiössä kovia röntgensäteitä käytettäessä, mikä helpottaa mittausjärjestelyä ja näytteenvaihtoa huomattavasti. Huotarin ja muiden mukaan tällöin on myös mahdollista käyttää märkiä tai nestemäisiä näytteitä joka on usein tarpeellista biologisten näytteiden tutkimuksessa. Lisäksi näytettä voidaan tutkia erilaisissa lämpötiloissa kryostaatin avulla tai vaikkapa korkeissa paineissa käyttäen timanttialasinta (*engl. diamond anvil cell*) [9].

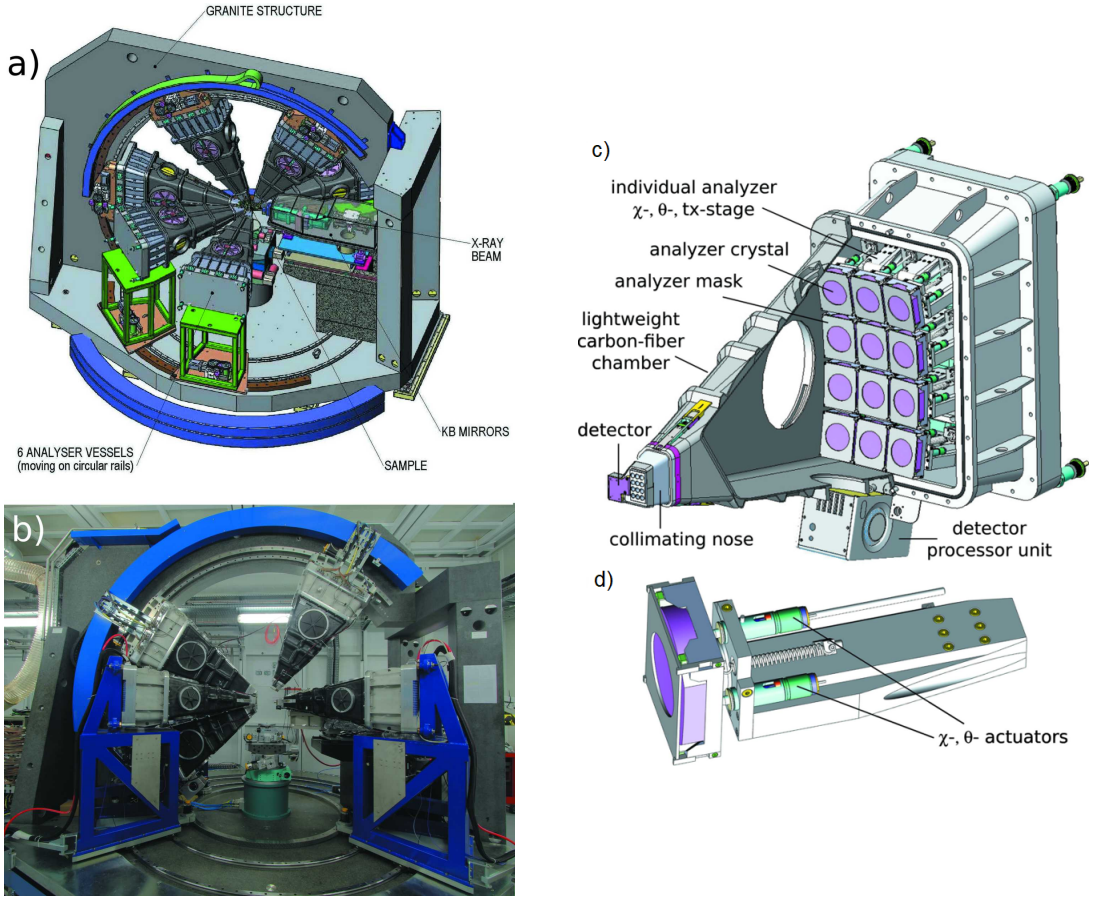
2.3 European Synchrotron Radiation Facility

Tässä tutkielmassa on käytän dataa, joka on tuotettu European Synchrotron Radiation Facilityssä (ESRF) Grenoblessa Ranskassa. S. Huotari, Ch. J. Sahle ja muut ovat artikkelissaan "A large-solid-angle X-ray Raman scattering spectrometer at ID20 of the European Synchrotron Radiation Facility" [9] kertoneet ESRF:n synkrotronin ja ID20-sädelinjan toiminnasta. ESRF:n synkrotroni [13] on ympärysmitaltaan 844 metriä ja sen kiihdyttimen elektronilinjan elektronien energia on 6 GeV. Jokainen synktrotronin suora osuus koostuu yhdestä 26 mm periodisesta undulaattorista ja kolmesta vaihdettavasta undulaattorista, joiden periodit vaihtelevat 26:sta 32 millimetriin. Synktrotronissa syntynyttä säteilyä käsitellään erinäisin keinoin ennen kuin sillä säteilytetään näytettä. Huotarin, Sahlen ja muiden mukaan fotonisädettä kollimoidaan ensin peilijärjestelmän kanssa, joka samalla vähentää seuraavan komponentin, nestetypellä jäähdytetystä piikiteestä muodostuvan esimonokromaattorin, lämpökuorma. Esimonokromaattorin jälkeen on mahdollista käyttää myös jälkimonokromaattoreita, joista useimmat ovat erilaisia piikiteitä. Monokromaattoreiden tarkoitus on päästää läpi vain kapean aallonpituusalueen säteilyä. Artikkelin kirjoittajat sanovat, että ilman jälkimonokromaattereita tyypilliset näytteisiin osuvat fotonivuot ovat luokkaa $10^{12} - 10^{13}$ fotonia sekunnissa fotonienergioilla n. 10 ± 3 keV.

Kuva 6: Ilmakuva ESRF:n laitoksesta. Kuva on lähteestä [13], `https://www.esrf.eu/files/live/sites/www/files/about/press-room/ESRF-MAY-2015_07.jpg`, luettu 15.5.2020

Artikkelin kirjoittajien mukaan säteily kohdistetaan näytteeseen kahden Kirkpatrick-Baez -peilin avulla. Tällöin näytealueella säde muodostaa 8 µm ×16 µm kokoisen pisteen. Spektrometri koostuu kuudesta analysaattorikammiosta, joista neljä havaitsee näytteestä etusuunnassa sironnutta säteilyä ja kaksi havaitsee taaksepäin sironnutta säteilyä. Suurin osa säteilystä menee kuitenkin näytteen läpi siroamatta. Säteilyn sirottua näytteestä se osuu kollimaattoriin, joka rajaa säteilyn yhteen analysaattorinkammion kahdestatoista piikiteestä. Kirjoittajien mukaan hiilikuidusta valmistetussa analysaattorikammiossa ylläpidetään yhden millibarin painetta minimoidakseen ilmasta aiheutuvat absorptiot ja haitallinen sironta. Analysaattorikiteet ovat Si(nn0)-kiteitä jotka ovat taivutettu siten että niiden taivutussäde on 1.0 metriä. Analysaattorikiteitä on mahdollista kääntää ja liikuttaa kammion sisällä.

Kullakin 72:sta analysaattorikiteestä on sille ominainen ω_2 ja $\mathbf{q}_2$, jotka mahdollistavat energian- ja liikemääränsiirron määrittämisen. Tämän tutkielman puitteissa keskityn tarkastelemaan vain etusuunnassa sironnutta säteilyä.



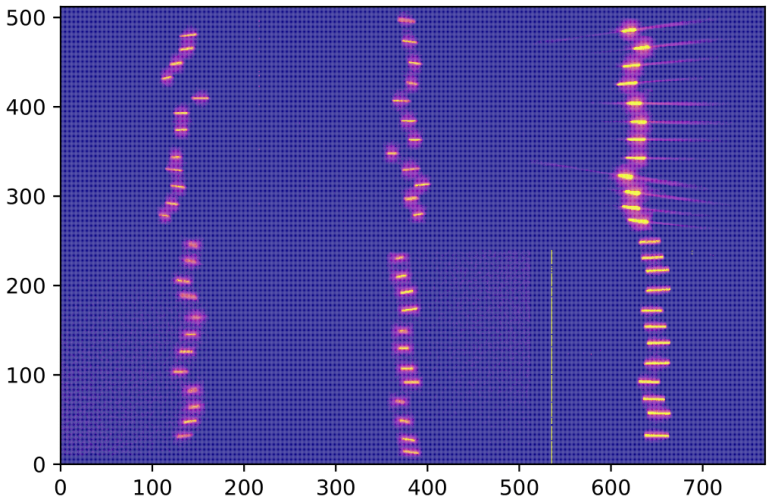
Kuva 7: a) Kaavakuva koko ID20-sädelinjan röntgenramansirontaspektrometrin rakenteesta. Koelaite koostuu kuudesta erikseen liikuteltavasta kideanalysaattorikammiosta joista kussakin on kaksitoista analysaattorikidettä. b) Valokuva spektrometristä kiinnitettynä kehikkoonsa. c) Läpileikkaus kideanalysaattorikammiosta, jossa on kaksitoista analysaattorikidettä. d) Piirros yhdestä analysaattorikiteestä. Kuvat ovat lähteestä [9] ja ne on julkaistu Open Access/Creative Commons -lisenssillä `https://creativecommons.org/licenses/by/4.0/deed.fi`.

Huotarin, Sahlen ja muiden mukaan spektrometrin energiaresoluution ei-resonoiville epäelastisille röntgenspektrometriakokeille olevan $\sim 0,3-2,0$ eV riippuen käytetystä

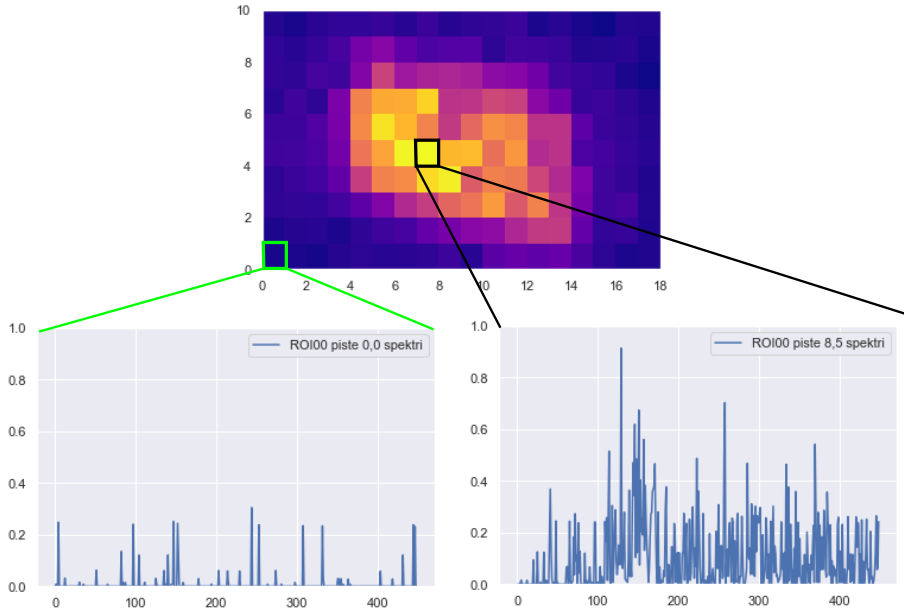
fotonienenergiasta, analysaattorikiteen kiderakenteesta ja mahdollisesta analysaattorikiteen maskista.

2.4 Datan esittely

ESRF:n röntgenramanspektrometri tuottaa valtavia määriä dataa. Koska ilmaisin on sekä paikka- että energiaherkkä, syntynyt data on kolmiulotteista. Kukin ilmaisin ottaa kuvan, jonka jokaisen pikselin takana on energiaspektri (ks. kuvat 8 ja 9). Suurin osa näistä pikseleistä ei kuitenkaan sisällä hyödyllistä dataa vaan pelkkää taustakohinaa. Kuvassa 8 oikeanpuoleisimpien tarkastelualueiden läheisyydessä on havaittavissa haitallista ilmasta sironnutta säteilyä, joka pitää poistaa dataa käsiteltäessä. Yksi tapa hakea kaikki tarkastelualueet (*engl. Region of Interest, ROI*) ja suodattaa kohinaa pois on käyttää meadiaanifiltteriä, jossa hylätään alueet joiden pikselien intesiteettit eivät ylitä tiettyä kynnysarvoa [10].



Kuva 8: Kuuden eri ilmaisimen detektorikuva. Aktiivisimpien aluiden läheisyydestä valitaan neliskanttiset tarkastelualueet, joiden spektrejä analysoidaan. Kaksi oikeanpuoleisinta detektorikuvaa ovat havainnoineet näytteessä tapahtuvaa takaisinsirontaa, jota en käsittele tässä työssä.



Kuva 9: Eräs tarkastelualueista. Jokainen pikseli on energiaherkkä ilmaisin, joten jokaisen pikselin ”alla” on energiaspektri.

Tätä tutkielmaa varten olen saanut käytettäväkseni European Synchrotron Radiation Facility:ssa (ESRF) kerättyä spektrometridataa [14]. Spektrometri on mitannut kahden eri näytteen, nestemäisen ja kiinteän veden, spektrejä viidentoista minuutin erissä. Tavallisesti useita mittauksia tehdään peräkkäin ja niiden spektrit lasketaan yhteen. Useimmiten mittausaika on kuudesta kahteentoista tuntia. Spektrometrin signaalin kohinasuhde paranee suhteessa mittausajan neliöjuureen, joten pidemmällä mittausajalla saadaan selvempi spektri [10].

Kullekin analysaattorikiteelle i elastisen viivan $E_0^{(i)}$ paikka mitataan energialla $\omega_1^{(i)}$, (ks. kuva 5), jonka avulla määritetään energiansiirtoakselin nollakohta. Koska elastisen viivan paikka on helppo määrittää ja koska tiedetään, että fotonien energiansiirto on silloin nolla, sen avulla on helppo kalibroida epäelastinen sirontaspektri. Elastisen viivan energian ja sisäkuoren virityksien väliset energiaerot pysyvät vakioina vaikka sisään tulevaan tai sironneen röntgensäteilyn energioita muutettaisiin. Eli sironneen säteilyn absoluuttisella aallonpituudella ei ole niinkään väliä kuin sisäkuoren spektrin ja elastisen viivan välinen erotus. Kunkin kiteen energiansiirto kalibroidaan tätä tietoa käyttäen.

Detektorien energiaspektri on jaettu 450-osaiseksi datavektoriksi, jossa jokainen datapiste vastaa havaittujen fotonien määrää tietyllä energia-alueella. Energia-alueet eivät ole kaikkien detektorien välillä samat, joten dataa käsiteltäessä kaikki spektrit on interpoloitava yhdelle energia-akselille.

Kuvassa 9 on esitetty kaksi spektriä yhdeltä tarkastelualueelta. Ensimmäinen spektri, joka on otettu aivan tarkastelualueen reunalta, ei sisällä suurikaan informaatiota ja sekin peittyisi kohinan alle. Tämä on esimerkki taustan spektristä. Toisessa spektrissä sen sijaan on selvä rakenne ja se sisältää selvästä enemmän informaatiota. Tämä on esimerkki näytteen spektristä.

Tieteellisesti kiinnostavia havaintoja tehdään useasta mittausajosta kerrallaan laskemalla ne yhteen. Kaikki spektrit eivät kuitenkaan sisällä muuta kuin kohinaa,

joten niiden pois suodattaminen on tarpeellista. Tavallisesti tutkija joutuu valitsemaan käsin ne spektrit, jotka hän haluaa laskea yhteen. Tämä on kuitenkin aikaavievää ja puuduttavaa työtä, varsinkin kun luokiteltavia datavektoreita saattaa olla tuhansia. Tämän tutkielman tehtävänä on automatisoida prosessi siten, ettei tutkijan täytyisi luokitella koko dataa käsin, vaan hän luokittelisi vain pienen osan kaikista spektreistä antaen jonkin seuraavista malleista luokitella loput spektrit automaattisesti.

3 Tekoälymenetelmät spektrien luokitteluun

Olen valinnut kaksi eri koneoppimisen muotoa automaattiseen spektrien luokitteluun. Molemmat mallit, tukivektorikone ja neuroverkko, perustuvat koulutusjoukkoihin, joissa spektrit on yhdistetty niitä vastaaviin käsinluokiteltuihin luokkiin. Mallit pyrkivät erottelemaan näytteen ja taustan spektrit toisistaan koulutusjoukon avulla ja ennustamaan spektrejä, joita mallit eivät ole aiemmin nähneet. Kyseessä on siis binäärinen luokitteluongelma, jossa mallit pyrkivät tekemään kuvauksen $\mathbb{R}^D \mapsto \{0, 1\}$ ja näin ollen luokittelemaan spektrejä. Tämä vastaa näytettä edustavien pikseleiden valintaa.

3.1 Tukivektorikone

Tukivektorikone (*engl. Support Vector Machine, SVM*) on monipuolinen ohjattu (*engl. supervised*) koneoppimismalli, joka kykenee sekä lineaariseen, että epälineaariseen luokitteluun, regressioon ja poikkeavien havaintojen (*engl. outlier*) tunnistukseen [15]. Tässä tutkielmassa keskityn luokitteluun kahden eri luokan välillä, eli binääriseen luokitteluun. Lisäksi seuraan Marc Peter Deisenrothin ja muiden matemaattista formulointia ja notaatiota heidän kirjastaan "Mathematics of Machine Learning" [16].

Tehtävässäni D on spektrin energiapisteiden lukumäärä ja $\mathbf{x}_n$ spektri järjestettynä datavektoriksi. Binäärisessä luokittelussa pyritään kuvaamaan vektori yhteen kahdesta luokasta $\{0, 1\}$ tai $\{-1, 1\}$ [16].

Tukivektorikoneen toiminta on helpointa kuvata esittämällä se geometrisesti. Kuvassa 10 data on lineaarisesti erotettavissa kahteen eri luokkaan. Tukivektorikone pyrkii rakentamaan mahdollisimman leveän kaistan näiden kahden luokan välille, siten että kaikki datapisteet ovat kaistan ulkopuolella, paitsi ne jotka määrittävät *tukivektorit* [15].

Tukivektorikoneen kouluttamiseksi koulutusdatan tulee olla jo valmiiksi luoki-

teltu syöte- ja tulostepareihin (*engl. input, output pairs*) $\{\mathbf{x}_n, \mathrm{y}_n\}$. Tukivektorikone luo kahden luokan pisteiden välille D-ulotteisessa avaruudessa niitä erottavan hypertason, joka on $D-1$ -ulotteinen aliavaruus. Malli tekee ennusteita katsomalla kummalle puolelle tätä hypertasoa ennustettava piste osuu [16]. Yksinkertaisin tapa jakaa avaruus kahteen osaan on käyttää lineaarista hyperpintaa eli hypertasoa.

Moniulotteisessa avaruudessa kahden vektorin $\mathbf{x}_i$ ja $\mathbf{x}_j$ samanlaisuutta voidaan tarkastella niiden sisätulon $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ kautta, sillä sisätulo kuvaa kahden vektorin projektiota toisilleen. Deisenroth ja muut määrittävät tukivektorikoneessa tapahtuvat kuvaukset seuraavasti:

$$f : \mathbb{R}^D \to \mathbb{R} \tag{15}$$

$$\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle + b, \tag{16}$$

mikä parametrisoidaan vektorilla $\mathbf{w} \in \mathbb{R}^D$ ja skalaarilla $b \in \mathbb{R}$. Syntyvät hypertasot ovat siis affiineja aliavaruuksia, eli niiden origoa ei ole kiinnitetty. Näin ollen Deisenroth ja muut määrittävät hypertason, joka erottaa kaksi luokkaa toisistaan

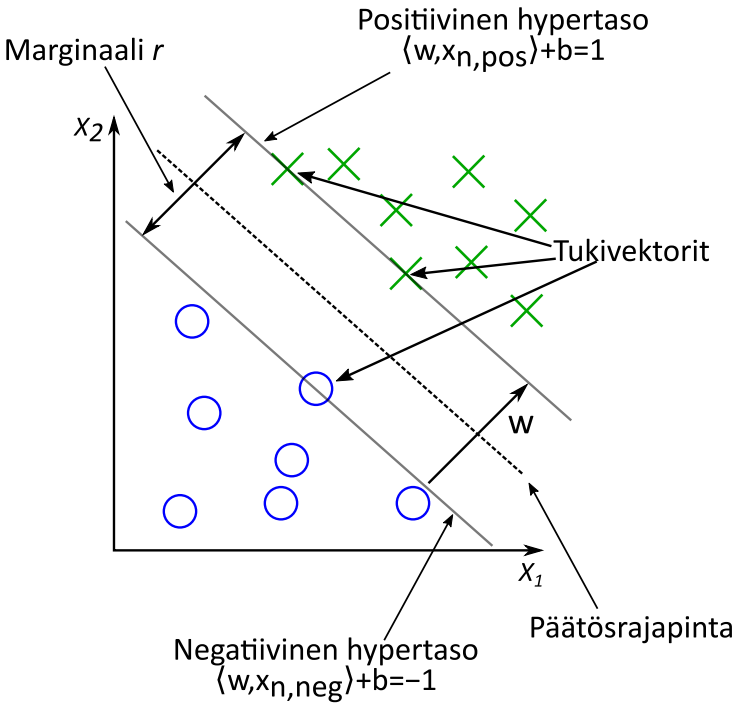
$$\{\mathbf{x} \in \mathbb{R}^D : f(\mathbf{x}) = 0\}. \tag{17}$$

Parametri $\mathbf{w}$ on hypertason normaalivektori, joka on ortogonaalinen jokaisen hypertason vektorin kanssa, ja b on hypertason leikkauspiste. Deisenrothin ja muiden mukaan hypertasoa käytetään pisteen luokittelemiseksi positiiviseen tai negatiivisen luokkaan riippuen siitä, kummalla hypertason puolella piste sijaitsee. Sen lisäksi että yhtälö (17) määrittelee hypertason, se myös määrittää suunnan hypertasolta. Kaikki hypertasolla olevat pisteet saavat arvon nolla ja hypertasolta poikkeavat pisteet ovat joko sen positiivisella tai negatiivisella puolella. Suunta hypertason suhteen vastaa pisteen etumerkkiä. Jos $f(\mathbf{x}) \geq 0$ niin piste kuuluu positiiviseen luokkaan 1 ja muulloin se kuuluu negatiiviseen luokkaan -1.

Luokittelijaa koulutetaan Deisenrothin mukaan vaatimalla että positiiviseen luokkaan kuuluville pisteille $f(\mathbf{x}) \geq 0$ ja negatiiviseen luokkaan kuuluville pisteille $f(\mathbf{x}) < 0$

$$\langle \mathbf{w}, \mathbf{x}_n \rangle + b \geq 0, \text{ kun } y_n = 1 \tag{18}$$

$$\langle \mathbf{w}, \mathbf{x}_n \rangle + b < 0, \text{ kun } y_n = -1 \tag{19}$$



Kuva 10: Tukivektorikone rakentaa kahden eri luokan välille hypertason ja pyrkii maksimoimaan marginaalin r leveyden. Pisteet jotka ovat marginaalin ulkopuolella eivät vaikuta tukivektorikoneen luokitteluun, joten koulutuspisteiden lisääminen ei välttämättä paranna tukivektorikoneen suorituskykyä. Tukivektorikone tekee ennusteita arvioimalla kummalla puolella päätösrajapintaa ennustettava piste sijaitsee. Kuva pohjautuu lähteeseen [17].

Tällä määritelmällä kuitenkin on olemassa ääretön määrä hypertasoja, jotka pystyvät erottelemaan lineaarisesti separoituvan datajoukon. Ratkaisun Deisenroth kertoo olevan marginaalien optimoiminen. Marginaali r on tarkastelupisteen $\mathbf{x}$ etäisyys sen ortogonaaliprojektiosta $\mathbf{x}'$ hypertasolle. Sebastian Raschka ja Vahid Mirjalili ovat kirjassaan "Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and Tensorflow, 2nd Edition" [17] asettaneet marginaalin arvoksi $r = 1$. Eli positiivinen ja negatiivinen hypertaso ovat kumpikin marginaalin $r = 1$ päässä päätösrajapinnasta mutta eri suunnissa. Tällöin positiivinen ja negatiivinen hypertaso voidaan ilmaista kuten kaavoissa (18) ja (19) mutta seuraavin muutoksin

$$\langle \mathbf{w}, \mathbf{x}_{n,pos} \rangle + b = 1, \text{ kun } y_n = 1 \tag{20}$$

$$\langle \mathbf{w}, \mathbf{x}_{n,neg} \rangle + b = -1, \text{ kun } y_n = -1. \tag{21}$$

Raschka on vähentänyt nämä kaksi lineerista yhtälöä toisistaan. Tämän lisäksi käytän Deisenrothin esittämää sisätulon bilineaarisuutta, jolloin

$$\langle \mathbf{w}, \mathbf{x}_{n,pos} \rangle - \langle \mathbf{w}, \mathbf{x}_{n,neg} \rangle = 2 \tag{22}$$

$$\langle \mathbf{w}, \mathbf{x}_{n,pos} - \mathbf{x}_{n,neg} \rangle = 2, \tag{23}$$

jonka Raschka on normalisoinut jakamalla puolittain painovektorin $\mathbf{w}$ normilla

$$\|\mathbf{w}\| = \langle \mathbf{w}, \mathbf{w} \rangle^{\frac{1}{2}} = \sqrt{\sum_{i=1}^{n} w_i^2} \tag{24}$$

$$\frac{\langle \mathbf{w}, \mathbf{x}_{n,pos} - \mathbf{x}_{n,neg} \rangle}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \tag{25}$$

Raschkan mukaan yhtälön vasen puoli voidaan tulkita positiivisen ja negatiivisen hypertason välisenä etäisyytenä, eli marginaalina. Aurélien Géron esittää kirjassaan "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" [15], että

painovektorin normi $\|\mathbf{w}\|$ on pyrittävä minimoimaan marginaalin maksimoimiseksi. Tämä nähdään myös selvästi yhtälöstä (25). Hänen mukaansa optimointialgoritmien on helpompi käyttää funktiota, joka on kaikkialla derivoituva, joten tavallisesti minimoidaan termiä $\frac{1}{2}\|\mathbf{w}\|^2$, jonka gradientti on aina $\mathbf{w}$ kun taas $\|\mathbf{w}\|$ ei ole derivoituva kun $\mathbf{w} = 0$.

Géronin mukaan näillä ehdoilla toimiva tukivektorikone tunnetaan kovan marginaalin (*engl. hard margin*) luokittelijana, sillä se ei salli yhtään väärää luokittelua koulutusvaiheessa. Eli mikäli datapisteet eivät ole täydellisesti lineaarisesti separoitavissa hypertasolla, kovan marginaalin tukivektorikone ei kykene suppenemaan ja luomaan luokittelijarajapintaa. Jos kuitenkin halutaan generoida hypertaso, joka sallii väärät luokittelut mutta pyrkii minimoimaan ne, voidaan käyttää pehmeän marginaalin (*engl. soft margin*) luokittelua, jolloin yhtälöön lisätään alijäämämuuttuja (*engl. slack variable*) ζ. Raschkan mukaan väärät luokittelut voidaan sallia antamalla niistä sakkoa jollain tietyllä sakkofunktiolla ja pyrkimällä minimoimaan tämän arvoa painovektorin normin ohella. Tällöin voidaan luokitella dataa jossa osa eri luokkiin kuuluvista datapisteistä ovat jakautuneet päällekäin. Yksittäiselle datapisteelle voidaan määritellä sakkofunktio

$$\langle \mathbf{w}, \mathbf{x}_{i,pos} \rangle + b \geq 1 - \zeta_i, \text{ kun } y_i = 1 \tag{26}$$

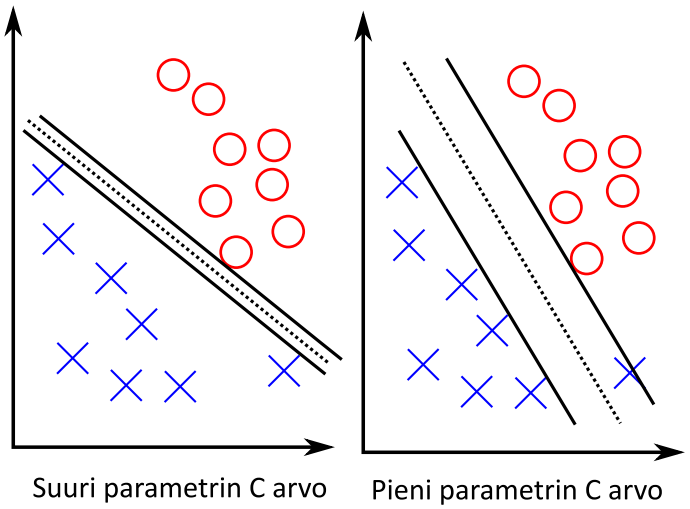
$$\langle \mathbf{w}, \mathbf{x}_{i,neg} \rangle + b \leq -1 + \zeta_i, \text{ kun } y_i = -1 \tag{27}$$

kun $i = 1, 2...n$, ja n on datajoukon näytteiden määrä. Raschkan ja Géronin mukaan tällöin minimoitavaksi suureeksi J tulee

$$J = \frac{1}{2}\|\mathbf{w}\|^2 + C\left(\sum_{i=1}^{n} \zeta_i\right), \tag{28}$$

missä muuttuja C määrää väärästä luokittelusta syntyvän häviön painoarvon. Kun C on suuri, väärästä luokittelusta häviötä syntyy paljon, jolloin termiä optimoides-

sa virheiden määrä pyritään minimoimaan marginaalien leveyden kustannuksella, kuten on havainnollistettu kuvassa 11.

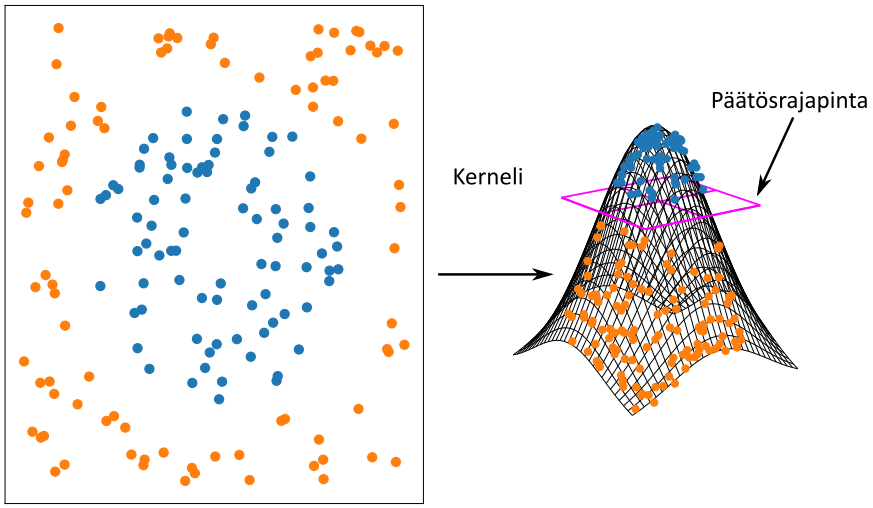


Kuva 11: Parametrin C vaikutus hypertason rakentamiseen. Vasemmalla häviön painoarvo on suuri, joten marginaalin leveyttä uhrataan kaikkien pisteiden luokittelemiseksi oikein. Oikealla häviön painoarvo on pieni, jolloin yksi piste päädytään luokittelemaan väärin. Vastapainoksi tällöin marginaalista saadaan leveä. Kuva pohjautuu lähteeseen [17].

Molemmat kovan ja pehmeän marginaalin ongelmat ovat Géronin mukaan konvekseja neliöllisiä optimointiongelmia (*engl. Quadratic Programming, QP*), joiden ratkaisuun on olemassa monia algoritmeja.

3.1.1 Kernelifunktiot

Géron kertoo, että tukivektorikoneet pystyvät luokittelemaan myös sellaisia datajoukkoja, joissa kahteen eri joukkoon kuuluvia datapisteitä ei sellaisenaan voida erotella toisistaan. Tällöin hänen mukaan datajoukko voidaan nostaa korkeampaan ulottuvuuteen jollain kuvauksella ja tällöin sen luokat voidaan separoida toisistaan (ks. kuva 12). Hän antaa esimerkkinä toisen asteen polynomimuunnoksen ϕ, joka



Kuva 12: Kernelitempulla voidaan erotella kaksi luokkaa jotka eivät ole sellaisenaan lineaarisesti eroteltavissa päätösrajapinnalla (*decision surface*). Menettelyssä pisteet kuvataan korkeampaan ulottuvuuteen jollain kernelifunktiolla.

nostaa kaksiulotteisen datajoukon kolmanteen ulottuvuuteen.

$$\phi(\mathbf{x}) = \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix} \tag{29}$$

Tätä kuvausta ei täydy kuitenkaan tehdä kaikkiin datapisteisiin erikseen, sillä tilanteessa on mahdollista käyttää *kernelitemppua*. Kahden vektorin $\mathbf{a}$ ja $\mathbf{b}$ pistetuloa laskettaessa muunnoksen voi jättää kokonaan laskematta sillä hän johtaa toisen asteen muutokselle että

$$\langle \phi(\mathbf{a}), \phi(\mathbf{b}) \rangle = \langle \mathbf{a}, \mathbf{b} \rangle^2. \tag{30}$$

Annettulle toisen asteen ϕ:lle

$$K(\mathbf{a}, \mathbf{b}) = \langle \mathbf{a}, \mathbf{b} \rangle^2 \tag{31}$$

Géron sanoo, että kernelifunktio $K(\mathbf{a}, \mathbf{b})$ on sellainen funktio, joka pystyy laskemaan jonkin muunnettujen vektorien $\phi(\mathbf{a})$ ja $\phi(\mathbf{b})$ pistetulon vain alkuperäisten vektorien $\mathbf{a}$ ja $\mathbf{b}$ avulla ilman että muunnoksia $\phi(\mathbf{a})$ ja $\phi(\mathbf{b})$ täytyy laskea.

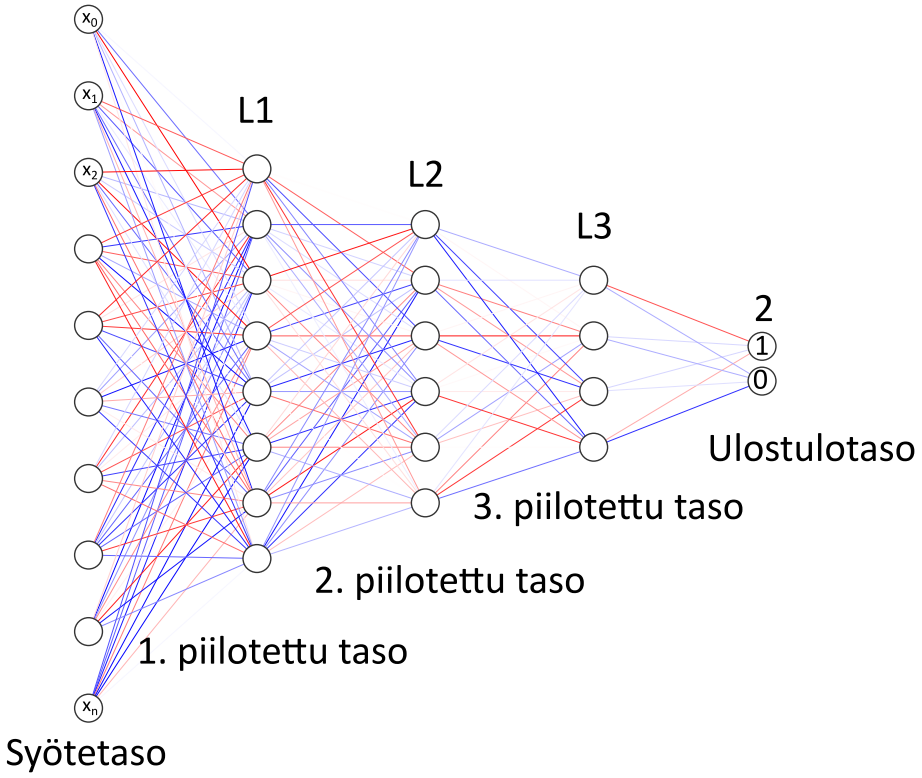
Taulukko 2: Eräitä Géronin esittelemiä kernelifunktioita [15].

Muunnos	Kernelifunktio
Lineaarinen	$K(\mathbf{a}, \mathbf{b}) = \langle \mathbf{a}, \mathbf{b} \rangle$
Polynomaalinen	$K(\mathbf{a}, \mathbf{b}) = (\gamma \langle \mathbf{a}, \mathbf{b} \rangle^2 + r)^d$
Gaussin radiaalikantafunktio RBF	$K(\mathbf{a}, \mathbf{b}) = \exp(-\gamma \lvert \mathbf{a} - \mathbf{b} \rvert^2)$

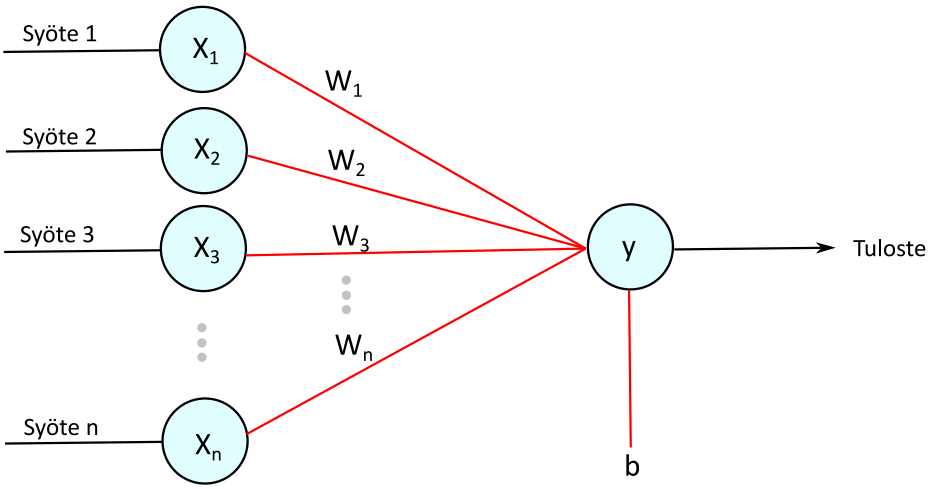
Polynomaaliselle kernelifunktiolle hyperparametri d on polynomin aste ja r on vakiotermi. Hyperparametri γ on polynomaalisessa kernelifunktiossa kerroin joka muuttaa polynomin jyrkkyyttä. Gaussin radiaalikantafunktiolla γ vaikuttaa kuvauksessa tehtyjen gaussin käyrän muotoisien muunnoksien jyrkkyyteen ja toimii samalla ikään kuin regularisaatioparametrinä [15]. Géronin mukaan lineaarista kernelifunktiota kannattaa kokeilla ensimmäisenä ja mikäli datajoukko ei ole liian suuri gaussin radiaalikantafunktio toimii hyvin useimmissa tapauksissa.

3.2 Neuroverkko

Toinen malli, jota olen käyttänyt spektrien luokitteluun, perustuu neuroverkkoihin. Neuroverkot ovat saaneet inspiraationsa aivojen arkkitehtuurista ja biologisten neuroneiden toiminnasta. Nykyään kuitenkin neuroverkot ovat alkaneet jättää biologista analogiaa jälkeensä kehittyen omaan suuntaansa [15]. Michael Nielsen kertoo kirjassaan Neural Networks and Deep Learning [18] jokaisen neuroverkon ytimessä olevan perseptroni, joka muistuttaa biologista neuronia.



Kuva 13: Neuroverkon havainnointikuva. Syötetasolla neuroverkko ottaa vastaan datavektorin, jossa $x_1, x_2, \cdots, x_n$ ovat spektrin intensiteetit eri energiakanavilla. Ensimmäisen piilotetun tason aktivaatiot riippuvat painoarvoista ja vakiotermeistä. Painoarvot on esitetty kuvassa eri värein, punaiset painoarvot ovat positiivisia ja siniset negatiivisia. Viivan paksuus kuvaa painoarvon suuruutta. Toisen tason aktivaatiot riippuvat ensimmäisen tason aktivaatioista ja niin edelleen. Datavektori luokitellaan kuuluvaksi siihen luokkaan, jota edustava tulostetason neuroni aktivoituu enemmän. Kuvan neuroverkko on piirretty palvelussa [19].

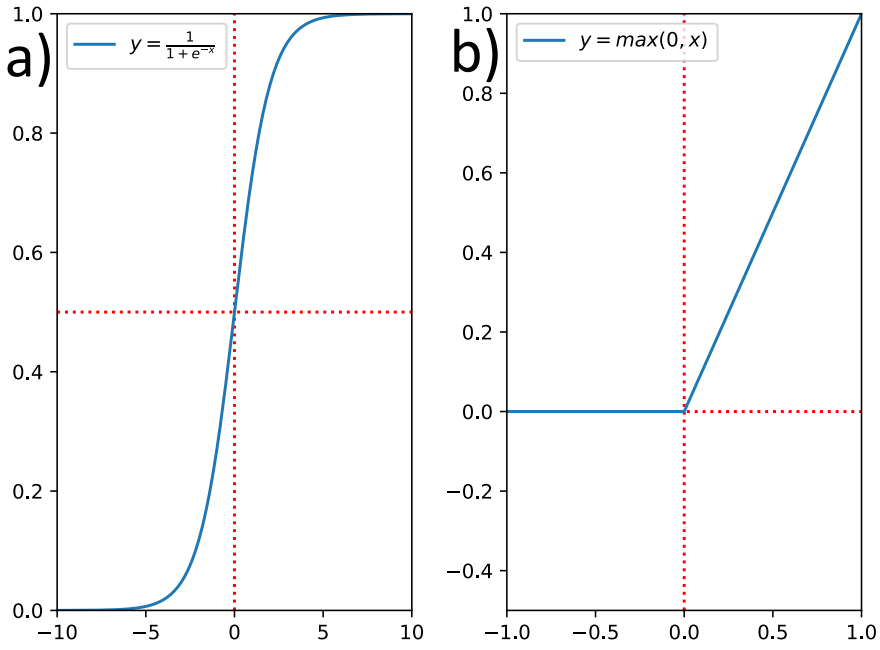


Kuva 14: Perseptronin rakenteen havainnointikuva. Kuva pohjautuu lähteeseen [20].

Perseptroni saa Nielsenin mukaan monta syötettä (*engl. input*) $x_1, x_2, ...x_n$ ja jokaiselle syötteelle on asetettu oma painoarvonsa (*engl. weight*) $w_1, w_2, ...w_n$, kuva 14. Nielesenin mukaan vanhat biologiaan pohjautuvat neuronit laskivat vain painotetun summan syötteistä ja painoarvoista lisäten niihin vakiotermin (*engl. bias*), jonka avulla määritettiin summalle neuronin aktivaatioraja.

$$\text{tuloste} = \begin{cases} 0, & \text{jos } \langle \mathbf{w}_n, \mathbf{x}_n \rangle + b \leq 0 \\ 1, & \text{jos } \langle \mathbf{w}_n, \mathbf{x}_n \rangle + b > 0, \end{cases} \tag{32}$$

missä $\mathbf{w}$ on neuronin painoarvot, $\mathbf{x}$ on neuronille syötetty datavektori ja b on vakiotermi.



Kuva 15: Sigmoidi- ja ReLU-aktivaatiofunktiot. Sigmoidifunktio asettaa minkä vain reaaliluvun nollan ja yhden välille. ReLU-funktio on nolla syötteen ollessa negatiivinen ja syötteen suuruinen syötteen ollessa positiivinen.

Aurélien Géronin [15] mukaan nykyaikaisissa neuroverkoissa neuronin painotettu summa syötetään neuronin *aktivaatiofunktiolle*, joka määrää miten paljon neuroni aktivoituu. Perinteisesti aktivaatiofunktiona on käytetty sigmoidifunktiota (ks. kuva 15a), joka muistuttaa Géronin mukaan biologisen neuronin toimintaa. Ian Goodfellow ja muut suosittelevat kirjassaan Deep Learning [21] nykyisissä neuroverkoissa vakiosuosituksena käytettävän ReLU, (*Rectified Linear Unit*) -funktiota, (ks. kuva 15b), joka määritellään

$$f(x) = max(0, x). \tag{33}$$

Heidän suosituksensa perustuu siihen, että usealla ReLU-aktivaatiofunktiolla voidaan kuvata epälineaarisia funktioita lineaarisilla komponenteilla, jolloin ne ovat

helposti optimoitavissa gradienttipohjaisilla optimointimenetelmillä.

Neuroverkko koostuu ainakin yhdestä kerroksesta, joka sisältää useita perseptroneja. Nielsen tiivistää, että neuroverkon kerroksen toiminta voidaan kirjoittaa lineaarialgebran keinoin asettaen kaikki painoarvot matriisiksi, aktivaatiot aktivaatiovektoriksi ja tekemällä vakiotermeistä oma vektorinsa.

$$h^{(i+1)}(\mathbf{X}^{(i)}) = f\left(\begin{bmatrix} w_{0,0}^{(i)} & w_{0,1}^{(i)} & \cdots & \cdots & w_{0,n}^{(i)} \\ w_{1,0}^{(i)} & w_{1,1}^{(i)} & \cdots & \cdots & w_{1,n}^{(i)} \\ \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ w_{n,0}^{(i)} & w_{n,1}^{(i)} & \cdots & \cdots & w_{n,n}^{(i)} \end{bmatrix} \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ \vdots \\ x_n^{(i)} \end{bmatrix} + \begin{bmatrix} b_0^{(i)} \\ b_1^{(i)} \\ \vdots \\ \vdots \\ b_n^{(i)} \end{bmatrix}\right), \tag{34}$$

missä $h^{(i+1)}(\mathbf{X}^{(i)})$ on kerroksen $i+1$ kaikki aktivaatiot käyttäen edellisen kerroksen i aktivaatioita. Aktivaatiofunktion f Nielsen on sanonut tulkittavan siten, että aktivaatiofunktiota käytetään jokaisen lopullisen vektorin elementeistä. Toistamalla tätä laskua voidaan laskea neuroverkon lopulliset aktivaatiot. Aurélien Géron on tiivistänyt tämän muotoon

$$h^{(i+1)}(\mathbf{X}^{(i)}) = f(\mathbf{W}^{(i)}\mathbf{X}^{(i)} + \mathbf{b}^{(i)}) \tag{35}$$

3.2.1 Sakkofunktio

Neuroverkko pitää kouluttaa, sillä sopivien painoarvojen ja vakiotermien määrittäminen manuaalisesti ei ole järkevää tai edes mahdollista. Nielsenin mukaan tämä tapahtuu alustamalla neuroverkko satunnaisilla painoarvoilla ja vakiotermeillä ja syöttämällä sille yksi datamatriisi $\mathbf{X}$ joka koostuu spektrien datavektoreista. Neuroverkko ei tietenkään tällöin tee hyvää ennustetta lopullisista tuloksista, mutta määrittämällä *sakkofunktio* (*engl. cost function/loss function*) voidaan tarkastella virheen suuruutta.

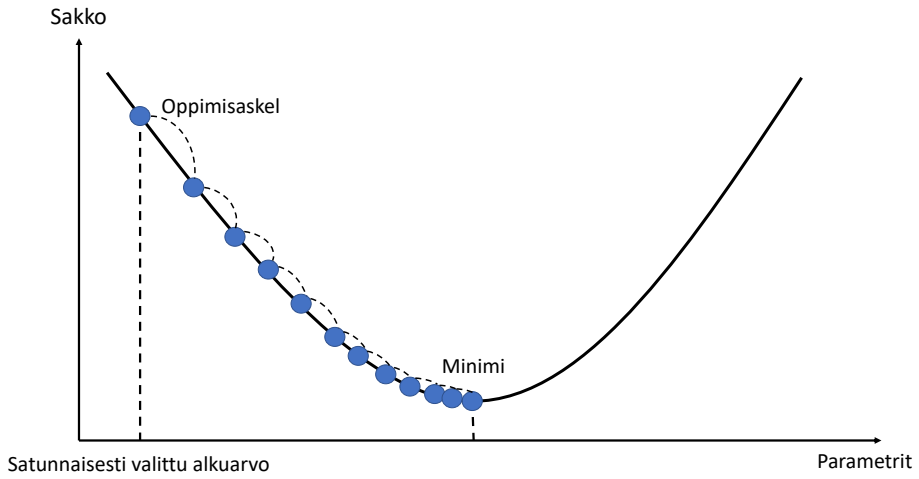
Sebastian Rashcka [17] antaa esimerkkinä neliöllisten virheiden sakkofunktion (*engl. Sum of Squared Errors, SSE*), jonka argumenttejä ovat painomatriisi $\mathbf{W}$ ja vakiotermivektori $\mathbf{b}$.

$$J(\mathbf{W}, \mathbf{b}) = \frac{1}{2} \sum_{i=1}^{n} (y_i + f(\mathbf{x}_i))^2 \tag{36}$$

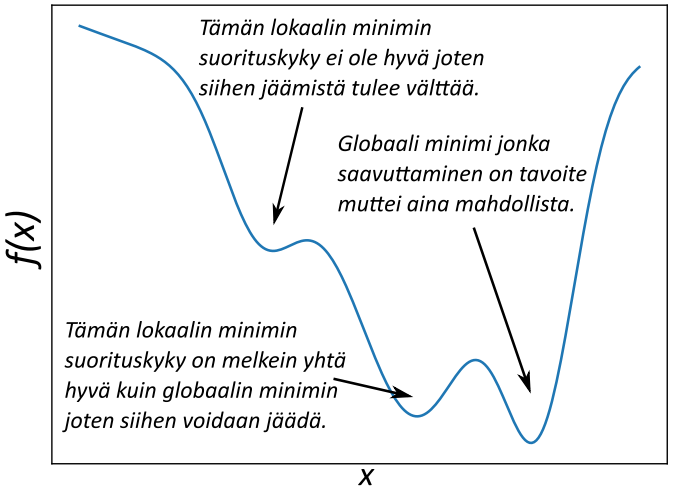
Funktion $f(\mathbf{x}_i)$ arvo on käytetyn aktivaatiofunktion tekemä ennuste ja y_i on datavektorin $\mathbf{x}_i$ todellinen luokitus. Sakkofunktio antaa virheen kullekin yksittäiselle datavektorille $\mathbf{x}_i$, tässä tapauksessa spektrille, ja laskee ne yhteen. Sakkofunktion minimoiminen tarkoittaa neuroverkon kouluttamista. Tätä varten täytyy laskea sakkofunktion gradientti.

3.2.2 Gradienttien käyttö

Sakkofunktion gradientin avulla voidaan määrittää mihin suuntaan mallin parametrejä pitää muuttaa, jotta virheen määrä pienenisi. Myös tukivektorikoneen ratkaisualgoritmi laskee kaavan (36) mukaisesti $J(\mathbf{w}, \zeta)$ gradientteja [15]. Sakkofunktion negatiivinen gradientti $-\nabla J(\mathbf{W}, \mathbf{b})$ osoittaa siihen suuntaan mihin sakkofunktio pienenee nopeimmiten. Gradienttimenetelmässä (*engl. gradient descent*) parametrejä muutetaan negatiivisen gradientin osoittamaan suuntaan jolloin sakkofunktion arvo on pienempi. Tätä toistetaan iteratiivisesti kunnes sakkofunktio on saavuttanut minimin eikä sen arvo enää muutu.



Kuva 16: Gradienttimenetelmässä mallin parametrit ovat aluksi satunnaisesti alustettuja ja niiden arvoja säädetään toistamiseen sakkofunktion minimin löytämiseksi. Askelpituus (*learning step*) on riippuvainen sakkofunktion gradientin suuruudesta. Mitä lähemmäs minimiä päästään, sitä pienempiä muutoksia parametreihin tehdään. Kuva pohjautuu lähteeseen [15].



Kuva 17: Gradienttimenetelmä saattaa olla löytämättä globaalia minimiä ja jäädä jumiin johonkin lokaaleista minimeistä. Kuva pohjautuu lähteeseen [21].

Ian Goodfellowin ja muiden [21] mukaan globaalin minimin löytäminen ei ole kuitenkaan varmaa, sillä sakkofunktion gradientti voi ohjata parametrejä pelkästään lokaaliin minimiin. Tällöin neuroverkko ei saavuta sen parhaita mahdollisia arvoja vaan jää jumiin. Goodfellow ja muut sanovat kuitenkin että useimmiten on hyväksyttävää, että ratkaisut eivät ole globaalisti optimeja kunhan sakkofunktion arvo on niissä huomattavasti ympäristöään pienempi.

François Chollet kirjassaan "Deep Learning with Python" [22] mainitsee kuitenkin, että koko koulutusjoukon käyttäminen ja sen gradientin laskeminen vaatii liikaa resursseja gradienttimenetelmälle sillä tavallisissa neuroverkoissa on parametrejä on satoja tai tuhansia. Aurélien Géron esittelee tähän ratkaisuksi stokastisen gradienttimenetelmän, jossa gradientti lasketaan kerrallaan vain yhdelle satunnaisesti valitulle koulutusjoukon instanssille. Gradientti on tällöin paljon helpompi laskea, ja painoarvojen ja vakiotermien muuttaminen on nopeampaa. Haittapuolena sakkofunktion käyttäyminen on myös epäsäännöllisempää. Géron huomauttaa kuitenkin että sakkofunktion satunnainen käyttäytyminen saattaa saada sen hyppäämään ulos lokaalista minimistä, jolloin sillä on parempi mahdollisuus löytää globaaliminimi. Stokastisessa gradienttimenetelmässä sakkofunktion arvot pienenevät ja kasvavat, mutta niiden keskiarvo lähestyy minimiä suhteellisen nopeasti. Se ei kuitenkaan asetu sinne, joten täydellistä minimiarvoa ei löydy vaan jotain sen läheisyydestä. Yksi ratkaisu tähän on Géronin mukaan parametreihin tehtyjen muutoksien pienentäminen pikkuhiljaa algoritmin edetessä.

3.2.3 Vastavirta-algoritmi

Neuroverkon parametrien muuttaminen tehdään käyttäen vastavirta-algoritmia (*engl. backpropagation*), jossa neuroverkko kuljetaan ikään kuin takaperin selvittäen mitkä painoarvot ja vakiotermit vaikuttivat eniten virheen suuruuteen. Parametrejä muutetaan siten, että virhe vähenee. Neuroverkkoa koulutettaessa tätä vaihetta toiste-

taan useita kertoja. Aurélien Géron kuvailee vastavirta-algoritmin toimintaa seuraavasti:

1. Koulutusjoukko jaetaan pienempiin eriin (*engl. mini-batch*) ja yksi eristä syötetään syötekerrokselle. Syötekerroksen jälkeen syötteet siirtyvät piilotetuille kerroksille, jotka aktivoituvat omien parametriensä mukaan. Tulostetaso aktivoituu viimeiseksi, ja sen aktivaatiot ovat ennusteita. Kaikki aktivaatiot, painoarvot ja vakiotermit pidetään muistissa sillä niitä tarvitaan vastavirta-algoritmin gradientin laskemisessa. Tämä on eteenpäinkulkuvaihe (*engl. forward pass*).

2. Algoritmi mittaa neuroverkon tulosteen virheen sakkofunktion avulla vertaamalla ennustettuja ja todellisia tuloksia.

3. Seuraavaksi algoritmi laskee kuinka paljon kukin tulostetason kytkentä edelliseen piilotettuun tasoon aiheutti virhettä. Tämä tapahtuu käyttämällä differentiaalilaskennan ketjusääntöä.

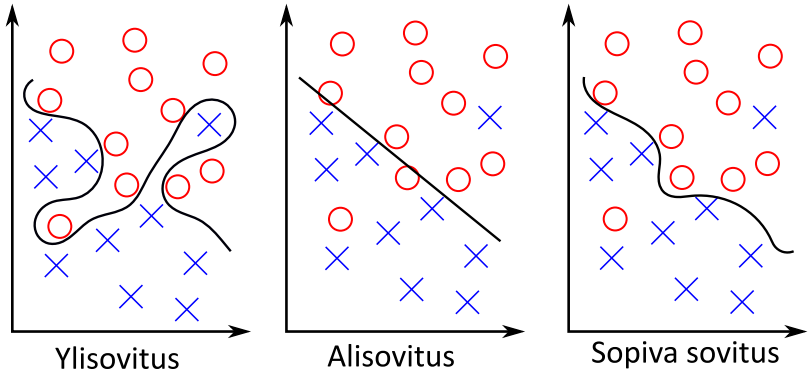
4. Tämän jälkeen algoritmi laskee edellisen piilotetun tason kytkentöjen virheen gradientin ketjusääntöä käyttäen. Näin jatketaan, kunnes algoritmi saavuttaa syötetason. Algoritmi kulkee ikään kuin vastavirtaan neuroverkon lävitse laskien kytkentöjen virheen. Tästä syystä sitä kutsutaan vastavirta-algoritmiksi (*engl. backpropagation*).

5. Algoritmi saa näin kaikkien kytkentöjen painoarvojen ja vakiotermien virheen gradientin. Gradienttimenetelmää käyttäen jokaista painoarvoa ja vakiotermiä muutetaan siten että virheen määrä laskee.

Tätä algoritmia toistetaan useita kertoja kunnes sakkofunktion on löytänyt minimin. Saadun minimin ei kuitenkaan tiedetä varmuudella olevan globaali minimi.

3.2.4 Regularisaatio

Neuroverkoissa on kymmeniätuhansia tai jopa miljoonia parametreja. Yhdessä niiden kyky kuvata hyvinkin monimutkaisia funktioita on erinomainen. Tavallisesti kuitenkin funktio, jota neuroverkko pyrkii approksimoimaan ei ole kuitenkaan niin monimutkainen kuin mitä neuroverkko pystyisi mallintamaan. Voi käydä niin että neuroverkko ylisovittaa koulutusjoukkoon eikä siten yleisty hyvin [15]. Tähän ratkaisuna on regularisaatio, joka pyrkii vähentämään mallin monimutkaisuutta [23].



Kuva 18: Neuroverkko saattaa yli- tai alisovittaa dataan. Neuroverkoilla ylisovittaminen on tavallista, joten sen välttämiseksi tulee käyttää regularisaatiota joka pyrkii yksinkertaistamaan mallia.

Regularisaatiossa sakkofunktioon $J(\mathbf{W}, \mathbf{b})$ lisätään termi, joka pyrkii minimoimaan painovektorin $\mathbf{W}$ suuruutta [23]. Tällöin yksittäisten aktivaatioiden suuruus pienenee ja mallista tulee yksinkertaisempi. Käyttäen L_2-regularisaatiota gradienttimenetelmällä minimoitavaksi termiksi tulee

$$J(\mathbf{W}, \mathbf{b}) = \frac{1}{2} \sum_{i=1}^{n} (y_i + f(\mathbf{x}_i))^2 + \lambda \sum_{i=1}^{n} ||\mathbf{w}_i||^2, \tag{37}$$

missä λ on regularisaatioparametri, jonka suuruutta muuttamalla mallin monimutkaisuutta voidaan säädellä [23]. Neuroverkot kykenevät sovittamaan hyvinkin monimutkaiseen dataan ongelmitta, mutta ilman regularisaatiota ylisovittamisen riski on suuri.

4 Spektrien luokittelu tekoälyllä

4.1 Mallit

Toteutin tukivektorikoneen ja neuroverkon Python-ohjelmointikielellä. Tukivektorikone pohjautuu pitkälti scikit-learn-pakettiin [24] ja neuroverkon toteutus perustuu Googlen Tensorflow-koneoppimispakettiin [25]. Lisäksi osa kuvaajien piirtämiseen käytetystä lähdekoodista on peräisin Matplotlib-paketista ja sen dokumentaatista [26]. Ohjelman lähdekoodi ja käyttämäni data on esitetty liitteessä 1.

4.1.1 Tukivektorikoneen toteutus

Tukivektorikone pohjautui scikit-learn-pakettin [24] tukivektorikoneimplementaatioon. Kirjoitin pakettia käyttäen seuraavan metodin, joka ottaa argumentteinaan koulutusjoukon, testijoukon, ennustemoodin, käytettävän häviötermin kertoimen ja valitun kernelifunktion.

```
1  from sklearn import svm
2  def Support_Vector_Machine(X_train, Y_train, X_test,Y_test, cost=1.0,
3  probabilistic=True, kernel_choice='rbf', return_model=False):
4
5      #initialize the model and fit the training data
6      clf=svm.SVC(C=cost,kernel=kernel_choice,probability=probabilistic)
7      model=clf.fit(X_train, Y_train)
8      #evaluate the performance using a test set
9      confidence = clf.score(X_test, Y_test)
10     #predict the test data
11     predictions=clf.predict_proba(X_test)
12     #if the model is needed then it can be returned as well for
13     #future predictions
14     if return_model:
15         return predictions, confidence, model
16     else:
17         return predictions, confidence
```

Metodille syötetään koulutusjoukko (`X_train`), sitä vastaavat oikeat luokittelut (`Y_train`), testijoukko (`X_test`) ja sitä vastaavat oikeat luokittelut (`Y_test`). Näi-

den lisäksi metodille kerrotaan, että luokitellaanko testidatapisteet automaattisesti, vaiko annetaan pisteen todennäköisyys kuulua kuhunkin luokkaan (`probabilistic=True`), mitä kernelifunktiota käytetään ja mikä on häviöfunktion parametrin C arvo (`cost`). Jos tukivektorikonetta käytetään todennäköisyysmoodissa niin se palauttaa ennusteelle todennäköisyyden jolla datapiste kuuluu kuhunkin luokkaan. Jos tarvitaan absoluuttisia ennusteita näytteen spektrien [1] ja taustan spektrien [0] välillä, on luokittelu tehtävä suurimman todennäköisyyden mukaan. Metodi palauttaa testijoukon ennusteet, sen arvion mallin hyvyydestä sekä tarvittaessa mallin, jota voi käyttää ennusteiden tekemiseen metodin ulkopuolella mikäli `return_model=True`. Tukivektorikoneen lähdekoodi on osittain peräisin lähteistä [15] ja [24] .

4.1.2 Neuroverkon toteutus

Tein neuroverkon käyttäen Googlen TensorFlow-pakettia [25] sekä François Cholletin keras-pakettia [27] joka on osaltaan rajapinta TensorFlow:n alemman tason metodeihin.

```
import tensorflow as tf
from tensorflow import keras
def Neural_network(X_train, X_test, Y_train, Y_test,
                   L1,
                   L2,
                   L3,
                   activation_function='relu',
                   reg_param=0.01,
                   optimizer_param='SGD',
                   epoch_n=10,
                   ):
        #if both L2 and L3 are 0 then the model has only one layer
    if L3==0 and L2==0:
         model = keras.Sequential([
        keras.layers.Dense(L1, activation=activation_function,
        kernel_regularizer=keras.regularizers.l2(l=reg_param)),
        keras.layers.Dense(2, activation='softmax')
    ])
```

```
    # if L3 is 0 then the model has two layers
    elif L3==0:
        model = keras.Sequential([
            keras.layers.Dense(L1, activation=activation_function,
            kernel_regularizer=keras.regularizers.l2(l=reg_param)),
            keras.layers.Dense(L2, activation=activation_function,
            kernel_regularizer=keras.regularizers.l2(l=reg_param)),
            keras.layers.Dense(2, activation='softmax')
        ])
    #otherwise the model has three layers
    else:
         model = keras.Sequential([
        keras.layers.Dense(L1, activation=activation_function,
        kernel_regularizer=keras.regularizers.l2(l=reg_param)),
        keras.layers.Dense(L2, activation=activation_function,
        kernel_regularizer=keras.regularizers.l2(l=reg_param)),
        keras.layers.Dense(L3, activation=activation_function,
        kernel_regularizer=keras.regularizers.l2(l=reg_param)),
        keras.layers.Dense(2, activation='softmax')
    ])

    #model is compiled here. The metrics[get_f1] correnspponds
    #to a method that
    #calculates the f1-score and the model uses that as a performance
    #metric instead of 'accuracy' or other metrics-
    model.compile(optimizer=optimizer_param,
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(
                  from_logits=False),
                  metrics=[get_f1])


    model.fit(X_train, Y_train, epochs=epoch_n)

    test_loss, test_acc = model.evaluate(X_test,  Y_test, verbose=2)

    print('\nTest accuracy:', test_acc)

    probability_model = tf.keras.Sequential([model,
                                             tf.keras.layers.Softmax()])
    predictions = probability_model.predict(X_test)

    correctly_predicted=evaluate_predictions(
    Y_test,binary_predictions_from_probabilistic(predictions ))
    return predictions, correctly_predicted, probability_model
```

Tämä metodin argumentteja ovat koulutusjoukot (`X_train, Y_train`) ja testijoukot (`X_test, Y_test`) samaan tapaan kuin tukivektorikoneessa. Argumentit `L1, L2, L3` ovat piilotettujen tason neuronien lukumäärä. Piilotettujen tasojen määrää voidaan vähentää asettamalla toisen ja/tai kolmannen kerroksen neuroneiden määräksi 0. Aurélien Géronin [15] mukaan jo yksi piilotettu taso pystyisi approksimoimaan mitä tahansa funktiota, mutta se ei ole yleensä järkevää sillä neuroneita tarvittaisiin erittäin paljon, eikä malli todennäköisesti yleistyisi hyvin. Hän valaisee, että piilotettujen tasojen määrä riippuu sovelluskohteesta mutta se on tavallisesti yhdestä viiteen. Käytän neuroverkossani yhdestä kolmeen piilotettua tasoa. Géronin mukaan neuroneiden määrä laskee usein piilotetulta tasolta toiselle mutta hän huomauttaa nykyään monessa neuroverkossa olevan vakiomäärä neuroneita jokaisella piilotetulla tasolla. Mallissani neuroneiden aktivaatiofunktiota on mahdollista vaihtaa, mutta sen oletusarvo on ReLU, koska se on kaikkein tavallisin aktivaatiofunktio nykyään [17]. Tulosteessa olen käyttänyt kahta neuronia, jotka vastaavat kahta mahdollista luokittelun luokkaa eli näytteen ja taustan spektriä. Tulostetasolla olen käyttänyt Softmax-aktivaatiofunktiota, joka antaa kunkin luokan todennäköisyyden välillä $[0, 1]$ ja pitää huolen että todennäköisyyksien summa on 1 [15]. Softmax-aktivaatiofunktio

$$p(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{38}$$

antaa kunkin luokan todennäköisyyden neuronien aktivaatiosta. Neuroneiden aktivaatiot itsessään eivät anna todennäköisyyksiä, sillä niiden summa voi olla suurempi kuin 1. Tällöin Softmax-aktivaatiofunktio muuttaa aktivaatiot todennäköisyyksiksi riippuen kunkin luokan aktivaatioiden keskinäisistä suhteista. Tavallisesti Softmax-aktivaatiofunktiota käytetään monen luokan luokittelun tulosteeen aktivaatiofunktiona [17], mutta päädyin käyttämään sitä binääriseen luokitteluongelmaan. Tavallisesti binäärisessä luokitteluongelmassa käytetään sigmoidi-funktiota, mutta

halusin mahdollistaa mallilleni spektrien luokittelun useampaan kuin kahteen luokkaan. Vaikka tässä tutkielmassa kyseistä tapausta en käsittelekään, olen ottanut tämän huomioon mahdollisten tulevaisuuden sovelluskohteiden vuoksi.

Regularisaatioparametrin `reg_param=0.01` arvoa muuttamalla voidaan mallin regulariaation suuruutta säädellä. Mitä suurempi regularisaatioparametrin on, sitä yksinkertaisemman mallin neuroverkko tuottaa. Tämä pyrkii vähentämään ylisovittamista. Myös optimointialgoritmia `optimizer_param='SGD'` on mallissani mahdollista muuttaa, mutta koska käytetty datajoukko on varsin pieni, käytännössä mikä tahansa gradienttimenetelmä olisi toimiva. Olen valinnut käyttää stokastista gradienttimenetelmää sen nopeuden takia.

Aurélien Géron [15] suosittelee binäärisessä luokittelussa käytettävän ristientropiasakkofunktiota (*engl. cross entropy*) eli tässä tapauksessa binääristä ristientropiaa. Olen kuitenkin käyttänyt sakkofunktiona kategorista ristientropiaa, joka sallisi useammankin kuin kahden luokan käytön luokittelijalla. Mikäli luokkia on vain kaksi on se identtinen binäärisen ristientropian kanssa [28]. Käytetty `sparse_categorical_cross_entropy` -sakkofunktio tekee juuri tämän ja osa `sparse` tarkoittaa että luokkien ei tarvitse olla ”one-hot”-enkoodattuja, jossa luokka 0 kirjoitettaisiin $[1, 0]$ ja luokka 1 kirjoitettaisiin $[0, 1]$ [29].

Käytettynä metriikkana, eli mallin hyvyyden parametrina, käytin F_1-arvoa, jonka esittelen tarkemmin seuraavassa osiossa. F_1-arvo ei ole enää yksi Tensorflown sisäänrakennetuista metriikoista, joten käytin koodissani erillistä metodia F_1-arvon laskemiseksi, joka on peräisin Aakash Goelin artikkelista [30]. Neuroverkon lähdekoodi on osittain peräisin lähteistä [15] ja [31].

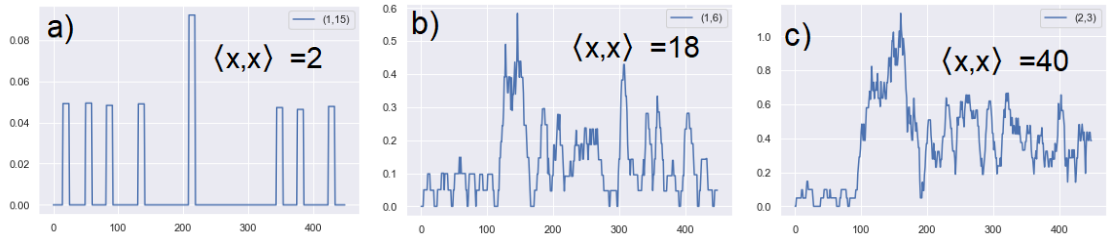
4.2 Datan käsittely ja koulutusjoukon luominen

Tukivektorikone ja neuroverkko tarvitsevat koulutusjoukon, jossa spektrin datavektori on yhdistetty sitä vastaavaan luokkaan. Ohjelmassani on kaksi tapaa luoda

koulutusjoukko. Kummassakin tapauksessa ohjelma piirtää yhden spektrin kuvaajan siten, että spektrin kohinaa on vähennetty Pythonin scipy-paketin l-filtterin avulla [32]. Tämä helpottaa käsin tehtävää luokittelua, sillä piikin rakenne voi olla vaikea nähdä kohinan alta. Luokittelu tapahtuu kirjoittamalla komentoriville 0 tai 1 riippuen siitä, onko kyseinen spektri peräisin taustasta vai näytteestä. Luokitus ja sitä vastaava spektri tallennetaan omiin listoihinsa, jotka lopulta tallennetaan omiksi tiedostoikseen.

Koulutusjoukko on mahdollista luoda joko yhdelle tarkastelualueelle kerrallaan, jolloin kyseisen alueen kaikki spektrit luokitellaan käsin. Tämä vaatii tavallisesti 100-300 luokittelua. Tämän ongelmana on kuitenkin se, että näytteestä peräisin olevia spektrejä saattaa olla huomattavasti vähemmän kuin taustan spektrejä mikä tekisi koulutusjoukosta epätasapainoisen. Lisäksi yhden tarkastelualueen spektrit saattavat poiketa toisten tarkastelualueiden spektreistä, ja mallien yleistyminen koko dataan saattaa näin menetellen kärsiä.

Toinen tapa rakentaa koulutusjoukko on valita jokaisesta tarkastelualueesta muutama spektri jotka luokitellaan. Jos spektrit valittaisiin tarkastelualueelta satunnaisesti, taustan spektrien suhde näytteen spektreihin olisi huomattavasti suurempi. Eli pisteitä ei voida valita satunnaisesti. Tiedämme kuitenkin, että pienen lasketataajuuden spektri ei voi olla näytteestä. Eli näytteen spektrin datavektorin pituuden itseisarvon pitää olla varsin suuri. Tästä syystä spektrit, joiden pistetulo, eli normin neliö, itsensä kanssa on pieni, voidaan automaattisesti luokitella taustan spektreiksi. Pitää kuitenkin, että suuri pistetulon $\langle \mathbf{x}, \mathbf{x} \rangle = ||\mathbf{x}||^2$ arvo ei takaa sitä, että $\mathbf{x}$ on peräisin näytteestä.



Kuva 19: a) Kun vektorin pistetulo itsensä kanssa on pieni spektri ei sisällä hyödyllistä informaatiota. Tämä spektri on selvästi taustan spektri. b) Suuremmalla pistetulon arvolla spektrin muodot alkavat erottua paremmin. Tämänkaltaisten spektrien käsin tehtävässä luokittelussa on vaikea olla johdonmukainen. c) Spektrit joilla on suurimmat pistetulot ovat useimmiten näytteestä ja niiden luokittelu on helppoa.

Olen toteuttanut tämän laskemalla kunkin kiinnostavan alueen spektrien pistetulot ja suodattanut pois tietyn osuuden (40%) niistä spektreistä, joiden pistetulot olivat pienimpiä. Jäljelle jäävistä spektreistä valitaan parametrin $n = 2 - 8$ spektriä luokiteltavaksi tasaisin välimatkoin niiden normien neliöiden perusteella. Kun $n = 2$, suodatuksen jälkeisistä spektreistä luokitellaan käsin kaksi spektriä, se jonka arvo on suurin ja se jonka arvo on pienin. Kasvattamalla parametrin n arvoa voidaan näiden kahden spektrin välistä valita lisää spektrejä luokiteltavaksi perustuen niiden normin neliöön.

Tällä tavalla jokaisesta tarkastelualueesta voidaan valita spektrejä koulutusjoukkoon ja samalla myös taustan ja näytteen spektrien suhde muuttuu paremmaksi.

Aurélien Géronin [15] mukaan sekä neuroverkko että tukivektorikone ovat malleina molemmat herkkiä datan skaalalle. Tästä syystä hänen mukaansa datan arvojen skaalaaminen tietylle välille on tarpeellista. Hänen mukaansa yksi tavallisimmista tavoista normalisoida dataa on niin kutsuttu minimi-maksimiskaalaus, jossa kaikki arvot normalisoidaan välille [0,1] seuraavasti:

$$x' = \frac{x - \min(\mathbf{X})}{\max(\mathbf{X}) - \min(\mathbf{X})}. \tag{39}$$

Toteutuksessani maksimiarvo etsitään koko koulutusjoukosta eikä jokaisesta spektristä erikseen. Tällöin kaikkia spektrejä skaalataan samassa suhteessa.

Tässä sovelluskohteessa ohjatun oppimismallin ongelma on se, ettei koulutusjoukon luominen ole ongelmatonta. Ihmiselle spektrien luokitteleminen ei ole helppoa, toisin kuin vaikka kissojen ja koirien tunnistaminen valokuvista. Lisäksi spektrien hyvyys taustan ja näytteen spektrien välillä on ainakin jossain määrin liukuva suure. Erityisen selvät tapaukset on helppo tunnistaa, mutta niiden väliin jää suurin osa spektreistä jotka sisältävät sekä näytteen että taustan spektrien ominaisuuksia. Spektrin hyvyyden silmämääräinen määrittäminen koulutusjoukkoa luodessa on vaikeaa ja tämä johtaa siihen, että manuaalisen luokittelijan kriittisyys vaihtelee koulutusjoukon luonnin aikana. Omasta kokemuksestani koulutusjoukkoa luodessani välillä hyväksyin joitain spektrejä liian helposti näytteestä peräisin oleviksi ja taas välillä luokittelin vastaavia spektrejä taustan spektreiksi. Luokittelun aikana tapahtuu helposti väsymystä, ja jos luokiteltavien spektrien määrä on suuri syntyy helposti vinoumia luokittelussa.

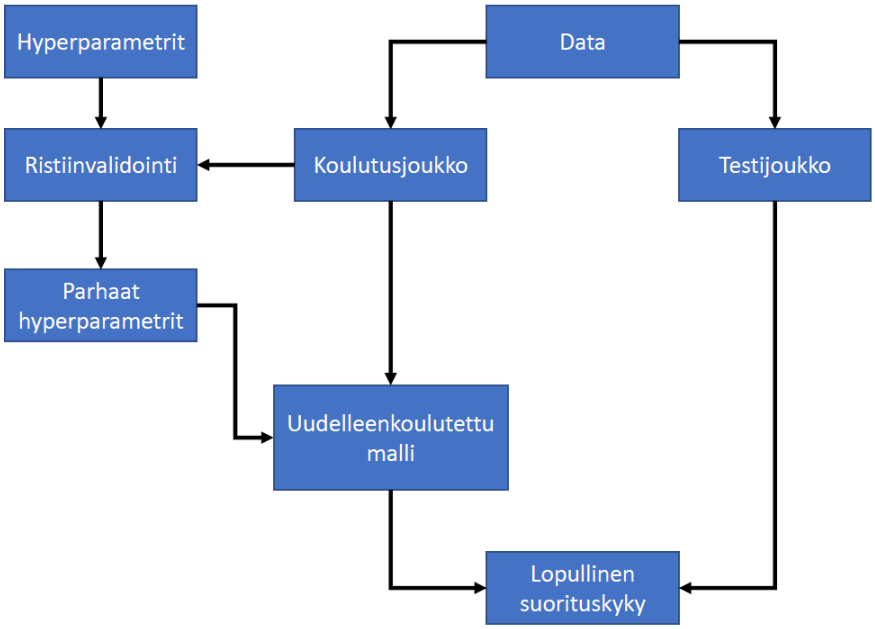
Tämä ongelma on kuitenkin sama, johon tutkija törmää hänen luokitellessaan spektrejä itse niiden yhteenlaskua varten. Hänkään ei ole täydellinen luokittelija, joten koneoppimismallin ei tarvitse pyrkiä täydellisyyten, vaan olemaan kilpailukykyinen tutkijan heuristiikan kanssa. Lisäksi muutamat väärät luokittelut koulutusjoukossa eivät automaattisesti pilaa mallia. Arvelen että koulutuksen virhe saattaa kasvaa, mutta testauksen virheen kasvua ei välttämättä tapahdu.

4.3 Mallin kouluttaminen ja validointi

Kirjassaan ”Deep Learning with Python” [22] François Chollet kertoo, että koko datajoukkoa ei tule käyttää mallin koulutukseen. Mallin suorituskykyä pitää kyetä arvioimaan jollain datalla ja hänen mukaansa tähän ei tule käyttää samaa dataa, jota käytettiin mallin kouluttamiseen. Cholletin mukaan tämä johtaa mallin yliso-

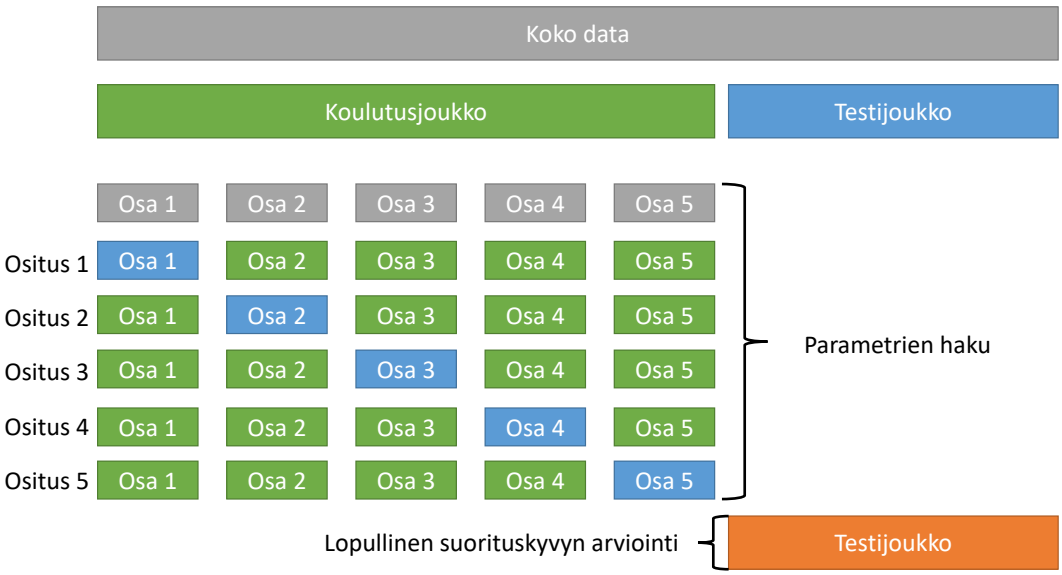
vittamiseen, jossa malli oppii koulutusjoukon ulkoa eikä yleisty datalle, jota se ei ole ennalta nähnyt. Siksi osa datajoukosta pitää jättää sivuun mallin suorituskyvyn testaamista varten.

Hänen mukaansa mallin suorituskyvyn arviointi vaatii datan jakamisen kolmeen osaan: koulutus-, validaatio- ja testijoukkoon. Malli koulutetaan koulutusjoukolla ja sen suorituskykyä arvioidaan validaatiojoukollla. Tätä vaihetta toistetaan tavallisesti useita eri kertoja käyttäen mallissa eri hyperparametrien yhdistelmillä. Kyseeseen tulevia hyperparametrejä ovat esimerkiksi neuroneiden määrää neuroverkon tasoilla, aktivaatiofunktiot tai tukivektorikoneen kohdalla häviöfunktion painoarvon suuruus tai käytetty kernelifunktio. Kun hyperparametrit jotka antavat parhaan tuloksen validaatiojoukolla on löydetty, malli koulutetaan niillä ja mallia testataan testijoukkoa käyttäen.



Kuva 20: Tavallinen tapa toteuttaa hyperparametrien haku. Parhaat hyperparametrit haetaan ristiinvalidointia käyttäen. Lopuksi mallia testataan datalla jota se ei ole aiemmin nähnyt sen suorituskyvyn arvioimiseksi. Kuva pohjautuu lähteeseen [33].

Validaatiojoukko on Cholletin mukaan tarpeellinen koska hyperparametrien valinnassa informaatiota vuotaa testaamiseen käytetystä validaatiojoukosta koulutusjoukkoon. Näin ollen malli voi saada informaatiota validaatiojoukosta ja lopulta ylisovittaa siihen. Mallin toiminnassa ollaan kuitenkin kiinnostuneita sen suorituskyvystä täysin ennaltatuntemattomalla datalla eli tässä tapauksessa testijoukkolla. Testijoukko on täysin erillään muusta ennen sen lopullista testaamista, joten malli ei voi saada siitä tietoa etukäteen ja siten ylisovittaa siihen.



Kuva 21: Ristiinvalidoinnin toimintaperiaate, kaikesta datasta ositetaan koulutusjoukko ja testijoukko. Koulutusjoukko ositetaan uudestaan useampaan osaan, joista jokainen toimii vuorollaan validaatiojoukkona kun malli koulutetaan muilla osituksilla. Kun jokainen ositus on toiminut validaatiojoukkona niiden suorituskyvyn keskiarvo tallennetaan kyseisen hyperparametriyhdistelmän suorituskyvyksi. Tätä vaihetta toistetaan useilla hyperparametriyhdistelmillä, kunnes malli koulutetaan käyttäen koko koulutusjoukkoa ja parhaita hyperparametrejä. Lopullinen suorituskyky arvioidaan käyttäen sivussa pidettyä testijoukkoa. Kuva pohjautuu lähteeseen [33].

Hyperparametrien valinnassa käytetään Cholletin mukaan tavallisesti K-ositettua ristiinvalidointia, jota kuva 21 havainnollistaa. Koulutusjoukko jaetaan K eri osaan, joista yksi jätetään sivuun validointijoukoksi ja malli koulutetaan lopuilla K-1 osalla. Mallin suorituskykyä arvioidaan validointijoukolla ja tulos tallennetaan. Tämän jälkeen toinen osa jätetään validointijoukoksi ja malli koulutetaan lopuilla osilla. Menettelyä toistetaan kunnes jokainen osa on ollut kerran validointijoukko. Tällöin mallin suorituskyvyn arvo on osavalidointien tuottama keskiarvo. Hyperparametrien optimoinnissa tätä vaihetta toistetaan kullekin hyperparametrien yhdistelmälle. Lopulta malli koulutetaan sillä hyperparametriyhdistelmällä jonka suorituskyvyn keskiarvo on paras ja mallia testataan testijoukkoa käyttäen.

Hyperparametrien etsiminen ristikkohaulla (*engl. grid search*) vaatii paljon laskenta-aikaa. Varsinkin neuroverkon kohdalla haku on raskas, sillä optimoitavia hyperparametreja ovat kerroksien ja niiden neuronien määrä, regularisaatioparametrin arvo sekä käytetty aktivaatiofunktio.

Mallin optimointia varten pitää valita jokin metriikka, jonka suhteen mallia optimoidaan. Kaikkein yksinkertaisin mittari on tarkkuusmetriikka (*engl. accuracy*), joka vertaa oikein ennustettujen ennusteiden suhdetta kaikkiin ennusteisiin. Aurélien Géronin [15] mukaan tarkkuus on kuitenkin helposti harhaanjohtava datajoukoilla, joissa luokkien keskinäiset suhteet ovat epätasapainossa. Hän on esitellyt sekaannusmatriisiin (*engl. confusion matrix*) perustuvan F_1-metriikan. Sekaannusmatriisissa todellisia luokkia verrataan ennustettuihin luokkiin.

Taulukko 3: Sekaannusmatriisi. Vasemmassa ylänurkassa on todelliset positiiviset tulokset ja oikeassa alanurkassa todelliset negatiiviset tulokset. Vasemmassa alanurkassa on väärät positiiviset ja oikeassa ylänurkassa väärät negatiiviset ennusteet. Taulukon luvut ovat parhaan tukivektorikoneen testijoukon ennusteiden tulokset. Täsmällisyys on $\frac{32}{33}$ ja saanti on $\frac{32}{36}$, jolloin sen F_1-arvo on $0,9275$.

		Ennustetut luokat		
		1	0	Yhteensä
Todelliset luokat	1	32	4	36
	0	1	40	41
	Yhteensä	33	44	77

Sekaannusmatriisin avulla Géron määrittää luokittelijan tarkkuuden (*engl. accuracy*), täsmällisyyden (*engl. precision*) ja saannin (*engl. recall*).

$$\text{tarkkuus} = \frac{\text{todelliset positiiviset} + \text{todelliset negatiiviset}}{\text{kaikkien todellisten ja väärien summa}} \tag{40}$$

$$\text{täsmällisyys} = \frac{\text{todelliset positiiviset}}{\text{todelliset positiiviset} + \text{väärät positiiviset}} \tag{41}$$

$$\text{saanti} = \frac{\text{todelliset positiiviset}}{\text{todelliset positiiviset} + \text{väärät negatiiviset}} \tag{42}$$

Täsmällisyyden arvo kasvaa kun väärien positiivisten määrä laskee. Samoin saannin arvo kasvaa kun väärien negatiivisten määrä laskee. Géronin mukaan täsmällisyyden ja saannin harmoninen keskiarvo on F_1-arvo

$$F_1 = \frac{2}{\frac{1}{\text{täsmällisyys}} + \frac{1}{\text{saanti}}} = 2 \cdot \frac{\text{täsmällisyys} \cdot \text{saanti}}{\text{täsmällisyys} + \text{saanti}}. \tag{43}$$

Géronin mukaan F1-arvon maksimointi suosii luokittelijoita, joiden täsmällisyys ja saanti ovat samankaltaisia. Spektrien luokittelua varten haluamme minimoida mahdollisuuksien mukaan vääriä negatiivisia ja vääriä positiivisia ennusteita. Arvokasta

informaatiota menetetään mikäli näytteen spektrejä luokitellaan virheellisesti taustan spektreiksi. Taustan spektrien luokittelemin näytteen sen sijaan lisää kohinan vaikutuksia summattuun spektriin. Géron sanoo, että täsmällisyyttä ja saantia ei voi yhtä aikaa nostaa mielivaltaisen hyviksi. Useimmiten täsmällisyyden nostaminen laskee saantia ja saannin nostaminen laskee täsmällisyyttä. Näin ollen F_1-metriikka tasapainoittelee täsmällisyyden ja saannin välillä soveltuen luokitteluun, jossa molempia pyritään maksimoimaan.

5 Tulokset

Lopullinen spektrien luokittelu tehtiin koulutusjoukolla jossa oli 384 käsin luokiteltua spektriä, kahdeksan jokaisesta tarkastelualueesta. Hyperparametrien ruudukkohaussa käytettiin arvoja joiden läpi käymiseen ei menisi kohtuutonta määrää aikaa. Tukivektorikoneella hyperparametrejä olivat sakkofunktion parametrin C arvot ja kernelifunktion tyyppi.

$$C = 1, 3, 5, 10, 25, 50, 75, 100, 200, 250, 400, 500, 750, 1000, 2000, 5000, 10000$$

$$\text{kernelifunktio} = \text{lineaarinen}, \text{polynomaalinen}, \text{Gaussin radiaalikantafunktio}$$

Neuroverkolle optimoitavia hyperparametreja oli huomattavasti enemmän ja niiden ruudukkohaku vei huomattavasti aikaa. Optimoitavia hyperparametreja olivat piilotettujen tason määrä, neuronien määrä kullakin piilotetulla tasolla, regularisaatioparametrin arvo ja aktivaatiofunktion tyyppi. Päätin käyttää vain kahta tavallisinta aktivaatiofunktiota, ReLUa ja sigmoidifunktiota. Hyperparametrien ruudukkohaku tehtiin erikseen sekä jää- että vesinäytteelle. Kumpikin koodin ajo vei noin kaksikymmentä tuntia, josta melkein kaikki aika kului neuroverkon hyperparametrien ruudukkohakuun.

$$\text{neuronien määrä 1. tasolla} = 600, 500, 450, 400, 350, 300, 250$$

$$\text{neuronien määrä 2. tasolla} = 350, 300, 250, 200, 150, 100, 75, 50, 25, 0$$

$$\text{neuronien määrä 3. tasolla} = 150, 100, 75, 50, 35, 25, 10, 0$$

$$\text{aktivaatiofunktio} = \text{ReLu}, \text{sigmoidi}$$

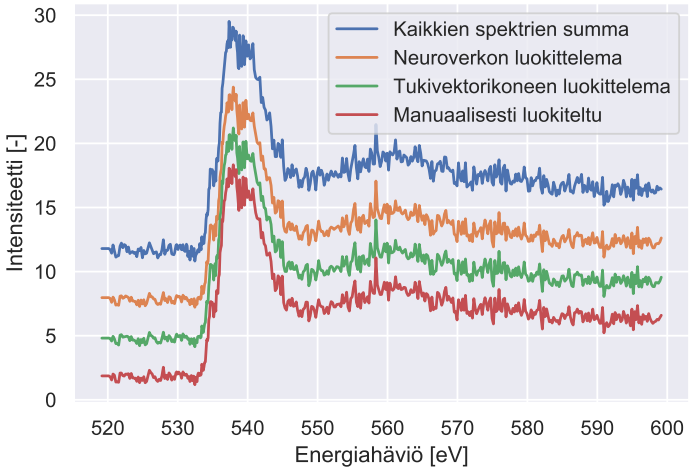
$$\text{regularisaatioparametri} = 10000, 1000, 100, 10, 1, 0.1, 0.01, 0.001, 0.0001$$

Tukivektorikoneella Gaussin radiaalikantafunktio oli paras kernelifunktio vesi-

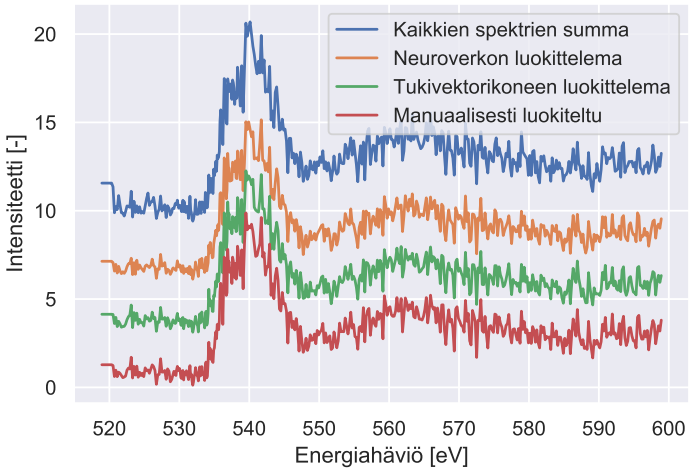
näytteelle mutta lineaarinen kantafunktio oli jäänäytteen luokittelussa paras. Sakkofunktion parametrin C arvo vaihteli varsin paljon, jäälle se oli 10000 mutta vesinäytteelle vain 10.

Neuroverkon kohdalla ReLU-aktivaatiofunktio oli järjestelmällisesti sigmoidifunktiota parempi aktivaatiofunktio. Tämä ei yllätä sillä esimerkiksi Géron [15] ja Chollet in [22] mukaan ReLU on nykypäivänä tavallisin aktivaatiofunktio joka toimii useimpiin tarkoituksiin erittäin hyvin. Parhaan F_1-arvon antanut hyperparametriyhdistelmä vaihteli kuitenkin näytteiden välillä. Jäänäytteellä piilotetuilla tasoilla 1., 2. ja 3. oli 600, 150 ja 10 neuronia kun taas vesinäytteellä parhaat neuronimäärät olivat 500, 300, 25. Mielestäni neuronien määristä on vaikea tehdä mitään johtopäätöksiä ja optimaalinen määrä neuroneita on useimmiten tapauskohtausta. Regularisaatioparametri oli molemmissa pieni, jäänäytteelle sen arvo oli 0,0001 ja vesinäytteelle se sain arvon 0,01. Mikäli mallilla olisi taipumusta ylisovittaa, niin mielestäni regularisaatioparametrin arvo olisi varsin suuri. Tässä tapauksessa regularisaatioparametrin arvot ovat pienet, joten en usko että mallilla on juurikaan taipumusta ylisovittaa koulutusjoukkoon. Tämä voi johtua siitä, että spektridata on varsin kohinaista.

Kun parhaat hyperparametrit oli löydetty niillä koulutettua mallia käytettiin uusien spektrien luokitteluun. Kummallekin mallille syötettiin yhden tarkastelualueen kaikki spektrit, jotka normalisoitiin luokittelua varten samaan tapaan kuin koulutusjoukko oli normalisoitu, eli välille [0,1]. Normalisoituja spektrejä, jotka oli luokiteltu hyviksi, vastaavat alkuperäiset spektrit summattiin yhteen. Olin luokitellut kyseisen tarkastelualueen spektrit myös käsin, ja näytteen spektreiksi luokittelemani spektrit summattiin yhteen, ks. kuvat 22 ja 23.

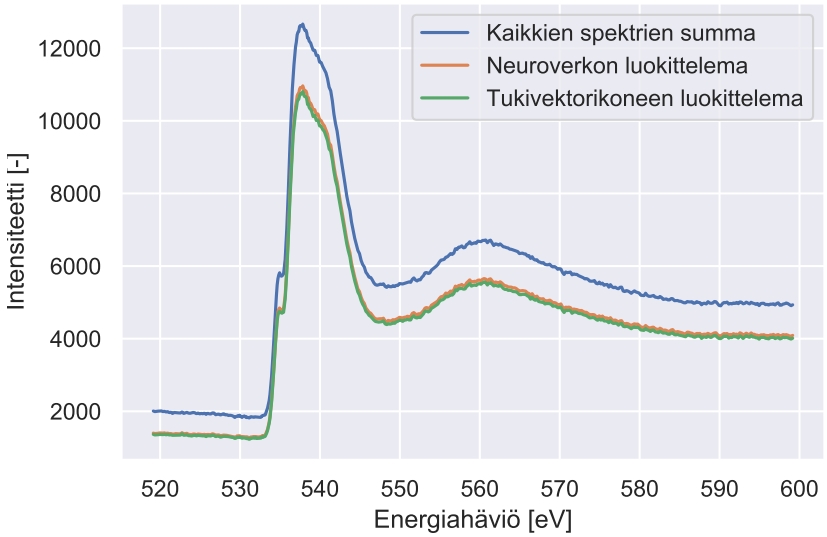


Kuva 22: Nestemäisen veden näytteestä sironneiden spektrien summa käsin luokiteltuna sekä tukivektorikoneen ja neuroverkon luokittelemana yhdeltä tarkastelualueelta. Kaikkien spektrien summa on lisätty vertailua varten ja sen, tukivektorikoneen ja neuroverkon summaspektrejä on kuvassa nostettu, jotta ne eroittuisivat paremmin muista spektreistä.

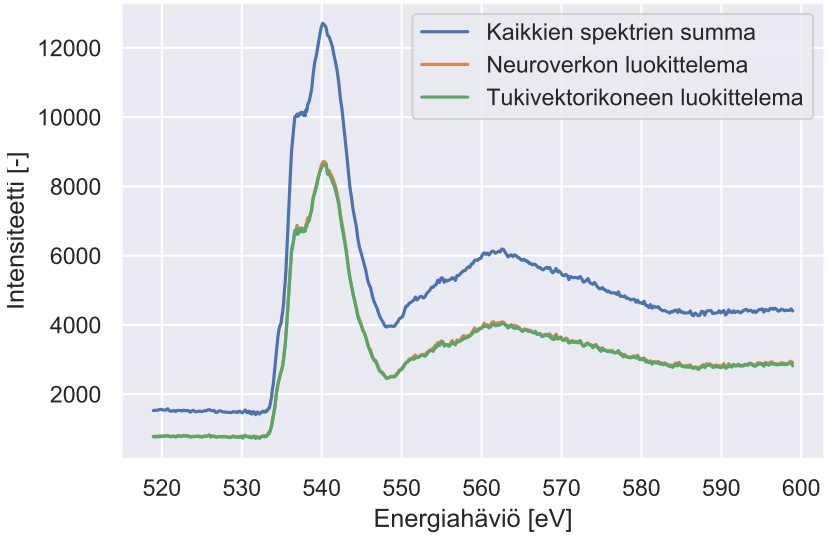


Kuva 23: Jään näytteestä sironneiden spektrien summa käsin luokiteltuna sekä tukivektorikoneen ja neuroverkon luokittelemana yhdeltä tarkastelualueelta. Kaikkien spektrien summa on lisätty vertailua varten ja sen, tukivektorikoneen ja neuroverkon summaspektrejä on kuvassa nostettu, jotta ne eroittuisivat paremmin muista spektreistä.

Kukin spektrien summa on esitetty vesinäytteelle kuvassa 22 ja jäänäytteelle kuvassa 23. Molemmilla malleilla tehdyt summaspektrit vastaavat käsin luokiteltua spektriä hyvin läheisesti. Koska taustaa antavat spektrit ovat useimmiten intensiteetiltään pieniä, niidenkin ottaminen mukaan summaamiseen ei muuta spektrin muotoa, jota hyvien spektrien kertaluokkia suurempi intensiteetti dominoi. Taustan spektrit ovat kuitenkin satunnaiskohinaa, joka saattaa piilottaa spektrien pienempiä ominaisuuksia alleen. Tosin jos taustalla olisi piikki näytteen piikin kanssa samalla energia-alueella, piikkejä ei voitaisi erottaa toisistaan.



Kuva 24: Kaikkien spektrien, jotka tukivektorikone ja neuroverkko ovat luokitelleet sironneen näytteestä, summa nestemäisen veden näytteellä. Luokiteltavia spektrejä on yli kymmenentuhatta, joten käsin luokiteltua summaspektriä ei ole tehty.

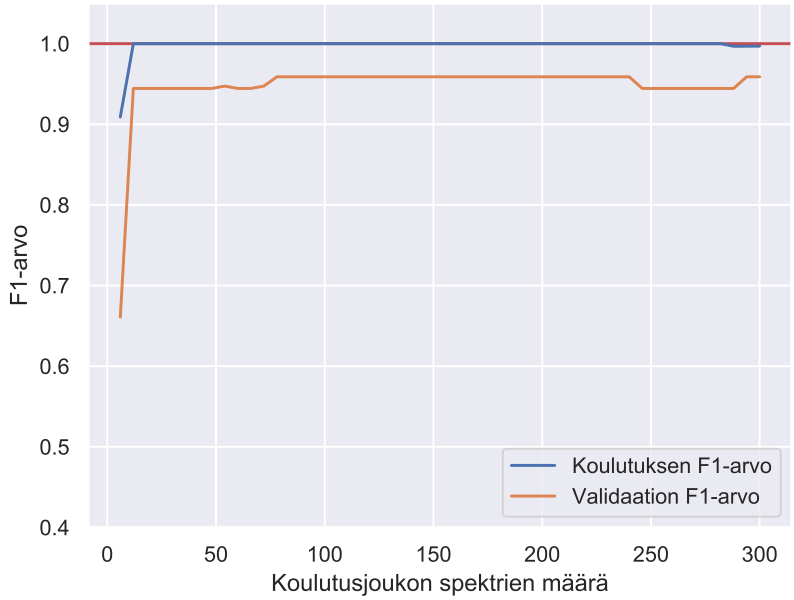


Kuva 25: Kaikkien spektrien, jotka tukivektorikone ja neuroverkko ovat luokitelleet sironneen näytteestä, summa jäänäytteellä.

Kuvissa 24 ja 25 on esitetty kummankin luokittelijan luomat summaspektrit koko datasta verrattuna kaikkien pikselien sokeaan yhteenlaskuun. Molemmille näytteille tukivektorikone ja neuroverkkoa loivat käytännössä identtiset spektrit. Mielestäni mallit luokittelevat käytönnässä samat spektrit näytteen spektreiksi ja samat taustan spektreiksi ja uskon pienten erojen tulevan rajatapausten käsittelyssä. Tukivektorikone ja neuroverkko ovat molemmat onnistuneet vähentämään taustan vaikutusta kaikkien spektrien summaamiseen verrattuna, tämä on nähtävissä kuvien 24 ja 25 spektrien energia-alueella 520-530 eV.

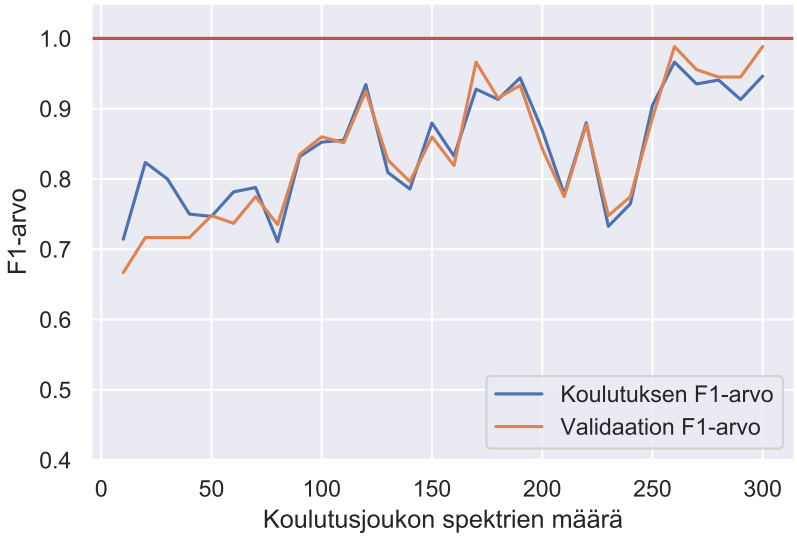
5.1 Oppimiskäyrät ja koulutusjoukon suuruuden vaikutus summaspektreihin

Parhaita hyperparametriyhdistelmiä käyttäen tutkin molempien mallien oppimiskäyrien käyttäytymistä, jotka on esitetty kuvissa 26 ja 27. Oppimiskäyrissä mallin suorituskykyä (F1-arvoa) koulutuksen aikana ja testijoukolla arvioidaan, kun koulutusjoukon spektrien määrää kasvatetaan. Odotuksena on, että koulutuksen ja validaation F1-arvot eivät kasva enää tietyn pisteen jälkeen, eli malli ei hyödy koulutusjoukon kasvattamisesta. Lisäksi koulutuksen ja validaation performanssien pitäisi olla samaa luokkaa, jotta voidaan todeta että malli ei ylisovita koulutusjoukon dataan [17].



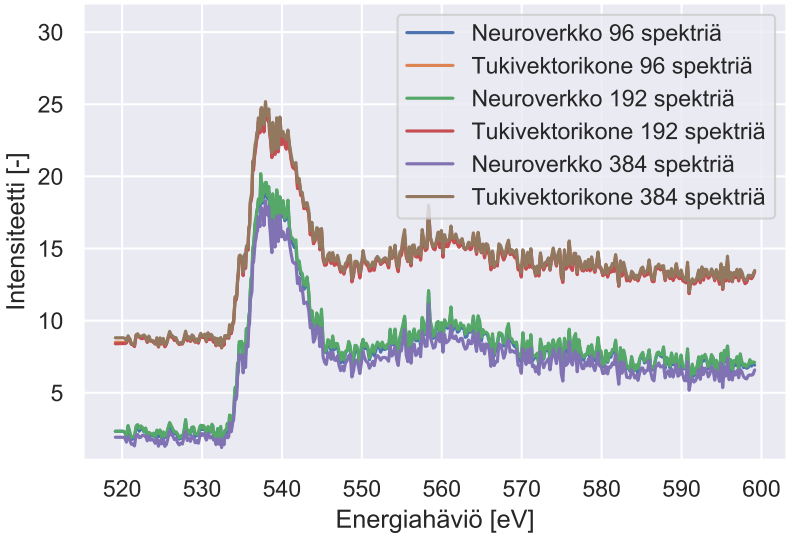
Kuva 26: Tukivektorikoneen oppimiskäyrä nestemäisen veden näytteellä. Paras mahdollinen F_1-arvo on 1, joka on piirretty punaisella viivalla.

Oppimiskäyristä havaitaan, että tukivektorikoneen F_1-arvo on erittäin hyvä jo pienillä koulutusjoukoilla. Pienille koulutusjoukoille on mahdollista luoda hypertaso sopivan kernelimuunnoksen kanssa, jolloin kaikki koulutuksen datapisteet luokitellaan oikein. Malli siis ylisovittaa dataa, ja huomataan, että kun koulutusjoukossa on n. 290 datapistettä kaikkia koulutusjoukon pisteitä ei voida enää luokitella oikein. Koulutusjoukon datapisteet ovat kuitenkin valittu satunnaisesti kaikesta datasta, joten niiden jakauma on suurin piirtein sama kuin validaation datapisteiden. Näin ollen ylisovittamisesta huolimatta tukivektorikone luokittelee dataa, jota se ei ole aiemmin nähnyt varsin tehokkaasti. Tukivektorikone toimii pienelläkin koulutusjoukolla hyvin, sillä teoriassa vain kaksi pistettä, jotka toimivat tukivektoreina (ks. kuva 10), riittää määrittämään hypertason joka luokittelee tehokkaasti. Samasta syystä koulutusjoukon kasvattaminen ei juurikaan paranna suorituskykyä, sillä pisteet jotka ovat marginaalin ulkopuolella eivät vaikuta hypertasoon lainkaan.

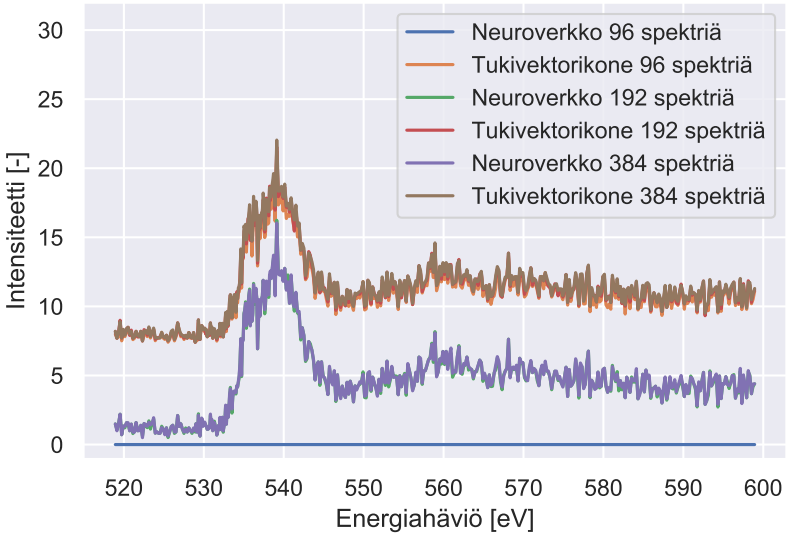


Kuva 27: Neuroverkon oppimiskäyrä nestemäisen veden näytteellä. Paras mahdollinen F_1-arvo on 1, joka on piirretty punaisella viivalla.

Neuroverkon oppimiskäyrä kertoo erilaisen tarinan. Neuroverkko vaatii huomattavasti enemmän dataa ennen kuin koulutuksen tai validaation F1-arvo lähestyy 0,9:ää ja siinä on havaittavissa huomattavaa heilahtelua. Satunnaisuus mielestäni tämä johtuu siitä, että toisin kuin tukivektorikoneessa, jokainen lisätty datapiste muuttaa neuroverkon parametrien, painoarvojen ja vakiotermien, suuruutta. Riippuen satunnaisesti koulutusjoukkoon valituista datapisteistä, neuroverkko tekee siihen mahdollisesti hyvinkin erilaisen sovituksen. Lisäksi neuroverkon painoarvot ja vakiotermit alustetaan jokaisella koulutuskerralla satunnaisesti. Vasta kun dataa on tarpeeksi, neuroverkon F1-arvo alkaa tasoittua hieman. Kuitenkaan tällä koulutusjoukolla ei päästä tilanteeseen jossa koulutuksen ja validaation F1-arvo olisi tasoittunut, joten malli todennäköisesti hyötyisi koulutusjoukon kasvattamisesta.



Kuva 28: Yhden tarkastelualueen tukivektorikoneen (ylemmät spektrit) ja neuroverkon (alemmat spektrit) näytteen spektreiksi luokiteltujen spektrien summa eri kolmella erisuuruisella koulutusjoukolla nestemäisen veden näytteellä.



Kuva 29: Yhden tarkastelualueen tukivektorikoneen (ylemmät spektrit) ja neuroverkon (alemmat spektrit) näytteen spektreiksi luokiteltujen spektrien summa eri kolmella erisuuruisella koulutusjoukolla jäänäytteellä.

Kuvissa 28 ja 29 samojen tarkastelualueen spektrit kuin kuvissa 22 ja 23 ovat luokiteltu parhaita hyperparametreja käyttäen erikokoisilla koulutusjoukoilla. Tarkastelualueita eteensironnalle on 48 ja käytetyissä koulutusjoukoissa kustakin tarkastelualueesta on luokiteltu käsin 2, 4 ja 8 spektriä. Näin ollen koulutusjoukkojen koot olivat 96, 192 ja 384 spektriä. Jää- ja vesinäytteelle tukivektorikone teki jokaisella koulutusjoukolla käytännössä identtisen spektrin. Neuroverkolla oli pientä hajontaa vesinäytteen kohdalla, mutta jäänäytteellä havaitaan selvästi, että 96 spektriä ei riittänyt neuroverkon koulutusjoukoksi, sillä neuroverkko on luokitellut kaikki spektrit taustan spektreiksi. Suuremmilla koulutusjoukoilla neuroverkko on kuitenkin luokitellut spektrit samalla kuin tukivektorikone. Teen tästä päätelmän, että tukivektorikone toimii varsin hyvin pienilläkin koulutusjoukoilla, tai ainakaan sen toiminta ei muutu koulutusjoukkoa kasvattamalla. Neuroverkko vaatii sen sijaan suuremman koulutusjoukon ja koulutusjoukon satunnaisesti valitut spektrit vaikuttavat sen hyvyyteen merkittävästi.

6 Päätelmät

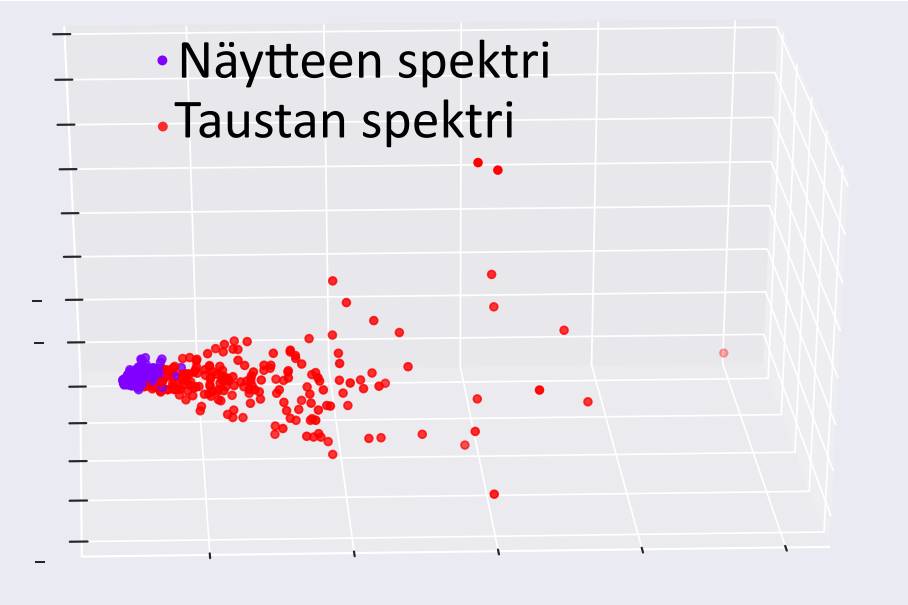
Mielestäni on selvää, että koneoppimista voidaan käyttää spektrien manuaalisen luokittelun korvaajana ainakin tässä sovelluskohteessa. Kuvat 22 ja 23 ovat selvä merkki siitä, että tukivektorikoneen ja neuroverkkon luomat summaspektrit pääsevät hyvin lähelle käsin luokiteltua vastinettaan. Teen tästä päätelmän, että molempien koneoppimismallien luokkittelun hyvyydet ovat ainakin käytetyille tarkastelualueille tutkijan tekemään luokitteluun verrattavissa olevia.

Muodoltaan vesispektri vastaa hyvin paljon L.-Å. Näslundin ja muiden artikkelissa "X-ray Absorption Spectroscopy Measurements of Liquid Water" [34] esitettyä röntgenramanspektriä. Spektrin vasemmalta reunalta löytyvä hapen esi-piikki (*engl. preedge peak*) on artikkelissa esitetyllä energianalueella 535 eV. Heidän mittauksensa oli tehty Stanford Synchrotron Radioation Laboratoryssa Yhdysvalloissa. Koneoppimismallien luomien summaspektrien muodot ovat hyvin samankaltaisia heidän spektriinsä, joten teen päätelmän että tuottamani nestemäisen veden summaspektri on validi.

John Tse ja muut ovat artikkelissaan "X-Ray Raman Spectroscopic Study of Water in the Condensed Phases" [35] esittäneet XRS-spektrin 1h-jäälle, josta tämän työn jäänäyte koostui [14]. Molempien spektrien muoto on tässäkin tapauksessa hyvin samankaltainen, ja molemmissa on selvästi havaittavissa vaakatasossa oleva kohta n. 536-538 eV energia-alueella.

Mielestäni on mielenkiintoista kuinka hyvin tukivektorikone toimii tähän luokitteluongelmaan. Näytteen ja taustan spektrit ovat selvästi eroteltavissa 450-ulotteisessa avaruudessa. Tukivektorikone ei myöskään tarvitse suuria koulutusjoukkoja vaan jo muutama kymmenen datapistettä riittää varsin hyvän hypertason rakentamiseen. Tämä johtuu siitä, että uusien datapisteiden lisääminen ei vaikuta hypertasoon mikäli uudet datapisteet ovat marginaalien ulkopuolella. Teoriassa kaksi spektriä, yksi näytteen spektri ja yksi taustan spektri, riittäisivät luokittelevan hypertason

rakentamiseen. Tukivektorikone on myös laskennallisesti tehokas, sillä sen ruudukkohaku vei alle puoli minuuttia verrattuna neuroverkon lähes kahteenkymmeneen tuntiin. Koulutusjoukon täydellinen sovittaminen vaikuttaa mielestäni hieman omituiselta, joten päädyin tarkastelemaan koko datajoukkoa pääkomponenttianalyysin (*engl. principal component analysis, PCA*) keinoin. Pääkomponenttianalyysissä data kuvataan alempaan ulottuvuuteen siten, että mahdollisimman suuri osa sen alkuperäisestä varianssista säilyy [15].



Kuva 30: 450-ulotteiset spektrit kuvastettuna kolmeen ulottuvuuteen säilyttäen mahdollisimman paljon niiden alkuperäisestä varianssista. Havaitaan että näytteen spektrit ovat varsin hyvin eroteltavissa taustan spektreistä.

Kuvassa 30 on esitetty spektrien 450-ulotteisten datavektoreiden projektio kolmeen ulottuvuuteen siten, että mahdollisimman paljon alkuperäisestä varianssista säilyy. Havaitaan selvästi että jopa tässä projektiossa näytteen spektrit ovat varsin hyvin lineaarisesti separoitavissa taustan spektreistä. Tämä on mielestäni varsin mielenkiintoinen löytö sen takia tukivektorikone näyttää toimivan tähän ongelmaan hyvin. Lisäksi tämä muistuttaa datan esitarkastelun tärkeydestä. Pääkomponenttia-

nalyysillä voidaan saada helposti intuitiota datan rakenteesta ennen kuin järeämpiä keinoja, kuten koneoppimista, käytetään.

Neuroverkko kykenee selvästi varsin hyvään luokitteluun, mutta se vaatii suurempia koulutusjoukkoja kuin tukivektorikone. Tässä sovelluskohteessa olisi toivottavaa ettei tutkija joutuisi luokittelemaan liikaa spektrejä käsin koulutusjoukkoa luodessaan. Neuroverkko on myös varsin herkkä uusien spektrien lisäämiselle, joten sen performanssi heittelee varsin paljon eri koulutusjoukkojen välillä. Neuroverkon suurimmat ongelmat ovat hyperparametrien suuri määrä, joiden ruudukkohakuun menee huomattavasti aikaa, sen kouluttamiseen kuluva aika, joka on tukivektorikoneeseen verrattuna huomattavasti pidempi, ja sen yleinen monimutkaisuus. Neuroverkkojen toiminta on varsin monimutkaista ja niiden käyttäminen ei ole yhtä helppoa kuin tukivektorikoneen. Tukivektorikonetta on helpompi käyttää niin kutsuttuna mustana laatikkona, jolloin dataa syötetään algoritmille joka tekee ennusteet, ilman että tukivektorikoneesta täytyy ymmärtää muuta kuin geometrinen intuitio luokittelevasta hypertasosta.

Tähän sovelluskohteeseen siis pidän tukivektorikonetta parempana tai ainakin helpompana ratkaisuna, joka ei ole ainakaan mielestäni neuroverkkoa huonompi. Sen toteuttaminen scikit-learn -paketilla on helppoa ja se on tehokas luokittelija tähän ongelmaan. Lisäksi se tukee kuvan 30 pääkomponenttianalyysin intuitiota siitä, että näytteen spektrit ovat eroteltavissa lineaarisesti näytteen spektreistä. En kuitenkaan kykene sanomaan miten hyvin se toimisi tilanteessa jossa spektreillä on useampi kuin kaksi luokkaa. Aurélien Géronin [15] kertoo että tukivektorikoneita pystyy käyttämään myös useampaa kuin kahta luokkaa sisältävän datan erotteluun. Neuroverkoilla tämä on mahdollista ja koodini tukee useamman luokan käyttämistä kouluttamiseen ja ennustamiseen suhteellisen pienillä muutoksilla.

En käsitellyt tässä työssä kuvassa 8 kahden oikeanpuoleisimman analysaattorin takaisinsirontaspektrejä. Takaisinsironta oli tämän tutkielman tutkimuskysymyksen

ulkopuolella, mutta tulevaisuudessa myöskin sitä voisi käsitellä koneoppimismenetelmin. Myöskin muita koneoppimisen muotoja voitaisiin tutkia, joista yhtenä potentiaalisena kandidaattina pidän konvoluutioneuroverkkoja, jotka ovat hyviä tunnistamaan erilaisia rakenteita datassa [15]. Tavallisesti niitä on käytetty kuvien tunnistamiseen mutta uskoisin yksiulotteisen konvoluution toimivan spektrien luokitteluun varsin hyvin.

7 Kiitokset

Haluan kiittää ohjaajaani Johannes Niskasta erinomaisesta ohjauksesta tämän tutkielman tekemisessä ja Christoph Sahlea, joka auttoi meitä useaan otteseen käytetyn datan ymmärtämisessä ja käsittelyssä. Haluan myös kiittää ystäviäni ja perhettäni tuesta ja kiinnostuksesta tutkielmaani kohtaan näinä vaikeina aikoina jolloin tämä tutkielma kirjoitettiin.

Viitteet

[1] Alessandro Bettini. *Introduction to Elementary Particle Physics*. Cambridge University Press, 2008. DOI: 10.1017/CBO9780511809019.

[2] Gordon Kane. *Modern Elementary Particle Physics: Explaining and Extending the Standard Model*. 2. painos. Cambridge University Press, 2017. DOI: 10.1017/9781316691434.

[3] Richard M. Martin. *Electronic Structure: Basic Theory and Practical Methods*. Cambridge University Press, 2004. DOI: 10.1017/CBO9780511805769.

[4] David J. Griffiths ja Darrell F. Schroeter. *Introduction to Quantum Mechanics*. 3. painos. Cambridge University Press, 2018. DOI: 10.1017/9781316995433.

[5] Hugh Young ja Roger Freedman. *University Physics with Modern Physics in SI Units*. Pearson, 2019. ISBN: 978-0-321-50062-5.

[6] Martin Chaplin. *Water Structure and Science*. URL: http://www1.lsbu.ac.uk/water/h2o_orbitals.html (viitattu 28. 04. 2020).

[7] P. W. Atkins ja R. S. Friedman. *Molecular quantum mechanics*. 5th ed. Oxford University Press, 2011.

[8] David Attwood. *Soft X-Rays and Extreme Ultraviolet Radiation: Principles and Applications*. Cambridge University Press, 1999. DOI: 10.1017/CBO9781139164429.

[9] S. Huotari et al. "A large-solid-angle X-ray Raman scattering spectrometer at ID20 of the European Synchrotron Radiation Facility". English. *Journal of Synchrotron Radiation* 24 (maaliskuu 2017), s. 521–530. ISSN: 0909-0495. DOI: 10.1107/S1600577516020579.

[10] Christoph Sahle et al. "Planning, performing and analyzing X-ray Raman scattering experiments". English. *Journal of Synchrotron Radiation* 22.2 (maaliskuu 2015), s. 400–409. ISSN: 0909-0495. DOI: 10.1107/S1600577514027581.

[11] Simo Huotari et al. "Direct tomography with chemical-bond contrast". English. *Nature Materials* 10 (heinäkuu 2011), s. 489–493. ISSN: 1476-1122. DOI: 10.1038/nmat3031.

[12] Winfried Schülke. *Electron Dynamics by Inelastic X-Ray Scattering*. Oxford University Press, tammikuu 2007. ISBN: 9780198510178.

[13] *European Syncrotron Radiation Facility, ESRF*. 2020. URL: http://www.esrf.eu/home.html (viitattu 02. 05. 2020).

[14] Johannes Niskanen et al. "Compatibility of quantitative X-ray spectroscopy with continuous distribution models of water at ambient conditions". *Proceedings of the National Academy of Sciences* 116 (helmikuu 2019), s. 201815701. DOI: 10.1073/pnas.1815701116.

[15] A. Géron. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Incorporated, 2019. ISBN: 9781492032649.

[16] Marc Peter Deisenroth, A. Aldo Faisal ja Cheng Soon Ong. *Mathematics for Machine Learning*. Cambridge University Press, 2020. DOI: 10.1017/9781108679930.

[17] Sebastian Raschka ja Vahid Mirjalili. *Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-Learn, and TensorFlow, 2nd Edition*. 2nd. Packt Publishing, 2017. ISBN: 1787125939.

[18] Michael A. Nielsen. *Neural Networks and Deep Learning*. misc. 2018. URL: http://neuralnetworksanddeeplearning.com/.

[19] Alexander LeNail. "NN-SVG: Publication-Ready Neural Network Architecture Schematics". *Journal of Open Source Software* 4.33 (2019), s. 747. DOI: 10.21105/joss.00747. URL: https://doi.org/10.21105/joss.00747.

[20] Arunava. *The Perceptron*. URL: https://towardsdatascience.com/the-perceptron-3af34c84838 (viitattu 20. 05. 2020).

[21] Ian Goodfellow, Yoshua Bengio ja Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN: 0262035618.

[22] Francois Chollet. *Deep Learning with Python*. 1st. USA: Manning Publications Co., 2017. ISBN: 1617294438.

[23] Marco Peixeiro. *How to Improve a Neural Network With Regularization*. URL: https://towardsdatascience.com/how-to-improve-a-neural-network-with-regularization-8a18ecda9fe3 (viitattu 02. 06. 2020).

[24] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". *Journal of Machine Learning Research* 12 (2011), s. 2825–2830.

[25] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[26] J. D. Hunter. "Matplotlib: A 2D graphics environment". *Computing in Science & Engineering* 9.3 (2007), s. 90–95. DOI: 10.1109/MCSE.2007.55.

[27] François Chollet et al. *Keras*. https://keras.io. 2015.

[28] Raúl Gómez. *Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names*. URL: https://gombru.github.io/2018/05/23/cross_entropy_loss/ (viitattu 21. 05. 2020).

[29] *Keras API reference / Losses / Probabilistic losses*. URL: https://keras.io/api/losses/probabilistic_losses/ (viitattu 21. 05. 2020).

[30] Aakash Goel. *How to add function (Get F1-score) in Keras metrics and record F1 value after each epoch?* URL: https://medium.com/@aakashgoel12/how-to-add-user-defined-function-get-f1-score-in-keras-metrics-3013f979ce0d (viitattu 21. 05. 2020).

[31] *Basic classification: Classify images of clothing*. URL: https://www.tensorflow.org/tutorials/keras/classification (viitattu 23. 05. 2020).

[32] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. *Nature Methods* 17 (2020), s. 261–272. DOI: https://doi.org/10.1038/s41592-019-0686-2.

[33] F. Pedregosa et al. *scikit-learn User Guide 3.1. Cross-validation: evaluating estimator performance*. 2020. URL: https://scikit-learn.org/stable/modules/cross_validation.html (viitattu 22. 04. 2020).

[34] L.-Å. Näslund et al. “X-ray Absorption Spectroscopy Measurements of Liquid Water”. *The Journal of Physical Chemistry B* 109.28 (2005). PMID: 16852732, s. 13835–13839. DOI: 10.1021/jp052046q. eprint: https://doi.org/10.1021/jp052046q. URL: https://doi.org/10.1021/jp052046q.

[35] John Tse et al. “X-Ray Raman Spectroscopic Study of Water in the Condensed Phases”. *Physical review letters* 100 (huhtikuu 2008), s. 095502. DOI: 10.1103/PHYSREVLETT.100.095502.

8 Liitteet

1. Tiedosto `XRS_spectra_classifier.py` on Python-ohjelmointikielellä kirjoitettu ohjelma, joka käsittelee dataa, luo koulutusjoukot ja käyttää sekä tukivektorikonetta että neuroverkkoa spektrien luokitteluun. Sen pitää olla samassa kansiossa muiden liitetiedostojen kanssa.

2. Tiedostot `liquid_water_apr20_AI_E0.h5` ja `solid_water_apr20_AI_E0.h5` ovat .h5-tiedostoja, jotka sisältävät spektrometridataa vesi- ja jäänäytteille. Kumpikin sisältää usean skannauksen sekä ilmaisimien energia-aluuet ja niiden sijainnit.

3. Tiedostot `liquid_spectra_15.txt, liquid_labels_15.txt` ja `solid_spectra_15.txt, solid_labels_15.txt` ovat yhden ilmaisimen havaitsemat spektrit yhdeltä tarkastelualueelta ja spektrejä vastaavat manuaaliset luokittelut näytteen tai taustan spektreiksi vesi- ja jäänäytteille.

4. Tiedostot `liquid_E_axis_ALL02_0.4.txt`, `liquid_E_axis_ALL04_0.4.txt`, `liquid_E_axis_ALL08_0.4.txt`, `solid_E_axis_ALL02_0.4.txt`, `solid_E_axis_ALL04_0.4.txt` ja `solid_E_axis_ALL08_0.4.txt` ovat energia-akselit kullekin eri kokoiselle koulutusjoukolle sekä vesi- että jäänäytteelle.

5. Tiedostot `liquid_labels_ALL02_0.4.txt`, `liquid_labels_ALL04_0.4.txt`, `liquid_labels_ALL08_0.4.txt`, `solid_labels_ALL02_0.4.txt`, `solid_labels_ALL04_0.4.txt` ja `solid_labels_ALL08_0.4.txt` ovat eri suuruisten koulutusjoukkojen manuaaliset luokittelut sekä vesi- että jäänäytteelle.

6. Tiedostot `liquid_spectra_ALL02_0.4.txt`, `liquid_spectra_ALL04_0.4.txt`, `liquid_spectra_ALL08_0.4.txt`, `solid_spectra_ALL02_0.4.txt`, `solid_spectra_ALL04_0.4.txt` ja `solid_spectra_ALL08_0.4.txt` ovat eri suuruisten koulutusjoukkojen spektrit sekä vesi- että jäänäytteelle.

Liitetiedostot ovat toistaiseksi saatavilla myös GitHub-kansiostani `https://github.com/KimmoPyyhtia/XRS_spectra_classifier`