Hahmontunnistus terästeollisuudessa

Manu Hietaniemi

27. toukokuuta 2014

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Tiedekunta/Osasto Fakultet/Sektion – Faculty Matemaattis-luonnontieteellinen tiedekunta	Laitos/Institution– Department Matematiikan ja tilastotieteen laitos	
Tekijä/Författare – Author Manu Hietaniemi		
Työn nimi / Arbetets titel – Title Hahmontunnistus terästeollisuudessa		
Oppiaine /Läroämne – Subject Tietokoneavusteinen matematiikka		
Työn laji/Arbetets art – Level Pro gradu -tutkielma	Aika/Datum – Month and year 28.5.2014	Sivumäärä/ Sidoantal – Number of pages 39

Tiivistelmä/Referat – Abstract

Terästeollisuudessa teräsnauhojen valmistuksessa esiintyvät ongelmat tuotannossa aiheuttavat suuria taloudellisia tappioita, joten näiden ongelmien ennustaminen ennen niiden esiintymistä olisi taloudellisesti hyvin merkittävää. Olemassa olevien fysikaalisten mallien avulla ennustaminen ei ole ollut mahdollista, joten tutkimuksessa on hyödynnetty ennustavia matemaattisia luokittimia, joiden avulla näitä ongelmallisia tilanteita ollaan pyritty ennustamaan.

Tutkielma käsittelee matemaattista teoriaa ja menetelmiä, joita on hyödynnetty aiemmin julkaistussa tutkimuksessa “Defect prediction in hot strip rolling using ANN and SVM”. Keskeisinä menetelminä tutkimuksessa on käytetty erilaisia datan esikäsittelymenetelmiä, kuten klassisia tilastollista analyysiä, piirteiden valintamenetelmiä, esimerkikisi itsestäänjärjestäytyviä karttoja, sekä ennustukseen neuroverkkoa ja tukivektorikonetta.

Tutkimuksen aineisto on peräisin teräsnauhojen lämpövalssausprosessista Ruukin terästehtaalta Raahesta. Alkuperäinen tutkimus toteutettiin Oulun Yliopiston tietokonetekniikan laboratoriossa yhteistyössä tutkimusryhmän ja Ruukin asiantuntijoiden kanssa. Tutkimuksen pohjalta ollaan pystytty osoittamaan, että tukivektorikoneen ja neuroverkon ennustetarkkuus on hyvin lähellä toisiaan kyseiselle aineistolle.

Avainsanat – Nyckelord – Keywords
Hahmontunnistus, neuroverkko, tukivektorikone, piirteiden valinta, terästeollisuus

Säilytyspaikka – Förvaringställe – Where deposited

Muita tietoja – Övriga uppgifter – Additional information

Sisältö

Luku 1

Johdanto

Vain mielikuvitus on rajana, kun alamme tarkastella säännönmukaisuuksia ympärillämme, jotka voivat olla esimerkiksi aikaan, paikkaan tai tilaan kytkeytyneitä. Monia näistä säännönmukaisuuksia sisältävistä ilmiöistä on myös mahdollista mitata erilaisten fysikaalisten mittalaitteiden avulla. Mittaustulosten analysoinnilla, kuten niiden graafisella hahmottamisella, voidaan jopa silmämääräisesti havaita joidenkin havaintojen noudattavan tiettyjä säännönmukaisuuksia ja näistä säännönmukaisuuksista on mahdollista tehdä erilaisia johtopäätöksiä. Esimerkiksi 1600-luvun vaihteessa Tycho Brahen astronomisten mittausten ja havaintojen perusteella silloinen apulaisensa, astrofyysikko Johannes Kepler, kehitti empiirisiä malleja kuvaamaan planeettojen liikeratoja. Näiden tarkastelujen pohjalta syntyivät klassisesta mekaniikasta tutut *Keplerin lait*.

Nykypäivänä on olemassa lukuisia tekniikoita, joiden avulla säännönmukaisuuksia on mahdollista mallintaa, tulkita ja jopa ennustaa ja tällaisia tekniikoita hyödynnetään ja kehitetään *hahmontunnistustutkimuksessa*. Käytännön sovelluksissa hahmontunnistukseen hyödynnetään automaattista tietojenkäsittelyä, koska tutkittavat havaintoainestot eli *datat* sisältävät usein useita tuhansia havaintoja. Hahmontunnistustutkimuksen piiriin kuuluvat muun muassa hahmontunnistus ja koneoppiminen sekä tietokonenäkö ja konenäkö, joista tässä työssä käsitellään hahmontunnistusta ja koneoppimista. Jotta hahmontunnistus on koneoppimisen keinoin mahdollista, on havainnot kyettävä lokeroimaan jonkin ominaisuuden mukaan, kuten havaintoaineistossa esiintyvien mittaustietojen perusteella. Toisaalta näitä mittaustietoja on mahdollista esikäsitellä esimerkiksi tilastollisin menetelmin, jolloin havannoille saadaan jalostettua lisäominaisuuksia. Näistä aineiston ominaisuuksista eli mittaustiedoista ja mahdollisesti niistä jalostetuista lisätiedoista käytetään nimitystä *piirteet*.

Tässä työssä havaintojen lokerointia tehdään luokittimien avulla, joiden tarkoitus on luokitella havainnot onnistuneisiin ja virhekoodillisiin luokkiin niiden lähtötietojen perusteella. Luokittelijat, kuten *neuroverkko* tai *tuki-*

vektorikone, pystyvät laskennallisin menetelmin löytämään säännönmukaisuuksia niille syötetystä datasta. Data koostuu havannoista, jotka tulevat siitä ympäristöstä mihin luokittelijaa halutaan hyödyntää. Havannoiksi luetaan esimerkiksi terästeollisuudessa toimivan tuotantolinjaston valmistamat teräsnauhat. Näistä teräsnauhoista tallennetaan tietokantaan koko valmistusprosessin ajan lukuisia mittaus- ja mallinnustietoja, kuten aikaleimat, kemialliset koostumukset, lämpötila- ja dimensiotiedot. Siis jokainen havainto sisältää lukuisia piirteitä, joiden avulla luokittelijan on mahdollista karakterisoida sitä.

Luokittelijoiden toiminta perustuu tilastolliseen oppimisteoriaan. Matematiikan näkökulmasta luokittelija voidaan ajatella kuvauksena syötteeltä vasteelle eli tässä konseptissa havainnolta sen lopputulokselle. Adaptiivisille luokittelijoille on ominaista, että käyttäjän ei tarvitse tuntea syötteen ja vasteen välistä funktiota, vaan luokittelija voidaan opettaa tunnistamaan niiden välinen yhteys. Täten on mahdollista löytää relaatio monimutkaisen epälineaarisen syötteen ja vasteen välille, jonka analyyttinen ratkaiseminen olisi mahdotonta. Luokittelijan opettaminen tapahtuu valitun aineiston pohjalta ja onnistuneesti toteutettuna on mahdollista ennustaa sen toimintaympäristössä olevien, opetusaineistosta erillisten, havaintojen lopputulosta.

Datan esikäsittely on olennaista luokittelijan optimaalisen toiminnan kannalta. Esikäsittelyllä voidaan vaikuttaa piirteiden määrään ja laatuun, jotka taas vaikuttavat suoraan siihen, kuinka tehokkasti valittu luokittelualgoritmi toimii. Esimerkiksi yksinkertaiset toimenpiteet, kuten diskreettien piirteiden jakaminen binäärisiksi ja jatkuvien piirteiden normalisointi voivat olla kriittisiä toimenpiteitä luokittelijan optimaalisen toiminnan takaamiseksi. On olemassa myös lukuisia erilaisia piirteiden valintamenetelmiä eli algorimeja, joiden avulla on mahdollista erotella aineistosta olennaisia piirteitä, joita hyödynnetään luokittelussa. Epäolennaisten piirteiden poistaminen nopeuttaa luokittelijan opetusalgoritmin toimintaa ja pahimmillaan epärelevantit piirteet voivat myös heikentää luokittelutarkkuutta.

Adaptiivisia luokittelijoita ja hahmontunnistusta voidaan käyttää nykypäivänä lähes minkä tahansa prosessien mallinnuksessa ja sen tukena erityisesti teollisuuden sovelluksissa. Luokittelijoita on mahdollista käyttää itsenäisesti, mutta ne voidaan myös integroida osaksi tuotanto- tai mallinnusprosessia. Luokittelijoiden kyky mukautua itsenäisesti prosessissa tapahtuviin muutoksiin ja oppia niistä, tekee niistä ylivertaisen verrattuna perinteisempiin tilastollisiin tai fysikaalisiin malleihin.

Tässä työssä on tutkittu teräksen nauhavalssaukseen liittyvää aineistoa, johon edellä mainittuja konsepteja on sovellettu. Yksi suurimmista tappioita aiheuttavista tekijöistä teräksen nauhavalssauksessa on siinä esiintyvät virhekoodit, joiden ilmentyessä teräsnauhoja joudutaan tarkastamaan ja muokkaamaan jälkikäteen. Tuotannossa käytetään edistyksellisiä fysikaalisia malleja, joiden avulla tuotantoprosessia mallinnetaan tarkasti, mutta virhekoo-

dien esiintymistä ei tästä huolimatta onnistuta välttämään. Adaptiivisen luokittelijan implementointi tuotantolinjastoon on yksi konkreettinen mahdollisuus ennustaa ja estää näiden virhekoodien syntymistä.

Luku 2

Luokittelijat

Luokittelijoiden tehtävä on kategorisoida havaintoja tiettyjen sääntöjen ja säännönmukaisuuksien perusteella. Säännöt, joiden mukaan havainnot luokitellaan, voidaan määritellä suunnittelijan toimesta, mutta yleisempi käytäntö on hyödyntää luokittelijalle opetusalgoritmia, jonka avulla luokittelija oppii löytämään ainestossa esiintyvät säännönmukaisuudet. Opetusalgoritmia käytettäessä suunnittelijalla ei tarvitse olla valaistuneempaa aineston ominaisuuksista tai siinä esiintyvien piirteiden merkityksistä.

Havantoja on mahdollista luokitella niiden vasteluokan perusteella, jolloin kahden havainnon piirteiden arvot voivat näennäisesti olla hyvinkin erilaisia, vaikka ne edustaisivat samaa luokkaa. Tällaisia luokittelualgoritmeja edustavat esimerkiksi päätöspuut, lähimpien naapureiden menetelmät ja erilaiset Bayesin ja Gaussin menetelmiin perustuvat luokittimet, joista tässä työssä on käytetty neuroverkkoja ja tukivektorikonetta. Vastaavasti vasteluokkaan perustuvan luokittelijan opetusmenetelmää kutsutaan *valvotuksi oppimiseksi*.

Toisaalta luokittelu voidaan tehdä havaintojen parametrien samankaltaisuuden, esimerkiksi Euklidisen etäisyyden, perusteella, jolloin havaintojen vasteluokka ei ole tiedossa tai sitä ei huomioida. Tällaisella lähestymistavalla voidaan havaintoja kategorisoida haluttujen kriteerien mukaan samankaltaisiin ryppäisiin ja opetusmenetelmää kutsutaan *valvomattomaksi oppimiseksi*. Yksi käytetyimmistä valvomattomista luokittelumenetelmistä on *itsestään järjestäytyvät kartat*, joita on käsitelty Luvussa 3.

2.1 Neuroverkko

Kirjallisuudessa neuroverkko määritellään muun muassa seuraavasti: Neuroverkko on laajamittaisesti rinnakkain jaettu prosessori, joka koostuu yksinkertaisista käsittely-yksiköistä, ja jolla on luontainen taipumus tallentaa kokemuksellista tietoa, sekä kyky hyödyntää sitä [1]. Neuroverkko muistuttaa aivojen toimintaa kahdella tavalla: Älykkyys hankitaan sen ympäristöstä

eli aineistosta oppimisprosessin avulla ja toisaalta neuronien keskenäisiä yhteyksiä, joita kutsutaan synaptisiksi painokertoimiksi, käytetään tallentamaan opittua informaatiota.

Yllämainittuun oppimiseen käytetään opetusalgoritmia ja opetusdataa, joiden avulla verkon painokertoimet saadaan optimoitua. Painokertoimia voidaan halutessaan säätää manuaalisesti, jos suunnittelijalla on valistuneempaa ennakkotietoa käyttötarkoituksesta. Toisaalta neuroverkoille on mahdollista hyödyntää jatkuvaa oppimista, jolloin verkon on mahdolista mukautua muuttuvaan aineistoon ja oppia virheistään.

Simon Haykin mainitsee teoksessaan *Neural networks* neuroverkkojen yhdeksän etua verrattaessa tavanomaisiin tilastollisiin lähestymistapoihin.

1. Epälineaarisuus. Neuroverkoilla on kyky kuvata epälineaarisia ongelmia, koska yksittäisiä tai useampia piirteitä on mahdollista käsitellä useamman neuronin kautta. Tämä on erittäin hyödyllistä, kun on kyse soveltavista ja monimutkaisista ongelmista, jotka monesti ovat luonnostaan epälineaarisia.

2. Syöte-vaste kuvaus. Neuroverkko oppii syötteiden ja niihin linkittyvien vasteiden perusteella ja on kyvykäs säätämään neuronien painokertoimet käytetyn opetusaineiston perusteella. Täten neuroverkko ei tarvitse *a priori* tietoa, eli malleja tai oletuksia, käsiteltävästä aineistosta. Kyseistä ominaisuutta voidaan kutsua parametrittomaksi tilastolliseksi päättelyksi.

3. Adaptiivisuus. Neuroverkkojen ominainen piirre on uudelleenoppiminen, jolloin tiettyyn tarkoitukseen käytetty ennalta määritelty verkko voidaan uudelleenopettaa aineiston muuttuessa. On kuitenkin huomioitava, ettei tämä uudelleenoppiminen takaa robusteja tuloksia vaan voi johtaa jopa ongelmiin [2].

4. Luottamuksellinen vaste. Neuroverkko tuottaa, halutessaan, todennäköisyyteen perustuvan vasteen, josta ilmenee tehdyn päätöksen todennäköisyys. Kyseistä todennäköisyyttä voi hyödyntää epäselvissä luokittelutapauksissa.

5. Taustatietojen hyödyntäminen. Neuroverkkoon syötettävä informaatio on jokaisen neuronin käytettävissä, joten tämä tieto tulee luontaisesti hyödynnetyksi.

6. Virhetoleranssi. Yleensä neuroverkon rakenteet, pirteet, neuronit ja vaste, ovat laajasti kytkettynä toisiinsa, joten käyttöolosuhteiden tulee muuttua merkittävästi, jotta suorituskyky heikentyisi merkittävästi.

7. Laajamittaisesti toteutettavissa. Neuroverkko neuroneineen on luonteeltaan vahvasti rinnakkain toteutettu. Tämä mahdollistaa nopeam-

man laskentatehon ja näin se soveltuu käytettäväksi VLSI-teknologiassa.

8. Suunnittelun ja analysoinnin yhdenmukaisuus. Neuroverkot ovat universaaleja informaationkäsittelijöitä, sillä samoja merkintätapoja käytetään eri tieteenalojen sovelluksissa. Koska neuronit ovat käytössä kaikenlaisissa neuroverkkorakenteissa, voidaan eri alojen sovelluksia ja teorioita jakaa poikkitieteellisesti. Myös neuroverkkojen eri osia, moduuleja, voidaan yhdistellä mielivaltaisesti toisiinsa halutun lopputuloksen toivossa.

9. Neurobiologinen analogia. Neuroverkkojen suunnittelun pohjana on aivojen analogia, jonka valossa tiedon prosessointi tässä mielessä on fyysisesti mahdollista ja myös nopeaa ja tehokasta. Siinä missä neurobiologit käyttävät neuroverkkoja tulkitsemaan neurobiologisia ilmiöitä niin insinöörit hyödyntävät neurobiologiaa uusien ideoiden löytämiseen teknologioissa.

2.1.1 Neuronimalli

Informaationprosessointiyksiköt, joita käytetään neuroverkoissa kutsutaan neuroneiksi. Neuronit koostuvat painokertoimista, summausfunktioista, harhatermistä ja aktivointifunktioista. Syötesignaalin piirteille lasketaan painokertoimet ja mahdollinen harha, jotka summataan ja syötetään aktivointifunktiolle, jotta saavutetaan neuronille sopiva vaste.

Olkoon $x_1, x_2, \ldots, x_m$ syötteen $\mathbf{x}$ piirteet, $w_{k1}, w_{k2}, \ldots, w_{km}$ piirteitä vastaavat painokertoimet, niin neuroni voidaan määritellä seuraavin yhtälöin

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.1}$$

ja

$$y_k = \varphi(u_k + b_k) \tag{2.2}$$

Edellä kuvatuissa yhtälöissä $\varphi(\cdot)$ on neuronin aktivointifunktio, y_k vasteet, b_k harhat. Termiä u_k kutsutaan vasteen lineaarikombinaattoriksi ja edelleen harhan ja vasteen muodostama vasteen affini-muunnosta voidaan kutsua neuronin k kuvaamaksi indusoiduksi paikalliskentäksi v_k

$$v_k = u_k + b_k \tag{2.3}$$

Aktivointifunktion $\varphi(\cdot)$ valinta on keskeistä optimaalisen toiminnan kannalta. Kirjallisuudessa [1] esimerkkinä mainitaan kolme aktivointifunktion arkkityyppiä:

- Porrasfunktio on jatkuva aktivointifunktio, jolle

$$\varphi(v_k) = \begin{cases} 1 & , \text{jos } v_k \geq 0 \\ 0 & , \text{jos } v_k < 0 \end{cases} \tag{2.4}$$

  Vastaavasti vasteelle y_k saadaan

$$y_k = \begin{cases} 1 & , \text{jos } v_k \geq 0 \\ 0 & , \text{jos } v_k < 0 \end{cases} \tag{2.5}$$

- Voidaan määritellä paloittainlineaarinenfunktio, jolle

$$\varphi(v_k) = \begin{cases} 1 & , \text{jos } v_k \geq +\frac{1}{2} \\ v_k & , \text{jos } +\frac{1}{2} < v_k < -\frac{1}{2} \\ 0 & , \text{jos } v_k \leq -\frac{1}{2} \end{cases} \tag{2.6}$$

- Sigmoid-funktio on yleisimmin käytetty aktivointifunktio neuroverkoissa. Kyseessä on aidosti kasvava funktio, jolla on sekä lineaarisia, että epälineaarisia ominaisuuksia. Logistinen sigmoidfunktio määritellään

$$\varphi(v_k) = \frac{1}{1 + \exp(-av_k)} \tag{2.7}$$

  missä a on sigmoid-funktion jyrkkyyden määräävä parametri. Kun a lähestyy ääretöntä lähestyy sigmoid-funktion kuvaaja edellä mainittua porrasfunktiota. Sigmoid-funktio kuitenkin differentioituva toisin kuin porrasfunktio. Myös kuvajoukko

$$\varphi(v_k) = \begin{cases} 1 & , \text{jos } v_k > 0 \\ 0 & , \text{jos } v_k = 0 \\ -1 & , \text{jos } v_k < 0 \end{cases} \tag{2.8}$$

  eroaa porrasfunktion kuvajoukosta (2.4).

2.1.2 Neuroverkkoarkkitehtuurit

Yleisesti neuroverkko rakennetaan niin, että neuronit muodostavat yhden tai useampia neuronikerroksia syötekerroksen ja vastekerroksen väliin. Tällöin näistä väliin jäävistä neuronikerroksista käytetään nimitystä *piilokerros*, koska niiden arvot optimoidaan algoritmin toimesta, eikä niiden varsinaisista arvoista olla kiinnostuneita. Siis signaali tai syöte kulkee neuronien läpi, jolloin syötteen piirteet voivat olla kytkettynä neuroneihin mielivaltaisesti, ja edelleen neuronit voivat olla kytkettynä vasteeseen mielivaltaisesti.

- Yksikerroksinen neuroverkko koostuu nimensä mukaan ainoastaan yhdestä piilokerroksesta neuroneja. Tällöin syöte kytkeytyy mielivaltaisesti neuronikerroksen neuroneihin, jotka tuotavat signaalille vasteen.

- Monikerroksinen neuroverkko sisältää useampia piilokerroksia ja verrattuna yhteen piilokerrokseen voidaan useammalla saavuttaa korkeammanasteista hahmontunnistusta. Voidaan myös sanoa, että verkolle saadaan parempi yleistämiskyky, eikä se ole niin riippuvainen paikallisista ääriarvoista. Useammasta neuronikerroksesta on erityisesti hyötyä sillon, kun syötteenä olevan aineiston koko tai piirteiden määrä tai on suuri.

On osoitettu, että aineisto, joka sisältää n kappaletta piirteitä, voidaan luokitella täsmällisesti yhdellä piilokerroksella sisältäen $(n-1)$ kappaletta neuroneja. Toisaalta kahdella piilokerroksella voidaan saavuttaa lähes täsmällinen vastekuvaus käyttämällä vain $(n-2)/3$ kappaletta neuroneja [3]. Joten tässä työssä käytettiin hyväksi yhtä tai kahta piilokerrosta tapauskohtaisesti.

2.1.3 Neuroverkon opetusprosessi

Puhuttaessa neuroverkoista voidaan opetusprosessi määritellä seuraavasti [4]: "Oppiminen on prosessi, jossa neuroverkon vapaat parametrit mukautuvat sen toimintaympäristön mukaan. Opetusmenetelmät määritellään sen perusteella, missä ja miten parametrit muuttuvat."

Opetusprosessi voidaan jaotella kolmeen eri vaiheeseen:

1. Neuroverkon toimintaympäristö tuottaa sille ärsykkeitä.
2. Neuroverkon vapaita parametrejä säädetään ärsykkeiden mukaan.
3. Neuroverkon vaste muuttuu siihen tehtyjen muutosten perusteella.

Opetusalgoritmeja on lukuisia ja kaikilla on eriaisia ominaisuuksia. Näitä ovat esimerkiksi virheidenkorjaukseen perustuva-, muistiin perustuva-, Hebbian-, kilpaileva- ja Boltzmann -oppiminen. Jatkossa käsitellään virheenkorjaukseen perustuvaan opetusta, jossa opetusaineiston pohjalta syntyneen virheen perustella neuroverkon painokertoimia säädetään optimaaliseen suuntaan, ja prosessia toistetaan kunnes haluttu tarkkuus ollaan saavutettu.

Opetusprosessissa on huomioitava aineiston mahdollisesti sisältämä vaihtelu ja satunnaisuus eli havaintojen yksilöllisyys. Jos luokitin opetetaan liian tarkasti tunnistamaan opetusaineiston havainnot, saattaa luokittelijan virhe kasvaa käyttökelvottomaksi opetusaineistosta erilliselle havannoille. Toisin sanoen, opetusaineiston täydellinen interpolointi johtaa *ylioppimiseen* [5], ja täten luokittimen yleistämiskykyä on testattava testi- ja validointiaineistoilla. Yksi tehokas menetelmä välttää ylioppiminen on *ristiinvalidointi* [6].

2.1.4 Virheentakaisinjohtamismenetelmä

Oletaan, että aineiston syöte ja haluttu vaste tunnetaan. Virheentakaisinjohtamismenetelmän perusajatuksena on palauttaa vasteessa esiintyvä virhe takaisin verkolle ja säätää neuronien painokertoimia optimaaliseen suuntaan. Tyypillisesti opetusaineistoa iteroidaan verkon läpi useampia kertoja, jotta verkon ennuste paranee. Näistä iterointikierroksista käytetään nimitystä *epookki* ja merkitään yläindeksillä $x^{(\tau)}$.

Olkoon neuronin vastaanottamat syötevektorit $\mathbf{x}_i$, missä $i = 1, \ldots, n$. Lisäksi y_i neuronin vastesignaali. Nyt voidaan laskea vastevektorin sisältämät virheet $e_i(\mathbf{w})$ vertaamalla verkon tuottamaa vastetta y_i haluttuun vasteeseen t_i eli

$$e_i(\mathbf{w}) = y_i - t_i$$

Edellä lasketulla virhetermillä $e_i(\mathbf{w})$ voidaan korjata neuronien painokertoimia $\mathbf{w}$ optimaalisemmaksi. Tällöin minimoidaan virhefunktio $E(\mathbf{w})$ esimerkiksi käyttäen neliöllisten virheiden summaa

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{n} \|e_i(\mathbf{w})\|^2. \tag{2.9}$$

2.1.5 Opetusnopeusparametri

Kun tarkastellaan opetuksen tehokkuutta, voidaan puhua opetusnopeudesta. Sitä arvioidaan *opetusnopeusparametrin* η avulla ja sitä kuvaava niin kutsuttu delta-sääntö eli painokertoimen w_{kj} muutos määritellään

$$\Delta \mathbf{w}_{kj} = \eta \mathbf{e}_k \mathbf{x}_j, \tag{2.10}$$

missä $0 < \eta \leq 1$ on opetusnopeutta kuvaava positiivinen reaaliluku, ja nimeltään opetusnopeusparametri.

Yllä kuvatulla painokertoimen muutoksella $\Delta \mathbf{w}_{kj}^{(\tau)}$ on mahdollista päivittää seuraavan epookin painokerrointa

$$\mathbf{w}_{kj}^{(\tau+1)} = \mathbf{w}_{kj}^{(\tau)} + \Delta \mathbf{w}_{kj}^{(\tau)} \tag{2.11}$$

2.1.6 Gradientin hyödyntäminen virheenkorjauksessa

Gradientin avulla voidaan optimaalisen virhefunktion etsimistä nopeuttaa merkittävästi verrattuna approksimaatiohin, jotka perustuvat paikallisiin neliöllisiin menetelmiin, kuten aiemmassa (2.9).

Taylorin-kehitelmä on tärkeä työkalu virhefunktion gradienttien ja painokertoimien päivitysten laskemiseksi

$$E(\mathbf{w} + \boldsymbol{\Delta}\mathbf{w}) = E(\mathbf{w}) + \nabla E(\mathbf{w})^T \boldsymbol{\Delta}\mathbf{w} + \boldsymbol{\Delta}\mathbf{w}^T \nabla^2 E(\mathbf{w}) \boldsymbol{\Delta}\mathbf{w} + \ldots \tag{2.12}$$

Tässä $\nabla^2 E(\mathbf{w}) = \mathbf{H}$ on *Hessin matriisi* eli se on positiivisesti semidefiniitti matriisi

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 E}{\partial^2 w_1^2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_n} \\ \\ \frac{\partial^2 E}{\partial w_2 \partial w_1} & \frac{\partial^2 E}{\partial^2 w_2} & \cdots & \frac{\partial^2 E}{\partial w_2 \partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_n \partial w_1} & \frac{\partial^2 E}{\partial w_n \partial w_2} & \cdots & \frac{\partial^2 E}{\partial^2 w_n^2} \end{bmatrix}$$

Jatkossa hyödynnetään erityisesti Taylorin-kehitelmän

- Ensimmäisen asteen

$$E(\mathbf{w} + \mathbf{\Delta w}) \simeq E(\mathbf{w}) + \nabla E(\mathbf{w})^T \mathbf{\Delta w} \tag{2.13}$$

  ja

- Toisen asteen approksimaatiota

$$E(\mathbf{w} + \mathbf{\Delta w}) \simeq E(\mathbf{w}) + \nabla E(\mathbf{w})^T \mathbf{\Delta w} + \mathbf{\Delta w}^T \mathbf{H} \mathbf{\Delta w} \tag{2.14}$$

Optimoidessa tiettyä ongelmaa halutaan löytää painokerroinvektori $\mathbf{w}$, joka minimoi virhefunktion $E(\mathbf{w})$ arvon. Kun tarkastellaan virhefunktiota geometrisesti, voidaan se ajatella pintana painokerroinavaruudessa. Jos painokerrointa $\mathbf{w}$ muutetaan vähän $\mathbf{w} + \varepsilon\mathbf{w}$, niin virhefunktio muuttuu

$$\varepsilon E \simeq \varepsilon \mathbf{w}^T \nabla E(\mathbf{w}),$$

missä virhefunktion gradientti $\nabla E(\mathbf{w})$ osoittaa virhefunktion suurimmin kasvavaan suuntaan. Koska $E(\mathbf{w})$ on sileä ja jatkuva painokertoimen $\mathbf{w}$ funktio, sen pienin arvo löytyy painokerroinavaruuden pisteestä, jossa gradientti katoaa eli

$$\nabla E(\mathbf{w}) = 0, \tag{2.15}$$

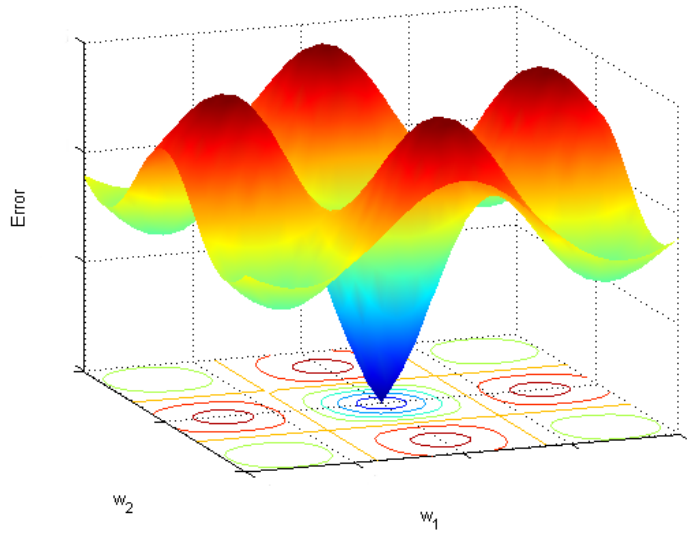
koska muutoin sitä voitaisiin muuttaa suuntaan $\nabla E(\mathbf{w})$ ja pienentää virhettä. Näitä pisteitä, joissa gradientti katoaa kutsutaan *satulapisteiksi* tai ääriarvoiksi *minima* tai *maxima*, joita on havainnollistettu kuvassa 2.1.

Ongelmana on, että näitä satulapisteitä on lukuisia ja virhefunktio voi olla hyvin epälineaarinen suhteessa painokerroinvektoreihin. Tästä syystä ei ole mahdollista löytää analyyttistä ratkaisua virhefunktiolle $\nabla E(\mathbf{w})$ ja on hyödynnettävä iteratiivisia numeerisia menetelmiä. Suurimmassa osassa näistä menetelmistä valitaan jokin mielivaltainen painokerroinvektori $\mathbf{w}^{(0)}$ ja muunnetaan tätä sopivaan suuntaan useampia epookkeja painokerroinavaruudessa

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \Delta \mathbf{w}^{(\tau)} \tag{2.16}$$

Näin edetessä on otettava huomioon, että virhefunktion arvon on pienennyttävä iterointikierrosten myötä, joten on voimassa ehto

$$E(\mathbf{w}^{(\tau+1)}) < E(\mathbf{w}^{(\tau)}) \tag{2.17}$$



Kuva 2.1: Virhefunktion kuvaaja painokerroinavaruudessa. Kun tarkastellaan virhefunktion minimointia, voidaan kuvasta nähdä useampia paikallisia minima-satulapisteitä, jotka sijaitsevat vihreiden ympyröiden kohdalla. Vastaavasti paikalliset maxima-satulapisteet on kuvattuna punaisella värillä. Keskellä tasoa on globaali minima-satulapiste, joka kuvattu sinisellä värillä.

Laskevan gradientin menetelmä

Gradienttia hyödyntävistä menetelmistä yksinkertaisin on *laskevan gradientin menetelmä*, jossa painokertoimen päivitys tehdään virhefunktion gradientin negatiiviseen suuntaan

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)}), \tag{2.18}$$

missä opetusnopeusparametri $\eta > 0$. Täten painokertoimen muutos yhtälölle (2.16) on

$$\Delta \mathbf{w}^{(\tau)} = -\eta \nabla E(\mathbf{w}^{(\tau)}) \tag{2.19}$$

Edelleen, jos hyödynnetään ensimmäisen asteen Taylorin-kehitelmän approksimaatiota (2.13) ja edellistä yhtälöä niin saadaan

$$\begin{aligned} E(\mathbf{w} + \Delta \mathbf{w}) &\simeq E(\mathbf{w}) + \nabla E(\mathbf{w})^T \Delta \mathbf{w} \\ &= E(\mathbf{w}) - \eta \|\nabla E(\mathbf{w})\|^2, \end{aligned} \tag{2.20}$$

kun $\eta > 0$ niin virhefunktio suppenee kohti optimaalista, mutta mahdollisesti vain paikallista, satulapistettä.

Erityisesti laskevan gradientin menetelmässä opetusnopeusparametrin η valinnalla on suuri vaikutus neuroverkon virhetermin suppenemiseen ja täten suorituskykyyn:

- Kun η on pieni, niin opetusalgoritmi toimii *liian vaimeasti*, jolloin painokertoimen kehityskaari on liian sileä.
- Kun η on suuri, niin opetusalgoritmi toimii *liian jyrkästi*, jolloin painokertoimen kehityskaari oskilloi liiaksi.
- Kun η ylittää tietyn kriittisen arvon niin algoritmi ei enään toimi tarkoituksenmukaisesti.

Menetelmä voi intuitiivisesti vaikuttaa tehokkaalta, koska painokertoimien päivitys tapahtuu juuri virhefunktion minimoivaan suuntaan. Menetelmä on kuitenkin tehoton juuri silloin, kun valmista aineistoa käytetään kokonaisuudessaan opetuksessa, koska virhefunktion gradientin laskemiseksi on käytettävä koko opetusaineistoa, joka on laskennallisesti työlästä. Tällaisissa tapauksissa voidaan käyttää tehokkaampia algoritmeja, kuten *skaalattua konjugaatti gradientti* tai *Levenberg-Marquardt* algoritmia.

Newtonin menetelmä

Newtonin menetelmä perustuu neliölliseen menetelmään, jossa minimoidaan virhefunktio $E(\mathbf{w})$. Menetelmä hyödyntää Taylorin kehitelmän toisen asteen approksimaatiota (2.14). Eli virhefunktio minimoituu, kun

$$\nabla E(\mathbf{w}) + \mathbf{H}\Delta\mathbf{w} = 0 \tag{2.21}$$

Ratkaistaan tämä painokertoimen muutoksen suhteen

$$\Delta\mathbf{w} = -\mathbf{H}\nabla E(\mathbf{w}) \tag{2.22}$$

Nyt voidaan painokertoimen päivitys $\mathbf{w}^{(\tau+1)}$ laskea yhtälöstä

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - (\mathbf{H}^{(\tau)})^{-1}\nabla E(\mathbf{w}^{(\tau)}) \tag{2.23}$$

Yleisesti ottaen Newtonin menetelmä suppenee asymptoottisesti nopeasti eikä siinä ilmenny suuren opetusnopeusparametrin aiheuttamaa niin kutsuttua liiallista oskillointia, kuten laskevan gradientin menetelmässä. Kuitenkin menetelmän mahdollisia kompastuskiviä on erityisesti

- Hessin matriisin $\mathbf{H}$ laskeminen useampien painokertoimien tapauksessa, joka on laskennallisesti työlästä.
- Ettei voida olla varmoja tällaisen positiivisesti definiitin matriisin $\mathbf{H}$ löytymisestä.
- Yhtälön suppeneminen kriittiseen pisteeseen vain silloin kuin virhefunktio on täydellisesti neliöllinen. Yleisesti näin ei voida olettaa.

Gauss-Newtonin menetelmä

Gauss-Newtonin menetelmässä käsitellään virhefunktiota neliövirheen pohjalta, kuten (2.9),

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{\tau}(e^{(i)})^2 \tag{2.24}$$

Havaintojen virheet $e^{(i)}$, missä $1 \leq i \leq \tau$, ovat painokertoimista $\mathbf{w}^{(i)}$ riippuvia, joten lineaarisen riippuvuuden nojalla voidaan määritellä

$$e_i'(\mathbf{w}) = e^{(i)} + \left[\frac{\partial e^{(i)}}{\partial \mathbf{w}}\right]^T_{\mathbf{w}=\mathbf{w}^{(\tau)}}(\mathbf{w} - \mathbf{w}^{(\tau)}), \text{ missä } i = 1, 2, \ldots, \tau \tag{2.25}$$

Yhtäpitävästi voidaan käyttää matriisimuotoa

$$\mathbf{e}'(\mathbf{w}^{(\tau)}) = \mathbf{e}^{(\tau)} + \mathbf{J}^{(\tau)}(\mathbf{w} - \mathbf{w}^{(\tau)}), \tag{2.26}$$

missä $\mathbf{e}^{(\tau)} = [e^{(1)}, e^{(2)}, \ldots, e^{(\tau)}]^T$ ja $\mathbf{J}^{(\tau)}$ on *Jacobin matriisi*, joka määritellään

$$\mathbf{J}^{(\tau)} = \begin{bmatrix} \frac{\partial e^{(1)}}{\partial w_1} & \frac{\partial e^{(1)}}{\partial w_2} & \cdots & \frac{\partial e^{(1)}}{\partial w_n} \\ \frac{\partial e^{(2)}}{\partial w_1} & \frac{\partial e^{(2)}}{\partial w_2} & \cdots & \frac{\partial e^{(2)}}{\partial w_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial e^{(\tau)}}{\partial w_1} & \frac{\partial e^{(\tau)}}{\partial w_2} & \cdots & \frac{\partial e^{(\tau)}}{\partial w_n} \end{bmatrix}_{\mathbf{w}=\mathbf{w}^{(\tau)}}$$

Täten Jacobin matriisi $\mathbf{J}^{(\tau)}$ on transpoosi virhetermien $e^{(i)}$ gradienteista koostuvalle matriisille

$$\nabla\mathbf{e}^{(\tau)} = [\nabla e^{(1)}, \ldots, \nabla e^{(\tau)}] \tag{2.27}$$

Nyt painokertoimen päivitys $\mathbf{w}^{(\tau+1)}$ tehdään neliöllisen virheen avulla

$$\mathbf{w}^{(\tau+1)} = \arg\min_{\mathbf{w}}\left[\frac{1}{2}\|\mathbf{e}'(\mathbf{w}^{(\tau)})\|^2\right] \tag{2.28}$$

Lasketaan yllä oleva neliö auki hyödyntäen aiempaa (2.26) saadaan

$$\begin{aligned}\frac{1}{2}\|\mathbf{e}'(\mathbf{w}^{(\tau)})\|^2 = \frac{1}{2}\|\mathbf{e}^{(\tau)}\|^2 &+ \mathbf{e}^{(\tau)^T}\mathbf{J}^{(\tau)}(\mathbf{w} - \mathbf{w}^{(\tau)}) \\ &+ \frac{1}{2}(\mathbf{w} - \mathbf{w}^{(\tau)})^T\mathbf{J}^{(\tau)^T}\mathbf{J}^{(\tau)}(\mathbf{w} - \mathbf{w}^{(\tau)})\end{aligned} \tag{2.29}$$

Tämä voidaan derivoida painokertoimen suhteen, jolloin saadaan

$$\mathbf{J}^{(\tau)^T}\mathbf{e}^{(\tau)} + \mathbf{J}^{(\tau)^T}\mathbf{J}^{(\tau)}(\mathbf{w} - \mathbf{w}^{(\tau)}) = \mathbf{0} \tag{2.30}$$

Tätä matriisiesitystä käyttäen päästään painokertoimen päivityksessä (2.28) eroon neliöllisestä virhetermin laskennasta ja se saa muodon

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \left[\mathbf{J}^{(\tau)^T}\mathbf{J}^{(\tau)}\right]^{-1}\mathbf{J}^{(\tau)^T}\mathbf{e}^{(\tau)} \tag{2.31}$$

2.1.7 Skaalattu konjugaattigradienttialgoritmi

Skaalattu konjugaattigradienttialgoritmi (SCG) poikkeaa aiemmin esitellyistä gradienttia hyödyntävistä menetelmästä erityisesti siinä, että virhefunktion päivitys tapahtuu gradientin konjugaatin suuntaan, eikä esimerkiksi gradientin minimoivaan suuntaan. Algoritmi on osoitettu hyvin tehokkaaksi verrattuna perinteisenpään takaisinjohtamisalgoritmiin ja sitä on onnistuneesti hyödynnetty muissakin tutkimuksissa [7] [8].

Algoritmissa käytetään hyväksi Newtonin menetelmän tulosta (2.22), jonka mukaan

$$\Delta \mathbf{w} = \mathbf{H}^{-1} \nabla E(\mathbf{w}). \tag{2.32}$$

Määritellään kantavektorien joukko $\mathbf{p}^{(1)}, \mathbf{p}^{(2)}, \ldots, \mathbf{p}^{(\tau)} \in \mathbb{R}^N$, jotka ovat konjugaatteja matriisille $\mathbf{H}$. Nyt $\mathbf{H}$ on positiivisesti definiitti ja

$$\begin{cases} {\mathbf{p}^{(k)}}^T \mathbf{H} \mathbf{p}^{(i)} = 0 & \text{, kun } k \neq i \\ {\mathbf{p}^{(k)}}^T \mathbf{H} \mathbf{p}^{(i)} > 0 & \text{, kun } k = i \end{cases} \text{, missä } 1 \leq k \text{ ja } i \leq \tau. \tag{2.33}$$

Valitaan $k = i = \tau$, niin ${\mathbf{p}^{(\tau)}}^T \mathbf{H} \mathbf{p}^{(\tau)} \neq 0$, ja merkitään $\mathbf{s}^{(\tau)} = \mathbf{H}\mathbf{p}^{(\tau)}$ ja $\mathbf{x}^{(\tau)} = {\mathbf{p}^{(\tau)}}^T \mathbf{H} \mathbf{p}^{(\tau)}$. Eli

$$\mathbf{x}^{(\tau)} = {\mathbf{p}^{(\tau)}}^T \mathbf{H} \mathbf{p}^{(\tau)} = {\mathbf{p}^{(\tau)}}^T \mathbf{s}^{(\tau)} \tag{2.34}$$

Seuraavaksi approksimoidaan termiä $\mathbf{s}^{(\tau)}$ epäsymmetrisesti [8]

$$\mathbf{s}^{(\tau)} = \frac{\nabla E(\mathbf{w}^{(\tau)} + \delta \mathbf{p}^{(\tau)}) - \nabla E(\mathbf{w}^{(\tau)})}{\delta^{(\tau)}} \quad \text{, missä } 0 < \delta^{(\tau)} < 1, \tag{2.35}$$

Yllä olevan approksimaation on alunperin johtanut Magnus Hestenes [9] konjugaatin suuntia hyödyntävien optimointimenetelmiin liittyen ja se on nimeltään *konjugaattigradienttimenetelmä*. Kuitenkin määrittelyn (2.33) nojalla approksimaation on oltava positiivinen, jotta matriisi $\mathbf{H}$ on positiivisesti definiitti, ja täten $\mathbf{x}^{(\tau)}$ täytyy olla positiivinen. Ongelma voidaan ratkaista lisäämällä skaalausparametri λ konjugaattigradienttimenetelmään. Kun $\mathbf{x}^{(\tau)} < 0$, kompensoi skaalausparametri λ tämän approksimaation (2.35) positiiviseksi. Täten $\mathbf{s}^{(\tau)}$ saa lopullisen muotonsa

$$\mathbf{s}^{(\tau)} = \frac{\nabla E(\mathbf{w}^{(\tau)} + \delta \mathbf{p}^{(\tau)}) - \nabla E(\mathbf{w}^{(\tau)})}{\delta^{(\tau)}} + \lambda^{(\tau)} \mathbf{p}^{(\tau)}. \tag{2.36}$$

Skaalatun konjugaattigradienttialgoritmin käytössä on päivitettävä painokertoimien $\mathbf{w}$ lisäksi kantavektoreita $\mathbf{p}$, jotka osoittavat virhefunktion minivoivan suunnan ja askeleen pituuden. Päivitykset tapahtuvat seuraavilla yhtälöillä:

$$\begin{cases} \mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \alpha^{(\tau)} \mathbf{p}^{(\tau)} \\ \mathbf{p}^{(\tau+1)} = \mathbf{r}^{(\tau)} + \beta^{(\tau)} \mathbf{p}^{(\tau)} \end{cases} \tag{2.37}$$

missä $\mathbf{r}^{(\tau)} = -\nabla E(\mathbf{w})$ ja

$$\alpha^{(\tau)} = \frac{\mathbf{p}^{(\tau)^T}\mathbf{r}^{(\tau)}}{\mathbf{p}^{(\tau)^T}\mathbf{s}^{(\tau)}} = \frac{\mathbf{p}^{(\tau)^T}\mathbf{r}^{(\tau)}}{\mathbf{x}^{(\tau)}} \tag{2.38}$$

sekä

$$\beta^{(\tau)} = \frac{(\mathbf{r}^{(\tau+1)} \cdot \mathbf{r}^{(\tau+1)}) - (\mathbf{r}^{(\tau+1)} \cdot \mathbf{r}^{(\tau)})}{\mathbf{x}^{(\tau)}} \tag{2.39}$$

2.1.8 Levenberg-Marquardt algoritmi

Levenberg-Marquardt (LM) algoritmi perustuu perinteisemmän Gauss-Newton menetelmän (2.31) laajennukseen. Gauss-Newton menetelmän

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \left[\mathbf{J}^{(\tau)^T}\mathbf{J}^{(\tau)}\right]^{-1}\mathbf{J}^{(\tau)^T}\mathbf{e}^{(\tau)}$$

ongelma on, että painokertoimen päivitykselle ei löydy ratkaisua silloin, kun termi $\mathbf{J}^T\mathbf{J}$ on singulaarinen ja käänteismatriisia $(\mathbf{J}^T\mathbf{J})^{-1}$ ei ole määritelty. Levenberg-Marquardt algoritmiin lisätään skalaarimatriisi $\lambda\mathbf{I}$, jonka summauksella saadaan termi aina $(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I})$ positiivisesti definiitiksi. Painokertoimen päivitys tehdään siis yhtälöllä

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \left[(\mathbf{J}^{(\tau)})^T\mathbf{J}^{(\tau)} + \mu\mathbf{I}\right]^{-1}(\mathbf{J}^{(\tau)})^T\mathbf{e}^{(\tau)} \tag{2.40}$$

Skalaarimatriisille $\mu\mathbf{I}$ on ominaista, että sen kasvaessa algoritmi lähestyy laskevan gradientin menetelmää (2.18), ja termin pienentyessä algoritmi suppenee kohti Gauss-Newton menetelmää (2.31).

2.1.9 Monikerrosperseptroni

Neuroverkkojen erikoistapausta, joka koostuu syötekerroksesta, yhdestä tai useammasta piilokerroksesta neuroneja sekä vastekerroksesta, ja jossa syötesignaali etenee verkon läpi suoraviivaisesti kerros kerrokselta, kutsutaan *monikerrosperseptroniksi* (MLP). Sen kolme keskeistä ominaispiirrettä ovat, että

- Neuronien aktivointifunktio on epälineaarinen ja sileä eli se on jatkuvasti differentioituva. Yleisesti käytetään aiemmin määriteltyä logistista sigmoid-funktiota (2.7).
- Neuroverkossa on yksi tai useampi piilokerros neuroneja. Näiden avulla on mahdollista eritella progressiivisesti enemmän syötteen tärkeitä ominaisuuksia ja täten käsitellä monimutkaisempia ongelmia.
- Verkon rakenteet eli piirteet, piilokerrokset ja vasteet, ovat vahvasti linkittyneitä toisiinsa. Tämän seurauksena rakenteen säätäminen tarkoittaa koko verkkorakenteen muokkaamista ja uudelleen opettamista.

Edellä mainitut seikat huomioiden on monikerrosperseptronin teoreettinen analysointi erittäin vaikeaa. Toisaalta, piilokerrosten olemassaolo tekee myös verkon havainnollistamisen monimutkaiseksi. Verkon opetusprosessissa valikoituu mitä piirteitä ja ominaisuuksia neuronit painottavat. Täten myös oppiminen on monimutkaisempaa, koska on käytävä läpi paljon laajempi kokonaisuus mahdollisia funktioita ja syötteen piirteiden rooleja. Tästä seuraa, että piirteiden valinta on hyvin keskeisessä roolissa verkkon suunnittelussa. Monikerrosperseptronilla on mahdollista ratkaista monimutkaisia ja -ulotteisia ongelmia, kun se opetetaan *valvotusti*, eli syötteen vaste tunnetaan, käyttäen suosittua virheen takaisinjohtamisalgoritmia.

2.2 Tukivektorikone

Tukivektorikone on neuroverkkoa uudempi ja monessa mielessä edistyneempi hahmontunnistusmenetelmä, joka on tehokas erityisesti kaksi vasteluokkaa sisältävälle aineistolle. Sen toiminta perustuu luokkien erottamiseen toisistaan hypertason avulla, jonka määrää opetusaineiston tietyt pisteet, joita kutsutaan tukivektoreiksi. Tavoitteena luokittelussa on maksimoida etäisyys luokkien pisteiden ja hypertason välillä, jotta luokittelijan toiminta optimoituu. Soveltaville aineistoille, jotka eivät ole separoituvia luokkien suhteen, tukivektorit on määritettävä siten, että hypertasolla jaettujen luokkien väliin jää mahdollisimman suuri etäisyys, mutta hypertason ja tukivektoreiden väliin jää mahdollisimman vähän luokkien pisteitä eli luokitteluvirhe minimoituu.

2.2.1 Optimaalinen separoiva hypertaso

Optimaalinen separoiva hypertaso jakaa pisteet $\mathbf{x}_i$ niiden vasteluokkien, esimerkiksi $y_i = \{-1, 1\}$, perusteella kahteen osaan maksimoiden samalla luokkien pisteiden ja hypertason välisen etäisyyden.

Hypertaso voidaan määritellä tason yhtälön ja sisätulon avulla

$$(\mathbf{w} \cdot \mathbf{x}_i) + b = 0, \tag{2.41}$$

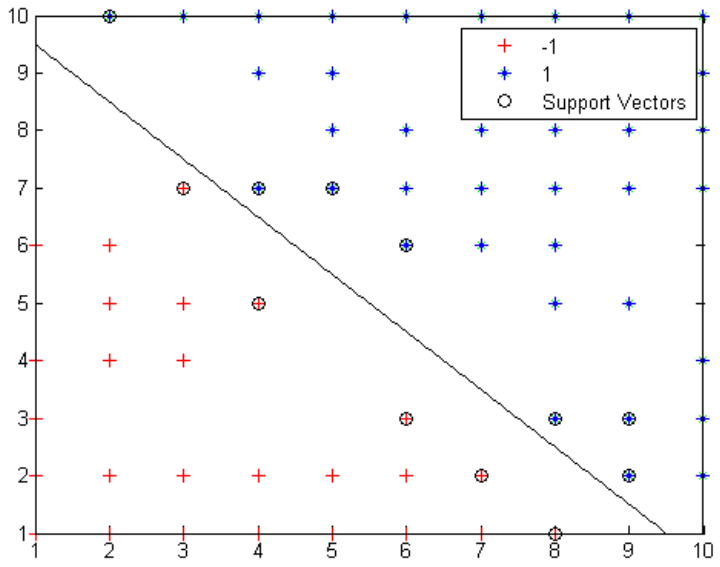
missä $\mathbf{w}$ on jonkun sisätuloavaruuden normaali ja $b \in \mathbb{R}$. Hypertaso on lineaarisesti separoiva, jos on olemassa $\mathbf{w} \in \mathbb{R}^n$ ja $b \in \mathbb{R}$, joille

$$\begin{cases} (\mathbf{w} \cdot \mathbf{x}_i) + b \geq 1 & , \text{kun } y_i = 1 \\ (\mathbf{w} \cdot \mathbf{x}_i) + b \leq 1 & , \text{kun } y_i = -1 \end{cases} \tag{2.42}$$

Yllä oleva yhtälöpari voidaan kirjoittaa yksinkertaisemmin muotoon

$$y_i(\mathbf{w} \cdot \mathbf{x}_i) + b \geq 1 \tag{2.43}$$

Nyt lineaarisesti separoivalle joukolle optimaalinen separoiva hypertaso maksimoi hypertason etäisyyden joukon pisteiden suhteen. Koska tason etäisyys



Kuva 2.2: Optimaalinen lineaarisesti separoiva hypertaso ja tukivektorit havainnollistettuna kahden luokan tapauksessa, jossa punaiset ja siniset pisteet edustavat kahteen eri luokkaan kuuluvia havaintoja.

lähimpään pisteeseen on $1/w$, täytyy optimoida hypertason yhtälö pisteiden etäisyyksien mukaan eli on optimoitava alla oleva yhtälöpari

$$\begin{cases} \min(\frac{1}{2}\mathbf{w} \cdot \mathbf{w}) \\ \text{s.e. } y_i(\mathbf{w} \cdot \mathbf{x}_i) + b \geq 1 \end{cases} \tag{2.44}$$

Nyt, jos on olemassa ratkaisu parille $(\mathbf{w}, b)$, niin ainakin yhdelle pisteelle $\mathbf{x}_i$ pätee $y_i(\mathbf{w} \cdot \mathbf{x}_i)+b = 1$. Täten ratkaisu on optimaalinen separoiva hypertaso.

2.2.2 Tukivektorit

Optimaalisen hypertason (2.44) ratkaisemisessa hyödynnetään *Lagrangen menetelmää*, jossa *Lagrangen kertoimien* $\alpha_i \geq 0$ avulla ratkaistaan *Lagrangen funktio*

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i\{y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b] - 1\} \tag{2.45}$$

Lagrangen funktio L on minimoitava muuttujien $\mathbf{w}$ ja b suhteen, mutta maksimoitava Lagrangen kertoimien suhteen $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$. Tämän nojalla saadaan yhtälö muotoon

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n} y_i \alpha_i = 0 \tag{2.46}$$

Eli

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{2.47}$$

Yllä olevan yhtälön toteuttavat ratkaisut, joille $\alpha_i \neq 0$, ovat niin kutsuttuja tukivektoreita. Nämä tukivektorit sijaitsevat kahden luokan havaintojen reunoilla lähimpänä hypertasoa, jota on havainnollistettu kuvassa 2.2.

Kun Lagrangen funktioon (2.45) sijoitetaan (2.46) ja (2.47) päästään eroon termeistä $\mathbf{w}$ ja b. Tätä yhtälöä kutsutaan *duaaliseksi optimointiongelmaksi*, missä on maksimoitava

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j), \tag{2.48}$$

jossa $\alpha_i \geq 0$ kaikille $i = 1, \ldots, n$ ja $\sum_{i=1}^{n} \alpha_i y_i = 0$.

Hypertason mukainen tukivektorikoneen *päätösfunktio* havainnoille määritellään

$$f(\mathbf{x}) = \operatorname{sgn}(\mathbf{w} \cdot \mathbf{x}) + b, \tag{2.49}$$

joka Lagrangen kertoimin ilmaistuna tulee muotoon

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i (\mathbf{x} \cdot \mathbf{x}_i) + b \right) \tag{2.50}$$

2.2.3 Piirreavaruus

Piirreavaruus mahdollistaa epälineaaristen piirteiden kuvaamisen lineaarisesti separoiville hypertasoille. On mahdollista kuvata mikä tahansa epälineaarinen funktio lineaarisesti separoituvaksi moniulotteiselle piirreavaruudelle, mikä johtaa lineaariseen luokitteluongelmaan.

$$\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \mapsto \Phi(\mathbf{x}) = \Phi_1(\mathbf{x}), \ldots, \Phi_n(\mathbf{x}) \tag{2.51}$$

Sisätulon avulla ilmaistuna tämä tulee muotoon

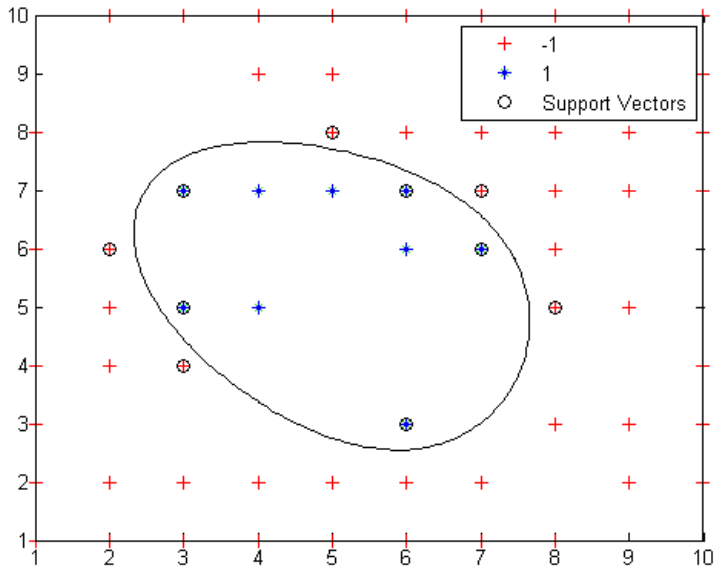
$$K(\mathbf{x}_i, \mathbf{x}) = (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x})) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}) \tag{2.52}$$

Funktiota $K(\cdot)$ kutsutaan *ytimeksi*, joka on symmetrinen funktio argumenttien suhteen eli $K(\mathbf{x}, \mathbf{x}_i) = K(\mathbf{x}_i, \mathbf{x})$. Yksinkertaisimillaan voidaan valita lineaarinen ydin, jolle

$$\Phi(\mathbf{x}) = \mathbf{x} \qquad \text{eli} \qquad K(\mathbf{x}_i, \mathbf{x}) = \mathbf{x}_i^T \mathbf{x} \tag{2.53}$$

Jotta voidaan kuvata epälineaarisia piirteitä lineaarisesti separoivilla tukivektoreilla, täytyy valita epälineaarinen kuvaus piirteille, joka on muotoa

$$f(\mathbf{x}) = \sum_{i=1}^{N} w_i \Phi_i(\mathbf{x}) + b \tag{2.54}$$



Kuva 2.3: Ydinfunktion avulla, joka hyödyntää korkeampi ulotteista piirreavaruutta, on mahdollista konstruoida optimaatinen separoiva hypertaso epälineaariselle päätösfunktiolle. Kuvassa punaiset ja siniset pisteet edustavat kahden eri luokan havaintoja.

missä $\Phi : X \mapsto F$ on epälineaarinen kuvaus syötteeltä piirreavaruudelle.

Käytännössä siis opetusdatan lineaarikombinaatioilla voidaan kuvata haluttu hypoteesi. Tukivektorikoneen päätösfunktio Lagrangen kertoimilla voidaan laajentaa ydinfunktiolle

$$f(\mathbf{x}) = \operatorname{sgn}\left[\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right] \tag{2.55}$$

Täten, jos löydetään sopiva ydinfunktio $K(\mathbf{x}_i, \mathbf{x})$, jolle sisätulo on määritelty, niin on mahdollista myös konstruoida tälle tukivektorikone.

2.2.4 Gaussinen ydinfunktio

Tukivektorikoneessa käytettävän ydinfunktion valinnalle on useita mahdollisuuksia, kuten polynomi-, hyperbolinen- tai *pyörähdyssymmetrinenfunktio*. Tarkastelaan viimeksi mainittua, koska se on yleisesti käytetty ja jokainen tällainen funktio on ainoastaan riippuvainen säteen Euklidisesta etäisyydestä sen ympyrän keskiöön nähden. Joten

$$\Phi(\mathbf{x}) = h(\|\mathbf{x} - \mathbf{x}_i\|) \tag{2.56}$$

Olkoon syöteverktorien joukko $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ ja näitä vastaavat vasteiden arvot $\{t_1, \ldots, t_n\}$. Tällöin on löydettävä sileä funktio $f(x)$ siten, että syötevektorit kuvautuvat niitä vastaaville vasteille eli $f(\mathbf{x}_i) = t_i$, jokaiselle $i = 1, \ldots, n$. Tällaiset funktiot löytyvät, kun valitaan lineaarikombinaatiot, joiden keskipisteinä toimivat syöteet

$$f(\mathbf{x}) = \sum_{i=1}^{n} w_i h(\|\mathbf{x} - \mathbf{x_i}\|), \tag{2.57}$$

missä kertoimet $\{w_i\}$ on ratkaistavissa pienimmän neliösumman menetelmällä. Koska vakioita on yhtä monta kuin muuttujia, ne vastaavat täsmälleen haluttuja vasteita. On kuitenkin pidettävä mielessä, että täydellinen interpolointi johtaa ylioppimiseen, eikä se täten ole tavoiteltavaa.

Jotta päästään duaalisesta päätöksenteosta todennäköisyyksiin pohjautuvaan hahmontunnistukseen, on mahdollista yhdistää Gaussin prosesseja tukivektorikoneen ytimen suunnitelussa. Gaussin prosessien näkökulmassa luovutaan perinteisestä parametrimallista ja määritellään ennalta todennäköisyysjakauma ydinfunktioille. Vaikka todennäköisyysjakaumalle voidaan määritellä ääretön määrä mahdollisia tapahtumia, niin käsitellään ainoastaan syötteitä $\mathbf{x_i}$ vastaavia vasteita ja täten kyseessä on äärellinen määrä funktioita. *Gaussinen ydinfunktio* määritellään

$$K(\mathbf{x}_i, \mathbf{x}) = \exp\left(- \frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2} \right), \tag{2.58}$$

jollekin $\sigma \in \mathbb{R}$.

2.2.5 Gaussista ydinfunktiota hyödyntävän tukivektorikoneen konstruointi

Tukivektorikoneen pioneerit, Vapnik ja Schölkopf et al., ovat käsitelleet artikkelissaan [10] ytimekkäästi, kuinka gaussista ydinfunktiota hyödyntävä tukivektorikone otetaan käyttöön. Seuraavassa konstruoidaan luokittelija

$$f(\mathbf{x}) = \operatorname{sgn}\left[\sum_{i=1}^{l} w_i \exp\left(- \frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{c_i} \right) + b \right] \tag{2.59}$$

missä b ja $c_i > 0$ ovat vakioita.

Ympyröiden keskipisteiden valinnalle $\mathbf{x}_i$ on useita vaihtoehtoja. Näistä kahta ääripäätä edustaa:

- Perinteisempi menettely on valita keskipisteet klusterointimenetelmällä, joka on riippumaton luokitteluongelmasta. Tällöin painokertoimien määrittämiseen voitaisiin käyttää esimerkiksi jotain aiemmassa esitettyä opetusalgoritmia.

- Edistyneempi lähestymistapa on etsiä tärkeimmät pisteet luokitteluongelman kannalta. Nämä pisteet on löydettävissä käyttämällä hyväksi tukivektorikonealgoritmia, joka perustuu tilastolliseen riskiteoriaan ja rakenteellisen riskinminimointiin.

Tukivektorikonealgoritmi on osoittautunut ylivertaiseksi verratuna perinteisempään lähestymistapaan [10]. Käytännössä on konstruoitava sopiva ydinfunktio, joka tässä tapauksessa on gaussinen pyörähdyssymmetrinenfunktio, ja säädettävä hyperparametrit optimaalisiksi, jotka on mahdollista löytää esimerkiksi ristiinvalidoinnilla.

On ratkaistava kahdensuuntainen luokitteluongelma, jossa vasteet edustavat jompaa kumpaa ääripäätä, olkoon ne $y_i \in \{-1, 1\}$. Olkoon lisäksi syötevektorit $\mathbf{x}_i$ ja funktioiden joukko $\{f_\alpha : \alpha \in \Lambda\}$ kuvaus
$f_\alpha : \mathbb{R}^N \to \{-1, 1\}$.

1. Rakenteelisen riskin minimointi.

   Oletaan aluksi, että meillä on osajoukko aineistosta
   $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_l, y_l)\}$, joiden määrittelyjoukot kuten yllä, ja jotka ovat Gaussin prosessin mukaisesta tuntemattomasta todennäköisyysjakaumasta $P(\mathbf{x}, y)$. Halutaan löytää funktio f_α, joka minimoi keskimääräisen virheen satunnaisesti valituille syötteille todennäköisyysjakaumasta $P(\mathbf{x}, y)$.

   $$R(\alpha) = \int \frac{1}{2}|f_\alpha(\mathbf{x}) - y| \; dP(\mathbf{x}, y) \tag{2.60}$$

   Ongelma on, että $R(\alpha)$ on tuntematon, sillä $P(\mathbf{x}, y)$ on tuntematon. Voidaan kuitenkin käyttää induktioperiaatetta riskin minimointiin, jolloin minimoidaan empiirinen riski

   $$R_{emp}(\alpha) = \frac{1}{l} \sum_{i=1}^{l} \; \frac{1}{2} \; |f_\alpha(\mathbf{x}_i) - y_i| \tag{2.61}$$

   Tämä lähestymistapa ei kuitenkaan ole optimaalinen eli se ei tuota pienintä mahdollista riskiä, kun käytössä on äärellinen määrä opetusaineistoa. Tähän ongelmaan on kehitetty *rakenteellinen riskinminimointiperiaate* [11]. Rakenteelliselle riskille on olemassa raja-arvo, jossa jollekin $\alpha \in \Lambda$ vähintään todennäköisyydellä $1 - \nu$, pätee

   $$R(\alpha) \leq R_{emp}(\alpha) + \phi\left[\frac{h}{l}\right] \tag{2.62}$$

   missä funktio ϕ määritellään

   $$\phi\left[\frac{h}{l}\right] = \sqrt{\frac{h\big(\log \frac{2l}{h} + 1\big) - \log(\nu/4)}{l}}$$

Edellä mainituissa yhtälöissä termiä h kutsutaan funktiojoukon Vapnik-Chervonenkis (VC) dimensioksi, joka kuvaa vektorikoneen toteuttamiskelpoisten funktioiden määrää. Binäärisessä luokitteluongelmassa h on maksimaalinen määrä pisteitä, jotka voidaan luokitella kahteen eri luokkaan sen funktioiden avulla eli 2^h.

Nyt yhtälön (2.62) nojalla, kun valitaan l kappaletta opetushavaintoja, voidaan riskiä hallita säätämällä kahta aiemmin mainittua ominaisuutta:

- Empiiristä riskiä $R_{emp}(\alpha)$, jonka suuruuteen vaikuttaa luokitteluun valittu funktio eli α.
- VC-dimensiota $h\{f_\alpha : \alpha \in \Lambda'\}$, missä Λ' on indeksijoukon Λ jokin osajoukko. VC-dimensio h on riippuvainen funktiojoukosta f_α, joten sen hallittavaan määrittämiseen hyödynnetään funktiojoukon sisäkkäisiä osajoukkoja $S_n := \{f_\alpha : \alpha \in \Lambda_n\}$, joille

  $$S_1 \subset S_2 \subset \cdots \subset S_n \subset \ldots \tag{2.63}$$

  ja joiden VC-dimensiot ovat vastaavasti

  $$h_1 \leq h_2 \leq \cdots \leq h_n \leq \ldots$$

Nyt rakenteellinen riskinminimointiperiaate valitsee funktion $f_{\alpha_l^n}$ osajoukosta $\{f_\alpha : \alpha \in \Lambda_n\}$, jolle riski on varmasti pienin yhtälön (2.62) nojalla.

2. Hypertasojen rakenne.

   Rakenteelliset muutokset eli funktiojoukon osajoukkojen S_n valinnat vaikuttavat separoituvien hypertasojen joukkoon ja täten myös opetusalgoritmiin.

   Olkoon sisätuloavaruus $\mathbf{Z} := \{\mathbf{z}_1, \ldots, \mathbf{z}_r\}$, jossa hypertasot kuvautuvat $\{\mathbf{z} \in \mathbf{Z}\} : (\mathbf{x} \cdot \mathbf{z}) + b = 0\}$. Kanoniselle parille $(\mathbf{w}, b)$ pätee

   $$\min_{i=1,\ldots,r} |(\mathbf{w} \cdot \mathbf{z}_i) + b| = 1 \tag{2.64}$$

   jossa $\mathbf{w}$ ja b valitaan siten, että lähin hypertason piste on etäisyydellä $\frac{1}{\|w\|}$.

   Määritellään seuraavaksi pallo $B_R(a) = \{\mathbf{z} \in \mathbf{Z} : \|\mathbf{z} - \mathbf{a}\| < R\}$, jonka säde R on pienin mahdollinen säde siten, että se sisältää pisteet $\mathbf{z}_1, \ldots, \mathbf{z}_r$. Lisäksi näille pisteille on määritelty päätösfunktio

   $$f_{\mathbf{w},b} = \operatorname{sgn}[(\mathbf{w} \cdot \mathbf{z}) + b] \tag{2.65}$$

Voidaan osoittaa, että hypertasojen joukolle on mahdollista määritellä rakenne. Tämä tulos perustuu ehtoon [12], jonka mukaan kanonisten hypertasojen joukolla $\{f_{\mathbf{w},b} : \|\mathbf{w}\| \leq A\}$ on VC-dimensio h, jolle pätee

$$h < R^2 A^2 + 1 \tag{2.66}$$

Ilman ehtoa $\|\mathbf{w}\| \leq A$ päädytään funktiojoukkoon, jonka VC-dimensio on $N+1$, missä N vastaa sisätuloavaruuden $\mathbf{Z}$ astetta. Täten ehdon ansiosta on tukivektorikoneella mahdollista käsitellä huomattavasti korkeamman asteisia dimensioita.

3. Tukivektorialgoritmi lineaariselle päätösfunktiolle

   Halutaan löytää funktio, jolle pätee

   $$f_{\mathbf{w},b}(\mathbf{z}_i) = y_i, \qquad i = 1, \ldots, l$$

   Tälle on myös voimassa yhtälön (2.64) nojalla

   $$y_i[(\mathbf{w} \cdot \mathbf{z}) + b] \geq 1, \qquad i = 1, \ldots, l \tag{2.67}$$

   Käytännön sovelluksissa kuitenkaan tällaista funktiota ei löydy, kuin aniharvoin, joten ehtoa (2.67) voidaan keventää ottamalla mukaan termi $\epsilon \geq 0$, jolloin yhtälö tulee muotoon

   $$y_i[(\mathbf{w} \cdot \mathbf{z}) + b] \geq 1 - \epsilon, \qquad i = 1, \ldots, l \tag{2.68}$$

   Nyt tukivektorialgorimin optimoinnissa on minimoitava raja-arvo rakenteelliselle riskille (2.62), jolloin sen on toteutettava ehto

   $$\zeta(\mathbf{w}, \epsilon) = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) + \gamma \sum_{i=1}^{l} \epsilon_i, \tag{2.69}$$

   missä γ on positiivinen vakiotermi, joka määräytyy funktiojoukon valinnan perusteella. Ehdon (2.66) nojalla termin $\frac{1}{2}(\mathbf{w} \cdot \mathbf{w})$ minimointi tarkoittaa VC-dimension minimointia eli myös rakenteellisen riskin (2.62) jälkimmäinen termi $\phi\left[\frac{h}{l}\right]$. Toisaalta $\sum_{i=1}^{l} \epsilon_i$ on yläraja opetusaineiston virheellisesti luokitelluille näytteille, joka myös vaikuttaa rakenteelliseen virheeseen (2.62), jolloin vakio γ on valittava sopivaksi.

   Optimoidessa raja-arvoa rakenteelliselle riskille otetaan käyttöön Lagrangen menetelmä (2.45) ja Kuhn-Tuckerin teoreemaa, jolloin saadaan esitys

   $$\mathbf{w} = \sum_{i=q}^{l} y_i \alpha_i \mathbf{z}_i, \tag{2.70}$$

jonka kertoimille α_i on löydettävä maksimaalinen $\alpha_i \neq 0$, jolloin syötteen havainnot $(\mathbf{z}_i, y_i)$ toteuttavat ehdon (2.68). Tällöin on löydetty tukivektorit $\mathbf{z}_i$. Maksimaalinen α_i saadaan ratkaistua yhtälöllä

$$W(\alpha) = \sum_{i=q}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j (\mathbf{z}_i \cdot \mathbf{z}_j), \tag{2.71}$$

missä

$$0 \geq \alpha_i \geq \gamma, \qquad i = 1, \ldots, l \quad \text{ja} \qquad \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{2.72}$$

Nyt sisätulon lineaarisuuden nojalla funktio (2.65) voidaan kirjoittaa muotoon

$$f(\mathbf{z}) = \operatorname{sgn}\left[\sum_{i=1}^{l} \alpha_i y_i \cdot (\mathbf{z} \cdot \mathbf{z}_i) + b \right] \tag{2.73}$$

4. Piirreavaruutta hyödyntävä tukivektorialgoritmi

   Edellä osoitettiin, kuinka tukivektorialgoritmi toimii lineaarille päätösfunktiolle. Konstruoidaan nyt tukivektorialgoritmi, jossa hyödynnetään piirreavaruutta.

   Olkoon syötteet kuten aiemmin ja kuvaus $\Phi : \mathbf{x}_i \mapsto \mathbf{z}_i$. Maksimoidakseen yhtälön (2.71) ja ratkaistakseen päätösfunktion (2.65) pitää pystyä ratkaisemaan sisätulo $(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i))$ korkeaulotteisessa piirreavaruudessa. Täytyy määrittää sopiva ydinfunktio siten, että

   $$K(\mathbf{x}, \mathbf{x}_i) = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i),$$

   jolloin päätösfunktio (2.55) tulee muotoon

   $$f(\mathbf{x}) = \operatorname{sgn}\left[\sum_{i=1}^{l} \alpha_i y_i \cdot K(\mathbf{x}, \mathbf{x}_i) + b \right] \tag{2.74}$$

   Funktionaalianalyysin Mercerin teoreeman nojalla voidaan positiivisesti definiitti ja jatkuva ydinfunktio $K(\mathbf{x}, \mathbf{x}_i)$ laajentaa tasaisesti suppenevaksi sarjaksi ominaisarvoja λ_j ja ominaisfunktioita ψ_j

   $$K(\mathbf{x}, \mathbf{x}_i) = \sum_{j=1}^{\infty} \lambda_j \psi_j(\mathbf{x}) \psi(\mathbf{x}_j), \tag{2.75}$$

   jotka vastaavat sisätuloa $(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i))$ ja kuvautuu
   $\Phi : \mathbf{x} \mapsto \left[\sqrt{\lambda_1}\psi_1(\mathbf{x}), \sqrt{\lambda_2}\psi_2(\mathbf{x}), \ldots \right]$. Tämän seurauksena voidaan tukivektorialgoritmin lineaarista tapausta soveltaa myös epälineaarisille tapauksille hyödyntäen sopivaa ydinfunktiota.

5. Gaussisen ydinfunktion implementointi

   Voidaan valitaan aiemmin esitelty sädepohjafunktio (2.59) ydinfunktioksi

$$K(\mathbf{x}, \mathbf{x}_i) = \exp\left(- \frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{c_i} \right) \tag{2.76}$$

   Nyt maksimoidaan sädepohjaydinfunktiollinen päätösfunktio neliöllisellä menetelmällä

$$W(\alpha) = \sum_{i=q}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j (\mathbf{z}_i \cdot \mathbf{z}_j),$$

   missä

$$0 \geq \alpha_i \geq \gamma, \qquad i = 1, \ldots, l \quad \text{ja} \qquad \sum_{i=1}^{l} \alpha_i y_i = 0$$

   Vakiota b kutsutaan *kynnysarvoksi* ja sen laskemiseksi hyödynnetään yhtälöä (2.68) ja tietoa, että tukivektoreille $\mathbf{x}_j$, joille $\epsilon = 0$ pätee

$$\sum_{i=1}^{l} \alpha_i y_i \cdot K(\mathbf{x}_j, \mathbf{x}_i) + b = y_j \tag{2.77}$$

Täten on konstruoitu tukivektorikone gaussisella ydinfunktiolla.

Luku 3

Piirteiden valintamenetelmät

Luokittelijalle syötettävät havainnot, koostuvat yleensä useista eri muuttujista, kuten esimerkkinä tässä työssä aikaleimoista, materiaali-, mittaus- ja mallinnustiedoista. Näistä useista muuttujista, jotka kohdistuvat yksittäisiin havaintoihin, käytetään nimistystä piirteet. Riippuen aineiston alkuperästä ja käyttötarkoituksesta näitä piirteitä voi olla liiankin paljon, eikä niiden tärkeydestä aineiston luokittelun kannalta ole välttämättä tietämystä.

Motivaatio piirteiden valintamenetelmien käyttöön tulee muun muassa siitä, että niillä on mahdollista parantaa luokittelijoiden ennustustarkkuutta, nopeuttaa niiden toimivuutta, ja yksinkertaistaa sekä tuottaa parempaa ymmärrystä prosessista, johon luokittimia hyödynnetään [13].

Tämän työn kannalta piirteiden valinta on oleellista erityisesti neuroverkkojen osalta, koska laskentateho on riippuvainen syötevektorien dimensiosta, toisin kuin tukivektorikoneessa. Kyseinen ongelma tunnetaan käsitteenä *dimensioiden kirous*, jonka alunperin esitteli Bellman vuonna 1957 [14]. Seuraavassa esitettyjen piirteiden valintamenetelmien avulla saavutettiin kaksi mielenkiintoista ryhmää, joita hyödynnettiin luokittelussa.

3.1 Piirteiden lisäys- ja poistomenetelmä

Piirteiden lisäys- ja poistomenetelmä, jäljempänä FBFS (Forward/backward feature selection algorithm), perustuu niin sanotusti kokeelliseen piirteiden valintaan ja menetelmä toteutettiin monikerrosperseptronilla Levenberg-Marquardt-algoritmilla.

Piirteiden joukosta valittiin yksi piirre kerrallaan luokitteluun sen perusteella kuinka paljon neuroverkon ennustustarkkuus parantui. Sitä mukaa, kun piirteitä hyväksyttiin luokitteluun, oli jokaisella kierroksella mahdollista tiputtaa jokin edellä valituista piirteistä pois aineistosta, jos poistaminen paransi tulosta. Vastaavaa menetelmää on käyetty menestyneesti sopivien piirteiden ja kontekstien määrittämiseen [15].

3.2 Itsejärjestäytyvät kartat

Samalle aineistolle tuotetussa edeltävässä tutkimuksessa [16] hyödynnettiin *itsejärjestäytyviä karttoja* (SOM) piirteiden valinnassa. Menetelmän kehittäjä on suomalainen Teuvo Kohonen, joka on myös kirjoittanut erittäin kattavan teoksen aiheen ympärille [17].

Itsejärjestäytyvät kartat perustuvat yksi- tai kaksiulotteiseen neuronimalliin, jossa neuronit voidaan ajatella diskreetteinä pisteinä tasossa muodostaen niin kutsutun kartan, toisin sanoen verkon tai hilan. Verkon oppiminen tapahtuu ilman valvontaa, joten havainnot luokitellaan pelkästään niiden piirteiden perusteella huomioimatta vasteluokkia. SOM:lla on mahdollista kuvata lukuisia piirteitä sisältävä aineisto kaksiulotteiselle kartalle säilyttäen havaintojen suhteelliset etäisyydet. Täten SOM on tehokas klusterointimenetelmä, jolla voidaan analysoida piirteiden käyttäytymistä toisiinsa nähden.

Opetuksessa havannoille lasketaan Euklidiset etäisyydet verkossa oleviin neuroneihin nähden. Neuroni, joka osuu lähimmäs syötettä valitaan parhaiten kuvaavaksi yksiköksi, jolloin tämän neuronin ja sen määrätyssä ympäristössä olevien neuronien painokertoimia säädetään kuvaamaan aineistoa paremmin. Opetuksen jälkeen kartasta voidaan visuaalisesti tulkita värikoodien perusteella esimerkiksi kuinka yksittäiset piirteet käyttäytyvät suhteessa muuhun piirrejoukkoon. Täten aineistosta on mahdollista esimerkiksi poistaa piirteitä, jotka selvästi näyttävät sekoittavan aineistoa eivätkä tuota selviä ryppäitä karttaan.

Luku 4

Aineisto ja tutkimusmenetelmät

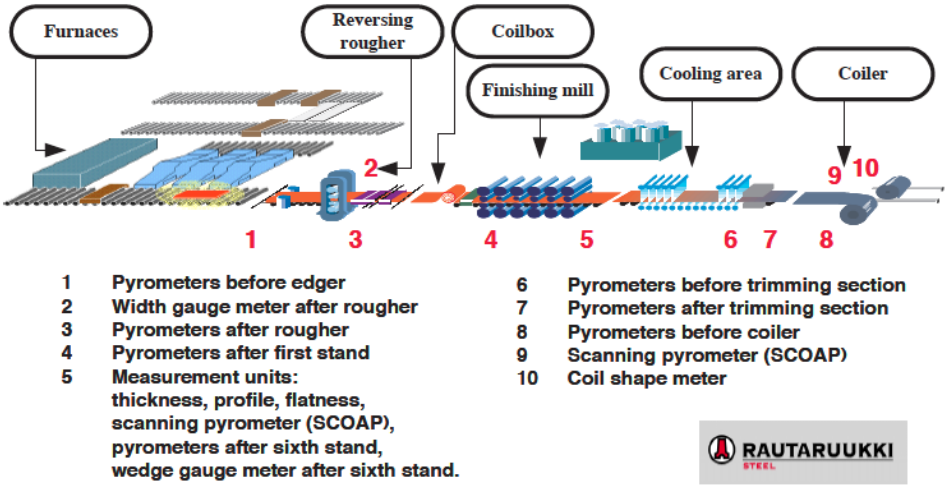
4.1 Tutkimusaineisto

Tutkimuksen aineisto käsittelee teräksen nauhavalssausta ja se on peräisin *Ruukin* (entinen *Rautaruukki*) terästehtaan tuotannonsuunnittelu- ja tuotannonvalvontajärjestelmistä Raahesta. Aineisto sisältää teräksen valssausprosessin lämpötila- ja dimensiotietoja ja näille mallinnettuja tavoite arvoja, sekä kemiallisia ominaisuuksia. Myös valssausprosessin eri vaiheista ja niiden kestosta on taltioitu aikaleimat.

Teräksen nauhavalssausprosessia voidaan karkeasti kuvailla seuraavasti:

- Kymmeniä tonneja painava teräsaihio kuumennetaan uuneissa tavoitelämpötilaansa valssausprosessia varten.
- Karkeassa esivalssauksessa teräs saatetaan tavoitepaksuuteensa ajamalla sitä valssikkaiden läpi edestakaisin.
- Lopullisessa valssauksessa aihio saa tarkat tavoitemittansa.
- Valssauksen jälkeen nauha jäähdytetään vesisuihkuilla, mikä on tärkeä vaihe teräsnauhan mekaanisten ominaisuuksien kannalta.
- Valmis nauha kelataan rullalle valmiiksi tuotteeksi.

Tuotantolinjaston, jota on havainnollistettu kuvassa 4.1, automatisoidut tietojärjestelmät hyödyntävät mittalaitteiden dataa ja mallintavat lämpövalssausprosessia sekä siihen liittyviä lukuisia parametreja tehokkaasti. Prosessin virheetön fysikaalinen mallintaminen on lähes mahdotonta, joten teräsnauhan valmistuksessa esiintyy herkästi erilaisia poikkeavuuksia. Automatisoitu tuotannonvalvontajärjestelmä merkitsee tietokantaan virhekoodin niille nauhoille, joiden lopputuloksessa on jotakin tavallisesta poikkeavaa.



Kuva 4.1: Ruukin tehtaan tuotantolinjasto havainnostettuna.

Näitä poikkeavuuksia on tuotannosta vastaavien henkilöiden tarkastettava ja niille on mahdollisesti suoritettava erilaisia toimenpiteitä jälkikäteen. Valmiin teräsnauhan tarkistaminen ja mahdollinen muokkaaminen on kallista ja aikaavievää, joten tutkimuksen tarkoituksena oli kehittää tuotantoa entisestään tuomalla hahmontunnistukseen perustuvaa koneälyllistä laadunvalvontaa linjastolle.

Alkuperäisessä aineistossa oli yli kaksisataa piirrettä, mutta edeltävän tutkimuksen [16] ja asiantuntijoiden tietämyksen perusteella lopullisia piirteitä jäi aineistoon 37 kappaletta. Osa näistä piirteistä oli luokkamuuttujia, jotka jaoteltiin binäärisiksi, toisin sanoin esimerkiksi viisi mahdollista arvoa sisältävästä luokkamuuttujasta luotiin 5 erillistä binääristä piirrettä. Täten käytössä oli 77 jatkuvaa tai binäärista piirrettä, joista

- 33 kappaletta materiaalitietoja,
- 9 kappaletta lämpötilatietoja,
- 12 kappaletta dimensiotietoja,
- 23 kappaletta valssaustietoja.

Havaintoja aineistossa oli 6004 kappaletta, joista 25.5% oli virhekoodillisia.

4.2 Tutkimusmenetelmät

4.2.1 Neuroverkot

Neuroverkot, joita hyödynnettiin tutkimuksessa olivat monikerrosperseptroneja (2.1.9), joiden painokertoimia päivitettiin virheentakaisinjohtamis-

menetelmällä (2.1.4). Opetusalgorimeista hyödynnettiin

- Levenberg-Marquardt -algortmia, koska tämä on osoitettu nopeimmaksi algoritmiksi keskisuurien neuroverkkojen opetuksessa [18] [3].
- Skaalattua konjugaattigradienttialgoritmia, koska se on täysin automatisoitu eli se ei vaadi käyttäjältä kriittistä näkemystä parametrien valinnasta, ja algoritmi on osoittautunut nopeammaksi kuin tavanomainen takaisinjohtamis- tai konjugaattigradienttialgoritmi [8].

Neuronien piilokerrosten lukumäärä rajoitettiin yhteen tai kahteen riippuen piirteiden lukumäärästä. Aktivointifunktiona käytettiin differentioituvaa logistista sigmoid -funktiota (2.7).

4.2.2 Tukivektorikone

Tukivektorikone toteutettiin gaussisella ydinfunktiolla (2.58). Optimaalisten hyperparametrien valinnassa haarukoitiin laajasti eri mahdollisuuksia, joiden tulokset varmennettiin ristiinvalidoinnilla, jotta optimaalisuus saavutettiin.

Luku 5

Tulokset

Aineiston esikäsittelyn ja piirteiden valintamenetelmien pohjalta luokitteluun valittiin kolme erillaista piirrekokoonpanoa, joita käytettiin luokittelussa:

- FBFS menetelmällä saavutettiin 12 piirrettä muodostava ryhmä.
- SOM menetelmän ryhmä koostui 16 kappaleesta piirteitä.
- Kaikki 77 piirrettä olivat kokonaisuudessaan yhtenä ryhmän.

Opetusaineiston vasteet oli jaoteltu binäärisesti siten, että virhekoodillinen havainto sai arvon 1 ja onnistunut arvon 0. Täten luokittelijoiden ennusteet sai arvoja väliltä 0 ja 1 eli $0 \leq y_i \leq 1$. Luokittelun päätöspisteeksi valittiin 0.5 eli ennusteen ollessa pienempi kuin $0,5$ merkittiin se onnistuneeksi ja päinvastoin arvoa 0.5 suuremmat ennusteet merkittiin virhekoodilliseksi eli

$$y_i = \begin{cases} 0 & \text{, jos } y_i \leq 0.5 \\ 1 & \text{, jos } y_i > 0.5 \end{cases} \tag{5.1}$$

Luokittelija	Piirrejoukko	Ennustusvirhe
Neuroverkko(LM)	FBFS (12 piirrettä)	17.5%
Neuroverkko(LM)	SOM (16 piirrettä)	17.6%
Neuroverkko(LM)	Kaikki (77 piirrettä)	17.7%
Neuroverkko(SCG)	FBFS (12 piirrettä)	17.8%
Neuroverkko(SCG)	SOM (16 piirrettä)	18.1%
Neuroverkko(SCG)	Kaikki (77 piirrettä)	17.6%
Tukivektorikone	FBFS (12 piirrettä)	19.8%
Tukivektorikone	SOM (16 piirrettä)	18.7%
Tukivektorikone	Kaikki (77 piirrettä)	18.1%

Taulukko 5.1: Luokittelijoiden ennustevirheet eri piirrejoukoilla

Luokittelijoiden tulokset eri piirrejoukoilla näkyvät taulukossa 5.1. Nähdään, että LM-algoritmilla opetettu neuroverkko tuotti pienimmän virheen ennusteille 12 piirrettä sisältävällä aineistolla. Tukivektorikone ja neuroverkko, joka opetettiin SCG-algorimilla, suoriutuvat parhaiten syötteenään kaikki 77 piirrettä. Nämä tulokset antavat vertailukohdan sille, kuinka luokittelijat suoriutuivat kyseisellä aineistolla.

Tulosten tarkempi analysointi kuitenkin osoitti, että suurin osa luokitteluvirheestä esiintyi virhekoodillisten teräsnauhojen luokassa $\{y_i = 1\}$. Virhekoodillisista havainnoista 45% oli luokiteltu väärin, kun taas onnistuneille nauhoille $\{y_i = 0\}$ luokitteluvirhe oli vain 5%. Tästä oli mahdollista päätellä, että liian suuri osa havainnoista luokitellaan onnistuneiksi $\{y_i = 0\}$.

Virhekoodillisten havaintojen ennustettavuuden parantamiseksi etsittiin ratkaisua säätämällä luokittelijoiden kynnysarvoa ja harhatermejä. Parametrien optimaalisuutta testattiin usealla aineiston osajoukolla, jotka olivat erillisiä opetusaineiston havainnoista, jotta löydettiin mahdollisimman geneeriset ja luotettavat parametrit. Taulukoissa 5.2 ja 5.3 on esitetty testiaineiston onnistuneiden ja virhekoodillisten havaintojen todelliset arvot riveillä ja ennusteet sarakkeilla. Taulukoista voi nähdä, että tämän menettelyn seurauksena aineiston ennusteiden kokonaisvirhe kasvoi huomattavasti, mutta väärinluokitellut havainnot olivat tasapainoisemmin jakautuneet luokkien kesken.

Neuroverkko	Onnistuneet	Virhekoodit	Havainnot	Ennustusvirhe
Onnistuneet	1556	610	2166	28,2%
Virhekoodilliset	206	557	763	27,0%
Ennusteet	1762	1167	2929	27,9%

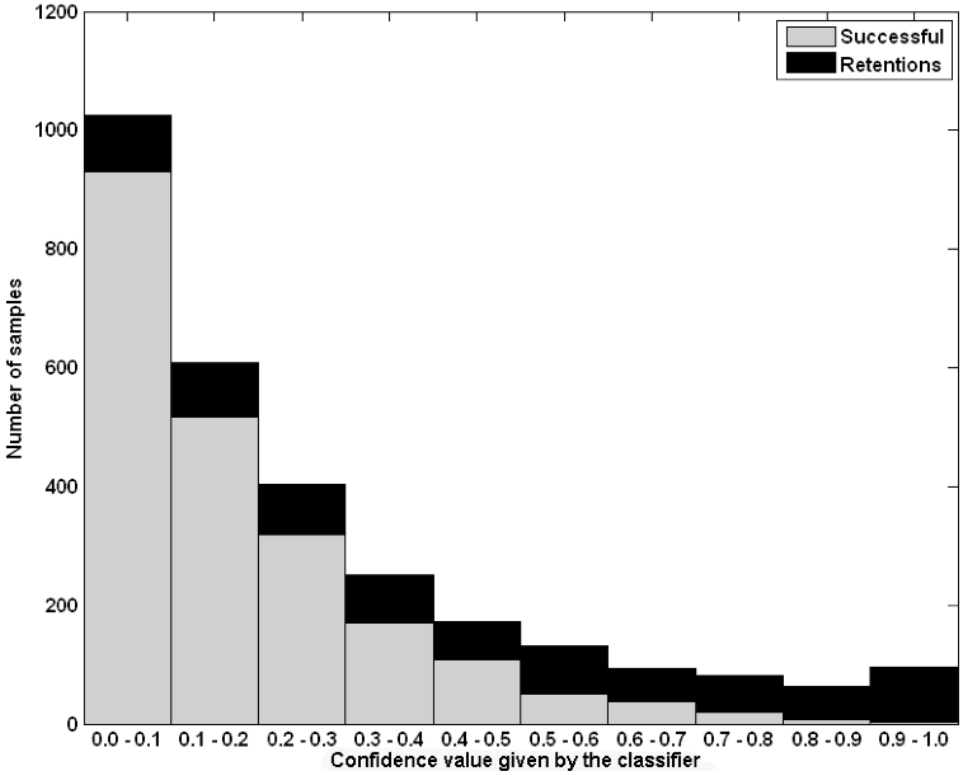
Taulukko 5.2: Neuroverkon ennustusvirhe onnistuneille havainnoille 28,2% ja virhekoodillisille havainnoille 27,0%, ja ennusteen kokonaisvirhe on 27,9%

Tukivektorikone	Onnistuneet	Virhekoodit	Havainnot	Ennustusvirhe
Onnistuneet	1620	546	2166	25,2%
Virhekoodilliset	222	541	763	29,1%
Ennusteet	1842	1087	2929	26,2%

Taulukko 5.3: Tukivektorikoneen ennustusvirhe onnistuneille havainnoille 25,2% ja virhekoodillisille havainnoille 29,1%, ja ennusteen kokonaisvirhe on 26,2%

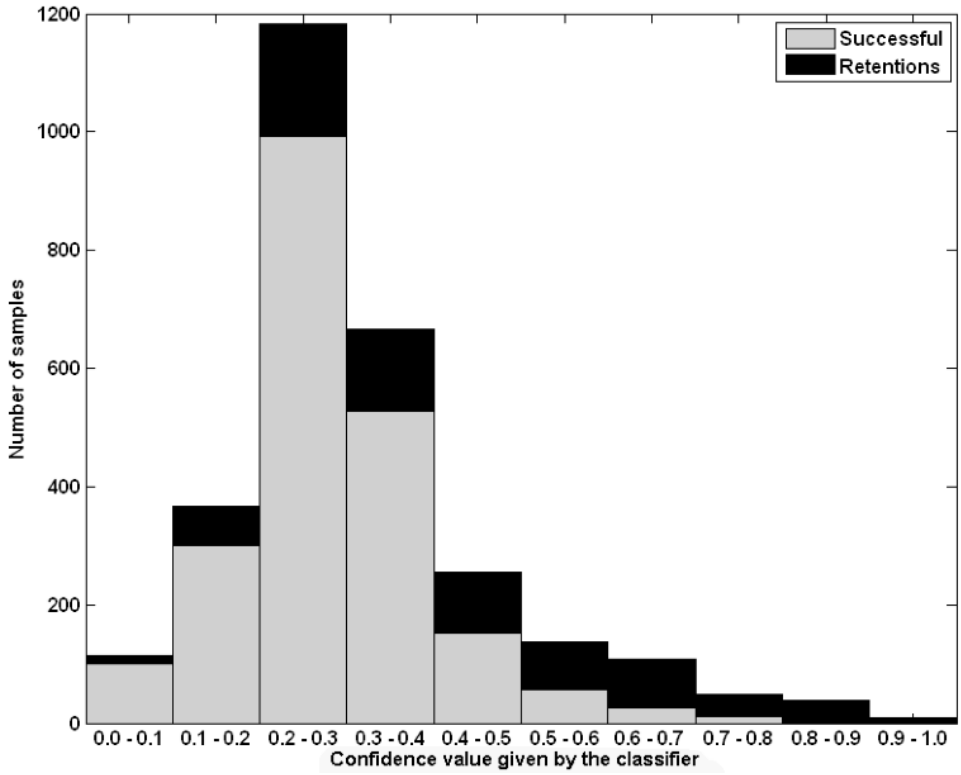
Luokittelun suhteellisen suuren kokonaisvirheen takia oli luovuttava binäärisestä päätösmallista (5.1) ja lähdettävä selvittämään virheiden jakautumista luokitteluvälille $y_i \in [0, 1]$. Luokittelijoiden ennusteet jaettiin aiemman päätösfunktion sijaan kymmeneen osaväliin, toisin sanoen $\{[0.0, 0.1[, [0.1, 0.2[, \ldots, [0.9, 1.0]\}$. Kuvissa 5.1 ja 5.2 näkyy neuroverkon (SCG)

ja tukivektorikoneen ennusteet luokiteltuna osaväleille. Kuvissa onnistuneet havainnot ovat harmaalla värillä ja virhekoodilliset mustalla, joten päätöstason arvolla 0.5 harmaiden näytteiden tulisi olla välillä $[0, 0.5]$ ja mustien välillä $]0.5, 1]$ tai muutoin ne ovat luokiteltu väärin. Kyseisellä analysointitekniikalla saatiin luottamustasot ennusteiden osaväleille, joista voidaan sanoa esimerkiksi neuroverkon tapauksessa, että luottamustasolle $]0.9, 1]$ osuva ennuste on 96% todennäköisyydellä virhekoodillinen havainto.



Kuva 5.1: Neuroverkon ennusteet kullakin luottamustasolla, missä värikoodeja vastaavat havainnot *successful* ja *retentions* tarkoittavat vastaavasti onnistunutta ja virhekoodillista havaintoa. *Number of samples* on havaintojen lukumäärä.

Aineiston kannalta suurin ongelma-alue on havainnot päätöstason 0.5 ympäristössä, jossa ennusteiden virheprosentti ylittää 25%. Neuroverkon tapauksessa tämä osaväli on leveämpi $[0.4, 0.7]$, kun tukivektorikoneelle se on $[0.4, 0.6]$. Kun edellä mainitut ongelma-alueille osuvat havainnot poistettiin luokittelusta eli vajaa neljännes aineistosta, niin tällöin luokittelun kokonaisvirheprosentti oli vain 10%.



Kuva 5.2: Tukivektorikoneen ennusteet kullakin luottamustasolla, missä värikoodeja vastaavat havainnot *successful* ja *retentions* tarkoittavat vastaavasti onnistunutta ja virhekoodillista havaintoa. *Number of samples* on havaintojen lukumäärä.

Luku 6

Pohdinta

Tutkimus toteutettiin Ruukin terästehtaalle, koska virhekoodillisten nauhojen tuottamat kustannukset ovat taloudellisesti merkittäviä. Adaptiivisen luokittelijan implementointi tuotantolinjastoon voisi olla hyvä tapa ehkäistä virhekoodillisten nauhojen tuotantoa, mutta luokittelun suuren epävarmuuden takia se ei ole taloudellisesti kannattavaa. On mahdollista, että tuotantolinjaston lukuisten mittalaitteiden ja mallinnuksen tuottaman tiedon lisäksi oleellista informaatiota puuttuu. Puutteellinen informaatio voitaisiin mahdollisesti tuottaa lisäämällä jokin mittalaite linjastoon. Toisaalta suuri osa informaatiosta, jota luokitteluun käytettiin, perustui valssausprosessin mallinnukseen hyödynnettyihin fysikaalisiin malleihin. Nämä fysikaaliset mallit voivat sisältää estimointivirheitä, joten mallinnuksen parantaminen voisi vähentää luokittelun virhettä.

Heikoista ennustustarkkuudesta huolimatta tutkimuksessa on pystytty osoittamaan, että luokittelijoiden ennustuskyky aineistolle on lähellä toisiaan, vaikka ennustusvirheiden jakaumat ovat erilaisia. Nämä erilaisuudet selittyvät luokittelijoiden toimintaperiaatteen pohjalta ja käytettyjen neuroverkkoalgoritmien erilaisuudella. Jakaumien tarkemman tarkastelun pohjalta olisi voinut olla mahdollista löytää homogeenisiä ryhmiä, joiden luokitteluvirhe olisi ollut pientä verrattuna saavutettuihin tuloksiin. Tietylle luottamustasolle sijoittuneiden ennusteiden klusterointi voisi olla yksi menetelmä, jolla homogeenisiä ryhmiä oltaisi saavutettu.

Kokonaan toisenlainen lähestymistapa olisi hyödyntää ajantasaista prosessiparametrien optimointia yhdistettynä luokittelualgoritmiin. Optimointi on mahdollista toteuttaa esimerkiksi käyttäen geneettisiä algoritmeja [20]. Optimoinnin jälkeen luokittelija voisi ennustaa valssauksen onnistumista, jolloin tuotantolinjaston parametrejä voitaisiin edelleen säätää onnistumisen takaamiseksi.

Luku 7

Sanasto

Dimensioiden kirous Curse of dimensionality

Epookki Epoch

Gaussinen ydinfunktio Gaussian radial basis function kernel

Hahmontunnistus Pattern recognition

Itsestään järjestäytyvät kartat Self-organizing maps

Konjugaattigradientti Conjugate gradient

Kynnysarvo Threshold

Laskevan gradientin menetelmä Gradient descent, Steepest descent method

Monikerrosperseptroni Multilayer perceptron

Neuroverkko Neural network

Opetusnopeusparametri Learning rate parameter

Piilokerros Hidden layer

Piirre Feature

Pyörähdyssymmetrinenfunktio Radial basis function

Päätösfunktio Decision function

Rakenteellinen riskinminimointiperiaate Structural risk minimization principle

Ristiinvalidointi Cross-validation

Satulapiste Saddle point

Skaalattu konjugaattigradientti Scaled conjugate gradient

Tukivektorikone Support vector machine

Valvottu oppiminen Supervised learning

Valvomaton oppiminen Unsupervised learning

Ydin Kernel

Ylioppiminen Overfitting

Kirjallisuutta

[1] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1998.

[2] Stephen Grossberg. *Neural networks and natural intelligence*. MIT, Cambridge, MA, 1988.

[3] Mark H. Beale Martin T. Hagan, Howard B. Demuth. *Neural Network Design*. MA: PWS Publishing, 1996.

[4] J. M. Mendel and R. W. McLaren. *A Prelude to Neural Networks*. Prentice Hall Press, Upper Saddle River, NJ, USA, 1994.

[5] Igor V. Tetko, David J. Livingstone, and Alexander I. Luik. Neural network studies. 1. comparison of overfitting and overtraining. *Journal of Chemical Information and Computer Sciences*, 35(5):826–833, 1995.

[6] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143. Morgan Kaufmann Publishers Inc., 1995.

[7] Debashis Nandi Haradhan Chel, Aurpan Majumder. Scaled conjugate gradient algorithm in neural network based approach for handwritten text recognition. In *Communications in Computer and Information Science Vol. 204*, pages 196–210. Springer, 2011.

[8] Martin F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993.

[9] Magnus R. Hestenes. Conjugate direction methods in optimization. In *Optimization Techniques Part 1*, volume 6 of *Lecture Notes in Control and Information Sciences*, pages 8–27. Springer Berlin Heidelberg, 1978.

[10] Bernhard Schölkopf, Kah Kay Sung, Christopher J. C. Burges, Federico Girosi, Partha Niyogi, Tomaso Poggio, and Vladimir Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE Transactions on Signal Processing*, 45(11):2758–2765, 1997.

[11] Vladimir Vapnik. *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[12] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[13] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[14] R. Bellman. *Dynamic programming*. Princeton University Press, Princeton, NJ, 1957.

[15] Wenqing Jiang and Antonio Ortega. Forward/backward adaptive context selection with applications to motion vector field encoding. In *ICIP 2*, pages 168–171, 1997.

[16] J. Elsilä, U.; Röning. Defect prediction in hot strip rolling. In *Ironmaking & Steelmaking, Volume 31, Number 3*, pages 241–248, 2004.

[17] Teuvo Kohonen. *Self-organizing maps*. Springer, 2001.

[18] W. H. Chen and J. Y. Shih. Comparison of support-vector machines and back propagation neural networks in forecasting the six major asian stock markets. In *Int. J. Electronic Finance, Vol. 1, No. 1*, pages 49–66, 2006.

[19] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[20] Bogdan Filipic and Erkki Laitinen. Model-based tuning of process parameters for steady-state steel casting. *Informatica (Slovenia)*, 29(4):491–496, 2005.