Takaisinlevittävän neuroverkon ja lähimmän naapurin menetelmän vertailu yksinkertaisessa tekoälysovelluksessa

Arttu Tamminen

Tampereen yliopisto
Informaatiotieteiden yksikkö/tko
Pro gradu -tutkielma
Ohjaaja: Martti Juhola
19.12.2014

Tampereen yliopisto
Informaatiotieteiden yksikkö/tko
Arttu Tamminen: Takaisinlevittävän neuroverkon ja lähimmän naapurin menetelmän vertailu yksinkertaisessa tekoälysovelluksessa
Pro gradu -tutkielma, 53 sivua
Joulukuu 2014

Tiivistelmä

Tutkielma käsittää kahden erilaisen luokittelumenetelmän vertailun tekoälysovelluksessa. Nämä menetelmät ovat takaisinlevittävä neuroverkko ja lähimmän naapurin menetelmä. Tutkimuksen tarkoitus on havainnollistaa menetelmien kykyä oppia hallitsemaan neljän suorana tunnetun pelin mekaniikkaa. Tutkimuksessa kävi ilmi että menetelmät eivät ole kovinkaan soveltuva pelin mekaniikan oppimiseen, ainakaan kovin vähäisellä opetusdatalla.

Purpose of this thesis is to compare two different machine learning methods: backpropagation neural network and nearest neighbour. Also demonstration of these two methods is included in this thesis, both learning and utilizing. Domain of learning and utilization was simple game called connect four. As result it became clear that methods were not suitable for this game, or at least with insufficient amount of training data.

Sisällys

1 Johdanto

Tässä tutkielmassa tarkastellaan kahden eri menetelmän soveltuvuutta tekoälysovelluksessa. Tekoälysovellus on melko yksinkertainen neljän suoraksi kutsuttu kahden pelaajan peli, jossa tarkoituksena on saavutta tilanne, jossa omat neljä pelimerkiä muodostavat jonon. Tutkielman tarkoitus on selvittää neuroverkon ja lähimmän naapuri -menetelmän kyvyt suoritua pelin strategian oppimisesta.

Tutkielma keskittyy neuroverkon, tarkemmin sanottuna takaisinpäinlevittävän neuroverkon toiminnan kuvaamiseen ja sen hyödyntämiseen neljän suora -pelissä. Lopullinen tarkoitus on kyetä luomaan neuroverkko, joka kykenee suoriutumaan kohtuullisesti vastapelaajan roolissa ihmispelaajaa vastaan. Luonnollisesti kuitenkin neuroverkon opetus- ja testausvaiheista saatava data on myös erittäin oleellinen osa tutkielmaa. Lähimmän naapurin -menetelmän tarkoitus tutkielmassa on toimia vertailukohteena neuroverkolle ja siksi sen käsittely ei ole aivan niin perustavanlaatuinen kuin neuroverkon.

Tutkielman lopullinen tarkoitus on tutkimustulosten valossa päätellä miten data on syytä esittää ja minkälaista sen tulisi olla, jotta sen avulla voidaan luoda mahdollisimman älykäs keinotekoinen pelaaja. Datan keräämisessa käytetyn algoritmin yksinkertaisuus kuitenkin rajoittaa tämän tavoitteen erinomaista saavuttamista. Lisäksi tutkielma kattaa vertailun neuroverkkototeuksen ja lähimmän naapurin menetelmän älykkyydessä. Tämä pyritään saavuttamaan asettamalla molemmat menetelmät pelaamaan vastakkain useita kertoja.

Luvussa 2 esittelen tutkielmassa sovellusalueena olevan neljän suora -pelin. Pelin säännöt ja mekaniikka esitellään yksinkertaisen strategia-esityksen kera. Luvussa 4 neuroverkko ja sen omaisuudet, kuten opetus- ja testausvaiheet esitellään perinpohjaisesti. Neuroverkon rakennusosa, neuroni, esitellään muutamissa erilaisissa muodoissa, kuitenkin keskittyen takaisinlevitettävässä neuroverkossa käytettyyn tietyn tyyppisen neuroniin. Myös lyhyt katsaus neuroverkkojen historiaan ja sovelluksiin tarjoillaan tässä kappaleessa. Luvussa 5 esitellään tutkielmassa käytettävän takaisinlevittävän neuroverkon opetus- ja testausvaiheet. Luvussa 6 esitellään lähimmän naapurin -menetelmä ja sen ominaisuudet. Kuten neuroverkkokappaleessa, opetus- ja testausvaiheet esitellään, kuten myös historia ja sovellukset. Luvussa 7 käsitellään menetelmien käyttämän datan kerääminen ja esiprosessointi. Datan keräämisessä käytetty yksinkertainen algoritmin toiminta esitellään niin sanallisesti kuin pseudokoodin muodossa. Luvussa 9 esitellään hyvin lyhyesti tekoälysovelluksessa käytetyt suunnittelumenetelmät, työkalut ja tekniikat. Koska kappaleen aihealue on melko etäinen tutkielman varsinaiseen aiheeseen nähden

käsitellään asiat hyvin suppeasti, niin tarkasti kuin se on sovelluksen ymmärtämisen kannalta tarpeellista. Luvussa 10 analysoidaan molemmista menetelmistä saadut tutkimustulokset ja pohditaan mahdollisia suunnitteluvirheitä ja parannusehdotuksia.

2 Neljän suora

Neljän suora -peli julkaistiin vuonna 1974 Milton Bradley -nimisen peleihin erikoistuneen yrityksen toimesta. Pelin säännöt ovat hyvin yksinkertaiset ja sen vuoksi se saattaa tuntua ensikertalaiselta hyvinkin helpolta. Pelaajien on tarkoitus saattaa neljä omaa pelimerkkiään yhtenäiseksi jonoksi. Jono voi olla joko pysty-, vaaka- tai diagonaalisuunnassa. Pelilauta koostuu useimmissa tapauksissa seitsemän sarakkeen ja kuuden rivin muodostamasta matriisista, johon pelaajat asettavat omia pelimerkkejään. Pelimerkkien asettaminen ei kuitenkaan ole täysin suoraviivaista, sillä pelilauta on asetettu pystysuoraan ja näin ollen pelimerkkejä voidaan asettaa laudalla vain laudan yläosasta. Edellisen järjestelyn ansiosta pelimerkit putoavat pelilautaa pitkin, kunnes törmäävät joko laudan alareunaan tai sarakkeessa jo olevaan pelimerkkiin. Pelaajat asettavat pelimerkkejä edellä kuvatulla tavalla pelilaudalle vuorotellen, kunnes jompikumpi pelaajista kykenee muodostamaan pelimerkeistään neljän oman pelimerkin käsittävän jonon tai vaihtoehtoisesti pelilauta täyttyy ja edellinen ehto ei tule voimaan.

2.1 Strategia

Yksinkertaisin strategia neljän suorassa on tavoitella vain ja ainoastaan omaa voittoa. Tämä on yksinkertaisin tapa ja tätä tapaa käyttämäni aineisto noudattaa. Pelaamisen tekee kuitenkin huomattavasti mielekkäämmäksi, mikäli pelaajat yrittävät oman voiton tavoittelun ohella estää toisen pelaajan voiton tavoittelu. Luonnollisesti on myös mahdollista pyrkiä ainoastaan estämään toisen pelaajan voiton tavoittelu, mutta tämä menetelmä ei useinkaan ole kovinkaan mielekäs. Aineistoani varten valitsin ensin mainitun menetelmän johtuen sen yksinkertaisuudesta ja näin ollen voin tutkimuksessani keskittyä neuroverkon ja lähimmän naapurin -menetelmän oppimiseen ja tulosten analysointiin. Edellä kuvattujen stategioiden perusteella voidaan helposti nähdä, ettei neljän suora ole kovinkaan monimutkainen tai kehittynyt peli, kuten esimerkiksi shakki tai go. Tästä seikasta huolimatta varsinainen pelaaminen on kohtalaisen viihdyttävää, mutta samalla tarkkaavaisuutta vaativaa, etenkin mikäli pelaaja käyttää kaikkia kolmea strategiata yhtäaikaa. Kolmen eri strategian yhtäaikanen hallinta vaatii hyvää huomiointikykyä ja kärsivällisyyttä.

Vaikkakin neljän suora käsittää kolme erilaista strategiaa pelin voittaminen riippuu myös ulkoisista tekijöistä. Näistä merkittävin on ihmispelaajan huomiokyvyn herpaantuminen, jolloin helposti saattaa jäädä huomaamatta vastapelaajalle

edullinen tilanne. Ja näin ollen pelaaja ei huomaa vuorollaan tehdä liikettä joka estää vastapelaajan voittomahdollisuuden. Tämä seikka korostuu erityisesti pelikertojen lukumäärän ja peliajan kasvaessa. Tekoälylle nämä seikat eivät ole juurikaan harmittavia, sillä tietokone kykenee hahmottamaan ja hallitsemaan kaikki mahdolliset pelitilanteet. Tästä huolimatta tekoälysovellus voi luonnollisesti tehdä myös virheitä ja johtuen neuroverkon luonteesta se saattaa tietyissä olosuhteissa olla hyvinkin virhealtis. Tämä ominaisuus ei kuitenkaan välttämättä ole aina tulkittavissa huonoksi ominaisuudeksi. Virheitä tekevä tekoäly voi esimerkiksi neljän suoran kaltaisessa pelissä tuoda lisäarvoa pelikokemukseen, jolloin tekoälyvastustaja saattaa tuntua aavistuksen enemmän inhimilliseltä. Kuitenkin tämä riippuu hyvin paljon virheen laadusta. Tekoälyn tekemä virhe saattaa vaikuttaa ihmispelaajalle siltä kuin tekoäly ei tuntisi sääntöjä tai muutoin vain pelaisi täysin väärin tai sitten tekoäly tekee inhimillisen virheen.

3 Koneoppiminen

Hahmontunnistus on menetelmä jonka avulla voidaan olemassa olevasta aineistosta etsiä tai muodostaa haluttuja hahmoja tai säännönmukaisuuksia. Koska hahmontunnistus on vain yleinen nimitys menetelmälle joka kykenee etsimään tai muodostamaan hahmoja tai säännömukaisuuksia, ei täten myöskään aineiston laadulle tarvitse asettaa erityisiä vaatimuksia. Aineisto voi hyvinkin olla esimerkiksi kuvallista, tekstuaalista tai nuumeerista, jolloin kussakin tapauksessa etsitään erilaisia hahmoja. Menetelmiä on useita erilaisia ja tässä tutkielmassa perehdytään kahteen: takaisin levittävään neuroverkkoon ja k-lähimmän naapurin -menetelmään. Menetelmät esitellään tarkemmin tuonnempana. Hahmontunnistuksessa aineistosta pyritään etsimään tiettyjä muuttujia, joiden avulla voidaan tarkasti erotella erilaiset hahmot toisistaan. Tehtävä ei ole lainkaan triviaali, sillä aina ei ole yksikäsitteistä se minkä laatuiset muuttujat pystyvät löytämään aineistosta hahmoja tarkimmin. Joskus aineiston koko saattaa olla hyvinkin suuri, mutta sen informaatiosisältö voi olla pieni. Tämä voi johtua useista syistä, kuten häiriöstä aineistossa tai jopa täysin turhasta sisällöstä aineistossa.

Neuroverkon ehkäpä kiinnostavin ominaisuus on sen kyky ammentaa tietämystä ympäristöstään ja siten kohentaa suorituskykyään oppimisen avulla. Kohennus tapahtuu tietyssä ajassa ja suorituskyvyn kohentuminen voidaan todentaa ennalta määritellyn menetelmän avulla. Oppiminen on iteratiivinen prosessi, jossa syötteiden ja syötteisiin liittyvien tulosten mukaisesti neuroverkon neuronien painoarvokertoimia muunnellaan. Parhaassa tapauksessa neuroverkko tulee jokaisella opetuskerralla entistä enemmän tietoiseksi omasta ympäristöstään. Haykin[1994] määrittelee oppimisen neuroverkkojen tapauksessa seuraavasti: ”Oppiminen on jatkuva prosessi jossa neuroverkon vapaita parametreja mukautetaan ympäristön stimulaation mukaisesti.” Edellä mainittu määritelmä voidaan pilkkoa seuraavan sekvenssin mukaiseksi. Neuroverkko stimuloituu ympäristön ärsykkeistä, joiden vuoksi neuroverkon sisäinen rakenne muuntuu, josta seuraa, että neuroverkko kykenee vastaamaan edellä mainittuihin ärsykkeisiiin uudella tavalla. Neuroverkon tapauksessa sisäisen rakenteen muutos tarkoittaa neuronien syötteiden painoarvokertoimien muutos. Painoarvokertoimien muutoksesta vastaa oppimisalgoritmi. Neuroverkkojen tapauksessa oppimisalgoritmeja on useita erilaisia, jotka ovat toisistaan poikkeavia. Kaikille oppimisalgoritmeille on kuitenkin yhteistä se, että ne muokkaavat neuroverkon painoarvokertoimia. Kaikilla oppimisalgoritmeilla on heikkoutensa ja vahvuutensa. Esittelen seuraavaksi Haykinin [1994] määrittelemää oppimissääntöä.

1. *Perceptron-oppiminen.* Perceptron-oppimisessa neuroni vastaanottaa syötevektorin, joka koostuu yhden tai useamman edeltävän neuronin tulosteesta. Neuroni tuottaa yhden tuloksen vastaanottamastaan syötevektorista ja tätä tulosta verrataan ennalta määriteltyyn tulokseen. Vertailu tapahtuu vähentämällä tavoitetuloksesta neuronin tuottama tulos. Edellisestä erotuksesta käytetään nimeä *virhesignaali.* Virhesignaali käynnistää proseduurin, joka on sarja neuronin synaptisten painoarvokertoimien korjauksia. Korjausten avulla neuronin painoarvokertoimet muuttuvat siten, että ekvivalenssi neuronin tuottama tuloksen ja ennaltamääritellyn tuloksen välillä toteutuu ajan funktiona. Korjaustoimenpiteitä sisältäviä sarjoja suoritetaan, kunnes neuroni on saavuttanut vakaan tilan ja oppimisprosessi voidaan päättää. Virhesignaalin käynnistämän sarjan painoarvokertoimien korjaavia toimenpiteitä. Yksittäinen toimenpide voidaan kuvata seuraavalla tavalla: neuronin synaptisille yhteyksille lasketaan painoarvokertoimen muutos ottamalla virhesignaalin ja neuronin tuottaman tuloksen tulo, joka kerrotaan määritellyllä vakiolla. Tätä vakiota kutsutaan *oppimisnopeusparametriksi.* Oppimisnopeusparametri määrittelee, kuinka nopeasti neuronin synaptisten painoarvokertoimien tulee muuttua ajan funktiona. Oppimisnopeusparametrin valinta on syytä tehdä huolellisesti, sillä huonosti valittu parametri voi heikentää ja hidastaa neuronin oppimista. Edellisen laskutoimituksen toimesta syntynyt painoarvokertoimen muutos lisätään nyt kaikkiin neuroniin saapuviin synaptisiin painoarvokertoimiin. Näin oppimissääntö joko heikentää tai korostaa neuroniin tulevia synaptisia yhteyksiä. Tätä oppimisprosessia kutsutaan myös *delta-säännöksi* tai *Widrow-Hoff-säännöksi.* Tämä oppimisalgoritmia käytetään takaisinlevittävässä neuroverkossa, joka käsitellään tarkemmin myöhemmässä vaiheessa tutkielmaa.

2. *Muistipohjainen oppiminen.* Muistipohjaisessa oppimisessa opetettava tietämys tallennetaan tietovarastoon pareittain, joissa yksi syöte vastaa yhtä tulosta. Syötteet eli opetusdata tallentaan yksi- tai useampiulotteisena vektoreina muistiin. Testivektori on edellä mainituista syötteistä poikkeva ja siten ”ennen näkemätön”. Testivektorista, toisin kuin opetusvektoreista, tunnetaan vain syöte mutta ei tulosta. Näin ollen testivektorille määritellään tulos käyttäen apuna opetusvektoreita. Toisin sanoen algoritmi etsii ja analysoi opetusvektorien joukosta testivektoria lähimpänä olevat vektorit [Haykin, 1994]. Kaikille muistipohjaisille opetusalgoritmeille on yhteistä se, että ne määrittävät jonkin tavan määritellä opetus- ja testivektorin etäi-

syyden [Haykin, 1994]. Lähimmän naapurin menetelmä edustaa muistipohjaista oppimisalgoritmi ja esittelen sen tarkemmin myöhemmässä vaiheessa tutkielmaa.

3. *Hebbin sääntö.* Hebbin oppiminen on vanhin ja tunnetuin oppimissääntö. Se on nimetty 1940-luvulla vaikuttaneesta neuropsykologi Donald Hebbin mukaan. Hebb määritteli oppimisen tapahtuman aivojen neuroneissa siten, että mikäli synaptisen yhtyeyden molemmissa päissä olevat neuronit aktivoituvat samanaikaisesti, synaptinen yhteys voimistuu. Kuitenkin 1970-luvulla oppimissääntö laajennettiin siten, että mikäli yhteyksien päissä olevat neuronit eivät aktivoituvat samanaikaisesti, yhteys neuronien välillä heikentyy.

4. *Kilpaileva oppiminen.* Nimensä mukaisesti kilpailevassa oppimisessa tuloskerroksen neuronit kilpailevat keskenään, siitä mikä neuroni aktivoituu. Toisin kuin esimerkiksi Hebbin oppimisessa, kilpailevassa oppimisessa vain ja ainoastaan yksi tuloskerroksen neuroni voi aktivoitua. Haykin [1994] esittelee kolme piirrettä, jotka koskettavat kilpailevaa oppimista hyödyntäviä neuroverkkoja. Ensimmäinen piirre määrittelee, että tuloskerroksen neuronit ovat kaikki samanlaisia, mutta niillä on satunnaisesti painoarvokertoimet synaptisilla yhteyksillä. Tästä johtuen tuloskerroksen neuronit reagoivat eri tavalla syötteisiin. Toinen piirre määrittelee, että neuronien voimakkuudelle on säädetty raja. Kolmas piirre määrittelee, että neuronit kilpailevat oikeudesta aktivoitua tietyllä syötteellä. Kilpailun voittavaa neuronia kutsutaan *winner-takes-all*-neuroniksi. Kuten muissakin oppimismenetelmissä kilpailevassa oppimisessa ovat kaikki neuronit kerrostensa neuronit yhteydessä seuraavan kerroksen neuroneihin. Poikkeavaksi kilpailevan oppimisen tekee se, että sen tuloskerroksen neuronit ovat yhteydessä myös toisiinsa siten ja että kukin neuroni on yhteydessä viereisiinsä neuroneihin. Tällä yhteydellä neuronit voivat hilitä naapurineuronien aktivoitumista. Lopulta jokaiselle neuronille lasketaan lopullinen tulos, joka koostuu neuronin edeltäviltä kerroksilta saaduista syötteistä ja tuloskerroksen naapurineuronien palautteesta. Suurimman lopputuloksen saanut neuroni aktivoituu ja muut neuronit menettävät aktivoitumisen mahdollisuuden. Tämän jälkeen aktivoituneen neuronin tulevien syötteiden painoarvokertoimia korostetaan ja näin ollen neuroverkko oppii tunnistamaan syötteen.

5. *Boltzmann-oppiminen* Boltzman-oppiminen perustuu tilastotieteellisiin menetelmiin. Boltzmann-oppimisessa neuronit toimivat hieman samalla tavalla

kuin kilpailevassa oppimisessa, sillä niillä voi olla vain kaksi tilaa; päällä tai pois. Näitä tiloja kuvataan binäärisesti numeroarvoilla 1 ja -1. Boltzmann-oppimista suuresti luonnehtii energiafunktio [Haykin, 1994], joka määräytyy neuroverkon kaikkien neuronien tilan mukaan. Energiafunktio lasketaan kaikkien neuronien tilojen tulona, jonka jälkeen jossain vaiheessa oppimisprosessia satunnaisen neuronin tila muunnetaan päinvastaiseksi käyttämällä energiafunktioon perustuvaa todennäköisyyttä. Lopulta neuroverkko saavuttaa vakaan tilan. Boltzmann-oppimisessa neuroneita on kahdenlaisia: näkyviä ja piilotettuja, hieman samaan tapaan kuin edellä kuvatuissa oppimissäännöissä. Näkyvät neuronit ovat yhteydessä ympäristöön ja piilotetut neuronit eivät ole. Tämä johtaa siihen, että näkyvien neuronien tilat ovat sidottu ympäristön syötteisiin, mutta piilotetut neuronit voivat vaihtaa tilaansa vapaasti.

3.1 Ohjattu ja ohjaamaton oppiminen

Haykinin [1994] mukaan ohjattua oppimista voidaan luonnehtia seuraavien elementtien avulla: ympäristö, opettaja ja opetettava järjestelmä, joka tunnetaan tässä yhteydessä nimellä neuroverkko. Ympäristö on informaatiojoukko, josta neuroverkolle opettettava tietämys hankitaan ja jalostetaan sopivaan muotoon. Ympäristöstä saatava tietämys esitetään syöte-vastaus-pareina, joita neuroverkko käsittelee opettaja-elementin välityksellä. Opettaja on jokin määräämätön älykäs toimija, joka osaa yhdistää ympäristöstä saatavat syötteet virheettömästä ympäristöstä saataviin vastauksiin ja näin muodostaa syöte-vastaus-pareja, joita neuroverkko kykenee käyttämään oppimismateriaalinaan. Opettajan luonnetta ei voida äärimmäisen tarkasti määritellä, vaan se voi olla esimerkiksi ympäristöstä saatavaan informaatioon perehtynyt asiantuntija tai jopa toinen koneoppimisen avulla harjaantunut järjestelmä. Opettajan valitsemat syöte-vastaus-parit edustavat optimaalista ratkaisua [Haykin, 1994]. Neuroverkolle esitetään yksi kerrallaan opettajan valitsemat syöte-vastaus-parit siten, että neuroverkko yrittää itse suoritutua syötteen käsittelystä ilman tietämystä syötteen vastauaparista. Tämän jälkeen tarkastellaan opettajan valitsemaa vastausta ja neuroverkon antamaa vastausta ja tarkastelun perusteella muokataan neuroverkon vapaita parametreja. Näin neuroverkko pyritään muokkaamaan siten, että se kykenee sisäistämään opettajan tietämyksen, emuloimaan opettajaa ja toimimaan itsenäisesti ympäristöstä saatavien syötteiden käsittelyssä.

Ohjaamattomassa oppimisessa ei tunneta edellä kuvattua opettajaelementtiä lain-

kaan, vaan neuroverkko toimii itsenäisesti sisäistäessään ympäristöstä saatavan informaation. Toisin sanoen neuroverkolle ei ole tarjolla opettajan valitsemia syöte-vastaus-pareja. Haykin [1994] jakaa ohjaamattoman oppimisen kahteen suuntaukseen jotka, tunnetaan nimillä vahvistettu oppiminen ja itseohjautuva oppiminen. Käsitellään nämä kaksi suuntausta edellä mainitussa järjestyksessä. Jung [1997] määrittelee vahvistetun oppimisen olevan yrityksen ja erehdyksen kautta tapahtuva toiminto, jossa oppiva järjestelmä, joka tässä tapauksessa on neuroverkko, oppii suorittamaan täsmällisiä toimenpiteitä käyttämällä hyödyksi ympäristöstään saamaansa palautetta. Ympäristö ymmärretään informaation tietovarannoksi, kuten ohjatussa oppimisessakin. Vahvistetussa oppimisessa ympäristö ei kuitenkaan opettajan avulla tarjoa neuroverkolle syöte-vastaus-pareja, vain ja ainoastaan syötteitä ja palautteita. Haykin [1994] täsmentää edellä esiteltyä prosessia siten, että ympäristöstä saatava palaute koostuu lyhytaikaisista ärsykkeiden sekvensseistä, jotka muotoutuvat heuristisiksi signaaleiksi, jotka neuroverkko kykenee hyödyntämään oppimisprosessissa. Itseohjautuvassa oppiminen on samankaltainen prosessi kuin edellä mainittu vahvistettu oppiminen, mutta heuristisien signaalien muodostuminen ei ole läsnä. Heuristisien signaalien puuttuminen on korvattu siten, että neuroverkon vapaat parametrit muunnetaan ympäristön tuottamien syötteiden mukaiseksi. Näin ollen neuroverkko optimoituu ympäristön informaatiolle ja se muodostamaan sisäisiä, esityksiä joiden avulla se kykenee poimimaan piirteitä syötteistä ja näin ollen suorittamaan sille määritellen tehtävän [Haykin, 1994].

4 Neuroverkot

Tässä luvussa selvitän hieman neuroverkkojen kehityshistoriaa ja selvitän neuroverkkojen yleistä rakennetta, niin tarkasti kuin se tutkielman kannalta on tarpeellista. Käytännössä tämä tarkoittaa monikerroksisen takaisinpäinlevittävän perceptron -neuroverkon toiminnan ja rakenteen kuvausta. Neuroverkko on pohjimmiltaan eräänlainen luokittelumenetelmä, jota voidaan helposti hyödyntää tekoälysovelluksissa sen ominaisuuksien ansioista. Näistä ominaisuuksista olennaisin lienee sen kyky oppia luokittelemaan ennalta tuntemantonta dataa tunnetun datan avulla. Toisin sanoen neuroverkko voidaan opettaa tunnetulla datajoukolla tunnistamaan tiettyjä luokitteluja datajoukosta, jonka jälkeen neuroverkko voi saada syötteenä tuntematoman datajoukon ja luokitella sen tuntemansa joukon avulla.

4.1 Historia

Alun perin neuroverkot luotiin matemaattisiksi malleiksi, joiden tehtävänä oli simuloida aivojen toimintaa. [Järvelin, 2008] Ensimmäiseksi neuroniksi voidaan luonnehtia Warren McCullochin ja Walter Pittsin 1940-luvulla kehittämää neuronia, joka on nimetty kehitäjiensä mukaan. McCulloch oli ammatiltaan psykiatri ja neuroanatomi, jonka asetti tavoitteekseen esittää ja mallintaa hermojärjestelmän toimintaa. Kun taas Pitts oli varsin lahjakas matemaatikko ja yhdistäessään kykynsä ja tietämyksensä he onnistuivat luomaan McCullochin tavoitteleman mallinnuksen hermojärjestelmästä [Haykin, 1994]. McCulloch-Pitts-neuronin syötteiden lukumäärää ei ole rajattu, mutta se kykenee antamaan tuloksena vain lineaarisesti erottuvat kaksi hahmoa. Tästä johtuen McCulloch-Pitts-neuroneja yhdistelemällä voidaan luoda verkko, joka kykenee mallintamaan loogisia funktiota. Edellä mainittua verkkoa kutsutaan nimellä perceptron-malli, joka mahdollistaa useamman lineaarisesti erottuvan hahmon tunnistamisen. McCulloch-Pitts-neuronit ja sitä myöten niistä koostuvat verkot ovat kuitenkin rajoittuneita johtuen niiden binäärisestä luonteesta ja opetusmenetelmän puutteista[Fausett, 1994]. Puutteistaan ja rajoituksistaan huolimatta McCulloch-Pitts-neuroniesitys toimi esikuvana ja innoittajana tuleville neuroverkkojen sukupolville.

Vuonna 1949 Donald Olding Hebb julkaisi kirjansa ”The Organization of Behavior”, jossa kirjoittaja kuvaa, kuinka fysiologisen oppimisen seurauksena aivojen neuronien yhteydet muodostuvat ja jalostuvat. Hebb kuvaa teoksessaan myös neuronien välisien synapsien voimistumisen tapahtuvan neuronien toistuvan ak-

tivoinnin yhteydessä. Teoksen merkittävyys ilmestyessään oli neuroverkkosovellusten kannalta kuitenkin harmittavan pieni, sillä neuroverkkoarkkitehdit jättivät teoksen esittelemät havainnot huomiotta [Haykin, 1994]. Kuitenkaan Hebbin teos ei jäänyt täysin huomiotta, sillä vuonna 1956 kirjoittivat Rocherster, Holland, Haibt ja Duda tekstin "Test on a cell assembly theory of the action of the brain, using a large digital computer". Tekstissä käsiteltiin digitaalisen tietokoneiden käyttöönottoa neuroverkkosovellusten luomisessa ja suorittamisessa, käyttäen Hebbin esittämää oppimisprosessia, jossa synapsit voimistuvat. Teksti kuitenkin antaa ymmärtää, etteivät tulokset olleet kovinkaan onnistuneita, mutta myöhemmin samana vuonna Uttley täydensi tutkimusta lisäämällä synapsit, joiden voimakkuutta voitiin muokata. Uttleyn tutkimus paljasti, että muuttuvien synapsien ansiosta on todennäköisesti mahdollista saavuttaa neuroverkko joka pystyy luokittelemaan yksinkertaisia binäärijoukkoja määriteltyihin luokkiin [Haykin, 1994].

15 vuotta McCullochin ja Pitts julkaiseman jälkeen Rosenblatt julkaisi uudentyyppisen lähestymistavan hahmontunnistusongelmaan julkaisussaan "Perceptron". Julkaisussa Rosenblatt esittelee yksikerroksisia neuroverkkoja, joissa neuroneilla on kiinteät painoarvot, mutta neuronien välisillä yhteyksillä (synapseilla) painoarvot ovat muokattavissa. Neuronien väliset painoarvot suppenevat iteratiivisessa opetuksessa kohti arvoja, jotka mahdollistavat käsillä olevan ongelman ratkaisun. Tämä menetelmä todettiin tehokkaammaksi kuin Hebbin esittelemä tapa. Rosenblattin ansiosta kiinnostus ja innostus neuroverkkoja kohtaan kasvoi [Fausett, 1994]. Tästä huolimatta vuonna 1969 Minsky ja Papert kirjoittivat kirjan nimeltä "Perceptrons" jossa he elenganteilla matemaattisilla menetelmillä todistivat, että on olemassa perustavanlaatuisia rajoituksia jotka määrittelevät, kuinka monimutkaisista toimenpiteistä yksikerroksiset neuroverkot voivat suoriutua [Haykin, 1994].

1960-luvulla tunnettiin myös toinen opetusmenetelmä, joka oli hieman erilainen kuin perceptronin opetusmenetelmä. 1960 Widrow ja Hoff esittelivät opetusalgoritmin, joka tunnetaan nimellä pienimmän neliösumman menetelmä. Algoritmi poikkeaa perceptronin opetusalgoritmista siten, että se säätelee neuronien välisiä painoarvoja siten, että neuroverkon tuottaman tuloksen ja halutun tuloksen erotus suppenee. Perceptronin opetusalgoritmi taas säätelee painoarvoja, kun neuroverkon antama tulos ei ole haluttu. Myöhemmät ja tehokkaammat monikerroksisien ja takaisinlevittävien neuroverkkojen opetusalgoritmit perustuvat tähän opetusalgoritmiin [Fausett, 1994].

Minskyn ja muiden tutkijoiden löytämien puutteiden ja rajoitusten vuoksi 1970-

lukua voidaan luonnehtia melko hiljaisiksi vuosiksi neuroverkkojen kehityksen kannalta [Fausett, 1994]. Tästä huolimatta kehitys kuitenkin eteni. 1970-luvun tärkeimpänä saavutuksena voidaan pitää itseorganisoituvia karttoja. Itseorganisoituvat kartat ovat ohjaamattomaan oppimiseen perustuvia neuroverkkomalleja [Järvelin, 2008]. Teuvo Kohosta voidaan pitää merkittävänä tutkijana ja kehittäjänä itseorganisoituvien karttojen saralla. Kohosen karttoja on käytetty muun muassa puheentunnistuksessa ja hahmontunnistuksessa.

4.2 Biologinen lähtökohta

Neuroverkkojen lähtökohtana voidaan pitää biologisia aivosoluista koostuvia verkostoja, jotka paremmin tunnetaan nimellä aivot. Neuronit ja aivosolut jakavat monia samoja ominaisuuksia ja näin ollen neuronia voidaan pitää eräänlaisena simulaationa aivosolusta. Molemmat vastaanottavat syötteitä, prosessoivat syötteet ja prosessoinnin tuloksena välittävät tuloksen eteenpäin. Aivot ovat monimutkainen, ei-lineaarinen ja rinnakkainen informaation käsittely-yksikkö [Haykin, 1994]. Aivot kykenevät organisoimaan neuroneita suorittamaan haluttujan toimintoja, kuten esimerkiksi lihasten toiminta tai hahmontunnitus. Aivoissa arvellaan olevan noin 10 miljardia neuronia ja niiden välillä 60 biljoonaa yhteyttä [Haykin, 1994]]. Aivoissa nämä neuronit koostuvat kolmesta osasta: tuojahaarakeet, soma ja viejähaarake. Kuten nimikin jo antaa ymmärtää, tuojahaarakeet tuovat informaatiota somaan muista neuroneista ja viejähaarake välittää neuronissa prosessoitua informaatiota eteenpäin muihin neuroneihin. Haarakkeissa välitttyvä informaatio on sähköisiä impulsseja, jotka välittyvät kemiallisesti synapsien välityksellä seuraaviin viejähaarakkeisiin [Haykin, 1994]. Soma vastaanottaa tuojahaarakkeiden välittämiä signaaleja ja laskee yhteen saamansa signaalit. Yhteenlaskun menetelmää ei ole vielä täysin kyetty selvittämään. Mikäli tämä yhteenlaskettu arvo ylittää tietyn raja-arvon, neuroni aktivoituu ja viejähaarake välittää signaalin eteenpäin. Edellä mainittu raja-arvo tunnetaan myös nimellä soman lepojännite ja sen määräytyminen ei ole vielä täysin selvitetty [Järvelin, 2008].
Haykinin [1994] mukaan ihmisen hermojärjestelmä voidaan esittää kolmitasoisena. Hermojärjestelmän keskimmäisenä osana on aivot, joita voidaan luonnehtia neuroverkkona. Tämä neuroverkko vastaanottaa jatkuvasti informaatiota esityksen kahdelta muulta järjestelmän osalta. Nämä kaksi muuta osaa ovat reseptorit ja efektorit. Neuroverkko käsittelee ja välittää saamansa informaatiota reseptorien ja efektorien välissä. Reseptorit vastaanottavat ärsykkeitä ihmisen kehosta tai ulkomaailmasta ja muuntaa ne sähköisiksi impulsseiksi ja välittää ne neu-

roverkolle. Efektorit taas vastaanottavat sähköisiä impulsseja neuroverkolta ja muuntavat ne vasteiksi. Neuroverkko ei kuitenkaan ole yksi yhtenäinen osa, vaan pikemminkin kokoelma osakokonaisuuksia jotka ovat erikoistuneet suorittamaan tiettyjä toimintoja, kuten näkö- tai kuulotoiminnot [Järvelin, 2008].
Merkittävin ero neuroverkkojen ja aivojen toiminnassa on suorituskyky. Edes nykyaikaiset tietokoneet eivät pysty kovinkaan sujuvasti käsittelemään useita asioita samanaikaisesti, jolloin neuroverkkojen tehokkuus jää huomattavasti jälkeen aivojen toiminasta. Aivot kykenevät suorittamaan laskutoimituksia rinnakkain, kun taas tietokoneet eivät kykene suoriutumaan tästä kovinkaan hyvin. Toisaalta taas tietokonetoteutuksen etuna voidaan, että se pitää suorittaa yksittäisiä laskutoimituksia hyvin lyhyessä ajassa, kun taas aivosolut ovat kohtalaisen hitaita suorittamaan laskutoimituksia. Tästä huolimatta on hyvin epätodennäköistä, että neuroverkko kykenee lähitulevaisuudessa ylittämään aivojen toimintakapasiteettia. Ehkäpä kuitenkaan neuroverkkojen ei ole edes tarkoitus mallintaa aivojen toimintaa kokonaisuudessaan, vaan toimia yksittäisten osaprosessien suorittajina. Kuten jo sanottua, aivot ja neuroverkot koostuvat useista rinnakkain toimivista aivosoluista ja neuroneista. Fausettin[1994] mukaan tämä mahdollistaa kohtalaisen hyvän vikasietoisuuden, jolloin muutamien neuronien tai aivosolujen tuhoutuminen ei merkittävästi vaikuta verkon tai aivojen toimintaan. Vikasietoisuus Fausettin [1994] mukaan tämä voidaan ymmärtää myös verkon ja aivojen kykynä ymmärtää ennalta tunnetuista poikkeavia syötteitä. Vaikka neuroverkkojen lähtökohtana voidaan pitää aivojen biologista mallia, ovat monet neuroverkkototeutukset erkaantuneet alkuperäisistä lähtökohdista. Tämä johtaa siihen, että on järkevämpää tarkastella neuroverkkoja matemaattisina malleina [Järvelin, 2008]. Tästä huolimatta Haykin [1994] esittelee yhdeksän merkittävää syytä neuroverkkojen käyttöön ja soveltamiseen.

1. *Epälinaariuus.* Johtuen neuronin epälineaarisesta luonteesta, myös toisiinsa liitetyistä neuroneista koostuva neuroverkko on epälineaarinen. Epälineaarisuus on hyvin tärkeä ominaisuus, erityisesti jos neuroverkon vastaanottamat syötteet ovat epälineaarisia.

2. *Syöte-tulos-kuvaus.* Ohjatussa opetuksessa opetusmateriaalin sisältämä informaatio tallennetaan neuroverkon painoarvokertoimiin. Jokainen opetusmateriaalin sisältämä yksittäinen ja uniikki osajoukko sisältää sisältää tuloksen, jonka neuroverkon halutaan antavan tulokseksi. Opetusmateriaalia syötettäessä neuroverkko osaa muunnella painoarvokertoimiaan siten, että verkon antama tulos poikkeaa mahdollisimman vähän opetusmateriaalin

osajoukossa määriteltystä tuloksesta. Tätä jatketaan, kunnes neuroverkko saavuttaa vakaan tilan, jonka mittaamiseen on aiemmin määritelty tietyt kriteerit. Näin neuroverkko kykenee luomaan kuvauksen syöteellensä ja antamallensa tulokselle.

3. *Mukautuvuus.* Neuroverkoilla on luontainen kyky mukauttaa painoarvokertoimiaan syötteiden muuttuessa. Erityisesti tämä ilmenee siinä, että neuroverkko on melko vaivattomasti uudelleenkoulutettavissa huomioimaan vähänpätöiset muutokset syötetteissään. On mahdollista myös suunnitella neuroverkkototeutus joka kykenee mukautumaan vaihteleviin tilanteisiin ilman valvottua opetusta.

4. *Ilmeinen vastaus* Hahmontunnistuksen yhteydessä neuroverkko voidaan valjastaa luokittelun lisäksi myös tarjoamaan lisäinformaatiota esimerkiksi vastauksen todennäköisyyden muodossa. Tällöin epävarmoissa tilanteissa neuroverkko voi ilmaista epävarmuutensa ja näin ollen neuroverkon antamaan vastaukseen voidaan suhtautua tietyllä varauksella. Näin neuroverkon suorituksen laatua voidaan kohentaa.

5. *Kontekstuaalinen informaatio.* Tietämys esitetään neuroverkoissa pelkästään verkon rakenteen ja sen painoarvokertoimien avulla. Jokainen neuroni on jollain tavalla vaikutuksessa muiden neuronien toimiin.

6. *Vikasietoisuus.* Neuroverkossa tietämys on siroteltu koko verkon alueelle. Tästä syystä neuroverkko on kohtuullisen vikasietoinen siinä mielessä, että mikäli neuroni tai sen yhteydet muihin neuronoihin häviävät, ei lopputulema ole välttämättä kovinkaan katastrofaalinen. Edellä kuvatussa tilanteessa neuroverkon kyky tunnistaa tai luokitella hieman heikenee, eikä suinkaan lamaannu kokonaan.

7. *Soveltuvuus usealle suorittimelle.* Nykyaikaiset tietokonelaitteistot tukevat jossain määrin rinnakkaisia prosesseja, ja luonnollisesti kasvattamalla suoritinyksiköiden määrää rinnakkaisien prosessien määrää voidaan ennestään kasvattaa. Tämä teknologia soveltuu neuroverkoille varsin mainiosti, mikäli useita neuroverkkoja halutaan aivojen tapaan yhdistää suuremmiksi kokonaisuuksiksi.

8. *Yhdenmukainen analyysi ja suunnittelu.* Erilaiset neuroverkkosovellukset jakavat melko paljon samoja piirteitä ja näin ollen niiden suunnitelua ja toteutuksia voidaan käyttää hyväksi myös täysin erilaisissa neuroverkkosovelluksissa.

9. *Analogia neurobiologiaan.* Neuroverkot muistuttavat rakenteensa ja toimintansa puolesta aivoja, jotka taas on havaittu toimivan nopeasti ja melko luotettavasti. Kun tämä analogia neurobiologiaan säilytetään, voidaan olla varmoja että neuroverkot tulevat jonain päivänä olevaan varsin menestyvä ja kehittyvä tieteenhaara.

4.3 Käyttökohteet

Neuroverkkojen monimuotoisuuden vuoksi niiden käyttökohteet ovat käytännössä rajattomat. Erityisesti saatavilla olevien syötteiden ollessa hyvin monimuotoista neuroverkot voivat osoittautua hyvin käyttökelpoisiksi menetelmiksi. Fausett [1994] esittelee muutamia mahdollisia käyttökohdetta neuroverkoille. Esimerkiksi signaalin käsittelyssä neuroverkko voidaan opettaa suodattamaan kohinaa tai konenäköjärjestelmässä voidaan neuroverkko opettaa tunnistamaan tiettyjä muotoja ja hahmoja. Lääketieteessä neuroverkkoja on opetettu tunnistamaan sairauksia tai vaivoja oireiden perusteella. Neuroverkko on oppinut Sejnowskin ja Rosenbergin [1986] vuonna 1986 tekemässä projektissa puhumaan. Projekti tunnetaan nimellä NETtalk, jossa neuroverkolle opetettiin tietty sanasto ja sanaston sanoihin liittyvät ääntämysohjeet. Näin oli mahdollista muodostaa sanastoa käyttämällä kokoelma virkkeitä, jotka neuroverkko kykeni lukemaan ääneen puheensyntetisaattorin avulla. Tekoälyn tutkimusta tämä ei kuitenkaan edusta, sillä verkkototeutus vain oppi tuottamaan kohtalaisen oikeankuuloista ääntelyä tietyille merkkijonoille. Toisin sanoen neuroverkko ei oppinut mitään sanojen ja termien semantiikasta. Luonnollisesti esimerkiksi osakekauppoihin tai muihin talouteen liittyviin ongelmiin neuroverkkoja voidaan soveltaa. Edellä mainitut asiat perustuvat joillakin osin tulevien tapahtumien ennustamiseen nykyisen tilanteen valossa. Tästä seikasta johtuen neuroverkko soveltuu hyvin tämänkaltaisiin tehtäviin.

Neuroverkkojen hyödyntäminen videopelien tekoälyteknologiana ei ole saavuttanut kovinkaan suurta suosiota [Charles and McGlinchy, 2004], mutta muutamia esimerkkejä kuitenkin on olemassa. Nämä pelit ovat useissa tapauksissa hidastempoisia ja vuoropohjaisia pelejä, kuten Mastermind, Othello ja Backgammon.

Edellä kuvatut kriteerit täyttää myös go-lauta peli, josta Thompsonin [2005] tekemässä tutkimuksessa pyrittiin luomaan neuroverkkoa käyttävä tekoäly go-lautapeliin. Tämän tutkielman pyrkii jossain määrin noudattelemaan Thompsonin tutkielman rakennetta, kuitenkin eri aihealueessa ja pyrkimyksenä on täydentää Thompsonin tutkimustuloksia lisäämällä vertailumenetelmäksi lähimmän naapurin -menetelmän. Ehkäpä merkittävin syy siihen, että neuroverkot eivät ole kovinkaan suosittu menetelmä on, että nykyaikaisten videopelien tekoälyratkaisu saattaa piillä niiden opetusvaiheessa. Opetusvaihe saattaa viedä huomattavan kauan aikaa ja opetusdatan hankkiminen voi olla haasteellista ja myös hyvin aikaa vievää. Neuroverkkojen käyttö videopeleissä saattaisi tuoda lisäarvoa, sillä neuroverkon opetusvaiheen taakasta osa voidaan siirtää pelaajan vastuulle. Näin ollen pelaajan toimista voitaisiin muodostaa opetusdataa, jota käyttämällä neuroverkon oppiminen kuvastaisi pelaajan omaa osaamista ja oppimista. Tämä myös osaltaan vaikuttaa pelin oppimiskynnyksen muodostumiseen. Kuitenkin vaarana saattaa piillä pelaajan tuottama opetusdata. Tämä data ei välttämättä ole kovinkaan kelvollista ja näin ollen verkko oppii pelaamaan yhtä ”huonosti” kuin itse pelaaja ja näin ollen videopeli saattaa muuttua huomattavan epämiellyttäväksi kokemukseksi. Tämän kaltaisen ongelman ratkaisuna voitaisiin kuitenkin käyttää tekoälyratkaisua, jossa sovelletaan tavanomaisia tekoälymenetelmiä ja neuroverkkoja.
Erittäin mielenkiintoinen ja ehkäpä hieman epätavanomainen sovellusalue neuroverkolle on esitetty Barcelos Tronton [2008] tutkielmassa, jossa neuroverkkoa käytettiin arvioimaan ohjelmistoprojektien työmääriä. Tutkielman syötedatana käytettiin melko tunnettua COCOMO-dataa. Data sisältää muutamien kymmenien ohjelmistoprojektien erilaisia attribuutteja, kuten koko, aikataulutus ja vaatimusten haavoittuvuus. Tutkielman tulokset olivat melko valoisia ja nähtävissä oli, että neuroverkkototeutuksia voitiin soveltaa melko hyvin tämänkin kaltaisiin sovellusalueisiin. Suurin ongelma kuitenkin oli syötedatan vähäinen määrä ja epähomogeenisyys.

4.4 Neuroni

Neuroverkko käsittelee sille annettua syötteitä neuroneiksi kutsutuissa alkioissa, jotka jakavat tiettyjä samoja ominaisuuksia elollisten olentojen aivoissa olevien neuronien kanssa [Fausett, 1994]. Toisin sanoen neuroni on melko yksinkertainen laskentayksikkö, joka vastaanottaa yhden tai useamman syötetiedon, painottaa ne painoarvokertoimien mukaan, yhteenlaskee tulokset ja välittää summan

aktivointifunktiolle. Aktivaatiofunktion ominaisuuksista riippuu, kuinka neuroni suhtautuu yhteenlaskettuun tulokseen ja välittää tästä uuden syötteen eteenpäin. Neuroni siis käsittää seuraavat osat:

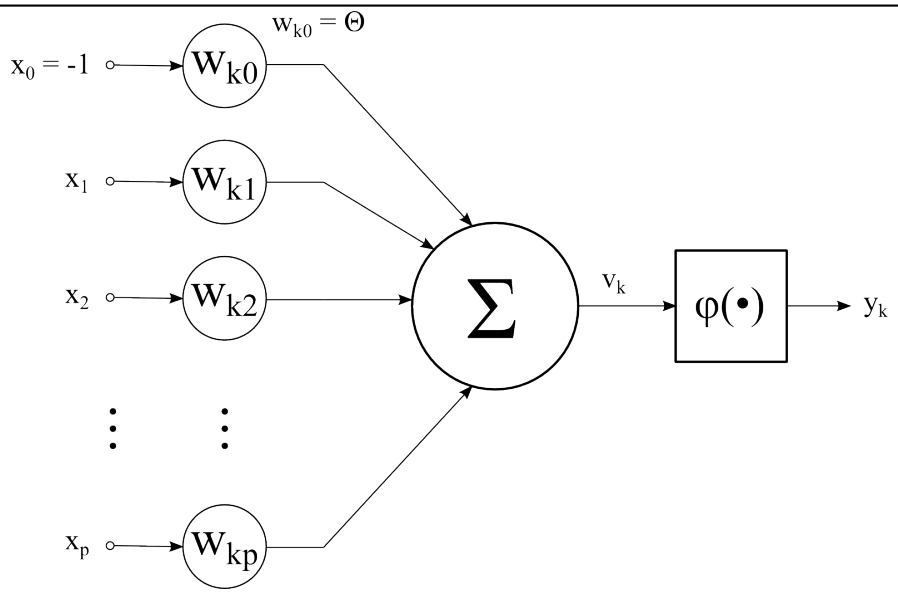
1. *Synapsien joukko* on joukko numeerista dataa jotka saapuvat neuronille joko ympäröivästä maailmasta syötteinä tai edeltävän kerroksen neuroneilta. Jokaiseen synapsiin liittyy painoarvokerroin. Synapsilta saapuva numeerinen data kerrotaan kyseisen synapsin painoarvokertoimella. Tämän avulla voidaan suuremmilla painoarvokertoimilla korostaa haluttuja synapseja. Synapsien rinnalla neuroni vastaanottaa myös vakioarvoisen siirtotermin, jonka arvo on 1 ja sillä on myös painoarvokerroin. Vaikka siirtotermi ei ole täysin verrattavissa synapseihin, on kuitenkin yksinkertaisuuden vuoksi syytä määritellä se kuten synapsi, mutta määritellä se se ulkopuoliseksi vakiosyötteeksi. [Järvelin, 2008].

2. *Summain* on elementti jonka tehtävänä on laskea yhteen neuroniin saapuvat syötteet jotka ovat kerrottu painoarvokertoimillaan. Summain voidaan ymmärtää lineaarisena kombinaattorina [Haykin, 1994]. Edellinen kuvaus voidaan esittää matemaattisesti kaavalla:

$$v_k = \sum_{j=0}^{p} w_{kj} x_j \tag{4.1}$$

3. *Aktivaatiofunktio* on funktio joka määrittää syötetiedon ja tuloksen keskinäisen riippuvuuden. Aktivointifunktio voidaan myös ymmärtää eräänlaisena rajoittimena, joka rajoittaa neuronin tuloksen halutulle arvoalueelle[Haykin, 1994]. Aktivointifunktiota on olemassa erilaisia ja aktivointifunktion valinta on syytä tehdä käsillä olevan ongelman tai tehtävän laadun mukaan. Mutta kuitenkin usemmissa tapauksissa epälineaarinen aktivointifunktio on paras valinta. Saavuttaaksemme monikerrosneuroverkkojen hyödyt ja edut on käytettävä epälinearisia aktivointifunktioita, sillä käyttämällä lineaarisia aktivointifunktiota monikerrosneuroverkossa lopputulos on täysin sama kuin käyttäisimme yksikerrosneuroverkkoa [Fausett, 1994].

Esittelen seuraavaksi muutamia tunnetuimpia ja käytetyimpiä aktivointifunktioita. *Askelfunktio* tunnetaan myös nimellä *Heavysiden funktio*. [Järvelin, 2008] Heavysiden funktiota aktivointifunktiona käyttävä neuroni voi saada siis vain kaksi



Kuva 4.1 Neuroni

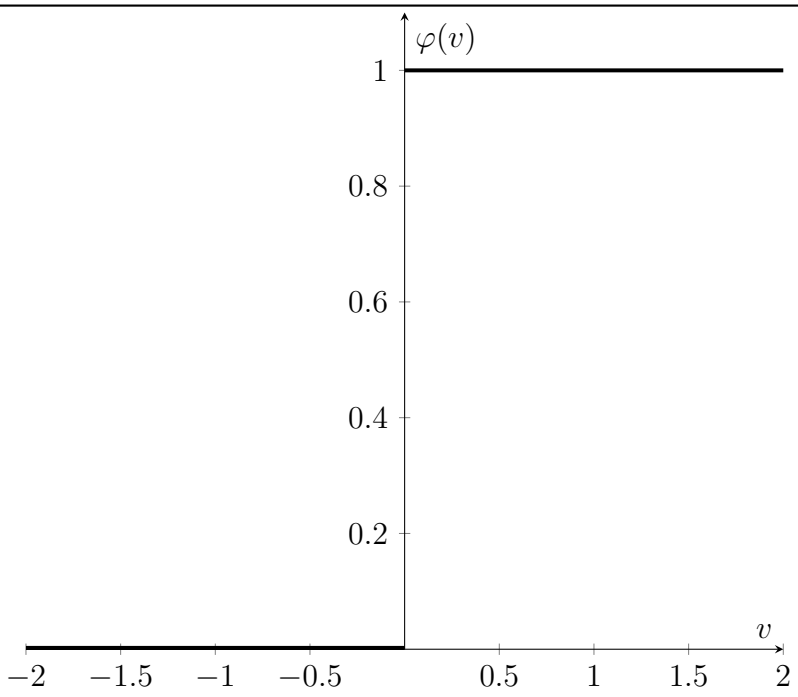
tilaa: aktivoitu ja deaktivoitu, hyvin samalla tavalla kuin biologinen neuroni. Askelfunktio määritellään seuraavasti:

$$\varphi(v) = \begin{cases} 1 & \text{jos } v \geq 1 \\ 0 & \text{jos } v < 0 \end{cases} \tag{4.2}$$

Aiemmin mainittu McCulloch-Pitts-neuroni käyttää lineaarista askelfunktiota, mikä selittää, miksi kyseiset neuronit kykenevät vain lineaariseen kahden hahmon erotteluun. Aktivointifunktion valinta ei rajoitu vain mainittuun yhteen funktioon, vaan voi olla käytännössä mikä tahansa funktio. Kuitenkin tietyt funktion ominaisuudet, kuten jatkuvuus ja derivoituvuus, tuovat lisäarvoa etevämpien ja monimutkaisempien neuroverkkojen käytössä. Ehkäpä käytetyin tällainen esimerkki derivoituvasta ja jatkuvasta funktiosta on sigmoidifunktio, jonka arvoalue voidaan skaalata halutulle alueelle.[Fausett, 1994]

Paloittain lineaarinen funktio on hieman monipuolisempi kuin askelfunktio. Paloittain lineaarinen funktio on lineaarinen, mutta sitä voidaan luonnehtia approksimaatioksi epälineaarisesta funktiosta. [Haykin, 1994]. Paloittain lineaarinen funktio määritellään seuraavasti:

$$\varphi(v) = \begin{cases} 1 & \text{jos } x \geq 0 \\ v & \text{jos } \frac{1}{2} > v > -\frac{1}{2} \\ 0 & \text{jos } x < 0 \end{cases} \tag{4.3}$$



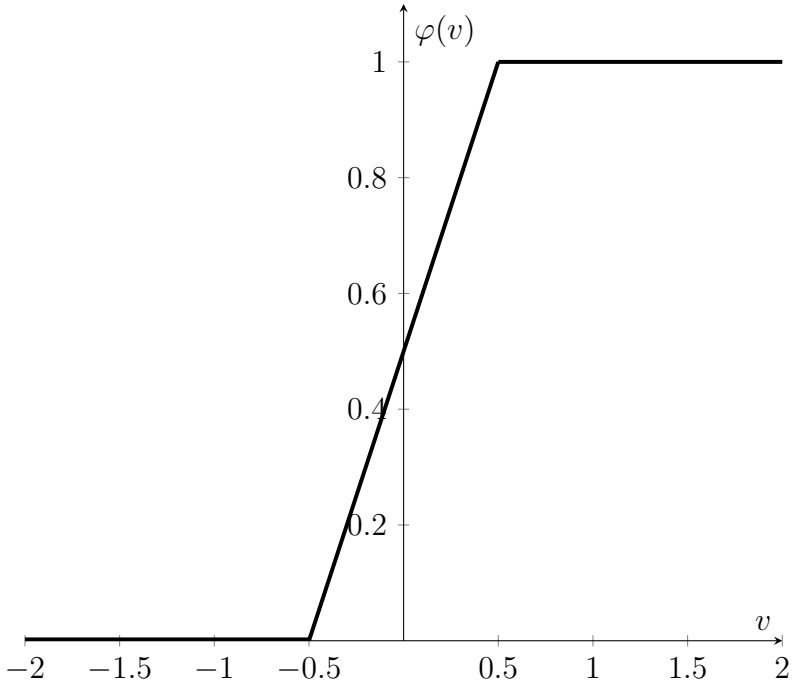
Kuva 4.2 Askelfunktion kuvaaja

Kuvaaja voidaan nähdä kuvassa 4.3.
Sigmoidifunktio on epälineaarinen aktivointifunktio jonka, kuvaaja nähdään kuvassa 4.4. Tämä sigmoidifunktio sijoittuu arvoalueelle (0,1), joka tarkoittaa myös sitä, että neuronin syötteet ja tulokset rajoittuvat tälle alueelle. Sigmoidifunktio on yksi käytetyimpiä aktivointifunktioita [Haykin, 1994]. Sigmoidifunktio voidaan esittää esimerkiksi seuraavalla kaavalla:

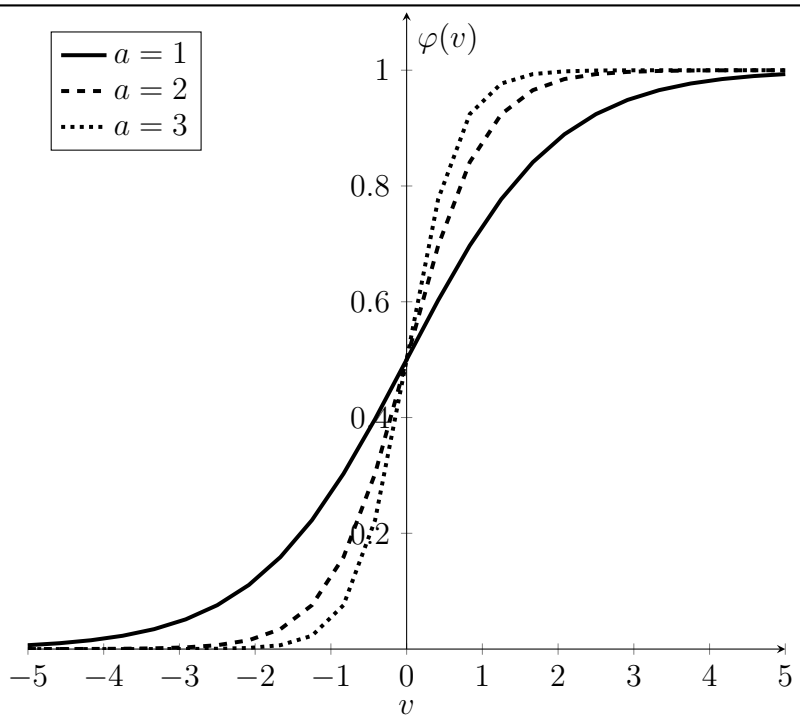
$$\varphi(v) = \frac{1}{1 + e^{-av}} \tag{4.4}$$

Funktion jyrkkyyttä voidaan säädellä parametrin a arvoa vaihtelemalla, tämä parametri tunnetaan nimellä *jyrkkyysparametri*. Mikäli jyrkkyysparametri kasvaa äärettömän suureksi, muuntuu sigmoidifunktio askelfunktioksi [Haykin, 1994]. Sigmoidifunktio on askelfunktiota joustavampi erilaisille syötteille, sillä askelfunktio käsittelee binäärisiä syötteitä 0 ja 1, kun taas sigmoidifunktio käsittelee reaalilukuja väliltä $(0, 1)$ [Haykin, 1994]. Sigmoidifunktio on kasvava ja derivoituva funktio, jonka ansiosta se soveltuu mainiosti myöhemmin mainittavan takaisinpäin levittävän neuroverkon aktivointifunktioksi. Mikäli sigmoidifunktion arvoaluetta halutaan laajentaa välille (-1,1), käytetään seuraavan mukaista esitystä:

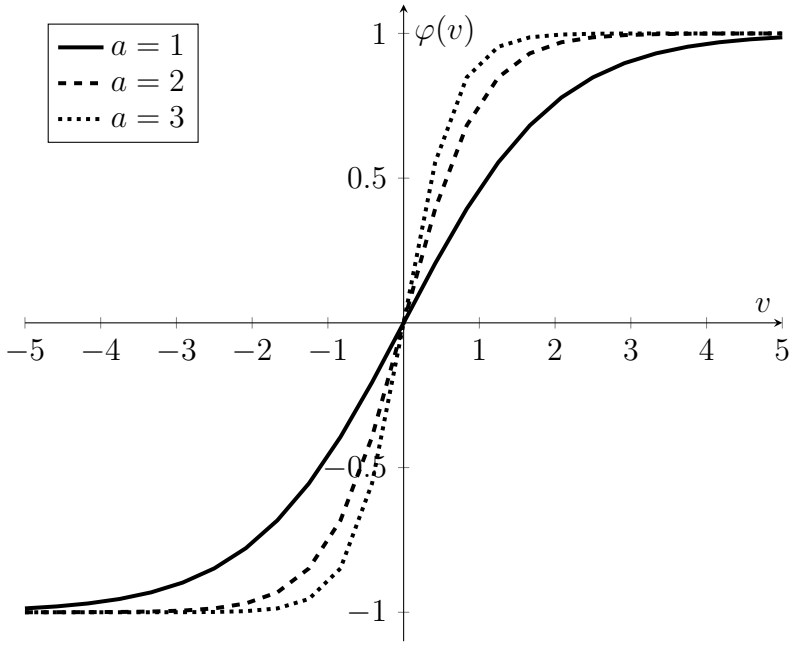
$$\varphi(v) = \frac{1 - e^{-av}}{1 + e^{-av}} \tag{4.5}$$



Kuva 4.3 Paloittain lineaarinen funktion kuvaaja



Kuva 4.4 Sigmoidifunktion kuvaaja



Kuva 4.5 Bipolaarisen sigmoidifunktion kuvaaja

Edellisen kaavan määrittelemää aktivointifunktiota kutsutaan bipolaariseksi sigmoidifunktioksi. Kuvaajasta 4.5 voidaan nähdä, että funktion arvovalue on $(-1, 1)$. Bipolaarisen sigmoidifunktiolla on luonnollisesti samat ominaisuudet kuin sigmoidifunktiolla koskien jyrkkyysparametria, jatkuvuutta ja derivoituvuutta.
Näin ollen voidaan aktivointifunktion valinnalla vaikuttaa siihen, luokitteleeko neuroni lineaarisesti vai epälineaarisesti. Aktivointifunktion valinta on siis tehtävä aina ongelmakohtaisesti. Tässä tutkielmassa pysytellään bipolaarisen sigmoidifunktion käytössä, sillä se mahdollistaa jäljempänä esitellyn takaisinlevittävän neuroverkon käytön.

4.5 Tietämyksen esittäminen

Seuraan tässä kappaleessa Haykinin [1994] määrittelemää tietämyksen esittämisen kaavaa. Tietämys voidaan ymmärtää neuroverkkojen tapauksessa tallennettuna informaationa tai malleina, joita neuroverkko käyttää hyväksi tulkitessaan, ennakoidessaan ja ollessaan muutoin vuorovaikutuksessa ympäristön kanssa. Neuroverkot ovat hyvin monipuolinen työkalu, jotka voivat käsitellä hyvin erilaatuista informaatiota. Tästä johtuen on ensiarvoisen tärkeää, että tietämys on esitetty neuroverkolle sopivassa muodossa ja että tietämys koskettaa ongelmia, joita varten neuroverkko suunnitellaan. Tästä syystä neuroverkkoja suunnitellessa on

pidettävä mielessä, mitä tietoa käsillä olevasta ongelmasta on välittömästä saatavilla ja minkälaista informaatiota neuroverkko tulee toiminnassaan vastaanottamaan. Ensin mainittua tietoa voidaan usein pitää melko selkeänä ja helposti suunnittelussa hyväksi käytettävissä olevana. Kun taas jälkimmäinen ei aina välttämättä ole suunnitteluvaiheessa niinkään yksikäsitteistä. Tästä samasta informaatiosta ammennetaan neuroverkon opetuksessa käytettävä opetus- ja testausdata.
Opetus- ja testausdata koostuu pareittain määritellyistä syötteistä ja tuloksista: syöte ja sitä vastaava tulos. Kuten edellä on mainittu opetus- ja testausdata edustavat tietämystä käsillä olevan ongelman ratkaisusta ja ratkaisuun tarvittavasta informaatiosta. Juuri tämä seikka tekee neuroverkoista varsin mielenkiintoisen ongelmanratkaisumenetelmän. Neuroverkon suunnittelu perustuu suoraan käsillä olevaan informaatioon.
Haykin [1994] esittelee muutamia määritelmiä neuroverkolle soveltuvasta ja hyvin määritellystä informaatiosta. Samankaltaisten syötteiden tulisi muodostaa samankaltaisia esityksiä (painoarvokertoimet) verkon sisällä, toisin sanoen verkon tulisi tuottaa saman vastaus samankaltaisille syötteille. Syötteiden samankaltaisuus voidaan määritellä monella eri tavalla ja on hyvin syötteen laadusta riippuvaista, kuinka määritellään samankaltaisuus. Luonnollisesti taas hyvin erilaisten syötteiden tulisi muodostaa erilaisia esityksiä verkon sisällä ja täten tuottaa erilaisia tuloksia. Tiedossa oleva informaatio ja muuttumaton informaatio tulee jollain tapaa sisällyttää neuroverkon rakenteeseen, jolloin neuroverkon ei tarvitse opetella sitä. Näin säästytään tarpeettomalta opetukselta ja neuroverkon rakenne pysyy yksinkertaisena. Nämä seikat mahdollistavat nopean suorituksen, pienemmän tilankäytön sekä minimoivat virhepäätelmien tekemistä.

4.6 Opetus

Tässä kappaleessa perehdytään Heavysiden funktiota aktivointifunktiona käyttävän neuronin opetuksen tarkasteluun. Myöhemmässä vaiheessa perehdytään sigmoidi-funktiota käyttävän neuronin opetukseen. Neuronin opettaminen perustuu neuronin painoarvojen muokkaamiseen. Muokkaamisen tarve voidaan määritellä neuroverkon tekemän virheen perusteella. Tämän kaltainen opetusmenetelmä tunnetaan ”ohjattuna opetuksena”, sillä siinä verkon kehittäjällä on ennaltatunnettua opetusdataa ja jokaiselle opetusdatan yksikölle oikein määritelty luokittelu. Kuten edellä on kuvattu neuroni saa syötteenä syötedataa, painottaa syötteet painoarvoilla, summaa yhteen painoarvotetut syötteet ja välittää

summan aktivointifunktiolle. Opetusvaiheen alussa on luonnollisesti määriteltävä painoarvoille alkuarvot. Painoarvojen alkuarvojen valinta vaikuttaa verkon oppimisnopeuteen ja näin ollen alkuarvojen valintaan on kehitetty erilaisia menetelmiä. Yksinkertaisin menetelmä on kuitenkin valita painoarvokertoimet satunnaisesti. Tämä menetelmä on käytössä tässä tutkielmassa. Seuraan Järvelinin [2008] esittämää neuronin opetusalgoritmia, jossa opetusdatajoukko jaetaan kahteen joukkoon, jotka kuuluvat eri luokkiin. Algoritmin tarkoituksen on iteratiivisesti muokata neuronin painoarvoja, siten että molemmat opetusdatajoukot pystytään luokittelemaan oikein. Painoarvojen muokkaaminen tapahtuu seuraavasti:

1. Opetusdatajoukosta valitaan ensimmäinen alkio, joka syötetään neuronille, joka suorittaa painoarvotuksen, summauksen ja summan välityksen aktivointifunktiolle.

2. Neuronin antamaa tulosta verrataan kyseisen opetusdatajoukosta valitun alkion tunnettuun luokitteluun ja mikäli luokittelu tapahtui oikein, siirrytään valitsemaan uusi alkio opetusdatajoukosta.

3. Mikäli luokittelu tapahtui väärin, muokataan neuronin kaikkia painoarvoja opetuskertoimen avulla. Opetuskerroin voi olla ennalta määritelty vakio tai opetuksen edetessä muuttuva arvo. Opetuskertoimen valinta vaikuttaa neuronin oppimisnopeuteen ja toisinaan myös kykyyn oppia.

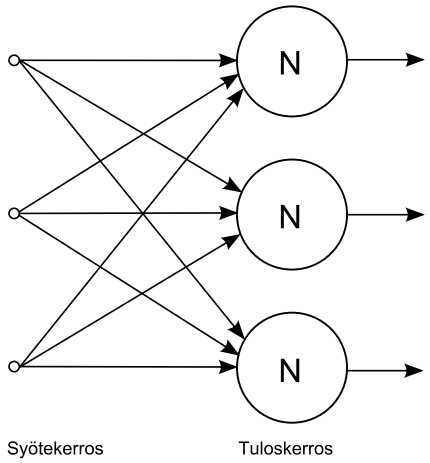
Edellä kuvattua algoritmia toistetaan, kunnes kaikki opetusdatan yksiköt ovat kerran syötetty neuronille. Tässä vaiheessa opetus voidaan lopettaa, mutta mikään ei takaa että neuroni kykenee nyt luokittelemaan oikein. Tästä syystä opetusdata saatetaan syöttää neuronille useita kertoja. Tähän aiheeseen palataan tarkemmin taaksepäinlevittävien verkkojen yhteydessä.

4.7 Rakenne

Useissa käytännön ongelmissa yksi neuroni ei kykene suoriutumaan annetusta luokittelutehtävästä, sillä se kykenee luokittelemaan saamansa syötedatan kahteen joukkoon. Yksinkertaisin esimerkki tästä lienee looginen operaatio ”poisulkeva tai”, joka voidaan logiikassa esittää ja-, tai- ja ei-operaatioden avulla. Nämä taas voidaan toteuttaa yksittäisillä neuroneilla. Tästä johtuen on useissa tapauksissa tarpeellista luoda usean neuronin verkko.

On olemassa erilaisia neuroverkkorakenteita, mutta kuitenkin kaikille verkoille yhteisiä piirteitä ovat syötekerros ja tuloskerros. Useimmin käytetty verkko muodostuu kerroksista, joihin kuuluvat syötekerros, piilokerrokset ja tuloskerros. Syötekerroksen tehtävänä on nimensä mukaisesti vastaanottaa syötteet ja välittää ne seuraavan kerroksen neuroneille. Nämä taas välittävät tuloksensa seuraavalla kerrokselle, ja niin edelleen kunnes saavutaan tuloskerrokseen. Edellä kuvattua signaalien välitystä kutsutaan eteenpäinsyöttämiseksi. Verkon kyky luokitella vaihtelevia määriä eri luokkia riippuu tuloskerroksen neuronien määrästä, sillä kukin neuroni kykenee luokittelemaan kaksi tapausta [Järvelin, 2008]. Haykin [1994] esitettelee neljä yleisintä neuroverkkorakennetta seuraavasti:

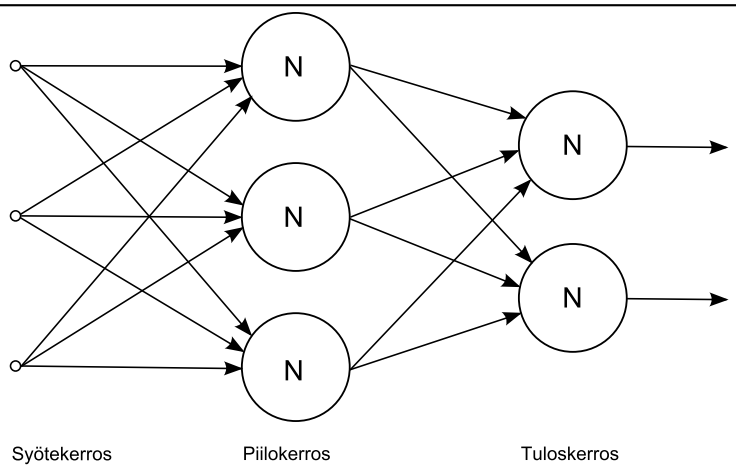
Yksikerroksinen eteenpäinsyöttävä verkko on kaikista yksinkertaisin neuroverkkorakenne. Verkko koostuu syötekerroksesta ja tuloskerroksesta. Tuloskerroksen ei tulkita koostuvan neuroneista, sillä niissä ei tapahdu painoarvokertoimimen muutoksia, toisin sanoen ne vain vastaanottavat syötedatan ja välittävät sen tuloskerrokselle. Syötekerroksen syötteiden lukumäärä on yksi tai useampia. Tuloskerros taas koostuu neuroneista, jotka saavat syöteensä syötekerrokselta. Kuten syötekerroksen syötteiden lukumäärä on määritelty yhdeksi tai useammaksi, myös tuloskerroksen neuronien määrä on määritelty yhdeksi tai useammaksi. Tuloskerroksen neuronien painoarvokertoimet voivat muuttua, joten verkon tietämys tallentuu suoraan tuloskerroksen neuroneihin.



Kuva 4.6 Yksikerroksinen neuroverkko

Useampikerroksinen eteenpäinsyöttävä verkko on huomattavasti monimutkaisempi kuin yksikerroksinen. Tässä neuroverkkorakenteessa kerroksia voi olla kaksi tai useampia. Kuten muutkin neuroverkkorakenteet, useampikerroksinen neuroverk-

ko koostuu syötekerroksesta ja tuloskerroksesta. Kuten edelläkin, syötteiden ja tulosneuronien lukumäärä on määritelty yhdeksi tai suuremmaksi. Yksikerroksiseen neuroverkkoon nähden eroavaisuus löytyy piilokerroksesta tai -kerroksista. Piilokerros koostuu neuroneista, joiden lukumäärä on määritelty yhdeksi tai useammaksi ja piilokerrosten lukumäärää on määritelty yhdeksi tai useammaksi. Käyttämällä piilokerroksia voidaan neuroverkkorakenteen kykyä oppia monimutkaisempia tehtäviä edistää. Neuroverkon omaksuma tietämys taltioituu piilokerroksen tai piilokerrosten neuronien painoarvokertoimiin. Useissa tapauksissa kaikilla syötteillä ja neuroneilla on yhteys seuraavan kerroksen kaikkiin neuroneihin. Kuitenkin on luonnollisesti myös mahdollista toteuttaa verkko, jossa osa näistä yhteyksistä puuttuu. Edellä kuvattua verkkototeutusta kutsutaan osittain kytketyksi verkoksi. Tämän kaltaisilla verkoilla voidaan syötteistä korostaa tiettyjä ominaisuuksia. *Toistuva verkko* on neuroverkkorakenne, jossa on vähintään yksi



Kuva 4.7 Useampikerroksinen neuroverkko

palautesilmukka. Tämän silmukan tehtävänä on syöttää verkon tulosarvot takaisin syöttöarvoiksi. Toistuvalla verkolla on samoja ominaisuuksia kuin kahdella edellä mainitulla: syötteiden ja tulosneuronien lukumäärä on määritelty yhdeksi tai suuremmaksi. Verkko voi sisältää tai olla sisältämättä piilokerroksia. Toistuva verkko sisältää myös kontekstineuroneita, joihin edellä mainitut silmukat liittyvät. Fauseet [1994] esittelee hilaverkkorakenteen, joka opetetaan tunnistamaan tietyn äärellisen automaatin generoimia vaihtelevan mittaisia sanoja.

Hilaverkko on edellisistä verkoista siten poikkeava, että sen syötteet ja tulosneuronit on järjestelty hieman eri tavalla. Kuitenkin syötteiden ja tulosneuronien lukumääriin päätee sama määritelmä, yksi tai useampia. Myös piilokerrosten lu-

kumäärää on määritelty nollaksi tai usemmaksi. Käytännössä hilaverkko on sama asia kuin useampikerroksinen verkko, mutta tulosneuronit on järjestelty riveihin ja sarakkeisiin.
Kaikille neuroverkkorakenteille on yhteistä, että neuroneihin tulevilla syötteillä on painoarvokerroin. Painoarvokertoimia muuntelemalla neuroverkko saavuttaa kykynsä suorittaa sille määritellyn tehtävän. On mahdollista suunnitella neuroverkko, jossa painoarvokertoimet ovat kiinteät ja niitä ei muuteta suunnittelun jälkeen, näin ollen verkko ei opi uutta suunnitelun jälkeen. Mikäli painoarvokertoimia muunnellaan verkon suorituksen aikana, verkko oppii sille esitettyn informaation ja siihen liittyvän tuloksen kuvauksen. On olemassa erilaisia tapoja muunnella painoarvokertoimia, ja tätä kutsutaan verkon opettamiseksi. Tästä johtuen voidaan kuvitella, että neuroverkko tallentaa oppimansa informaation neuronien painoarvokertoimiin.

5 Takaisinlevittävä neuroverkko

Luvussa 4 esiteltiin yleisesti lähes kaikkia neuroverkkoja yhdistäviä tekijöitä ja ominaisuuksia. Tässä luvussa perehdytään tarkemmin takaisinlevittävään neuroverkkototeutukseen, jota tässä tutkielmassa hyödynnetään. Takaisinlevittävä neuroverkko koostuu yksittäisistä perceptron-neuroneista jotka ovat verkostoitu usean perceptronin verkoksi. Näin kuvattuna takaisinlevittävä neuroverkko ei juurikaan eroa edellä kuvatuista neuroverkoistarakenteista, mutta takaisinlevittävän neuroverkon ehkäpä tärkein ominaisuus piilee sen opetusmenetelmässä. Edellä kuvatussa neuronin opetusmenetelmässä perehdyttiin neuroniin, jonka aktivointifunktiona käytettiin Heavysiden funktiota. Funktion ominaisuuksien puutteesta johtuen niistä koostuvan verkko voi koostua vain ja ainoastaan syöte- ja tuloskerroksista. Tämä saattaa osoittautua kriittiseksi puutteeksi ongelmien ratkaisuissa. Kuten jo aiemmin on mainittu, neuronin aktivointifunktio voi olla käytännössä mikä tahansa funktio, voidaan neuronin aktivointifunktio vaihtaa jatkuvasta derivoituvaan funktioon [Järvelin, 2008]. Edellä esitelty sigmoidifunktio on jatkuvasta derivoituva ja epälineaarinen funktio ja siten soveltuva useakerroksiseen neuroverkkoon. Useakerroksinen, takaisinlevittävä neuroverkko soveltuu mainiosti monimutkaisienkin ongelmien ratkaisujen löytämiseen ja siksi tässä tutkielmassa keskitytään juuri tämän kaltaisen neuroverkon sovelluksen tutkimiseen.

5.1 Opetus

Noudattelen Fausettin [1994] esittelemää monikerroksisille neuroverkoille soveltuvaa takaisinlevitysmenetelmää. Takaisinlevitys on eräs useakerroksisten neuroverkkojen opetusmenetelmä, joka on saavuttanut kohtalaisen suuren suosion [Järvelin, 2008]. Kuten yksittäisen neuronin opetuksessa, verkon toteuttaja tarvitsee opetusdataa. Opetusdatan alkioiden luokittelu on tunnettu, jolloin kyse on ohjatusta oppimisesta. Oppimisen tarkoituksena on saavutta tasapainotila kahden toiminnin välillä. Nämä toiminnot ovat opetettavan datan tunnistaminen, jota voidaan luonnehtia muistiksi, ja kyky antaa uudella syötteille hyväksyttävä vastaus, jota voidaan luonnehtia yleistämiseksi. Oppimisen aikana neuroverkon rakennetta muokataan säätämällä neuronien välisiä painoarvoja ja näin ollen "tallennetaan"neuroverkon muistiin syöteet ja niiden vastaukset. Oppiminen siis käsittää kolme vaihetta: opetusyksiköiden syöttäminen neuroverkolle, virhekertoimien laskeminen ja painoarvokertoimien korjaus [Fausett, 1994]. Kuvasta 5.1 voidaan nähdä kuinka syötteet ja virhesignaalit liikkuvat eri suuntiin verkossa

opetuksen ollessa käynnissä.
Seuraavaksi esittelen opetusproseduurin yksityiskohtaisesti:

1. *Painoarvojen alustaminen.* Yksinkertaisin ratkaisu painoarvojen alustamiseen on asettaa painoarvo kertoimet pieniksi satunnaisiksi luvuiksi. On kuitenkin tärkeää huomata, että valittaessa liian suuret arvot neuronien välisit signaalit häviävät saturaatioalueelle jossa sigmoidifunktion derivaatalla on hyvin pieni arvo. Toisaalta liian pienet arvot taas hidastavat oppimista. Yleisesti ottaen painoarvot kannattaa alustaa arvoalueelle (-0.5, 0.5) [Fausett, 1994].

2. *Syöte.* Käydään läpi kaikki opetusdatan syöteyksiköt ja syötetään ne yksitellen neuroverkolle. Neuroverkko välittää syötteet eteenpäin seuraavaan piilokerrokseen, jossa neuronit laskevat yhteen syöteensä kaavan mukaisesti:

$$v_k = \sum_{j=0}^{p} w_{kj} x_j \tag{5.1}$$

   Lopulta summa suodatetaan aktivointifunktion läpi, joka on tässä tapauksessa on bilopaarinen sigmoidi funktio. Tämä vaihe suoritetaan piilokerrosten lukumäärän verran.

3. *Tulos.* Piilokerrokselta saapuvat signaalit yhteen lasketaan tuloskerroksessa ja summat välitetään tuloskerroksen neuronien aktivointifunktiolle.

4. *Virheen laskeminen.* Tuloskerroksen laskemia tuloksia verrataan ennalta määriteltyyn syöteyksikön tulokseen. Vertailu tapahtuu laskemalla paikallinen gradientti kaavalla:

$$\delta_k = (t_k - y_k)\varphi'(v_k) \tag{5.2}$$

   Jossa y_k on neuronin laskema tulos. Paikallista gradienttia hyödynnetään laskemalla painoarvokertomen muutos kaavalla:

$$\Delta w_{jk} = \alpha \delta_k z_j \tag{5.3}$$

   Jossa Δw_{jk} on neuronien j ja k välisen painoarvokertoimen muutos, α on opetuskerroin ja z_j on neuronista j saapunut tulos. Tämä tulos välitetään tuloskerrosta edeltävälle piilokerrokselle.

5. *Virhesignaalin välitys.* Kukin piilokerroksen neuroni vastaanottaa virhekertoimet tuloskerroksen neuroneilta. Neuronille j nämä virhekertoimet lasketaan yhteen kaavalla:

$$\delta_j = \sum_{k=0}^{m} \delta_k w_{jk} \tag{5.4}$$

Jossa m on piilokerroksen seuraavan kerroksen neuronien lukumäärä. Tämä seuraava kerros voi olls joko tuloskerros tai toinen piilokerros. Seuraavaksi lasketaan piilokerroksen neuronin j ja tuloskerroksen neuronin i välisen painoarvokertoimen muutos. Tämä tapahtuu kaavalla:

$$\Delta w_{ij} = \alpha \delta_j \varphi'(v_k) \tag{5.5}$$

Jossa α on opetuskerroin ja v_k neuroniin saapuneiden syötteiden summa, joka on laskettu kohdassa 2.

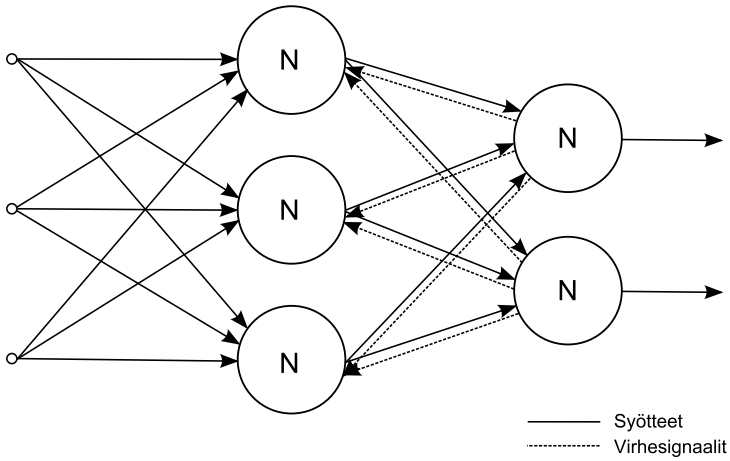
6. *Painoarvojen korjaus.* Seuraavaksi korjataan neuroverkon painoarvokertoimet edellä saatujen tulosten mukaisesti. Painoarvokertoimen muutos tapahtuu kaavalla:

$$w'_{jk} = w_{jk} + \Delta w_{jk} \tag{5.6}$$

Jossa w'_{jk} on neuronien j ja k välinen uusi painoarvokerroin ja w_{jk} on vanha painoarvokerroin.

Edellä olevan proseduuri kuvaa, kuinka yksi opetusyksikkö voidaan opettaa neuroverkolle. Kuitenkin on huomattava että, mikäli piilokerroksia on useampia kuin yksi, täytyy kohta 2 suorittaa piilokerrosten lukumäärän verran. Ja luonnollisesti kohdassa 5 tapahtuva virhesignaalin takaisinvälitys tapahtuu myös useammin mikäli piilokerroksia on useampia kuin yksi. Tämän jälkeen loput opetusdatan alkiot käsitellään samalla tavalla ja näin pyritään saavuttamaan neuroverkolle painoarvot, joilla se kykenee luokittelemaan annetun opetusdatan. Usein opetusdatan yhden kerran tapahtuva läpikäyminen ei ole riittävä toimenpide, vaan se suoritetaan useita kertoja. Kuitenkin jokaisella läpikäynnillä voidaan tehostaa algoritmin toimintaa vaihtamalla alkioiden järjestystä [Järvelin, 2008].Yhtä opetusdatan läpikäyntiä kutsutaan *epookiksi*. Epookkien lukumäärä voi vaihdella sadoista jopa kymmeniin tuhansiin, riippuen sovellusalueesta, painoarvojen alustuksesta, datan esitystavasta ynnä muista erilaisista muuttujista. Takaisinlevittävälle opetusalgoritmille ei ole määritelty pysäyttämishehtoa, mutta epookkien lukumäärää voidaan rajoittaa määrittelemällä hyväksyttävä virhearvo. Epookin jokaisen alkion

tuloksesta lasketaan virhearvo ja epookin lopuksi lasketaan keskiarvo näistä virheistä. Tätä keskiarvovirhettä verrataan ennalta määrättyyn tavoitevirhearvoon ja mikäli epookin keskiarvovirhe alittaa sen, voidaan opetus voidaan keskeyttää. Mikäli keskiarvovirhettä halutaan käyttää opetusalgoritmin keskeyttämisen ehtona, täytyy opetusyksiköiden joukko jakaa kahteen osaan: opetus- ja testausjoukkoon. Opetusjoukkoa käytetään verkon opettamiseen ja tietyn intervallin, esimerkiksi jokaisen epookin jälkeen, testausjoukkoa käytetään tutkimaan onko verkon palauttama virhearvo tai sen muutos riittävä syy keskeyttää opetus. Edellä on mainittu että, neuroverko opetus voidaan lopettaa, vasta kun neuroverkko on saavuttanut sovitun keskiarvovirheen, on hyvä huomata että, neuroverkko ei välttämättä aina saavuta sovittua keskiarvovirhettä. Näin ollen voidaan tarkastella keskiarvovirheen muuttumista. Keskiarvovirheen pienentyessä neuroverkko oppii ja opetusta jatketaan, mutta mikäli keskiarvovirhe rupeaa kasvamaan on verkko saavuttanut oppimiskapasiteettinsa rajat ja ryhtyy menettää kykynsä. Näin meneteltäessä pyritään minimoimaan *ylioppiminen* [Fausett, 1994]. Ylioppiminen tarkoittaa verkon yleistyskyvyn heikkenemistä, mihin saattaa myös johtaa liian suuren piilokerroksen käyttäminen. Ylioppiminen voi ilmetä myös, siten että verkko oppii opetusdatassa ilmenevää kohinaa ja näin ollen luokittelu tapahtuu väärin. Tästä syystä opetusdatan esiprosessointi on melko merkittävä toimenpide. Takaisinlevittävän neuroverkon testausvaihe on hyvin yksinkertai-



Kuva 5.1 Syötteiden ja virhesignaalien kulkusuunnat verkossa

nen. Testattavan datajoukon alkiot syötetään neuroverkolle ja tulosta verrataan ennalta tunnettuun tulokseen. Näin ollen voidaan nähdä onko opetus onnistunut riittävän hyvin. Mikäli verkon laatija määrittelee testausvaiheessa ilmenevän

virhekeskiarvon olevan riittävän pieni, voidaan verkko siirtää tuotantokäyttöön. Mikäli kuitenkin verkon arkkitehti kokee keskiarvovirheen liian suureksi, on verkon arkkitehtuuri suunniteltava uudelleen tai tehtävä muita toimenpiteitä verkon oppimisen hyväksi. Tämä voi tarkoittaa opetusdatan uudelleen esiprosessointia, lisädatan hankkimista tai painoarvokertoimien alustamista eri tavalla. Ylioppiminen tarkoittaa tilannetta jossa neuroverkkoa opetetaan liian kauan, jolloin neuroverkko omaksuu varsin hyvin opetusaineiston ja suoriutuu sen osajoukkojen luokittelusta varsin lahjakkaasti. Tämä puolestaan hävittää verkon kyvyn yleistää tietämyksenseä opetusaineiston ulkopuolella oleville tapauksille. Opetuksen yhteydessä ylioppiminen voidaan välttää kahdella tavalla. Näistä ensimmäisessä tarkkaillaan keskiarvovirheen muutosta opetuksen edetessä. Keskiarvovirheen muutoksen saavuttaessa tietty arvo epookkien välissä voidaan opetus keskyttää ja todeta neuroverkko riittävän oppineeksi. Keskiarvovirheen muutoksen arvoa jolla opetus voidaan keskeyttää voidaan pitää prosenttiarvoa välillä [0.05, 1]. Edellä kuvattua keskiarvovirheen muutosta voidaan luonnehtia neuroverkon oppimisnopeudeksi ja mikäli nopeus tantuu riittävän alhaiseksi voidaan tulkita verkon ammentaneen kaiken tietämyksen opetusaineistosta. Joissakin tapauksissa neuroverkko ei kuitenkaan saavuta maksimaalista luokittelukykyä käyttämällä edellä kuvattua lopetusehtoa, sillä opetuksen aikana verkko saattaa vaikuttaa siltä kuin se olisi saavuttanut maksimaalisen luokittelukyvyn varhaisessa vaiheessa, mutta todellisuudessa se saavutetaan vasta useita epookkeja myöhemmin. Toinen menetelmä välttää ylioppimista on ajaa testiaineisto opetuksen aikana jokaisen epookin jälkeen ja tarkkailla testituloksen kehittymistä. Mikäli verkon luokittelukyky kohentuu epookkien edetessä opetusta jatketaan, mutta mikäli luokittelukyky suppenee voidaan opetus keskeyttää. Tässä menetelmässä on sama haittapuoli kuin ensin mainitussa. Tämän haittapuolen eliminointiin voidaan käyttää menetelmää jossa neuroverkkoa opetetaan tarkoituksellisesti liian useita epookkeja ja taltioidaan tietyin epookki-intervallein keskiarvovirheen muutos ja opetuksen aikaisen testauksen tulokset. Näitä taltioituja tuloksia analysoimalla voidaan määritellä optimaalinen epookkimäärä neuroverkolle. Tämä tietysti johtaa siihen että, neuroverkko täytyy opettaa kahdesti. Kuitenkin näin voidaan välttää neuroverkon liian aikaisen opetuksen keskeytys keskiarvovirheen muutokseen perustuen.

6 Lähimmän naapurin menetelmä

Lähimmän naapurin menetelmä on neuroverkon tapaan luokittelumenetelmä, jonka avulla voidaan myös jaotella datajokko klustereihin. Lähimmän naapurin menetelmä on hieman yksinkertaisempi kuin neuroverkko ja toimii tässä tutkielmassa vertailumenetelmänä neuroverkolle. Lähimmän naapurin menetelmä on kohtuullisen yksinkertainen toteuttaa ja siitä syystä olen valinnut sen vertailumenetelmäksi. Lähimmän naapurin menetelmä luokittelee dataa saamansa opetusdatan mukaisiin luokkiin. Uuden tapauksen luokkaa määritettäessä valitaan tietty määrä jo tunnettuja luokitteluja uutta tapausta lähinnä olevia ennalta tunnettuja tapauksia ja valitaan uudelle tapaukselle luokka näiden perusteella. Menetelmä on kohtuullisen yksinkertainen ja näin ollen haasteellisin osuus on määritellä kysymyksessä olevalle informaatiojoukolla soveltuva etäisyysmitta, jonka avulla uusien tapauksien luokittelu tapahtuu. Numeerisille joukolle etäisyysmitan valitseminen on melko triviaalia, mutta esimerkiksi tässä tutkimuksessa informaatio ei koostu numeerisesta informaatiosta.

6.1 Käyttökohteet

Lähimmän naapurin menetelmiä käytetään erityisesti luokittelutehtävissä. Lähimmän naapurin menetelmät ovat hyvin joustavia toteuttaa ja siten ne soveltuvat erinomaisesti lähestulkoon kaikenlaisen datan luokitteluun. Lähimmän naapurin menetelmää on käytetty esimerkiksi hahmontunnistukseen, luonnollisen kielen analysointiin ja moniin muihin kohteisiin. Vaatimuksena on, että datayksiköiden vertailuun voidaan kehittää tai valita jokin järjellinen etäisyysmitta. Datayksiköt voivat olla hyvinkin vaihtelevanlaatuisia erilaissa käyttökohteissa ja tästä johtuen erilaisia etäisyysmittoja on tutkittu ja kehitetty erilaisiin tarpeisiin. Useimmissa tapauksissa kuitenkin edempänä esiteltävät etäisyysmitat voidaan todeta toimiviksi ratkaisuksi.

6.2 Opetus

Lähimmän naapurin menetelmän opetuksessa käytetään opetusjoukkoa, joka luodaan olemassa olevasta aineistosta. Aineisto voi olla hyvinkin vaihtelevanlaatuista ja saattaa vaatia sen esiprosessointia, jossa voidaan häivyttää poikkeavia havaintoja ja korvata puuttuvia havaintoja. Lähimmän naapurin menetelmää voidaan kutsua ”laiskaksi” menetelmäksi. Nimitys johtuu siitä, että opetusvaiheessa tunnetun datajoukon yksiköt tallennetaan tietämyskantaan. Näin ollen termin ope-

tusvaihe käyttäminen on hieman harhaan johtavaa, sillä varsinaisesti mitään oppimista ei tapahdu, kuten neuroverkon tapauksessa. Vasta varsinaisessa luokitteluvaiheessa tapahtuu, joissakin tilanteissa, laskennallisesti mahdollisesti vaativa toimenpide, joka määrittää uuden tapauksen luokittelun. [Iltanen, 2010] Tämä menettely takaa nopean opetusvaiheen, mutta hidastaa varsinaista testaus- ja käyttövaihetta. Lähimmän naapurin menetelmän tehokkuutta on yritetty parantaa erilaisin menetelmin. Tässä tutkielmassa ei juurikaan oteta kantaa suorituskykyyn, vaan paneudutaan ainoastaan tuloksiin ja niiden oikeellisuuteen. Kuitenkin edellisten ominaisuuksien vuoksi lähimmän naapurin menetelmä saattaa osoittautua käyttökelvottomaksi nopeutta vaativissa sovelluksissa, joissa käsitellään hyvin monimutkaista dataa.

Opetusjoukon tulisi olla mahdollisimman kattava ja monipuolinen [Larose, 2005] Sen tulisi sisältää riittäviä määriä monimutkaisia ja harvinaisempi muuttujakombinaatiota. Näin voidaan taata, että menetelmä kykenee luokittelemaan myös uusia tapauksia kelvollisesti. Tämä voidaan saavuttaa tasapainottamalla opetusjoukko lisäämällä sinne vähemmän tunnettuja muuttujakombinaatiota ja poistamalla tunnettuja muuttujakombinaatiota [Larose, 2005]. Tällä menetelmällä voidaan myös joissakin tapauksissa parantaa menetelmän kykyä luokitella menettämättä tarkkuutta. Opetusaineiston olisi syytä käsittää riittävä määrä luokiteltavalle aineistolle luontaisia havaintoja, jolloin luokittelija kykenee luokittelemaan tuntemattomat havainnot oikein. Kuitenkin luokittelijan tulisi kyetä luokitella oikein myös hieman poikkeavia havaintoja, jolloin opetusjoukon tulisi sisältää myös tämänkaltaisia havaintoja. Edellä mainittu seikka on melko kriittinen ongelma tehokkaan luokittelijan luomisessa. Suurella todennäköisyydellä monipuolinen opetusaineisto tuottaa enemmän oikein luokiteltuja havaintoja kuin vähemmän monipuolinen, mutta se ei myöskään kykene mukautumaan yhtä tehokkaasti uusiin havaintoihin kuin vähemmän monipuolinen. Tätä ongelmaa kutsutaan ”poikkemavarianssi-vaihtokaupaksi”, jossa luokittelijan laatijan on selvitettävä monipuolisen ja vähemmän monipuolisen luokittelija hyödyt ja haitat, ja pyrittävä laatimaan luokittelija, joka käsittää molempien tarpeellisia ominaisuuksia. Edellä mainittu toimenpide on erittäin tarpeellinen myös sen kannalta, että luokittelija ei ”ylioppi”. *Ylioppiminen* ilmenee siten, että luokittelija osaa liki täydellisesti luokitella testausaineiston, mutta ei osaa lainkaan luokitella uusia tapauksia oikein. Luonnollisesti ylioppimisen vastakohtana on *alioppiminen*, jossa luokittelija ei kykene luokittelemaan kumpaankaan; testausaineisto ja tuntematonta aineistoa oikein. Kuten jo edellä mainittu on luokittelija ja opetusaineisto määriteltävä siten, että luokittelija on riittävän monipuolinen ja samalla riittävän yksinkertainen. Tämä

ei ole lainkaan helppo tehtävä ja luokittelija saattaa saada tehtäväkseen luokitella täysin uudenlaisia havaintoja, jolloin luokittelija laatijan on ylläpidettävä luokittelijan kykyä suoritua tehtävästään. Kuitenkin lähimmän naapurin menetelmän tapauksessa suurempi vaikutus luokittelijan onnistuneeseen laatimiseen on etäisyysmitan valinnassa.

6.3 Etäisyysmitta

Lähimmän naapurin menetelmää käytettäesssä tärkein tehtävä on etäisyysmitan valinta. Etäisyysmitta voi olla käytännössä mikä tahansa metodi, jolla voidaan vertailla kahta datayksikköä toisiinsa ja selvittää, kuinka etäällä tai lähellä ne ovat toisiaan. Etäisyysmitan valinnassa on kuitenkin otettava muutamia seikkoja. Kuten sanottu etäisyysmitan tulee kyetä ilmaisemaan skalaarinen etäisyys kahden datayksikön välillä. Etäisyysmitan d täytyy toteuttaa seuraavat neljä ominaisuutta datayksiköille a, b ja c.

$$d(a, b) \geq 0 \tag{6.1}$$

$$d(a, b) = 0 \text{ joss } a = b \tag{6.2}$$

$$d(a, b) = d(b, a) \tag{6.3}$$

$$d(a, b) \leq d(a, b) + d(b, c) \tag{6.4}$$

Ensimmäinen vaatimus määrittelee, että etäisyys on aina positiivinen ja näin ollen järjellinen suure etäisyyksiä vertailessa. Toinen vaatimus määrittelee, että etäisyys on nolla jos ja vain jos vertailussa olevat havainnot ovat identtiset. Kolmas vaatimus määrittelee, että etäisyys on sama riippumatta havaintojen järjestyksestä. Neljäs vaatimus määrittelee, että mikäli käytetään kahden havainnon sijasta kolmea havaintoa, etäisyys ei voi olla lyhyempi kuin käytettäessä kahta havaintoa. Etäisyysmitan valinta riippuu käytettävän datan laadusta, esimerkiksi jatkuville muuttujille voidaan hyödyntää euklidista etäisyyttä joka toteuttaa edellä mainitut ominaisuudet. Tämä etäisyysmitta voidaan laskea seuravaalla kaavalla:

$$d(x, y) = \sqrt{\sum_{i=0}^{m} (x_i - y_i)^2} \tag{6.5}$$

Kategoriallisille muuttujille edellä kuvatun kaltaista etäisyysmittaa ei voida käyttää. Niiden etäisyyksien vertailuun esitellään menetelmä tuonnenpana. Mikäli luokittelussa käytetään useita muuttujia yhden sijasta, saattaa joskus olla tarpeellista normalisoida joitakin muuttujia. Normalisoinnin syynä voi olla esimerkiksi muuttujan huomattavasti laaajempi arvoalue. Mikäli tämänkaltainen muuttuja jätetään normalisoimatta, se saattaa peittää alleen muut muuttujat ja näin toimia liian vaikuttavana tekijänä luokittelussa. Jatkuville muuttujille voidaan suorittaa esimerkiksi min-max-normalisointi tai z-score-normalisointi. Normalisoinnit toteutetaan seuraavasti:

$$X' = \frac{X - \min(X)}{\max(X) - \min(X)} \tag{6.6}$$

$$X' = \frac{X - \mu}{\sigma} \tag{6.7}$$

Kategoriallisia muuttujia ei voida normalisoida edellä kuvattuja menetelmien avulla. Kategorisille muuttujille voidaan käyttää yksinkertaista samankaltaisuuden vertailua. Kahden kategorisen muuttujan etäisyys voidaan määritellä tutkimalla ekvivalenssia, muuttujien ollessa samat etäisyydeksi määritellään nolla ja mikäli ne ovat eri etäisyydeksi määritellään yksi. Useissa tapauksissa muuttujia on kuitenkin enemmän kuin yksi, jolloin muuttujien etäisyyden lasketaan yhteen, jolloin saadaan kahden yksikön etäisyys. Kuitenkaan data ei aina ole soveltuva edellä mainitulle etäisyysmitalle, kuten esimerkiksi tässä tutkielmassa, jossa data koostuu pikemminkin merkkijonoista. Merkkijonojen etäisyyksien vertailuun voidaan käyttää Hamming-etäisyysmittaa, joka vertailee merkkijonon merkkejä keskenään. Menetelmän vaatimuksena on, että merkkijonot ovat keskenään yhtä pitkät. Menetelmä on hyvin samankaltainen kuin kategoriallisien muuttujien etäisyyksien määrittelyn menetelmä. Tämä luonnollisesti vaatii, että datajoukon kaikki yksiköt ovat samanpituisia merkkijonoja. Hamming-etäisyys lasketaan kahdelle binääriselle tai bipolaariselle merkkijonolle vertailemalla niiden eroavaisuutta jokaiselle merkkijonon alkiolle [Fausett, 1994]. Esimerkiksi merkkijonojen ”10001110” ja ”00101100” etäisyys on kolme.

Hamming-etäisyys on valittu tässä tutkielmassa lähimmän naapurin menetelmän etäisyysmitaksi sen soveltuvuuden ja vaivattoman toteutuksen vuoksi. Hamming-etäisyyden kehitti Richard Hamming vuonna 1950 kehittäessään virheentarkistusmenetelmää telekommunikaatiojärjestelmään [Hamming, 1950].

6.4 Testaus ja luokittelijan käyttö uudella aineistolla

Kuten jo mainittu, lähimmän naapurin -menetelmä on ”laiska”, jolloin varsinainen ”oppiminen” tapahtuu vasta testausvaiheessa. Testausaineisto voidaan hankkia erottamalla opetusaineistosta tietty määrä havaintoja. Koska testausaineisto on irrotettu tunnetusta opetusaineistosta, luokittelijan laatijalla on tietämys siitä kuinka tämä testausdata tulisi luokitella oikein. Näin ollen testauksen jälkeen luokittelijan laatijalla on tieto siitä kuinka hyvin luokittelija suoritui ja näin ollen voi tarvittaessa esiprosessoida tai muuten kohentaa opetusaineistoa. Testausvaiheessa syöteaineiston yksittäiset alkiot käydään läpi ja jokaista verrataan opetusaineiston jokaiseen havaintoon käyttäen valittua etäisyysmittaa. Näin saadaan kullekin opetusaineiston havainnolle etäisyys. Kuitenkin tätä ennen on valittava arvo k, joka määrittää, kuinka monta lähintä havaintoa valitaan tarkasteluun. Parametrin k arvon valitsemille ei ole juurikaan kehitetty vallitsevaa ja hyväksi havaittua menetelmää [Hand et al., 2001] Toisin sanoen k:n mahdollisesti paras mahdollinen arvo on selvitettävä manuaaliseti kokeilemalla. Valitsemalla melko pieni k:n arvo voidaan välttyä poikkeavilta havainnoilta ja häiriöltä. Tämä voi kuitenkin johtaa yliopimiseen, jolloin menetelmä ei kykene yleistämään vaan oppii enneminkin yksittäisiä tapauksia [Larose, 2005]. Kuitenkin myös Larosen mukaan liian suuren k:n arvon valitseminen johtaa kiinnostavien datayksiköiden huomiotta jättämiseen. Tässä tutkielmassa k:n paras mahdollinen arvo selvitetään empiirisin tutkimusten perusteella. Kun k:n arvo on kiinnitetty voidaan aloittaa uuden aineiston testaus. Testaus tapahtuu seuraavasti:

1. Valitaan aineistosta luokittelematon yksikkö.

2. Lasketaan etäisyydet opetusjoukossa olevien yksiköiden ja valitun yksikön välillä.

3. Valitaan k kappaletta lähinnä olevia yksiköitä opetusjoukosta.

4. Valitaan joukosta luokittelu jonka esiintymä on suurin.

5. Asetetaan kohdassa 1. valitulle yksikölle edellisen kohdan perusteella luokittelu.

Kohdassa neljä tapahtuva valinta ei kuitenkaan aina ole täysin triviaali, sillä on mahdollisuus, jossa yhdenkään luokan esiintymä ei ole toisia suurempi. Näin ollen voidaan käyttää erilaisia menetelmiä luokan määrittelyksi. Yksinkertaisin tapa on

valita satunnainen luokka joukosta, mutta tämä ei kuitenkaan ole kovinkaan kehittynyt menetelmä. Menetelmää voidaan luonnehtia eräänlaiseksi äänestämiseksi, jossa jokaisella valitulla yksiköllä on käytössään yksi ääni, jolla se äänestää omaa luokitustaa. Tämä ei kuitenkaan välttämättä tuota tyydyttäviä tuloksia. Esimerkkinä on tilanne, jossa tarkasteltavaa yksikköä lähinnä olevalla yksiköllä on tietty luokitus, mutta kaksi tai useampaa kauempana olevaa yksikköä edustavat eri luokitusta kuin lähimänpänä oleva. Tässä tilanteessa uusi yksikkö saa luokituksen näiden etäisempien yksiköiden luokituksen perusteella vaikka ne ovatkin, ehkäpä jopa huomattavasti etäämmällä kuin lähimpänä oleva yksikkö. Tämän kaltaisessa tilanteessa saavutetaan parempia tuloksia laskemalla painoarvot jokaiselle joukon yksikölle etäisyyyden avulla ja näin ollen korostetaan kaikista lähimpänä olevia yksiköitä [Iltanen, 2010]. Lähimpänä oleville yksiköille annetaan suurimmat painoarvot, jotka toimivat kertoimina äänestyksessä. Painoarvokerroin voidaan laskea muun muassa seuravalla tavalla: On kuitenkin huomioitava, että tämä saattaa johtaa tilanteeseen jossa tarkasteltava yksikkö ja vertailtava yksikkö ovat identtiset, jolloin etäisyys olisi nolla. Tällöin matemaattisesti ajatellen painoarvon laskeminen olisi mahdotonta. Useasti on kuitenkin tapana tällöin merkitä nämä näiden yksiköiden painoarvo verrattain suureksi, jolloin on hyvin todennäköistä että uusi yksikkö luokitellaan niiden mukaisesti. Menetelmä myös vähentää tasapelitilanteiden määrää, jolloin satunnaisesta luokittelusta päästään eroon [Larose, 2005].

Havaintojen sisältäessä useita muuttujia on lähimmän naapurin menetelmää käytettäessä tarpeen määritellä kunkin muuttujan tarpeellisuus. Jotkin muuttujat saattavat olla täysin merkityksettömiä, kun taas toiset voivat hyvinkin merkittäviä. Muuttujien merkittävyys voidaan määritellä puhtaan aiheen tuntemuksen perusteella tai laskennallisesti ristiinvalidoimalla muuttujat. Käyttämällä asiantuntijuutta voidaan myös suodattaa pois muuttujia, jotka saattavat vääristää luokittelua. Näin lähimmän naapurin luokittelukykyä voidaan parantaa entisestään. Edellä on esitetty opetusvaiheen vievän aikaa O(n), mutta testausvaihe käyttää useamman aikayksikön. Testausvaiheessa uuden luokittelemattoman yksikön etäisyydet kaikkiin tietämyskannassa oleviin yksiköihin on selvitettävä. Tästä johtuu, että aikavaatimus on hieman korkeampi kuin O(mn), sillä jokaisen yksikön kohdalla täytyy suorittaa myös etäisyysmittaan perustuva laskenta. Tämä lisää testausvaiheen aikavaatimusta. Erittäin suurilla opetusjoukoilla tämä voi osoittautua melko hitaaksi toimeenpiteeksi.

7 Aineiston esiprosessointi ja kerääminen

Kuten muutkin tiedonlouhintamenetelmät, neuroverkot ja lähimmän naapurin menetelmät tarvitsevat aineistoa, jonka pohjalta ne kykenevät hahmottamaan tai luokittelemaan havaintoja uudesta ja tuntemattomasta aineistosta. Aineiston laadusta riippuen saattaa olla tarpeellista suorittaa esiprosessointia. Esiprosesoinnin ansiosta tiedonlouhinta aineistosta nopeutuu ja saadaan tarkempia tuloksia. Näin ollen on hyvin tärkeää häivyttää aineistosta poikkeavat havainnot ja korvata puuttuvat havainnot [Iltanen, 2010]. Aineiston esiprosessoinnilla voidaan aineistosta hävittää tarpeettomia havaintoja ja näin ollen pienentää aineiston kokoa, joka luonnollisesti vaikuttaa tiedonlohinta-algoritmien nopeuteen, mutta myös joissain tapauksissa menetelmän käyttämisellä voidaan saavuttaa tarkempia luokittelutuloksia. Tilastollisin menetelmin voidaan määritellä havaintojen muuttujien väliset riiipuvuudet ja näin ollen karsia tarpeettomia muuttujia aineistosta. Aineiston esiprosessointiin voidaan käyttää erilaisia menetelmiä, jotka soveltuvat kulloinkin kyseessä oleville havainnoille. Poikkeavien havaintojen paikantaminen on melko vaivatonta ja niiden negatiivista vaikutusta luokitteluun voidaan vaimentaa esimerkiksi aineiston normalisoinnilla.

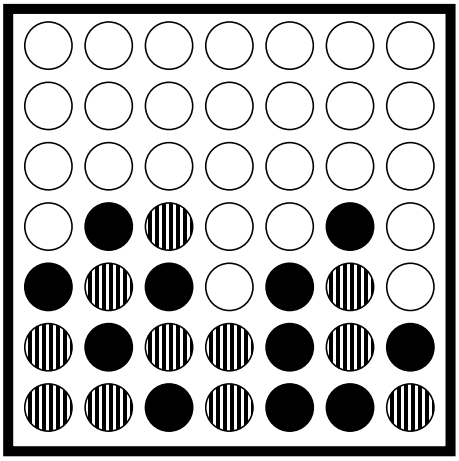
Tutkimukseni vaatii huomattavia määriä opetus- ja testausdataa, joten koin tarpeelliseksi kehittää tätä varten sovelluksen, joka kykenee generoimaan dataa ja tilastoimaan neuroverkon oppimista ja testausta. Sovellus on tehty hyödyntäen Microsoftin .NET-kehitysympäristöä graafisen käyttöliittymän toteuttamiseen ja AForge.Net-neuroverkkokirjastoa. Molemmat tuotteet ovat saatavilla ilmaiseksi ja ovat vapaasti verkosta ladattavissa. Datan generoiminen tapahtuu luomalla satunnaisia tilanteita pelialustalla, jonka jälkeen algoritmi määrittelee, mikä olisi optimaalinen seuraava siirto. Algoritmi hyvin yksinkertainen, sillä se huomioi ainoastaan yhden pelaajan siirrot ja pyrkii vain ja ainoastaan voittoon. Näin ollen se ei lainkaan pyri estämään vastapelaajan toimia.

Kuitenkin koin myös tarpeelliseksi generoida realistisia pelitilanteita joissa pyrin puolueettomasti pelaamaan itseäni vastaan siten että, molemmille tekoälytoteutuksille syntyisi riittävästi opetusdataa.

7.1 Opetus- ja testausaineiston hankkiminen

Molempien menetelmien käyttämä aineisto, niin opetus- ja testausvaiheessa, on generoitu kirjoittajan luomalla yksinkertaisella algoritmilla. Algoritmi kykenee havaitsemaan pelilaudan tilanteet ja päättelemään, miten tulisi seuraavaksi toi-

mia, jotta seuraava siirto veisi tilanteen lähemmäksi voittoa. Algoritmi on hyvin yksinkertainen ja naivi, sillä se ei ota lainkaan huomioon vastapelaajan toimia, vaan pyrkii voittavaan tilanteeseen ottamatta huomioon mahdollisia vastapelaajan, sitä estäviä, toimia. Myöskään algoritmi ei kykene havaitsemaan tulevia pelitilanteita ja toiminaan sen mukaisesti, toisin kuin ihmispelaaja. Tämä seikka ei kuitenkaan estä tutkimasta neuroverkon ja lähimmän naapurin menetelmän oppimista ja tulosten tarkastelua. Syötedata muodostetaan satunnaisista tilanteista,



Kuva 7.1 Eräs tilanne neljän suora -pelissä

josta esimerkki voidaan nähdä kuvassa 7.1 pelilaudalla ja tähän liittyvän tulosdatan algoritmi muodostaa tarkkailemalla tilannetta pelilaudalla ja päättelemällä, mikä olisi mahdollisimman suotuisa seuraava siirto. Kuitenkin jos tilanne on sen kaltainen, että mikään siirto ei ole merkittävän suotuisa, algoritmi valitsee satunnaisen siirron. Algoritmi päättelee siirron suotuisuuden tarkastelemalla siirtoa seuraavia tapahtumia. Seuraavat tapahtumat voidaan jakaa neljään kategoriaan:

1. Siirto tuottaa tilanteen, jossa pelaaja saa yhden pelimerkin jonoon. Näin ollen siirto pisteytetään nollan pisteen arvoiseksi.

2. Siirto tuottaa tilanteen, jossa pelaaja saa kaksi pelimerkkiä jonoon. Näin ollen siirto pisteytetään yhden pisteen arvoiseksi.

3. Siirto tuottaa tilanteen, jossa pelaaja saa kolme pelimerkkiä jonoon. Näin ollen siirto pisteytetään kahden pisteen arvoiseksi.

4. Siirto tuottaa tilanteen, jossa pelaajaa saa neljä pelimerkkiä jonoon. Näin ollen siirto pisteytetään kolmen pisteen arvoiseksi.

Algoritmi kokeilee sijoittaa uuden pelimerkin jokaiseen sarakkeeseen ja arvio tilanteen, jonka jälkeen jokaisen sarakkeen tilanne pisteytetään. Korkeimman pistemäärän saanut sarake valitaan seuraavaksi siirroksi. Mikäli kaikki sarakkeet saavat pisteytyksestä arvon nolla, valitaan satunnainen sarake, kun taas mikäli joku sarake saa pisteytyksestä arvon kolme, se valitaan välittömästi. Toisin sanoen algoritmi aloittaa sijoittamalla tyhjälle pelilaudalle ensimmäisen pelimerkin. Tyhjälle pelilaudalle mihin tahansa sarakkeeseen asetettu pelimerkki saa pisteytyksen nolla, algoritmi valitsee satunnaisen sarakkeen. Tätä jatketaan kunnes jompi kumpi lopetusehdoista täyttyy; pelilaudalle ei mahdu uusi pelimerkkejä tai toinen pelaajista saa neljä omaa pelimerkkiään jonoon. Edellisten ehtojen täyttyessä pelilauta tyhjennetään merkeistä ja algoritmi ryhtyy luomaan uusia tilanteita tyhjälle pelilaudalle. Tätä jatketaan kunnes on tarvittava määrä opetus- ja testausdataa on generoitu. Näin ollen opetusdataan muodostuu enemmän pelin alku- ja keskivaiheen tilanteita. Tämä saattaa osaltaan vaikuttaa neuroverkon oppimiseen, mutta toisaalta peli käsittää enemmän juuri näitä tilanteita kuin loppuvaiheen tilanteita. Kuten jo aiemmin on mainittu, menetelmä ei ole kovinkaan sofistikoitunut, saati älykäs. Kaikesta huolimattta algoritmi tuottaa hyvin kelvollista dataa neuroverkon ja lähimmän naapurin -menetelmän käyttöön.
Algoritmia voidaan soveltaa kuvassa 1 näkyvään tilanteeseen. Tässä tilanteessa mikäli seuraava asetettava pelimerkki olisi musta, se pisteytettäisiin seuraavasti:

1. sarake antaa yhden pisteen, koska siihen asetettava musta pelimerkki muodostaa kahden pelimerkin jonon.

2. sarake antaa myös yhden pisteen samasta syystä kuin edellinen sarake.

3. sarakkeeseen sovelletaan samoja sääntöjä kuin kahteen edeltävään sarakkeeseen.

4. sarake antaa kaksi pistettä, koska siihen asetettava musta pelimerkki muodostaa kolmen pelimerkin jonon.

5. sarake antaa kolme pistettä, koska siihen asetettava musta pelimerkki muodostaa neljän pelimerkin jonon. Algoritmi päättyy tähän sarakkeeseen, sillä kolme pistettä on paras mahdollinen tulos. Näin ollen algoritmi päättelee tämän olevan paras mahdollinen seuraava siirto. Tämä väite ei kuitenkaan pidä paikkaansa.

6. sarake antaa yhden pisteen samoista syistä kuin kaksi ensimmäistä saraketta.

7. sarake antaa myös yhden pisteen kuten edellinen sarake.

Opetus- ja testidata esitetään 42 alkion pituisena lukutaulukkona, jossa tyhjät kohdat ja pelaajien pelimerkit esitetään eri lukuarvoilla eri tutkimustapauksissa. Nämä tutkimustapaukset käsitellään luvussa ?Datan esiprosessointi?. Näin data saa yksinkertaisen muodon, joka soveltuu mainiosti niin neuroverkolle, kuin lähimmän naapurin -menetelmälle sovellettavaksi. Tulosdata esitetään neuroverkon tapauksessa seitsemän alkion pituisella taulukolla, jossa kukin alkio esittää yhtä saraketta. Alkio voi saada arvon väliltä [0, 1] ja suurimman arvon saanut alkio määrittää suotuisimman seuraavan liikkeen sarakkeen. Näin ollen edellä mainitulle algoritmin tuottama opetus- ja testausdata on helposti käytettävissä.
Syötedataa voidaan generoida vaihtelevia määriä, riippuen neuroverkon rakenteesta. Mutta useimmissa tapauksessa voidaan karkeana sääntönätarvittavan syötedatan määrä laskea kertomalla vapaiden muuttujien eli opetettavien kertoimieb määrä kymmenellä. Näin ollen syötedatan määrä riippuu neuroverkon piilokerroksien neuronien ja syötteiden lukumäärästä. Kuitenkin useimmissa tapauksissa piilokerroksia ei ole tarpeellista olla kuin yksi kappale, sillä pahimassa tapauksessa useamman piilokerroksen käyttäminen vain hidastaa ja jopa estää neuroverkon oppimista. Tutkimuksessa tutkitaan muutamia erilaisia neuroverkkorakenteita, jotta voidaan selvittää, onko verkon rakenteella ja opetusaineiston esitystavalla merkittäviä vaikutuksia.

7.2 Aineiston erilaiset esitystavat

Opetus- ja testausdata ovat molemmille menetelmille soveltuvassa muodossa jo valmiiksi, joten on syytä perehtyä datan varsinaiseen sisältöön. Koska erilaiset pelitilanteet esitetään käyttäen jokaiselle pelilaudan paikalle numeerista arvoa väliltä [-1, 1], voidaan sama data esittää usealla eri tavalla. Olen tähän tutkielmaan valinnut kuusi erilaista esitystapaa, jolloin voidaan tuloksista nähdä, onko datan esitystavalla juurikaan merkitystä tuloksiin. Samaa menetelmää on aikaisemmin hyödyntänyt muun muassa Darby Thompson omassa tutkielmassaan [Thompson, 2005]. Taulukosta 7.1 nähdään kuinka kuudella eri tavalla voidaan pelilaudan tilanteen koodata. Toisin sanoen syötedata ainoastaan vaihtaa muotoaan, mutta tulosdata pysyy samanlaisena kaikissa eri tavoissa. Näin pyritään

Musta merkki	Harmaa merkki	Tyhjä paikka
1	-1	0
1	0	-1
-1	1	0
-1	0	1
0	-1	1
0	1	0

Taulukko 7.1 Opetus- ja testausaineiston erilaiset esitystavat

varmistamaan, riippuuko tutkimustulokset datan esitystavasta. Sikäli neuroverkon tapauksessa tämä on hyvinkin tärkeä asia tarkisteltavaksi, sillä neuroverkon oppimiskyky riippuu syötedatan ja tulosdatan riippuvuuksista [Thompson, 2005]. Lähimmän naapurin menetelmään ei vaikuta datan esitysmuoto.
Erittäin merkittävää on myös käytetyn opetusaineiston määrä ja sen monipuolisuus. Monipuolisuus on hyvinkin tärkeä asia, jotta voidaan välttää ylioppiminen, mutta myös on tärkeää, ettei luokittelija muodostu liian monimutkaiseksi liiallisen monipuolisuuden takia. Sovelluksen tuottama opetusaineiston on hyvin satunnaista, jolloin monipuolisuus varmasti tulee hyvin esille, mutta varsinaisia reaalimaailman tilanteita se ei todennäköisesti sisällä niin paljon kuin tarpeellista olisi.

8 Työkalujen valinta

Tutkielman tekemiseen kuului myös datan generoimiseen, testaukseen ja raportointiin soveltuva ohjelmisto, joten on tarpeellista lyhyesti kuvata sovelluksen toteuttaminen, suunnittelu ja käytetyt menetelmät.

8.1 Suunnittelu ja arkkitehtuuri

Tekoälysovelluksen suunnitelun lähtökohtana voidaan pitää yksinkertaistettua malli-näkymä-ohjain-arkkitehtuuria. Arkkitehtuurin perusajatus on erottaa käyttöliittymän ja sen tapahtuman varsinaisesti sovelluslogiikasta. Toisin sanoen, mallin tehtävä on säilyttää sovelluksessa käytettyä sovellusdataa ja tarjota tätä dataa muokkaavat loogiset operaatiot [Koskimies ja Mikkonen., 2005]. Näkymän tehtävä on esittää tämä sovellusdata käyttöliittymässä ja kontrollerin tehtävä on toimia tiedon välittäjänä mallin ja käyttöliittymän välissä. Tutkielman tekoälysovelluksessa suunnitelumallia on hieman yksinkertaistettu siirtämällä kontrollerin tehtävä näkymälle, jolloin näkymän on suoraan käsiteltävä sovellusdataa mallin tarjoamien operaatioiden avulla. Tämä yksinkertaistus on tehty täysin tarkoituksenmukaisesti, sillä kyseessä on kohtalaisen pieni ja yksinkertainen sovellus, jolloin kontrollerin käyttö olisi tuottanut turhaa vaivaa.

8.2 Microsoft .NET

Microsoft .NET on Microsoftin ohjelmistokehys, joka tarjoaa työkalut graafisten sovellusten tekemiseen Microsoftin käyttöjärjestelmille. Ohjelmistokehys on vuosien kehitystyön tulos ja se tarjoaa nykyaikaisien ohjelmointiympäristöjen tavoin automaattisen roskienkeruun ja monia muita korkeantason työkaluja kuten esimerkiksi kattavat tietorakenteet. Ohjelmistokehys tarjoaa myös mahdollisuuden kehittää sovelluksia muutamilla erilaisilla ohjelmointikielillä. Näin ollen ohjelmistokehys tarjoaa verrattomat mahdollisuudet tehokkaaseen sovelluskehitykseen Microsoftin käyttöjärjestelmissä.

8.3 AForge.NET

AForge.NET on LGPL versio 3.0:n alainen avoimen lähdekoodin ohjelmistokehys joka sisältää työkaluja useisiin tehtäviin. AForge.NET on toteutettu Microsoftin .NET -ohjelmistokehyksen avulla ja se on suunnattu tutkijoille ja kehittäjille, jotka työskentelevät tekoälyä ja konenäköä vaativien haasteiden paris-

sa[AForge.NET, 2013]. Näin ollen ohjelmistokehys sisältää monipuolisia työkaluja muun muassa kuvanprosessoinnin, videoprosessoinnin, neuroverkkojen, geneettisten algoritmien ja koneoppimisen parissa työskenteleville tutkijoille ja kehittäjille. Kuitenkin tässä tutkielmassa käytetään ainoastaan ohjelmistokehyksen tarjoamia neuroverkkotyökaluja. Ohjelmistokehyksen neuroverkkotyökalut kattavat taaksepäinlevittävän neuroverkon, delta-säännön neuroverkon, perceptron neuroverkon, itseohjautuvat neuroverkot sekä muutamia muita neuroverkkoja. Näin ollen AForge.NETin työkalujen määrä on varsin vaikuttava jo pelkästään neuroverkkojen osalta.

8.4 Lähimmän naapurin menetelmä

Lähimmän naapurin menetelmän käyttöön en löytänyt vaivattomasti käyttöönotettavaa ohjelmakirjastoa, jolloin ainoaksi vaihtoehdoksi jäi suunnitella ja laatia oma toteutus. Lähimmän naapurin menetelmän ollessa kohtalaisen yksinkertainen luokittelumenetelmä jäi varsinaiseksi haasteeksi etäisyysmitan suunnittelu ja toteutus. Tarkoitukseni oli pyrkiä suunnittelemaan ja toteuttamaan lähimmän naapurin menetelmä hyvin itsenäiseksi kokonaisuudeksi ja yleiseksi ratkaisuksi, jolloin se olisi voitu irroittaa sovelluksesta ja hyödyntää muissa vastaavissa tilanteissa. Ratkaisun yleistäminen koitui liian isoksi haasteeksi ja se oli vastakkain tutkielman alkuperäisiä tarkoitusperiä, joten jouduin hylkäämään tämän ajatuksen. Yleistyksen puuttuminen ei kuitenkaan vaikuta sovelluksen toimintaan millään tavoin.

9 Testaus ja tutkimus

Tässä luvussa esitellään ja analysoidaan erikokoisien neuroverkkojen opetuksen yhteydessä saadut tulokset. Opetuksen aikana tarkasteltiin keskiarvovirheen kehitystä ja muutosta. Opetuksen aikana myös neuroverkon luokittelukykyä tarkasteltiin testaamalla sitä opetusaineistosta erillisellä testausaineistolla. Koska tutkimuksessa käytettiin kuutta erikokoista neuroverkkoa, oli viisainta jakaa ne kahteen ryhmään: pienikokoiset ja suurikokoiset neuroverkot. Pienikokoiset neuroverkot käsittävät verkkorakenteet joissa olivat 5, 15 ja 25 piiloneuronia sisältäneet neuroverkot. Ja suurikokoiset verkkorakenteet käsittivät 35, 45 ja 55 piiloneuronia sisältäneet neuroverkot. Kaikki neuroverkot opetettiin opetuskertoimella 0.01. Opetusaineisto koostui 2000 pelitilanteesta ja testiaineisto 200 pelitilanteesta. Kutakin neuroverkkoa opetettiin 20000 epookin verran, joka mahdollisti opetuksen tarkastelun isommassa aikaikkunassa kuin verkon opetuksen kannalta oli tarpeen.

Opetuksen aikana taltioitiin keskivirheen kehitys ja muutos sekä kuinka virheettömästi neuroverkko suoritui testiaineistosta opetuksen aikana. Tätä dataa läpikäydessäni tarkoitukseni oli suodattaa tästä edellä mainitusta neuroverkkoparvesta parhaiten menestyvä neuroverkko. Menestymisen määrittelyssä käytin neuroverkon kykyä luokitella testiaineisto mahdollisimman hyvin, mutta myös tarkkailin keskivirheen muutosta jotta voisin huomata missä vaiheessa verkon oppiminen hidastuu siinä määrin että opetus voidaan keskeyttää. Oppimisen hitauden määrittelin kuten Järvelin[2006] ja siten valitsin epookkien välisen keskiarvovirheen muutoksen raja-arvoksi 0.05%. Taulukosta 9.1 voidaan nähdä minkä epookin kohdalla kukin neuroverkko saavutti tämän asetetun raja-arvon.

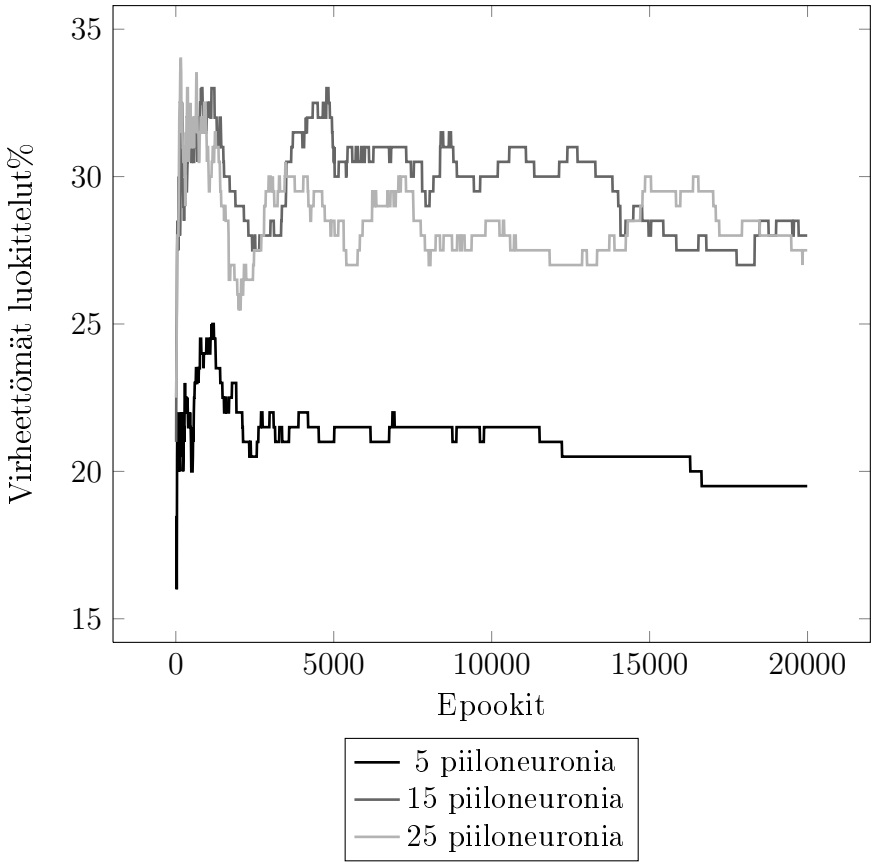
Piiloneuronit	Epookit	Testaustarkkuus
5	1220	24.5%
15	1450	31.0%
25	1880	26.5%
35	3180	35.0%
45	4740	38.5%
55	3790	33.0%

Taulukko 9.1 Erikokoiset neuroverkot, opetusepookkien määrät ja testaustarkkuus opetusepookkien lopussa

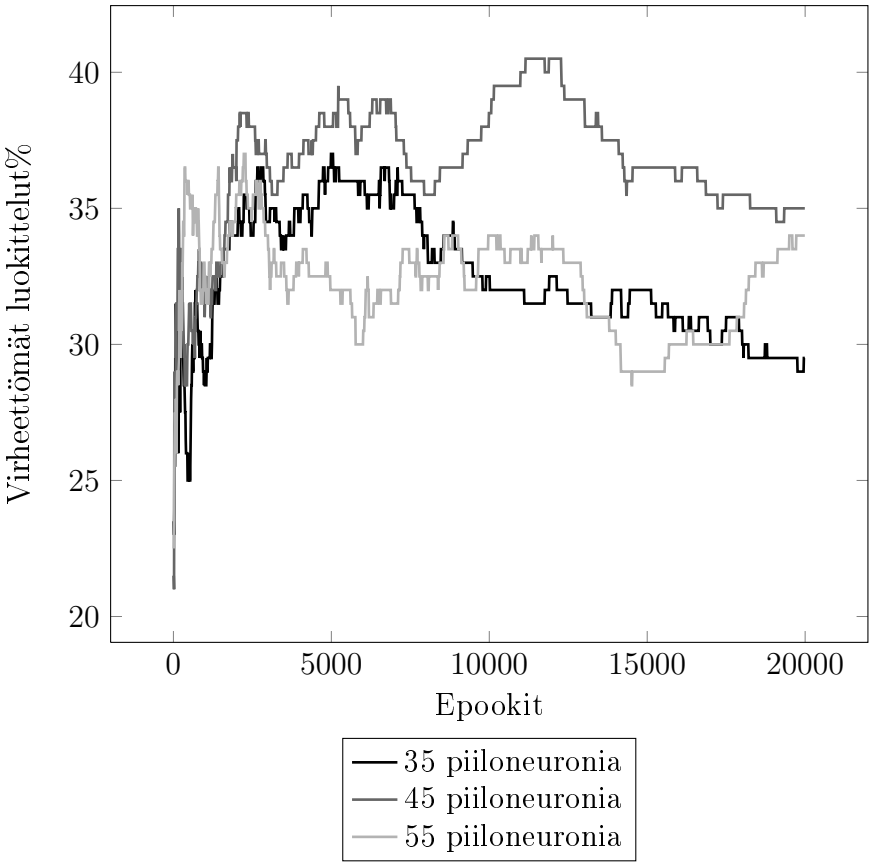
Aineiston tutkimalla oli siis mahdollista löytää kullekkin neuroverkolle optimaalinen määrä opetusepookkeja, mutta kuvaajista 9.1 ja 9.2 voidaan huomata muutamia erikoisia ja mielenkiintoisia tilanteita. Edellä mainitut kuvaajat kertovat kuinka neuroverkko kehittyi opetuksen aikana luokittelemaan testausaineiston täsmällisesti. Esimerkiksi kuvaajasta 9.1 voidaan huomata kuinka pienin, 5 piiloneuronin, verkko taltio opetusaineiston tietämyksen melko nopeasti, mutta samalla tavalla nopeasti sen kyky luokitella testausaineisto suppenee. Eikä se myöskään parhaimmillaankaan saavuta kovinkaan hyvää luokittelutulosta. Ehkäpä vieläkin kiinnostavampaa on huomata että hieman isommat verkot, 15 ja 25 piiloneuronia, saavuttavat parhaimmillaan jo huomattavasti paremmat tulokset, mutta myös että 25 piiloneuronin verkko saavuttaa hyvin varhaisessa vaiheessa parhaan tuloksensa ja sen jälkeen hieman taantuu ja taas palautuu paremmalle tasolle jolloin myös oppimisen nopeus tasaantuu. 25 piiloneuronin verkon opetuksessa on havaittavissa huomattavasti samankaltaisia piirteitä.

Isompien neuroverkkojen tapauksessa voidaan kuvaajasta 9.2 havaita että, verkoilla kestää hieman enemmän aikaa saavuttaa riittävän hyviä tuloksia testiaineiston luokittelussa. Kaavion perusteella isompien verkkojen oppimisprosessi on aavistuksen vähemmän mielenkiintoinen kuin pienempien verkkojen. Kuitenkin eräs varsin mielenkiintoinen ilmiö tapahtuu 45 piiloneuronin opetuksen yhteydessä: verkko saavuttaa parhaan tarkkuutensa keskiarvovirheen muutoksen perusteella 4740 epookin kohdalla, mutta kuvaajasta voidaan varsin helposti huomata että mikäli verkkoa opetetaankin noin 12000 epookkia, niin se saavuttaa vieläkin paremmin tarkkuuden. Tämä ilmiö kitetyttää neuroverkkojen opetuksen keskeyttämisen määritelemisen hankaluuden, sillä kuten sanottu keskiarvovirheen muutoksen mukaan opetus voitaisiin keskeyttää jo varhaisemmassa vaiheessa kuin olisi tarpeen. Näin ollen voidaan tulkita että, jotkut verkkorakenteet oppivat parhammaimman tarkkuutensa vain näennäisesti ja samalla hidastavat oppimistaan vaikka myöhemmässä vaiheessa tilanne voisi olla hyvin toinen.

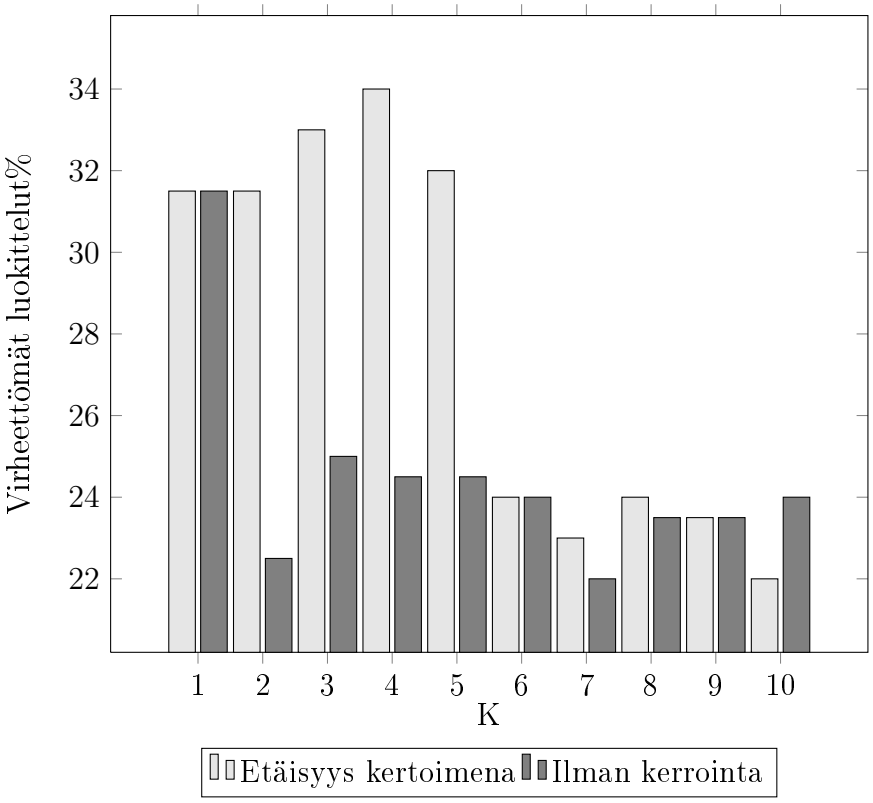
Kuitenkaan mikään neuroverkkoparven neuroverkoista ei saavuta kovinkaan mullistavan hyvää luokittelukykyä, parhaimmillaankin 45 piiloneuronin neuroverkko luokittelee oikein noin 40%. Toisessa ääripäässä 5 piiloneuronin neuroverkko saavuttaa parhaimmillaan 23% tuloksen luokittelussa. Lähimmän naapurin menetelmä kykenee parhaimmillaan luokittelemaan testiaineiston oikein noin 34% tarkkuudella. Huonoimmillaan lähimmän naapurin menetelmä saavuttaa vain noin 22% tarkkuuden, kuten kuvaajasta 9.3 nähdään. Näin ollen voidaan olettaa että, neuroverkkot ja lähimmän naapurin menetelmät ovat melko tavalla yhtä hyvin sisäistäneet opetusaineiston. Testitulokset osoittavat tutkimuksen ongelman ole-



Kuva 9.1 Pienikokoiset neuroverkot oppivat 20000 epookkia oppimisparametrina 0.01



Kuva 9.2 Suurikokoiset neuroverkot oppivat 20000 epookkia oppimisparametrina 0.01



Kuva 9.3 Lähimmän naapurin testaus

van melko vaikea koneoppimisalgoritmeille, kun neuroverkot ja lähimmän naapurin menetelmä tuottivat parhaimmillaankin vain 40% tarkkuuden.

10 Yhteenveto

Näiden tutkimustulosten varjossa voidaan melko varmasti sanoa että, neuroverkko ja lähimmän naapurin menetelmä eivät sovellu neljän suoran kaltaisen pelin tekoälyratkaisuksi kaikista parhaiten. Huomattavasti parempia tuloksia voitaisiin saavuttaa isommalla määrällä opetusaineistoa ja opetusaineiston karsimisella. Karsiminen täytyisi kohdistaa siten että, aineistosta hävitetään redundantteja pelitilanteita. Redundantteja pelitilanteita ovat liian samanlaiset pelitilanteet. Huomattavan paljon niitä esiintyy aloitustilanteiden joukossa. Erityisesti lähimmän naapurin menetelmä kärsii liian suuresta määrästä samanlaisia tilanteita, jolloin se ei kykene löytämään aineistosta riittävästi erilaisia, mutta samalla lähellä olevia pelitilanteita joista se voisi muodostaa tyydyttävän ratkaisun. Myös neuroverkolle nämä pelitilanteet ovat turmiollisia, sillä neuroverkko saattaa ylioppia ne ja näin hävittää kykynsä yleistää nämä tilanteet. Todellisten pelattujen pelien kautta saatu opetusaineisto oli tutkimuksen kannalta aivan liian pieni, mutta oli tutkimuksen puitteissa aivan liian aikaa vievää ja työlästä hankkia lisää aineistoa. Ja luonnollisesti aineistoa olisi pitänyt hankkia useilta eri henkilöiltä. Näin olisi voitu saada opetusaineistoa erilaisista pelityyleistä, joka varmasti olisi ollut hyödyllistä molempien menetelmien kannalta.

Neljän suoran luonteen tuntien olisi varmasti ollut tulosten kannalta parempi käyttää jotakin täysin muuta menetelmää. Shakissa ja muutamissa muissa klassisissa lautapeleissä on useasti käytetty päätöspuihin nojautuvia ratkaisuja jotka etsivät kussakin tilanteessa edullisimman tai parhaan reitin pelitilanteiden parvessa. Tämän kaltainen menetelmä olisi varmasti myös neljän suoralle otollisin mahdollinen teköälyratkaisu. Neuroverkko ja lähimmän naapurin menetelmä suorittavat päättelynsä aina silloisen pelitilanteen mukaan ja pyrkivät valitsemaan parhaan mahdollisen seuraavan siirron, joka ei välttämättä ole lainkaan laajemmalla aikaikkunalla paras vaihtoehto. Peliä pitäisi pystyä ennakoimaan muutamia siirtoja eteenpäin ja siten valitsemaan paras mahdollinen siirto. Luonnollisesti molempia menetelmiä voidaan käyttää tähän, mutta näkisin, että olisi varmasti tehokkaampaa kehittää matemaattinen esitys jolla paras siirto voitaisiin pistetyttää. Lisäksi mielenkiintoisia tuloksia voitaisiin saavuttaa yhdistämällä nämä menetelmät, jolloin niitä voitaisiin käyttää esimerkiksi satunnaisesti vuorotellen tai valitsemalla jollain heuristiikalla tilanteeseen sopivampi menetelmä. Kuitenkin tutkimus toi minulle henkilökohtaisesti paljon uutta tietämystä neuroverkkojen ja lähimmän naapurin menetelmän opettamisesta ja soveltamisesta käytännön ongelmiin. Pyrin jatkossakin keskittämään ajatukseni neuroverkkojen tutkimiseen

ja erityisesti itseoppivien neuroverkkojen alueeseen.

Viiteluettelo

[Barcelos Tronto et al., 2008] Iris Fabiana de Barcelos Tronto, José Demísio Simões da Silva and Nilson Sant' Anna. *An investigation of artificial neural networks based prediction systems in software project management.* The Journal of Systems and Software, 81, 356-367, 2008.

[Fausett, 1994] Laurene Fausett. *Fundamentals of Neural Networks.* Prentice Hall, New Jersey, 1994.

[Hamming, 1950] Richard Hamming. *Error detecting and error codes.* The Bell System Technical Journal, 14, 2 (April. 1950).

[Hand et al., 2001] David Hand, Heikki Mannila and Padhraic Smyth. *Principles of Data Mining.* MIT Press, 2001.

[Haykin, 1994] Simon Haykin. *Neural Networks: A Comprehensive Foundation.* Macmillian College Publishing Company, Inc, 1994.

[Iltanen, 2010] Kati Iltanen. *Tietämyksen muodostaminen, Kurssimateriaali.* Tietojenkäsittelytieteen laitos, Tampereen yliopisto, 2010.

[Jang et al, 1997] Jyh-Shing Roger Jang, Chuen-Tsai Sun and Eiji Mizutani. *Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence.* Prentice Hall. 1997).

[Järvelin, 2008] Antti Järvelin. *Neurolaskenta, Kurssimateriaali.* Tietojenkäsittelytieteen laitos, Tampereen yliopisto, 2008.

[Koskimies ja Mikkonen, 2005] Kai Koskimies, Tommi Mikkonen. *Ohjelmistoarkkitehtuurit.* Talentum Media, 2005.

[Larose, 2005] Daniel T. Larose. *Discovering Knowledge in Data: An Introduction to Data Mining.* Wiley, 2005.

[Sejnowski and Rosenberg, 1999] Terrence J. Sejnowski, Charles R. Rosenberg. *NETtalk: a parallel network that learns to read aloud.* The Johns Hopkins University Electrical Engineering and Computer Science Technical Report, 1986.

[Thompson, 2005] Darby Thompson. *Teaching a Neural Network to Play Konane. Draft.* Spring, 2005. Available as http://cs.brynmawr.edu/Theses/Thompson.pdf.