Neuroverkon hyödyntäminen Texas Hold'em limit -pelaajan seuraavan toiminnon ennustamisessa

Joona Tuominen

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Tietojenkäsittelyoppi
Pro gradu -tutkielma
Ohjaaja: Martti Juhola
Maaliskuu 2008

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Tietojenkäsittelyoppi
Joona Tuominen: Neuroverkon hyödyntäminen Texas Hold'em limit -pelaajan seuraavan toiminnon ennustamisessa
Pro gradu -tutkielma, 54 sivua
Maaliskuu 2008

Texas Hold'emissa, kuten muissakin pokerin varianteissa, yksi tärkeä menestyksekkään pelin osatekijä on vastustajan toiminnan ennakointi. Tutkin tässä työssä, miten luotettavaa tietoa opetettu neuroverkko voi antaa Texas Hold'em -pokeria pelaavan henkilön seuraavasta toiminnosta.

Neuroverkkoja on hyödynnetty laajasti erilaisten pelien tutkimuksessa, ja johdantoluvussa referoin muutamia esimerkkejä tällaisista tutkimuksista. Useimmat neuroverkkosovellukset keskittyvät vuoropohjaisiin ja matemaattiseen sekä loogiseen päättelykykyyn perustuviin peleihin, kuten pokeriin ja shakkiin. Neuroverkkojen oppimiskykyä voidaan hyödyntää kuitenkin myös esimerkiksi videopeleissä.

Johdannon jälkeen käyn lyhyesti läpi Texas Hold'em -pokerin säännöt, minkä jälkeen kerron perustiedot neuroverkoista ja niiden toiminnasta. Lisäksi käsittelen neuroverkkojen opettamista ja opettamisessa usein käytettyä takaisinlevitysalgoritmia (back-propagation algorithm). Esittelen myös muita neuroverkkomalleja itse käyttämäni ohella kootakseni laajemman yleisesityksen neuroverkoista.

Seuraavassa vaiheessa käsittelen neuroverkon käytännön toteutusta ja kerron, miten olen toteuttanut oman neuroverkkoni ja käsitellyt siinä käytettävää dataa. Käsittelen myös datankäsittelyyn liittyviä yleisiä ohjeita ja metodeja.

Seuraavaksi kerron neuroverkon opettamisesta, validoinnista ja testaamisesta, sekä tuloksista ja tulkinnoista, joihin neuroverkon opetuksen aikana päädyin. Olen testannut neuroverkkoani 87 erilaisella datajoukolla pyrkien kaiken aikaa parempaan tulokseen, ja lopulta työn tuloksista nähdään, että muodostamani neuroverkko osaa ratkaista oikein hieman yli puolet testattavista tapauksista, mikä on hyvin lähellä ennakkoon asettamaani tavoitetta.

Tulosten purkamisen yhteydessä pohdin myös, miten olisin voinut päästä parempaan tulokseen, ja minkälaisia jatkotutkimuksen aiheita neuroverkot erityisesti pokerin yhteydessä tarjoavat.

Avainsanat ja -sanonnat: tekoäly, neuroverkko, pokeri

Sisällys

Määritelmät ja lyhenteet

bluffata: Pokerissa bluffaaminen tarkoittaa sitä, että pelaaja yrittää saada vastustajan uskomaan oman kätensä olevan parempi kuin mitä se todellisuudessa on.

flop: Kolme ensimmäistä pöytään jaettavaa korttia Texas Hold'em -pelissä. Tässä työssä flopilla tarkoitetaan jaon aikana toisena tapahtuvaa pelikierrosta, jolla em. kortit jaetaan.

jakajan nappi: Vastaa englanninkielistä termiä button ja tarkoittaa merkkiä, joka osoittaa, kuka on jakovuorossa. Jakovuoro (ja samalla myös jakajan nappi) kiertävät pöytää myötäpäivään siten, että jakajaksi vaihtuu joka jaolla pöydässä seuraavana istuva pelaaja.

korottaa: Korotus vastaa tässä työssä englanninkielistä pokeritermiä raise ja merkitsee vuorossa olevan pelaajan tekemää korotusta. Korotus on mahdollinen tilanteessa, jossa joku edeltävistä pelaajista on panostanut.

livepokeri: Pokeri, jota pelataan kanssapelaajien kanssa fyysisesti samassa tilassa (yleensä saman pöydän ääressä). Vrt. nettipokeri.

luovuttaa: Luovuttaminen vastaa tässä työssä englanninkielistä pokeritermiä fold ja merkitsee tilannetta, jossa vuorossa oleva pelaaja luopuu kädestään kesken jaon.

maksaa: Maksu vastaa tässä työssä englanninkielistä pokeritermiä call ja merkitsee tilannetta, jossa vuorossa oleva pelaaja maksaa ennen häntä tehdyn panostuksen ja mahdolliset korotukset pysyäkseen jaossa mukana.

nettipokeri: Pokeri, jota pelataan internetin välityksellä virtuaalisissa pelihuoneissa. Vrt. livepokeri.

panostaa: Panostus vastaa tässä työssä englanninkielistä pokeritermiä bet ja merkitsee vuorossa olevan pelaajan tekemää panostusta. Panostus on mahdollinen tilanteessa, jossa kukaan aikaisempi pelaaja ei ole vielä panostanut kyseisellä kierroksella.

pelikierros: Texas Hold'em -pokerissa on jaon aikana neljä erillistä pelikierrosta: floppia edeltävä kierros (pre-flop), flop-kierros, turn-kierros ja river-kierros. Jokaisen

jaon aikana ei välttämättä pelata jokaista pelikierrosta, mutta ne pelataan aina edellä esitetyssä järjestyksessä.

pimeä kortti: Kts. taskukortti.

positio: Positiolla tarkoitetaan tässä työssä pelaajan istumapaikkaa pöydässä suhteessa jakajan nappiin. Positio on sitä parempi, mitä kauempana pelaaja on jakajan napista myötäpäivään laskettuna. Jakajan napin kohdalla pelaavalla on paras positio.

potti: Pottiin kuuluvat ne kaikki pelimerkit, jotka pelaajat ovat laittaneet pelipöytään panostaessaan, korottaessaan tai maksaessaan panostuksen tai korotuksen. Jaon voittaja voittaa jaon päätyttyä potin.

pre-flop: Jaon aikana ensimmäisenä pelattava pelikierros, jolla jaetaan jokaiselle pelaajalle kaksi taskukorttia.

river: Viides pöytään jaettava kortti Texas Hold'em -pelissä. Tässä työssä riverillä tarkoitetaan jaon aikana neljäntenä tapahtuvaa pelikierrosta, jolla em. kortti jaetaan.

showdown: Taskukorttien avaaminen ja jaon voittajan ratkaiseminen siinä vaiheessa, kun kaikki pelikierrokset on pelattu ja river-kierroksen panostukset ja korotukset suoritettu. Jos pelikierroksia ei pelata loppuun (kaikki pelaajat yhtä lukuun ottamatta ovat luopuneet kädestään pelin aikaisemmassa vaiheessa), showdownia ei suoriteta.

sokkopanos: Ennen jaon aloittamista jakajasta seuraava pelaaja asettaa pöytään pienen sokkopanoksen (small blind) ja pienestä sokkopanoksesta seuraava pelaaja asettaa ison sokkopanoksen (big blind). Nämä panokset toimivat pelin alussa ikään kuin panostuksena ja korotuksena, jotka seuraavien pelaajien pitää maksaa osallistuakseen floppiin.

sököttää: Sököttäminen vastaa tässä työssä englanninkielistä pokeritermiä check ja merkitsee tilannetta, jossa ennen vuorossa olevaa pelaajaa ei ole panostettu tai korotettu pelikierroksen aikana, eikä myöskään vuorossa oleva pelaaja panosta tai korota.

taskukortti: Pre-flop-kierroksella pelin alussa jokaiselle pelaajalla jaetaan kaksi taskukorttia, joita muut pelaajat eivät näe. Näiden taskukorttien ja pöytään jaettavien korttien avulla jokainen pelaaja muodostaa oman pelikätensä.

turn: Neljäs pöytään jaettava kortti Texas Hold’em -pelissä. Tässä työssä turnilla tarkoitetaan jaon aikana kolmantena tapahtuvaa pelikierrosta, jolla em. kortti jaetaan.

väri: Väri on pokerikäsi, jossa kaikki pelaajan kortit ovat samaa maata (herttaa, pataa, ristiä tai ruutua). Texas Hold’em -pokerissa viiden kortin käsi muodostetaan kahden taskukortin ja viiden pöytäkortin avulla.

1. Johdanto

Pokeri ja etenkin sen variantti Texas Hold'em on viime vuosina kasvattanut huimasti suosiotaan ympäri maailman – myös Suomessa. Osaltaan tätä suosion kasvua selittävät lukuisat internetissä toimivat pelipalvelut, joista suurin osa tarjoaa muiden pelien ohessa myös mahdollisuuden pelata pokeria verkon kautta ympäri maailman. Lisäksi Texas Hold'em on nopea peli oppia, mutta se tarjoaa kuitenkin haasteita myös kehittyneemmille pelaajille.

Vaikka Texas Hold'em muiden pokeripelien tavoin perustuukin paljolti matematiikkaan ja todennäköisyyksien sekä tilastojen hallintaan, myös vastapelaajien lukemisella on pelissä oma tärkeä roolinsa. Pelattaessa livepokeria toisten kanssa samassa tilassa pelaajia voidaan tulkita heidän käyttäytymisensä, eleidensä ja ilmeidensä kautta, mikä ei ole mahdollista internetin välityksellä pelattaessa. Nettipelissä pelaajien tulkitsemisen tulee perustua heidän toimintaansa erilaisissa pelitilanteissa, koska vastaavaa vastapelaajan fyysisen läsnäolon mukanaan tuomaa informaatiota kuin livetilanteessa ei ole käytettävissä.

Kehittyessään ja oppiessaan peliä pelaaja yleensä oppii lukemaan vastapelaajia paremmin heidän toimintojensa perusteella. Pokeri on kuitenkin hyvin monitahoinen peli, koska ammattimaiset pelaajat voivat käyttää hyvinkin vaihtelevaa pelityyliä ja olla taitavia bluffaamaan. Aina ei ole helppoa lukea vastustajaa tai muistaa kaikkea pelin aikana esiin tullutta informaatiota, ja lisäksi käsityksen muodostamista vaikeuttaa pelissä esiintyvien muuttujien suuri määrä. Tämän vuoksi olen rakentanut ja opettanut neuroverkon, joka pyrkii ennustamaan vastapelaajan seuraavaa toimintoa. Tässä työssä käyn läpi neuroverkon suunnittelun ja rakentamisen sekä sen antamat tulokset. Vastustajan seuraavan toiminnon ennustaminen voi tuoda pelaajalle huomattavaa etua, ja on mielenkiintoista nähdä, kuinka luotettavasti neuroverkko pystyy tämän tekemään.

Viisi mahdollista pelaajan toimintoa ovat luovutus, sököttäminen, maksu, panostus ja korotus. Keskityn tässä työssä tarkastelemaan pelaajien reagointia panostuksiin ja korotuksiin, joten toiminnoista työn kannalta relevantteja ovat luovutus, maksu ja korotus.

Tekoälyä ja neuroverkkoja on käytetty apuna erilaisissa peleissä paitsi oman oikeanlaisen toiminnan vahvistamisessa, myös kanssapelaajien toiminnan ennustamisessa. Aiheesta on tehty jonkin verran aikaisempia tutkimuksia, ja referoin niitä myöhemmin johdannossa.

Toisessa luvussa käyn lyhyesti läpi Texas Hold'em-pelin säännöt. Kolmannessa luvussa kerron yleisesti neuroverkoista ja niiden toiminnasta. Luvussa 4 käsittelen yleisesti neuroverkkosysteemin toteutusta. Luvussa 5 kuvailen käyttämäni datan käsittelyä. Luvussa 6 kerron oman neuroverkkoni käytännön opettamisesta ja

testaamisesta. Testaamisessa hyödynnän kymmenkertaista ristiinvalidointia, jolloin testijoukkona käytetään osuutta opetusjoukon datasta. Luvussa 7 käsittelen tulokset ja teen yhteenvedon.

1.1. Neuroverkkojen käyttö vastustajan mallintamiseen Texas Hold'em -pelissä

Omassa käytännön toteutuksessani olen käyttänyt lähteenä pääasiassa Davidsonin [1999] työtä. Davidson on käyttänyt tutkimuksessaan 19 syötearvoa, monikerroksista perceptronia (multi-layer perceptron, MLP) sekä takaisinlevitysalgoritmia. Kaikki painokertoimet saavat alkuarvokseen satunnaisluvun väliltä [-0,5, 0,5]. [Davidson, 1999]

Davidson käyttää opetuskertoimena tutkimuksessaan joko vakioarvoa 0,4 tai keskiarvovirhettä riippuen siitä, kumpi on pienempi. Näin hän uskoo välttävänsä liian rajut heilahdukset neuroverkon opetuksen aikana, jolloin suppeneminen kohti oikeita arvoja tapahtuu tasaisemmin. [Davidson, 1999]

Välttääkseen juuttumista paikallisiin ääriarvoihin (local peaks) Davidson käyttää momenttia pieniä esteitä varten. Tämän lisäksi hän on kehittänyt oman tekniikkansa, jossa hän tarkkailee, onko opetus mahdollisesti jumittunut, ja jos näin käy, hän lähettää neuroverkkoon signaalin, joka säätää painoarvoja satunnaisesti hyvin pieniä määriä. Davidson on havainnut tämän toimivaksi tavaksi estää opetuksen jumiutuminen ääriarvokohdissa. [Davidson, 1999]

Neuroverkon tarkkuuden Davidson määrittelee testijoukon avulla. Jos parhaan tuloksen saanut tulosneuroni vastaa oikeaa tulosta, luokittelua pidetään onnistuneena. Virhe neuroverkossa on laskettu halutun tuloksen ja saavutetun tuloksen välisen keskivirheen neliönä. Davidsonin neuroverkko osasi vain 25 epookin eli opetusiteraation jälkeen ennustaa 85 % testidatan esiintymistä oikein, mitä voidaan pitää varsin hyvänä tuloksena. [Davidson, 1999]

Suurimpia Davidsonin kohtaamista ongelmista oli nopeasti pelityyliään vaihtavien pelaajien toimintojen mallintamisen vaikeus, sillä he vaihtelevat tyyliään juuri siitä syystä, että heidän toimiensa ennakoiminen olisi vaikeampaa. Ratkaisuehdotukseksi tähän ongelmaan Davidson ehdottaa useamman yhtäaikaisen neuroverkon käyttöä saman pelaajan ennustamisessa. Nämä neuroverkot opetettaisiin erilaisella pelaajaan liittyvällä datalla käyttäen eri datamääriä ja ajallisesti eri hetkenä kerättyä dataa. [Davidson, 1999]

Davidson ottaa jatkotutkimuksen mahdollisuuksista ensimmäisenä esiin reaaliaikaisen internetissä toimivan mallintamissysteemin rakentamisen. Lisäksi Davidson on sitä mieltä, että hänen oma tutkimuksensa olisi voinut antaa parempia tuloksia, jos hän olisi käyttänyt enemmän kuin 19 muuttujaa. Hän myöntää kuitenkin, että mitä suurempaa määrää muuttujia käytetään, sitä enemmän on käytettävä aikaa muuttujien käsittelyyn, jotta se vahvistaisi neuroverkon ennustamiskykyä. Lisäksi Davidson suunnittelee erilaisten neuroverkkorakenteiden ja -tekniikoiden käyttöä

tarkoituksenaan saavuttaa parempia tuloksia. Loppupäätelmänä Davidson toteaa lyhyesti, että neuroverkot ovat osoittautuneet toimivaksi tavaksi mallintaa pokerinpelaajia. [Davidson, 1999]

1.2. Bayesilainen pokeri

Korb ja muut [1999] ovat tutkineet bayesialaisen verkon (Bayesian network) avulla muodostetun ohjelman käyttöä ohjelman oman pokerikäden, vastustajien käsien sekä vastustajien käyttäytymisen tulkinnassa. Kyseisessä tutkimuksessa on tutkittu viiden kortin avopokeria, joka eroaa Texas Hold'emista pelinä melko paljon, mutta jossa on tutkimuksellisesti paljon yhteisiä piirteitä: yritetään ennustaa asioita puutteellisen informaation avulla samankaltaisessa pelissä. [Korb et al., 1999]

Viiden kortin avopokerissa on yhteensä neljä pelikierrosta jakoa kohti, ja Korb ja muut käyttävät erillistä bayesilaista verkkoa jokaisella kierroksella. Kukin verkko mallintaa suhteet nykyisen käsityypin, lopullisen käsityypin ja vastustajan käyttäytymisen välillä. Verkko palauttaa yhden totuusarvon, joka kertoo sen, tuleeko verkkoa käyttävä ohjelma voittamaan pelin. Nykyinen käsityyppi mallinnetaan verkkoa käyttävän ohjelman osalta kokonaan, mutta koska vastustajilla on kyseisessä pelityypissä yksi taskukortti, vastustajien nykyistä käsityyppiä ei voi mallintaa kokonaan ennen showdownia. [Korb et al., 1999]

Korb ja muut käyttivät alkuperäisessä verkossaan 9 käsityypin jaottelua verrattaessa ohjelman ja pelaajien käsityyppejä. Tällä ei kuitenkaan päästy kovinkaan hyviin tuloksiin, minkä vuoksi käsityyppien määrää nostettiin 17:ään. Tutkittaessa pokeripelejä pelinaikaisten käsien perusteella ongelmaksi muodostuu juuri erilaisten käsien valtava määrä. Tietynlaista luokittelua voidaan tehdä, mutta jos esimerkiksi sekä ohjelmalla että pelaajalla on kädessään pari, ei ole lainkaan yhdentekevää, minkä arvoinen pari on. Lisäksi, jos molemmilla on samanarvoinen pari, seuraavaksi suurin kortti eli hai ratkaisee pelin voittajan. Tutkimuksissa näitä käsiä kuitenkin luokitellaan yleensä melko pelkistetysti, koska liian suuri muuttujien määrä vaikeuttaa oikean tuloksen muodostamista. Korb ja muut ovat helpottaneet tätä ongelmaa kartoittamalla eri käsien muodostumistodennäköisyyksiä pelin eri vaiheissa jakamalla 10 miljoonaa esimerkkikättä ja keräämällä näistä saadun datan talteen toimintamatriiseihin (action matrices). [Korb et al., 1999]

Korb ja muut ovat muodostaneet verkkoa käyttävälle ohjelmalleen tietynlaiset funktiot, joilla päätetään ohjelman toiminnot vastustajia vastaan. Kyseisten funktioiden kaavoissa on huomioitu pelattava kierros sekä voiton todennäköisyys, joka on laskettu em. toimintamatriisien avulla. Funktioita on testattu ja pyritty tarkentamaan stokastisen haun (stochastic search) avulla pelauttamalla ohjelmaa sääntöperustaista vastustajaa vastaan. Edellä mainittujen funktioiden käyttöön ohjelmassa liittyy myös jonkin verran satunnaisuutta, jolla on pyritty saamaan ohjelma bluffaamaan silloin tällöin. [Korb et al., 1999]

Bayesilaista verkkoa testattiin kahta automaattista vastustajaa vastaan: yksinkertaista todennäköisyyksien perusteella valintansa tekevää vastustajaa ja sääntöpohjaista vastustajaa vastaan. Testaus osoitti, että bayesilainen verkko päihittää nämä automatisoidut vastustajat helposti. Verkkoa testattiin myös verkkopeliin kutsuttuja ihmisvastustajia vastaan, jolloin bayesilaista verkkoa käyttävä ohjelma jäi toiseksi. Ihmisvastustajat onnistuivat kuitenkin keskimäärin voittamaan vain n. 63 panosyksikköä 450 peliä kohti, mikä ei ole kovin paljon. Korb ja kumppanit [1999] uskovatkin, että tietyin parannuksin heidän järjestelmänsä oppii päihittämään myös ihmisvastustajat.

1.3. Tietokonevastustajien kehittäminen yksinkertaistettuun pokeripeliin

Barone ja While [1998] ovat tutkineet evolutiivista tapaa saada ohjelma oppimaan yksinkertaistettua pokeria. Evolutiivisessa prosessissa yritetään sopeutua vallitsevaan, todennäköisesti kaiken aikaa muuttuvaan ympäristöön.

Barone ja While käyttävät autonomisia agentteja suorittamaan tiettyjä osatehtäviä oppimisprosessista perustuen tiettyihin pokerin pelaamisen tärkeisiin periaatteisiin. He osoittavat työssään, että nämä kehittyvät agentit muodostavat erilaisia tekniikoita reagoida monenlaisiin pelistrategioihin, joita niiden vastustajat (pelaajat) käyttävät, maksimoidakseen voittonsa kutakin vastustajatyyppiä vastaan. [Barone and While, 1998]

Geneettiset algoritmit käyttävät tietokoneissa evoluution toimintaperiaatteita luonnonvalinnan kautta tapahtuvaan ongelmienratkaisuun. Ratkaisuehdotusten populaatio kehittyy jonkin valintamekanismin vaikutuksesta, kunnes tietyt ratkaisuun vaadittavat kriteerit täyttyvät. Evolutiiviset algoritmit mallintavat ongelman ulkoisten piirteiden mukaan geneettisen materiaalin asemesta, ja riippumatta esiintyvistä geneettisistä muodonmuutoksista ulkoisissa piirteissä tapahtuva muutos seuraa Gaussin käyrää keskiarvovirheellä nolla ja jollain tietyllä keskihajonnalla. [Barone and While, 1998]

Pattie Maes on määritellyt agentin järjestelmäksi, jonka edellytetään täyttävän useita, usein monimutkaisia tavoitteita muuttuvassa ympäristössä. Agentit tunnistavat ympäristöään erilaisin sensorein ja vaikuttavat ympäristöönsä nk. aktuaattorien (actuator) kautta. Agentit jaetaan usein pienempiin komponentteihin, joista kukin hallitsee pientä, erillistä aliongelmaa. Näiden komponenttien keskinäinen kanssakäyminen antaa agentille persoonallisuuden ja kyvyn ratkaista useita tavoitteita. [Barone and While, 1998]

Maes on määritellyt kaksi asiaa, jotka pitää ratkaista suunniteltaessa autonomisia agentteja: 1) Kokemuksesta oppimisen ongelma: agenttien pitää pysyä tasapainossa nykyisen parhaan ratkaisun hyödyntämisen ja uusien, mahdollisesti parempien ratkaisujen välillä. 2) Toiminnan valinnan ongelma: agenttien täytyy päättää, kuinka täytetään useat, usein keskenään ristiriidassa olevat tavoitteet. Agentit tarvitsevat

päätösmekanismin käsittelemään ristiriitaisia toimia eri toimijakomponenteilta. [Barone and While, 1998]

Barone ja While lähestyvät pokerin pelaamiseen liittyvää ongelmaa siten, että tietokonepelaajalle haetaan vahvuutta muiden pelaajien eri pelityyleille ominaisista heikkouksista. He esittelevät työssään uuden viitekehyksen mukautuvan pokerinpelaajan suunnitteluun upottamalla implisiittiset evolutiivisen algoritmin oppimisen erityispiirteet itsenäiseen agenttiarkkitehtuuriin kehittääkseen konepelaajan, joka paljastaa vastustajien pelityylien heikkoudet ja maksimoi näin voittonsa. [Barone and While, 1998]

Barone ja While ratkovat edellä mainituista kahdesta perusongelmasta ensimmäistä, kokemuksesta oppimisen ongelmaa, käyttämällä pokeriagenttien populaatiota sekä evolutiivista (1 + 1) strategiaa oppimismekanismina. Kyseinen strategia toimii siten, että käytetään yhtä vanhempaa generoimaan yksi jälkeläinen seuraavan sukupolven muodostuessa vanhempien ja jälkeläisten yhdistetystä populaatiosta. [Barone and While, 1998]

Toista perusongelmaa, eli toiminnan valinnan ongelmaa, joka tässä tutkimuksessa merkitsee valintaa pokeritoimintojen luovutus, maksu ja korotus väliltä, Barone ja While ratkovat suunnittelemalla eri tavalla päteviä itsenäisiä komponentteja, joista jokainen vastaa yhtä pokeripelin perusperiaatetta. [Barone and While, 1998]

Kolme pokerin perusperiaatetta, joita Barone ja While käyttävät kunkin erillisen komponentin pätevyysalueena, ovat *käden vahvuusanalyysi*, *positio* sekä *riskien hallinta*. Lisäksi he käyttävät nk. ratkaisijaa (resolver), joka kuuntelee jokaista erillistä komponenttia, yhdistelee niiden ehdotukset ja päättää agentin globaalin aktuaattoritoiminnon. Ratkaisija asettaa tietyt painotukset jokaiselle ehdotukselle ja pyrkii kehittämään näitä painotuksia, jolloin saadaan aikaiseksi agentti, jonka yksittäiset komponentit sekä niiden suhteellinen merkitys kehittyvät ajan kuluessa. [Barone and While, 1998]

Jokainen komponentti tuo tietonsa ratkaisijalle panostusmonikkoina (betting tuple). Käden vahvuudesta vastaava komponentti voisi tuoda esimerkiksi melko vahvalla kädellä panostusmonikon (5 %, 10 %, 85 %), jossa 5 % tarkoittaa luovuttamistodennäköisyyttä, ja 10 % sekä 85 % vastaavasti maksun ja korotuksen todennäköisyyksiä. Ratkaisija käyttää näitä monikkoja sekä painotuksia määritellessään globaalin aktuaattoritoiminnon. [Barone and While, 1998]

Barone ja While peluuttivat agenttejaan kahdessa 10 hengen pöydässä, joissa molemmissa oli 5 pokeriagenttia ja 5 pelaajaa. Pöydät oli jaoteltu pelaajien karkean pelitapajaon mukaisesti käyttäen pelaajamäärittelyjä passiivinen / aggressiivinen sekä löysä / tiukka. Taulukosta 1 on nähtävissä em. pelaajatyyppien ristiintaulukointi siten, kuin Barone ja While sen esittävät. Testipöytien pelaajat olivat nk. löysässä pöydässä löysiä ja aggressiivisia, ja nk. tiukassa pöydässä tiukkoja ja passiivisia. Pokeriagenteilla

oli näissä pöydissä tavoitteena keskittyä pöydissä esiintyvien pelaajatyyppien heikkouksien paljastamiseen ja sitä kautta voittojen maksimoimiseen. [Barone and While, 1998]

	Löysä	**Tiukka**
Passiivinen	Pelaaja, joka yliarvioi kätensä arvon, mutta korottaa harvoin peläten suuria potteja.	Pelaaja, joka pelaa harvoja käsiä (yleensä vain kun kädellä on suuri voittomahdollisuus), mutta korottaa harvoin peläten suuria potteja.
Aggressiivinen	Pelaaja, joka yliarvioi kätensä arvon, ja joka korottaa usein kasvattaakseen mahdollisia voittojaan.	Pelaaja, joka pelaa harvoja käsiä (yleensä vain kun kädellä on suuri voittomahdollisuus), ja joka korottaa usein kasvattaakseen mahdollisia voittojaan.

Taulukko 1 [Barone and While, 1998]. Pokerin pelaajien luokittelu.

Barone ja While tutkivat hyvin yksinkertaistettua mallia pokerista. Peli oli korttien perusteella vastaava kuin Texas Hold'em (jokaisella on kaksi taskukorttia ja viisi yhteistä korttia, joista muodostetaan yksi viiden kortin käsi), mutta tutkimuksessa käsiteltiin vain viimeistä kierrosta, riveriä, jolloin osittaisen käden arvioiminen voitiin jättää pois laskuista. Evolutiivisessa prosessissa uuden sukupolven tuotantovaiheessa käytettiin aluksi Gaussin käyrän poikkeamaa 10. Tätä poikkeamaa vähennettiin ajan myötä. [Barone and While, 1998]

Löysässä pöydässä kehittyvien agenttien keskimääräinen pelimerkkien palautus ei ollut koskaan negatiivinen 25 sukupolven jälkeen. Pokeriagentit kehittyivät satunnaisesta tilasta palauttamaan tuottoa n. 2500 jaetun käden jälkeen. 40 sukupolven jälkeen kasvuvauhti hidastui huomattavasti, mutta yleinen nouseva trendi tuloksissa jatkui aina 500 sukupolveen asti. [Barone and While, 1998]

Tiukassa pöydässä saatiin jokseenkin samankaltaisia tuloksia kuin löysässä pöydässä. Aluksi agentit menestyivät huonosti, mutta ne kehittyivät ajan myötä. Voitot jäivät tässä pöydässä kuitenkin selvästi pienemmiksi, mikä on odotuksenmukaista tiukassa pöydässä. [Barone and While, 1998]

Barone ja While tutkivat myös komponentti kerrallaan agentin toimintaa tiukassa pöydässä saadakseen selville ratkaisijan asettamat painotukset eri komponenteille. Lopulliset painotukset antoivat käden voimakkuudelle 85 % painotuksen, positiolle 4 % painotuksen ja riskienhallinnalle 11 % painotuksen. Odottamattoman suurta painoarvoa käden voimakkuudelle he perustelivat sillä, että testaus oli rajoitettu yhteen pelikierrokseen. Useamman pelikierroksen käyttäminen olisi tarkoittanut sitä, että käden

vahvuutta arvioidaan vajaalla tiedolla, mikä olisi tuottanut agenteille vahvemman valintapaineen riskienhallinnan ja position suhteen. Löysässä pöydässä painotukset olivat samankaltaiset mutta vieläkin selkeämmin käden vahvuutta painottavat 95 % osuudella. [Barone and While, 1998]

Kolmas kokeilu, jota Barone ja While agenteillaan tuottivat, oli eri agenttien (löysässä pöydässä kehitetyn, tiukassa pöydässä kehitetyn ja kehittämättömän agentin) pelauttaminen eri pöydissä kuin missä niitä alun perin pelautettiin. Nämä kokeilut varmistivat odotetun tuloksen siitä, että löysien pelaajien heikkoudet on helpompi paljastaa kuin tiukkojen pelaajien. Lisäksi nähtiin, että tietynlaisessa pöydässä kehitetty agentti osasi pelata omassa pöydässään paremmin kuin toisentyyppisessä pöydässä. [Barone and While, 1998]

1.4. Vahvat strategiat ja vastastrategiat: mestaruustason tietokone-pokerinpelaajan rakentaminen

Johanson [2007] on tutkinut ammattilaistason pokerinpelaajan rakentamista tietokoneelle. Johanson tutkii neljää eri lähestymistapaa tehokkaan pokeriohjelman rakentamiseen: 1) Nashin tasapainostrategiaa, 2) vastustajan pelitapaan perustuvaa vastastrategiaa, 3) kahden edellisen kompromissia, vahvaa vastastrategiaa ja 4) usean aiemmin muodostetun agentin käyttöä joukkueena ja käytettävän agentin valintaa pelin aikana opitun perusteella. Tutkimuksessa tuloksena kehitettyjä ohjelmia on käytetty AAAI Computer Poker Competition -tapahtumassa kilpailemaan ihmispelaajia vastaan ensimmäisessä ihminen-kone-pokerikilpailussa.

Johanson tekee työnsä aikana vertailua jo tunnettujen tekniikoiden ja ohjelmien kanssa suhteessa omaan työhönsä. Hänen Nashin tasapainostrategiaan perustuva kokeilunsa antoi hyvät tulokset siitä syystä, että hän onnistui luomaan uusia tasapainostrategioita ratkaisemalla suuremman tasapainojoukon kuin aiemmin on ollut mahdollista. Näiden uusien strategioiden ansiosta hän onnistui peittoamaan kaikki aiemmin tunnetut pokeriagentit. Tässä lähestymistavassa Johansonilla oli yhtenä lähtökohtana luoda ohjelma, jolla olisi mahdollisimman vähän hyväksikäytettäviä ominaisuuksia sen sijaan, että ohjelma olisi aktiivisesti etsinyt muiden pelaajien heikkouksia. Vaikka tällä lähestymistavalla pystytään luomaan ohjelma, joka kykenee voittamaan useimmat vastustajat olemalla vaikeasti hyväksikäytettävä, siitä löydettiin myös heikkous: voitot jäävät pieniksi, koska siinä ei yritetä käyttää hyväksi vastustajien heikkouksia. [Johanson, 2007]

Vastastrategian luomisessa hankaluutena on erityisesti vastustajasta luodun mallin mahdollinen virheellisyys tai se, että vastustaja saattaa vaihtaa pelityyliään kesken kaiken. Tavoitteena Johansonilla olikin luoda vahva vastastrategia, joka kykenee käyttämään hyväkseen tiettyä vastustajaa tai vastustajien luokkaa samalla, kun pyritään pienentämään vastustajien mahdollisuutta hyödyntää ohjelman mahdollisia heikkouksia. Näin ohjelma kykenee voittamaan tietyn tyyppiset vastustajat, mutta ei kuitenkaan häviä

paljoa niitäkään vastustajatyyppejä vastaan, joita ei ole erikseen otettu huomioon ohjelman toiminnassa. [Johanson, 2007]

Edellisten lähestymistapojen yhdistelmä perustuu siihen, että vastustaja voidaan mallintaa riittävän tarkasti erilaisten vastastrategioiden avulla ja valita sen perusteella oikeanlainen vahva strategia, jota käytetään. Johanson onnistui luomaan yhdistelmän, joka pystyy käyttämään vastastrategioita melkein yhtä tehokkaasti kuin lähestymistavassa 2 heikentämättä kuitenkaan liiaksi lähestymistavan 1 vahvuutta yleisellä tasolla. Tämä saatiin aikaan asettamalla tietynlaisia rajoituksia lähestymistavan 1 huonoimman suorituskyvyn tuottamiin tilanteisiin. [Johanson, 2007]

Neljäs lähestymistapa on otettu mukaan siksi, että lähestymistavoissa 1-3 käytetyt tekniikat olettavat kukin vastustajasta jotain, kun taas kilpailuvastustajan, jota vastaan lopullista työtä ollaan testaamassa, pelitapoja ei tunneta. Näin ollen oikean lähestymistavan valinta etukäteen on mahdotonta. Vahvaa agenttia (lähestymistapa 1) käytetäänkin nyt muodostamaan joukkue ja käytetään meta-agenttia, joka valitsee, mitä lähestymistapaa kunkin käden kohdalla hyödynnetään. [Johanson, 2007]

Johansonin ohjelmat osallistuivat ensimmäiseen ihminen-kone -pokerikilpailuun, jossa useat pokeriagentit pelasivat kahta menestynyttä ihmispelaajaa, Phil Laakia ja Ali Eslamia vastaan. Paitsi että molemmat pelaajat ovat pokeriammattilaisia, heillä on molemmilla myös melko vahva tekninen tausta, josta on heille hyötyä ohjelmallisia pokerinpelaajia vastaan pelatessa. [Johanson, 2007]

Tulos tulkittiin joko koneen tai ihmisen voitoksi, jos tietyn pelimäärän jälkeen jompi kumpi osapuoli oli vähintään 25 pienen panostuksen verran edellä. Yhteensä neljästä erillisestä ottelusta Johansonin kehittämät koneet hävisivät 3 ottelua ja pelasivat yhden tasapelin. Kilpailun tulos oli siis selkeästi voitto ihmisille. [Johanson, 2007]

Kilpailussa seurattiin panosten jakautumisen lisäksi DIVAT-arvoa, joka pyrkii analysoimaan taito-osuutta ja poistamaan tuurielementin peleistä. Tätä arvoa analysoimalla Johansonin pokeriohjelmat olisivat pärjänneet paljon paremmin, voittaen kolme neljästä pelistä ja pelaten yhden tasapelin. Kahta omalla tyylillään pelaavaa pelaajaa vastaan pelaaminen sekä lopulta melko vähäinen käsien määrä heikensi Johansonin ohjelmien suoritusta. Johanson kuitenkin uskoi saaneensa riittävästi lisäinformaatiota ja jatkotutkimuksen kohteita, jotta vastaavissa kilpailuissa pystyttäisiin käyttämään tulevaisuudessa vielä tehokkaampia ja voitokkaampia ohjelmia. [Johanson, 2007]

1.5. Neuroverkkoagenttien kehittäminen NERO-videopelissä

Neuroverkkoja voidaan hyödyntää myös muissa peleissä kuin pokerissa. Pokerin ohella suosittuja sovellusaloja ovat shakki, go ja muut rajattuihin siirtoihin ja vuoroihin pohjautuvat pelit, mutta myös monipuolisemmilla ja epätäsmällisemmillä toimintavaihtoehdoilla toimivissa peleissä ja sovelluksissa voidaan hyödyntää neuroverkkoja.

Useimmissa nykyaikaisissa tietokonepeleissä pelaajan kohtaamat ei-pelaajahahmojen (non-player-character) eli tietokoneohjattujen hahmojen heikkoudet toistuvat aina uudestaan. Peli alkaa tuntua nopeasti tylsältä vastustajien tehdessä joka kerta samat liikkeet. Jos nämä hahmot voisivat oppia pelaajan toimintojen perusteella kehittämään käyttäytymistään tiettyyn suuntaan, peli pysyisi pidempään kiinnostavana. Stanley ja muut [2005] esittelevät työssään rt-NEAT-metodin (real-time NeuroEvolution of Augmenting Topologies), jolla voidaan kehittää monimutkaisia neuroverkkoja reaaliaikaisesti pelin pelaamisen aikana. Tämän konseptin esittelemiseksi pelaaja opettaa robottien joukkoa taistelemaan NeuroEvolving Robotic Operatives -pelissä (myöhemmin NERO).

Paitsi että tietokonepelit ovat suosittu viihdemuoto, ne tarjoavat myös oivan testiympäristön tekoälysovelluksille. Yksi perusongelma ei-pelaajahahmojen – joihin tässä myöhemmin viitataan sanalla agentti – opettamisessa on se, että pelin ennustettavuus ja ohjattavuus kärsii agenttien toiminnan muuttuessa. Yksi tapa lähestyä tätä ongelmaa on opettaa agentit pelin kehitysvaiheessa ja siirtää ne valmiiksi opetettuina mukaan lopullisen peliin. Tällöin agentit eivät kuitenkaan pysty enää muokkaamaan käytöstään tietyn pelaajan toiminnan perusteella. [Stanley et al., 2005]

Koska NEROn pitää oppia pelaamisen aikana, ohjattu oppiminen, kuten esimerkiksi takaisinlevitysalgoritmin käyttö tai päätöspuuoppiminen (decision tree learning), ei tule kyseeseen. Stanley ja muut listaavat viisi videopelien ominaisuutta, jotka vaikuttavat ratkaisevasti käytettävän oppimistekniikan valintaan: 1) suuri tila- ja toimintoavaruus, 2) eri agenttien eriävät käyttäytymismallit, 3) yksittäisen agentin toiminnan pysyvyys, 4) nopea omaksumiskyky, 5) aikaisempien tilojen muistaminen. [Stanley et al., 2005]

Edellä esitettyjen ominaisuuksien vuoksi Stanley ja muut hylkäävät perinteisen vahvistusoppimisen (reinforcement learning) tekniikat, kuten Q-oppimisen (Q-learning) ja tila-toiminto-palkkio-tila-toiminto -algoritmin (State-Action-Reward-State-Action algorithm, SARSA algorithm) NEROn opettamisessa, ja käyttävät niiden asemesta neuroevoluutiota (NeuroEvolution, NE), jolla edellä mainitut videopelien ominaisuudet pystytään toteuttamaan paremmin. [Stanley et al., 2005]

Suurin haaste on neuroevoluution käyttö reaaliaikaisesti pelin aikana. Stanley ja muut pitävät rtNEATia kuitenkin soveltuvana ratkaisuna tästä huolimatta. NEAT käyttää useita neuroevoluution metodeita opetukseen, ja koska se laajentaa etsintäavaruutta (search space) vain silloin kun se on hyödyllistä, se pystyy löytämään merkittävästi monimutkaisempia kontrollirakenteita kuin ennalta määritellyllä topologialla toimiva evoluutio. [Stanley et al., 2005]

NEAT perustuu kolmeen pääajatukseen: 1) Verkkorakenteen kehittäminen edellyttää joustavaa geneettistä koodausta. Jokaisessa genomissa on lista yhteysgeeneistä, jotka viittaavat kahteen yhdistettyyn solmugeeniin. Jokainen yhteysgeeni määrittää syötesolmun, tulossolmun, yhteyspainon ja innovaationumeron,

joka auttaa löytämään vastaavat geenit risteytyksen aikana. Mutaatiot voivat muuttaa painoarvoja tai neuroverkon rakennetta, kuten missä tahansa neuroevoluutiojärjestelmässä. 2) NEATissa jokainen yksilö kilpailee omassa evolutiivisessa lokerossaan koko populaatiossa kilpailemisen asemesta. Siten topologisilla innovaatioilla on aikaa optimoida rakenteensa, ennen kuin aletaan kilpailla muiden evolutiivisten lokeroiden kanssa. NEAT käyttää lisääntymisstrategiaa nimeltä eksplisiittinen sopivuusjako (explicit fitness sharing), jossa jokainen evolutiivisen lokeron sisällä esiintyvä yksilö hallitsee osuutta ko. lokeron evolutiivisesta sopivuudesta, eikä yksilö pääse ottamaan valtaa populaatiosta. 3) NEATissa lähtötilanne on yksinkertainen verkko, jossa ei ole piiloneuroneja. Uusia rakenteita esitetään sitä mukaa kun niitä mutaatioiden myötä syntyy, ja vain sopivuusarvioinnissa hyödyllisiksi todetut rakenteet selviävät. Näin NEAT tutkii minimimäärän painosuhteita ja löytää sopivan kompleksisuusasteen ongelmalle. [Stanley et al., 2005]

Reaaliaikainen evoluutio pelin aikana toteutetaan siten, että huonoiten toimiva yksilö korvataan parhaiden yksilöiden jälkeläisellä. Näin vältytään liian suurilta yhtäkkisiltä muutoksilta, ja tätä hienovaraista kehitystä voidaan jatkaa läpi koko pelin. Liian nuoria agentteja ei poisteta, sillä kehitystä ei tapahtuisi, jos uudet, kehittymättömät agentit poistettaisiin heti luomisen jälkeen. [Stanley et al., 2005]

NEROssa pelaaja opettaa robottien armeijaa, joka ei osaa aluksi mitään. Opetuksen tuloksia voidaan testata pelaamalla toista pelaajaa vastaan, minkä jälkeen pelaaja pyrkii luomaan oikeanlaisia opetusharjoituksia kehittääkseen heikoiksi havaittuja ominaisuuksia. Pelaaja luo harjoitukset asettamalla objekteja kentälle ja määrittelemällä tavoitteet liukuvalikoilla (slider). Erilaisia objektivaihtoehtoja ovat staattiset viholliset, vihollisen tykkitornit, liikkuvat tykkitornit sekä seinät. Liukuvalikoilla taas määritellään, kuinka paljon agenttia palkitaan tai rangaistaan mm. vihollisten lähestymisestä, kohteisiin osumisesta, ammutuksi tulemisesta, ystävien seuraamisesta ja tuhoutumisesta. Kaikkia näitä mitataan samalla asteikolla, ja agentin sopivuus lasketaan näiden komponenttien summana. [Stanley et al., 2005]

Pelin roboteilla on monenlaisia sensoreita, joilla ne mittaavat mm. muiden objektien läheisyyttä ja niiden toimintaa. Opetuksen aikana poistettavat robotit menevät pelikentälle sijoitettuun tehtaaseen, josta palaa seuraavan sukupolven robotti korvaamaan poistetun. Opetuksen alussa kentällä on 50 robottia, joita kontrolloidaan neuroverkolla, jossa on satunnaiset painoarvot eikä lainkaan piiloneuroneja. Robotit kehittyvät tästä lähtöasetelmasta nopeasti, ja pelaaja voi tallentaa opetustilanteita sekä testata robottiensa osaamista muiden pelaajien kouluttamia robotteja vastaan taistelumoodissa. [Stanley et al., 2005]

Taistelumoodissa kumpikin pelaaja käyttää 20 robottia, jotka voi olla koottu eri opetusryhmistä. Robotit taistelevat keskenään, kunnes toisen joukkueen robotit on tuhottu tai jäljellä olevat robotit eivät enää taistele. Se pelaaja, jolla on suurempi määrä

robotteja jäljellä, voittaa pelin. Taistelukenttä voi olla pelkkä tyhjä tila, tai se voi sisältää muureja tai muita esteitä. Stanleyn ja muiden testauksessa on käytetty tyhjää kenttää. [Stanley et al., 2005]

Yksi perustaktiikka NEROssa on etsiä vihollinen ja ampua tätä. Tämän ominaisuuden opetusta varten kentälle asetettiin yksi liikkumaton vihollinen, ja robottia palkittiin vihollisen lähestymisestä. Koska robotit katsovat tehtaasta kentälle tullessaan satunnaiseen suuntaan, 90 %:lla roboteista kului keskimäärin 99,7 sekuntia oppia lähestymään vihollista onnistuneesti. Robotteja opetettiin myös välttelemään vihollista. Pelaaja ohjasi vihollista peliohjaimella, ja robotti sai rangaistuksen, jos oli liian lähellä vihollista. Mielenkiintoinen havainto oli, että robotti pyrki tällöin vihollisen selustaan, jotta voisi tulla palkituksi vihollisen ampumisesta mutta ei saisi rangaistusta päästäessään vihollisen liian lähelle. Robotit saatiin myös väistämään tykkitulta palkitsemalla niitä vihollisen lähestymisestä ja rankaisemalla ammutuksi tulemisesta. Lisäksi robotit oppivat mm. suunnistamaan useiden seinien muodostamissa labyrinteissa sekä valitsemaan turvallisimman reitin halutun kohteen luokse. [Stanley et al., 2005]

Stanley ja muut huomasivat myös, että jotkin robottijoukkueet pärjäsivät tietynlaisia vastustajia vastaan paremmin, toisenlaisia vastaan huonommin. Haasteeksi muotoutuikin opettaa robottijoukko, joka pärjäisi kaikille vastustajatyypeille. Yksi parhaista robottijoukkueista opetettiin tarkkailemalla ilmiöitä, jotka toistuivat usein taisteluissa. Takaa-ajot päätyivät usein siihen, että vihollinen oli selkä seinää vasten kentän reunassa. Tällöin ei ollut mahdollista toteuttaa aiemmin hyväksi havaittua taktiikkaa ja kiertää vastustajan selän taakse. Robotit onnistuivat kuitenkin lähestymään seinää vasten olevia, kenttää kohti tähtääviä tykkejä silloin, kun ne ampuivat toiseen suuntaan. Vastaavasti robotit oppivat vetäytymään silloin, kun tykit ampuivat robotteja kohti. Tämä joukkue voitti ensimmäisen NERO-turnauksen käyttämällä edellä kuvattua strategiaa. [Stanley et al., 2005]

Ensimmäisen NERO-turnauksen osanottajat olivat yhtä mieltä siitä, että peli oli viihdyttävä. Ensimmäisen NERO-prototyypin, joka valmistui vuonna 2004, menestys tarjoaa potentiaalia kaupallisiin tuotteisiin nykyaikaisissa peleissä. Erityisen hyvin kyseinen tekniikka toimisi pitkään jatkuvissa peleissä, kuten massiivisissa, verkossa pelattavissa moninpeleissä (Massive Multiplayer Online Games, MMOGs), jotka kestävät kuukausia tai vuosia, ja joissa rtNEAT ehtii omaksua ja optimoida agenttien käyttäytymistä pitkäjaksoisesti. [Stanley et al., 2005]

2. Texas Hold'em

Texas Hold'em on tällä hetkellä suosituin kaikista pokerin lukuisista varianteista Pohjois-Amerikassa ja Euroopassa [PokerStars.com, 2008]. Osaltaan Texas Hold'emin suosiota selittää se, että sitä pelataan myös suurissa TV-turnauksissa, kuten esimerkiksi World Poker Tourissa. Lisäksi internet on tuonut jokaiselle yhteyden haltijalle mahdollisuuden pelata pokeria verkossa, ja suuri pelaajakunta on innostunut nimenomaan Texas Hold'em -peleistä.

Käyn tässä luvussa lyhyesti läpi Texas Hold'em -pokerivariantin säännöt. Oletan pokerin perussäännöt, esimerkiksi käsien arvojärjestyksen, tunnetuksi ja käyn läpi kohdassa 2.1. pelin aloituksen ja ensimmäisen kierroksen toiminnan, sillä se eroaa hieman myöhemmästä pelistä. Kohdassa 2.2. kerron loput pelin kulusta. Keskityn tässä kertomaan fixed limit (lyhyemmin yleensä vain limit) -pelistä, jossa korotusten määrää on rajoitettu. Tutkin tässä työssä nimenomaan limit-pelejä, koska niin kutsutuissa no limit -peleissä panostuksen tai korotuksen arvoa tai niiden määrää ei ole rajattu, mikä muuttaa pelin luonnetta ratkaisevasti. No limit -peliä on vaikeampi ennustaa paitsi sen vuoksi, että korotuksen arvo -muuttuja saa vaihtelevampia arvoja, myös sen vuoksi, että pelitaktiikat ovat moninaisempia verrattuna limit-peliin.

2.1. Pelin aloitus, pre-flop-kierros

Ennen jakamisen aloittamista jakajasta seuraavana istuva pelaaja asettaa pöytään ns. pienen sokkopanoksen, ja tästä seuraava pelaaja (myötäpäivään) ns. ison sokkopanoksen. Iso sokkopanos on yleensä kaksinkertainen pieneen sokkopanokseen verrattuna, ja näiden sokkopanosten – samoin kuin myöhemmin pelissä tapahtuvien panostusten ja korotusten – arvo on aina pelikohtaisesti määritelty. Jakovuoro ja sokkopanokset kiertävät myötäpäivään siten, että kukin pelaaja pöydässä joutuu vuorollaan laittamaan peliin ensin ison ja seuraavalla jaolla pienen sokkopanoksen.

Kun sokkopanokset on asetettu, jokaiselle pelaajalle jaetaan kaksi korttia pimeänä eli siten, että muut pelaajat eivät näe kyseisiä kortteja. Tätä pelin vaihetta kutsutaan nimellä pre-flop. Taskukorttien jakamisen jälkeen isoa sokkopanosta seuraava pelaaja aloittaa kierroksen. Hän voi tehdä yhden kolmesta toiminnosta: (1) luovuttaa eli heittää kortit pois ja luopua jaosta, (2) maksaa, jolloin hänen pitää laittaa pöytään ison sokkopanoksen verran merkkejä ja hän on yhä mukana pelissä tai (3) korottaa, jolloin hän laittaa pöytään ison sokkopanoksen ja korotuksen summan verran merkkejä. Pre-flop-kierroksella korotuksen arvo on yleensä sama kuin pieni sokkopanos.

Muilla pelaajilla on samat vaihtoehdot: luovuttaa, maksaa tai korottaa. Mikäli joku aikaisempi pelaaja on korottanut, maksuun tarvittava summa nousee luonnollisesti korotuksen verran. Kierroksen toimia jatketaan, kunnes kaikki pelissä mukana olevat

pelaajat – siis ne jotka eivät ole luovuttaneet – ovat maksaneet kaikki pyydetyt korotukset tai vähintään ison sokkopanoksen verran silloin, kun kukaan ei ole korottanut. Usein sallittujen korotusten määrää on rajoitettu, yleensä kolmeen korotuskertaan kussakin pelivaiheessa: pre-flopissa, flopissa, turnissa ja riverissä. Tämä tarkoittaa käytännössä sitä, että kolmanteen korotukseen ei voi enää uudelleen korottaa, vaan voi vain joko luovuttaa tai maksaa. Tämä sääntö on mukana siksi, ettei kaksi pelaajaa pystyisi jatkuvasti korottelemalla pudottamaan kolmatta pelaajaa pelistä.

2.2. Flop, turn, river ja korttien paljastus

Pre-flop-vaihetta seuraa kolme kierrosta, jotka on nimetty järjestyksessä flopiksi, turniksi ja riveriksi. Kun pre-flop on pelattu loppuun, eli kaikki pelaajat ovat joko maksaneet pyydetyn summan tai luovuttaneet, ja mikäli pelaajia on vielä jäljellä enemmän kuin yksi, jaetaan flop, eli kolme avointa korttia pöytään. Nämä kolme korttia ovat kaikkien pelaajien yhteisiä. Kun flop on jaettu, jakajan vasemmalla puolella oleva pelaaja aloittaa kierroksen. Hän voi nyt luovuttaa (mikä ei tosin yleensä tässä tilanteessa kannata, koska kukaan ei ole panostanut), sököttää eli passata panostusvuoronsa ja siirtää vuoron seuraavalle pelaajalle, tai panostaa. Näin jatketaan myötäpäivään pelaaja kerrallaan. Jos joku pelaajista panostaa tai korottaa, pitää muiden pelaajien maksaa pysyäkseen pelissä mukana. Korotus on myös aina mahdollinen, ellei korotuksia ole jo tehty kolmea kertaa (tai jotain muuta ennalta sovittua määrää) kyseisellä kierroksella. Kierrosta pelataan jälleen niin kauan, että jokainen jaossa mukana oleva pelaaja on vastannut jokaiseen panostukseen tai korotukseen joko luovuttamalla tai maksamalla.

Flopia seuraa kierros nimeltä turn, jolloin pöytään jaetaan yksi yhteinen kortti lisää. Tämän jälkeen pelataan panostuskierros samaan tapaan kuin flopissa sillä erotuksella, että turn- ja river-kierroksella panostuksen ja korotuksen arvo on yleensä kaksinkertainen aikaisempien kierrosten vastaaviin. River-kierroksella jaetaan taas yksi kaikkien pelaajien yhteinen kortti pöytään, ja pelataan panostuskierros kuten edellä. Kun river-kierroksen panostukset on pelattu, eli kaikki ovat luovuttaneet tai maksaneet viimeisimmän panostuksen tai korotuksen (on myös mahdollista, että kaikki sököttävät ja ovat pelissä mukana, jos yhtään panostusta ei tehdä), näytetään kortit. Potin voittaa se, joka pystyy muodostamaan parhaan viiden kortin pokerikäden omista kahdesta kortistaan ja pöydän viidestä kortista. Mikäli joku panostaa tai korottaa pelin aikaisemmassa vaiheessa ennen korttien paljastusta, eikä kukaan muu pelaaja maksa panostusta tai korotusta, viimeisimpänä panostanut tai korottanut voittaa potin.

3. Neuroverkko

Tässä luvussa kerron neuroverkoista siinä määrin kuin tähän työhön kuuluvan soveltavan tutkimuksen kannalta on tarpeellista.

3.1. Johdatus neuroverkkoihin

Keinotekoinen neuroverkko (artificial neural network, myöhemmin neuroverkko) on eräs tekoälyn sovellusalue. Tekoälytutkimuksessa on pyritty saamaan tietokoneet suorittamaan älyllisiä tehtäviä samaan tapaan kuin ihmiset. Tietyillä tekoälyn sovellusalueilla, kuten shakissa ja muissa rajattuihin vaihtoehtoihin ja tiettyyn logiikkaan perustuvissa tehtävissä, tietokoneiden tekoälyt on saatu toimimaan jopa ihmistä paremmin. Esimerkiksi IBM:n kehittämä shakkitietokone, Deep Blue, voitti vuonna 1997 shakin silloisen hallitsevan maailmanmestarin, Garry Kasparovin, normaaleilla turnauspeliajoilla 6 pelin turnauksessa pistein 3,5–2,5 [IBM Research, 2008].

Jokapäiväisessä elämässämme on kuitenkin paljon erilaisia, älykkyyttä vaativia tehtäviä, joihin tekoälysovellukset eivät vielä kykene, tai jotka ihmisaivot pystyvät suorittamaan tehokkaammin, nopeammin tai tarkemmin. Useimmiten ongelmia kohdataan sovellusalueissa, joissa käsiteltävän tiedon määrä on suuri. Vaikka tietokoneen laskentateho päihittääkin leikiten ihmisaivojen vastaavan, ihmisaivot pystyvät prosessoimaan rinnakkaisesti paljon paremmin kuin parhaatkaan nykyaikaiset tietokoneet [Juhola, 2000].

Esimerkkinä tällaisesta monimutkaista prosessointia vaativasta tehtävästä voisi mainita toimimisen liikenteessä. Ihminen pystyy kohtuullisen hyvin tekemään havaintoja ja toimimaan vallitsevan tilanteen mukaisesti esimerkiksi autoillessaan, mutta tekoälyllä varustetulle – esimerkiksi robotin ohjaamalle – kulkuneuvolle liikenteessä selviytyminen olisi vaikeaa. Havainnoitavien asioiden määrä on suuri, tilanteet muuttuvat nopeasti ja erilaisten toimintomahdollisuuksien määrä on valtava. Lisäksi näkökyvyn mallintaminen koneelle on hankalaa. Vaikeiden ja monimutkaisten tehtävien suorittaminen on tietokoneelle vaikeampaa kuin ihmiselle osaltaan siitä syystä, että vaikka tietokoneen laskentateho onkin parempi kuin ihmisen, ihmisaivoissa on valtava määrä laskentayksiköitä (n. 10^{10} kpl) ja ne toimivat hyvin rinnakkain (kullakin niistä on n. 10^4 yhteyttä muihin laskentayksiköihin) [Juhola, 2000].

Tekoälytutkimus on alkuvaiheessaan edennyt pitkälti rinnakkain ihmisaivojen tutkimisen kanssa. Myös neuroverkkojen kehitys perustui alun perin biologiseen esikuvaansa, ihmisaivojen toimintaan. Sittemmin ideat ovat kehittyneet neuroverkkojen puolella paljolti omaan suuntaansa, mutta perusta on edelleen peräisin biologisten hermosolujen toiminnan matemaattisesta mallintamisesta.

Neurolaskennassa hyödynnetään useita ihmisaivojen ominaisuuksia. Yksi näistä on oppiminen. Oppiminen neuroverkoissa tapahtuu samaan tapaan kuin ihmisen oppiminen: aluksi neuroverkko ei osaa juuri mitään, mutta kun neuroverkkoa opetetaan, sen osaaminen kasvaa. Onnistuessaan neuroverkot pyrkivät vahvistamaan onnistunutta suoritusta, mutta epäonnistuessaan ne pyrkivät korjaamaan virhettään onnistuakseen paremmin seuraavalla kerralla [Autio, 2003].

Toinen ihmisaivojen toiminnasta neuroverkkoihin siirretty ominaisuus on vikasietoisuus. Nykyinen lääketiede, neuropsykologia ja puheterapia mahdollistavat esimerkiksi aivovamman kärsineen ihmisen menettämän puhekyvyn palauttamisen jopa entiselle tasolleen. Epäkuntoon menneiden laskentayksiköiden tehtävien korvaaminen muilla laskentayksiköillä tai epäkuntoisten laskentayksiköiden uudelleenopettaminen palauttamalla niille tarvittavaa tietoa on tuttu ajattelumalli niin ihmisen kuin neuroverkkojenkin kuntoutuksessa. [Juhola, 2000]

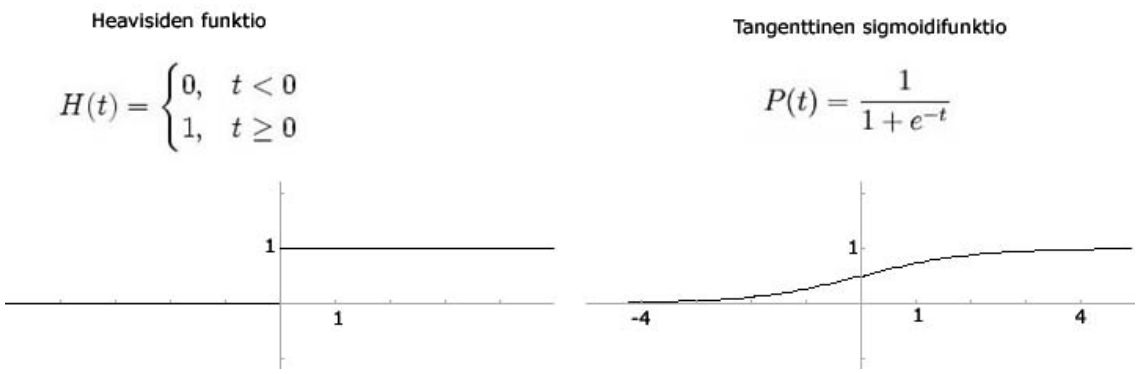
3.2. Neuronien toiminta

Keinotekoisten neuroverkkojen rakenne ja toiminta jäljittelevät ihmisaivojen rakennetta ja toimintaa. Neuroverkko sisältää useita laskentayksiköitä eli neuroneja sekä laskentayksiköiden välisiä yhteyksiä eli synapseja (synapse, edge). Neuroneilla ja synapseilla on lisäksi joitakin yksilöllisiä ominaisuuksia, jotka vaikuttavat neuroverkon toimintaan ja täten myös neuroverkon tuottamiin tuloksiin.

Neuroni saa toisilta neuroneilta syötteitä synapseja pitkin. Synapsille on asetettu oma painoarvonsa, joka määrittelee sen, kuinka tärkeä yhteys on kyseessä, ja joka heikentää tai vahvistaa synapsia pitkin neuronista toiseen kulkevaa signaalia. Mikäli kyseessä on tärkeä yhteys, eli sillä on suuri painoarvo, signaalia vahvistetaan, ja vähemmän tärkeän yhteyden kohdalla signaalia vastaavasti heikennetään. Syöteneuroneissa annetaan neuroverkolle tuotavat alkuarvot. Syöteneuronit eivät laske mitään eikä niillä ole kynnysarvoa (bias), ne vain välittävät arvoja eteenpäin seuraaville kerroksille.

Neuronin vastaanottama syöte lasketaan kaikkien siihen saapuvien syötteiden summana. Jokainen syöte muodostuu jonkin toisen neuronin tulossignaalista kerrottuna synapsin painokertoimella, ja mikäli kokonaissyötteen arvo ylittää neuronille asetetun sisäisen kynnysarvon, neuroni aktivoituu ja lähettää vasteen eteenpäin. Vasteen suuruus määräytyy käytettävän aktivaatiofunktion mukaan.

Aktivaatiofunktiot voidaan jakaa karkeasti kahteen eri luokkaan: lineaarisiin aktivaatiofunktioihin ja jatkuviin aktivaatiofunktioihin. Esimerkkinä lineaarisesta aktivaatiofunktiosta on Heavisiden funktio, ja esimerkkinä jatkuvasta aktivaatiofunktiosta tangenttinen sigmoidifunktio (myöhemmin sigmoidifunktio, kuva 1). Kuvassa näkyvissä kaavoissa t on neuroniin tulevan kokonaissyötteen arvo.



Kuva 1. Esimerkki lineaarisesta ja jatkuvasta aktivaatiofunktiosta.

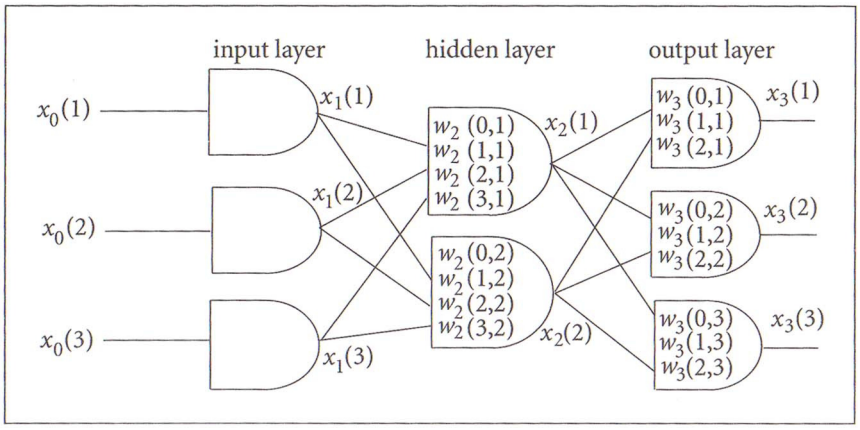
3.3. Neuroverkkojen rakenne

Neuronit muodostavat kerroksia, joissa voi olla vaihteleva määrä neuroneita. Perceptron on Frank Rosenblattin kehittämä yksinkertainen neuroverkko, jossa on syöteneuroneja sekä yksi tulosneuroni [Estebon, 1997]. Syöteneuronit sijaitsevat syötekerroksessa (input layer) ja tulosneuroni tuloskerroksessa (output layer). Monimutkaisemmissa neuroverkoissa, niin sanotuissa monikerroksisissa perceptron-neuroverkoissa, voi olla syöte- ja tuloskerroksen lisäksi myös piilokerroksia (hidden layer).

Synapsit muodostavat yhteyksiä ainoastaan peräkkäisten kerrosten välille. Mikäli syötekerroksen ja tuloskerroksen välillä on useita piilokerroksia, neuroverkon synapsien määrä kasvaa, sillä neuroneilla on jokaisen kerroksen välillä useita yhteyksiä. Tuloskerrosta edeltävän kerroksen – joka on siis syötekerros silloin kun neuroverkossa ei ole piilokerroksia – neuronit yhdistyvät tuloskerroksen neuroneihin synapseilla, joiden välittämien arvojen perusteella voidaan määritellä neuroverkon tietyillä syötearvoilla tuottama tulos. Neuroverkon kerroksien määrän ilmaisuun on olemassa erilaisia käytäntöjä. Esimerkiksi se, lasketaanko syötekerros mukaan neuroverkon kerroksiin saattaa vaihdella tapauskohtaisesti. Käytän tässä työssä tapaa, jossa myös syötekerros lasketaan. Näin ollen kuvassa 2 [Picton, 2000] esitetyssä neuroverkossa on kolme kerrosta.

Yleisesti ottaen yksinkertainen perceptron-verkko, jossa ei ole lainkaan piilokerroksia, pystyy ratkaisemaan vain ongelmia, joiden hahmot ovat lineaarisesti erottuvia [Juhola, 2000]. Tähän työhön liittyvässä sovelluksessa käytän monikerroksista perceptron-verkkoa.

Kuvassa 2 nähdään tyypillinen kolmekerroksinen neuroverkko kuten Picton [2000] on sen esittänyt. Kolmikerroksisessa neuroverkossa on syötekerros, tuloskerros ja yksi piilokerros. Pictonin esimerkissä syötekerros sisältää kolme, piilokerros kaksi ja tuloskerros kolme neuronia.



Kuva 2 [Picton, 2000]. Tyypillinen kolmikerroksinen perceptron.

Kuvan 2 neuroverkko on kokonaan yhdistetty, millä tarkoitetaan sitä, että jokaisessa kerroksessa kaikkien neuronien tulosarvot on yhdistetty seuraavan kerroksen jokaisen neuronin syötearvoon. Tämä on yleisin mutta ei ainoa tapa, jolla monikerroksinen perceptron voi olla yhdistetty [Picton, 2000].

Syötekerros on suoraan yhteydessä ulkopuoliseen maailmaan. Fyysisessä neuroverkossa syötekerros voisi muodostua esimerkiksi mittausantureista, jotka tuovat dataa neuroverkkoon. Tietokoneella ohjelmallisesti käytettävä neuroverkko voi saada syötetietonsa esimerkiksi käyttäjän manuaalisesti syöttämänä, tai suoraan tiedostosta. Syötekerroksien neuroneista tuodaan siis neuroverkolle syötearvot, mutta niissä itsessään ei tapahdu lainkaan prosessointia. [Picton, 2000]

Neuroverkon piilokerrosta kutsutaan piilokerrokseksi sen vuoksi, että kyseisen kerroksen syöte- ja tulosarvot eivät ole nähtävissä neuroverkon ulkopuolelta katsottaessa. Neuroverkossa voi olla useita piilokerroksia, mutta useimmissa tilanteissa yksi piilokerros riittää käsiteltävän ongelman ratkaisemiseen [Picton, 2000]. Piilokerroksissa tapahtuu neuroverkon varsinainen prosessointi, josta kerron lisää myöhemmin tässä luvussa.

Tuloskerros on yhteydessä neuroverkon ulkopuoliseen maailmaan ja prosessoi dataa kuten piilokerroksetkin. Kun tietyt syötearvot ajetaan opetetulle neuroverkolle, saadaan tuloskerroksesta selville kyseisiin syötearvoihin liittyvä tulos. [Picton, 2000]

Kuvassa 2 ei ole eritelty syöte- ja tulosarvoja erillisillä symboleilla, koska jokaisesta tietyn kerroksen tulosarvosta tulee seuraavan kerroksen syötearvo, ja ne pysyvät siis keskenään yhtä suurina. Kuvassa symboli x esittää arvoa, joka tulee perceptroniin tai lähtee perceptronista. Neuroverkon syötteillä on alaindeksi 0, ja muissa tapauksissa alaindeksi kertoo, minkä kerroksen tulosarvoja ko. arvot ovat. Suluissa oleva numeroarvo viittaa siihen, monennenko kyseisen kerroksen neuronin tulosarvosta on kyse. Esimerkiksi symboli $x_2(1)$ siis tarkoittaa toisen kerroksen ensimmäisen neuronin tulosarvoa, ja samalla kolmannen kerroksen neuroneille vietävää syötearvoa. [Picton, 2000]

Neuroverkon painoarvoja on merkitty kuvassa 2 kirjaimella w. Alaindeksi kertoo taas, minkä kerroksen painoarvosta on kyse. Suluissa olevista numeroista ensimmäinen kertoo, monenteenko edellisen kerroksen neuroniin paino on yhdistetty ja jälkimmäinen taas sen, monennenko saman kerroksen neuronin tulosarvoon kyseinen paino vaikuttaa. Esimerkiksi $w_2(2,1)$ ilmaisee siis toisen kerroksen painoa, joka on yhdistetty edellisen kerroksen tulosarvoon $x_1(2)$ ja joka on osallisena tulosarvoon $x_2(1)$. [Picton, 2000]

Kohdassa 3.4. kerron neuroverkon opettamisesta ja eräästä yleisestä monikerroksisen perceptronin oppimisalgoritmista, takaisinlevitysalgoritmista. Ohjelma, jolla tämän työn käytännön osuus on tehty, käyttää myös takaisinlevitystä oppimisalgoritminaan.

3.4. Neuroverkon suunnattu opettaminen ja takaisinlevitysalgoritmi

Neuroverkkojen opettaminen tapahtuu syöttämällä neuroverkolle dataa ja muuttamalla synapsien painoarvoja datan syöttämisen aikana. Tietyt syötearvot johtavat neuroverkolle määriteltyjen ominaisuuksien ja painoarvojen perusteella aina tietynlaiseen tulokseen opetuksen eri vaiheissa. Pääperiaate ohjatussa oppimisessa on se, että kun tiettyihin syötetietoihin liittyvä oikea tulos tiedetään ja datan tuottama tulos senhetkisessä neuroverkossa nähdään, pyritään ohjaamaan neuroverkon toimintaa oikeaan suuntaan. Jos saatu tulos on erilainen kuin pitäisi, painoarvoja muutetaan siten, että päästään kohti oikeaa tulosta. Neuroverkko oppii siis opetuksen aikana tekemiensä virheiden avulla. Neuroverkon oikeanlaista käyttäytymistä vahvistetaan samalla kun vääränlaista käyttäytymistä pyritään ehkäisemään.

Opetuksen alussa painoarvot on määritetty satunnaisesti, sillä tässä vaiheessa neuroverkko ei voi vielä osata mitään. Neuroverkossa esiintyvät painoarvot ovat alkuvaiheessa usein pieniä satunnaislukuja, eikä neuroverkko voi näin ollen muodostaa oikeita tuloksia automaattisesti. Neuroverkkojen käyttäminen perustuukin siihen, että neuroverkkoa opetetaan jollakin opetusaineistolla, jonka avulla se voi muodostaa painoarvot neuronien välisille yhteyksille opetusaineistoon perustuen.

Suunnatussa opettamisessa käytössä on jotain valmista dataa, johon kuuluvat sekä syötteet että syötteistä johdettavissa oleva tulos. Neuroverkko käy opetusvaiheen aikana tätä dataa läpi useita kertoja päivittäen samalla synapsien painoarvoja siten, että neuroverkon tuottama virhe olisi mahdollisimman pieni. Virhe lasketaan tässä odotetun tuloksen ja saavutetun tuloksen erotuksena. [Picton, 2000]

Opetusta jatketaan, kunnes opetuksen lopettamisehto täyttyy. Ehtona voi toimia esimerkiksi suoritettavien epookkien eli datan läpikäyntikertojen määrä, opetukseen käytetty aika tai keskineliövirheen neliöjuuren muutos.

Usein monikerroksisen perceptronin opetuksessa käytetään niin kutsuttua takaisinlevitysalgoritmia. Joskus takaisinlevitysalgoritmiin viitataan yleistettynä delta-sääntönä, sillä se on samankaltainen aikaisemmin kehitetyn, delta-säännöksi kutsutun opetusprosessin kanssa. Tätä alkuperäistä delta-sääntöä on käytetty ADALINE-

neuroverkon kanssa. Se oli ensimmäisiä neuroverkkoja, joiden opetuksessa voitiin hyödyntää painoarvoja. [Picton, 2000]

Perusajatus takaisinlevitysalgoritmin käytössä on se, että neuroverkon painoarvoja säädetään siten, että toivotun tuloksen ja saavutetun tuloksen eroa pyritään pienentämään. Jos saavutettu tulos on suurempi kuin haluttu tulos, painoarvoa säädetään siten, että tulo $w_i x_i$ antaa pienemmän tuloksen kuin aiemmin. Käytännössä siis jos x_i on negatiivinen, painoarvoa kasvatetaan ja jos x_i on positiivinen, painoarvoa pienennetään. Mikäli saavutettu tulos on pienempi kuin haluttu tulos, toimitaan päinvastoin. [Picton, 2000]

Sen, kuinka paljon painoarvoa säädetään, tulee olla suhteessa muuttujiin x_i, joka tarkoittaa siis käytännössä perceptroniin tulevaa syötearvoa, sekä δ, jolla merkitään virhettä halutun ja saavutetun tuloksen välillä. Kun käytetään sigmoidifunktiota, muutos painoarvossa w_i yksittäisessä perceptronissa on kaavan 1 mukainen. Kaavassa Δw_i tarkoittaa siis painoarvon muutosta ja η oppimiskerrointa, joka on käyttäjän määrittelemä vakio. P:llä merkitään opetusjoukon esiintymien (pattern) määrää, ja x_i sekä δ on määritelty kuten edellä. [Picton, 2000]

$$\Delta w_i = \frac{\eta}{P} \sum_{p=1}^{P} x_{ip} \delta_p$$

Kaava 1 [Picton, 2000]. Painoarvon muutos käytettäessä sigmoidifunktiota ja takaisinlevitysalgoritmia.

Painoarvoja päivitetään jokaisen opetusjoukon esiintymän jälkeen sen sijaan, että ne päivitettäisiin kerralla, kun koko opetusjoukko on käyty läpi. Tämä nopeuttaa neuroverkon oppimista, sillä opetusjoukko saattaa olla hyvin suuri. [Picton, 2000]

Mitchell [1997] esittää takaisinlevitysalgoritmin suorituksen neuroverkon muodostamisen ja satunnaisten alkupainojen asettamisen jälkeen nelivaiheisena proseduurina, joka suoritetaan jokaisen opetusesimerkin, {x, t} kohdalla kunnes ennalta määritetty lopetusehto täyttyy. Vektori x on muodostettu verkon syötearvoista ja vektori t verkon tavoiteltavista tulosarvoista. Proseduurin neljä vaihetta on esitetty seuraavasti:

1) Aluksi käsiteltävän opetusesimerkin syötearvot levitetään eteenpäin verkossa, jotta saadaan laskettua tulosarvo.

2) Seuraavaksi levitetään virhe takaperin verkon läpi. Jokaiselle verkon tulosneuronille lasketaan virhe perustuen tavoitellun tuloksen ja saavutetun tuloksen väliseen erotukseen.

3) Jatketaan taaksepäin verkossa ja lasketaan jokaiselle piiloneuronille virhe. Virhe lasketaan piilokerroksen ja tuloskerroksen välisten painoarvojen sekä tuloskerroksen virheiden summana.

4) Jokainen painoarvo päivitetään kaavan 2 mukaisesti. η on ennalta määritetty oppimiskerroin, δ on edellä laskettu virhe ja x on perceptroniin tuleva syötearvo.

$$\Delta w = \eta x \delta$$

Kaava 2 [Picton, 2000]. Yksittäisen painoarvon muutos.

3.5. Momentti ja muut neuroverkon variointimahdollisuudet

Monikerroksinen neuroverkko toteutetaan useimmiten yhdellä piilokerroksella käyttäen sigmoidifunktiota aktivaatiofunktiona [Picton, 2000]. Neuroverkkoa voidaan lisäksi muokata monin tavoin neuroverkon oppimisen nopeuttamiseksi.

Momenttitermin lisääminen painoarvojen säätämisen yhteyteen voi vaikuttaa neuroverkon oppimisnopeuteen. Aiemmin esitetyn mukaan perceptronin painoarvojen muutokset seuraavat kaavaa 2.

Painoarvon muutokseen voidaan lisätä vakiomuotoinen momenttitermi α, jolla kerrotaan nykyinen Δw:n arvo. Tällöin painoarvon muutos seuraa siis kaavaa 3.

$$(\Delta w)_{k+1} = \eta x \delta + \alpha(\Delta w)_k$$

Kaava 3 [Picton, 2000]. Yksittäisen painoarvon muutos käytettäessä momenttitermiä.

Oikean momentin arvon valitseminen on yrityksen ja erehdyksen kautta tehtävä valinta, vaikka usein valitaankin arvo, joka on pieni suhteessa arvoon ηxδ [Picton, 2000].

Toinen variointitapa on säätää tulosarvoalue väliltä [0,1] välille [-1,1]. Tämä saattaa nopeuttaa opetusta käytettävän funktion epäsymmetrisyyden vuoksi. Tämä uusi funktio, jota kutsutaan hyperboliseksi tangentiksi, tuottaa myös sigmoidifunktiota todennäköisemmin nollaa lähellä olevia arvoja laajennetun arvoalueensa vuoksi [Picton, 2000]. Uusi arvoalue voidaan ottaa käyttöön painottamalla sigmoidifunktio kaavan 4 tapaan.

$$y = \frac{2}{1+e^{-x}} - 1 = \frac{(1-e^{-x})}{(1+e^{-x})}$$

Kaava 4 [Picton, 2000]. Hyperbolinen tangentti.

Jos hyperbolista tangenttia käytetään aktivaatiofunktiona takaisinlevitysalgoritmin kanssa, pitää takaisinlevityssääntöä muokata siten, että y:n derivaatta x:n suhteen on kaavan 5 mukainen. Käytettäessä tätä muokattua takaisinlevityssääntöä oppimiskertoimen η arvo voidaan kaksinkertaistaa [Picton, 2000].

$$y' = 2y(1-y)$$

Kaava 5 [Picton, 2000]. Takaisinlevityssäännön muokkaus käytettäessä hyperbolista tangenttia.

Edellä on kuvattu sitä, miten neuroverkon opetus algoritmeineen toimii, ja miten sitä voidaan nopeuttaa. Oleellista on myös tarkastella, milloin neuroverkon opetus on aika lopettaa.

Opetuksen lopettamisaika voidaan määritellä usealla eri tavalla, mutta yksi yleisimmistä tavoista on tarkastella halutun tulosarvon ja saadun tulosarvon välisen keskineliövirheen arvoa. Tämän arvon tulisi pienentyä opetuksen aikana, ja kun tämä arvo saavuttaa riittävän alhaisen, ennalta sovitun arvon, voidaan opetus lopettaa. [Picton, 2000]

Edellä mainittua ennalta sovittavaa arvoa mietittäessä kannattaa kuitenkin pysyä kohtuudessa. Jos arvo asetetaan liian pieneksi, neuroverkon opetus kestää kauan ja voidaan ajautua yliopetukseksi nimitettyyn tilanteeseen. Yliopetuksella neuroverkon kyky yleistää dataa heikkenee, mikä sotii neuroverkon peruslähtökohtia vastaan, sillä datan yleistämiskyvyn tulisi olla neuroverkon perusominaisuus [Picton, 2000]. Silloin kun neuroverkko yliopetetaan, verkon sanotaan ylisovittavan (over-fit) opetusdatan. Palaan ylisovittamisen ongelmaan ja sen välttämiseen tarkemmin luvussa 6.

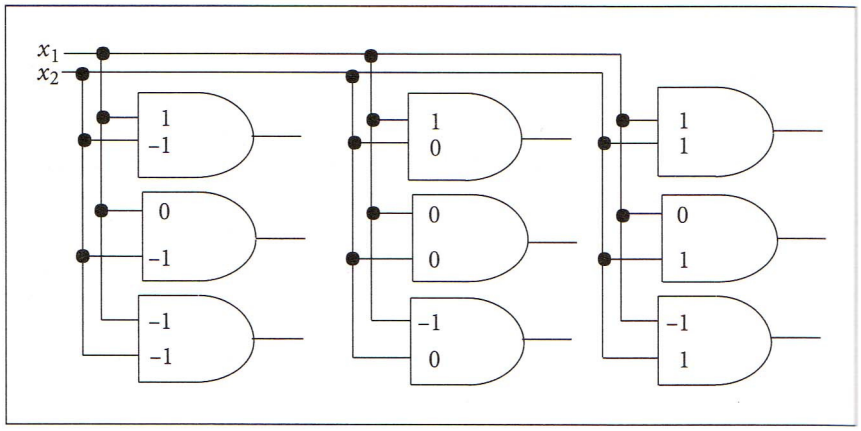
3.6. Muista neuroverkoista

Edellä kuvatun monikerroksisen perceptronin lisäksi on olemassa monenlaisia muita neuroverkkomalleja. Esittelen tässä niistä muutamia yleisimpiä.

3.6.1. Kohosen verkko

Kohosen verkko, joka on saanut nimensä kehittäjänsä Teuvo Kohosen mukaan, pyrkii tuottamaan itsejärjestyvän malliluokittelijan käyttäen Kohosen oppimista (Kohonen learning) painoarvojen säätämiseen. Tyypillisesti Kohosen verkko kostuu neuronien muodostamasta 2-ulotteisesta taulukosta, jossa jokainen syötearvo tuodaan jokaiseen neuroniin. Jokaisella neuronilla on omat painoarvonsa, joita verrataan saapuvaan syötearvoon, ja lähimpänä syötearvoa olevia painoarvoja vastaava neuroni antaa suurimman palautteen. Kohosen verkko on järjestetty siten, että painoarvoiltaan lähellä toisiaan olevat neuronit sijaitsevat myös fyysisesti lähekkäin verkossa. [Picton, 2000]

Hyvä esimerkki Kohosen neuroverkosta on itsejärjestyvä kartta (self organising map, SOM), jonka tavoitteena on tuottaa esitys, jossa painot vastaavat koordinaatteja jossain topologisessa järjestelmässä tai kartassa, ja yksittäiset neuroverkon elementit on järjestetty tietyllä tavalla (kuva 3). [Picton, 2000]



Kuva 3 [Picton, 2000]. Kaksiulotteinen kartta esitettynä kaksiulotteisena neuronitaulukkona.

Aluksi jokainen painoarvo Kohosen verkossa on asetettu satunnaisesti. Tämän jälkeen järjestelmä järjestää itse itsensä siten, että painot vastaavat koordinaatteja ja elementtien sijainnit koordinaattisen järjestelmän sijainteja. Tämän järjestyksen aikaansaamiseksi voidaan käyttää erilaisia metodeja, mutta usein käytetään euklidista etäisyysfunktiota. Tämä saadaan ottamalla neliöjuuri erotusten neliöiden summasta (kaava 6). Kaksiulotteisen ongelman tapauksessa etäisyys kullekin neuronille lasketaan kaavalla 7. [Picton, 2000]

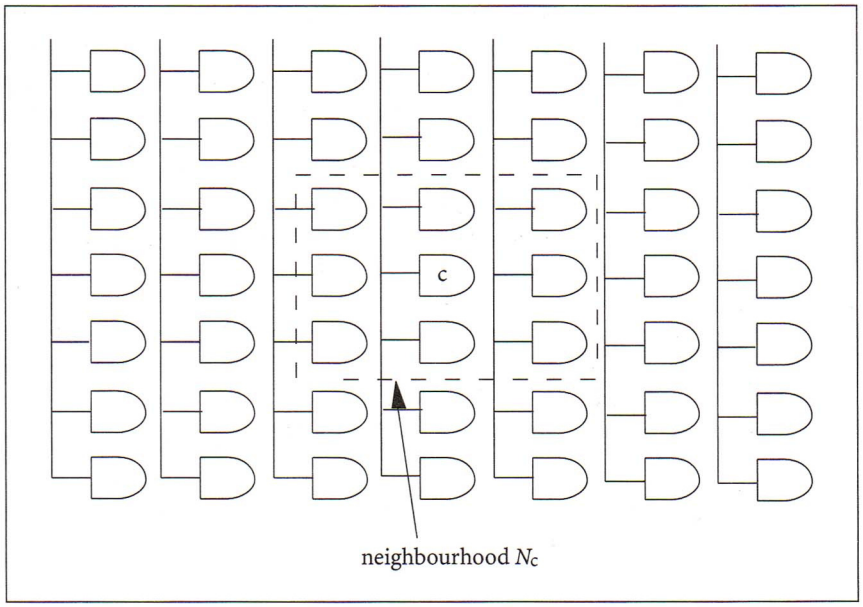
$$D_j = \sqrt{\Sigma_{i=1}^{n}(x_i - w_{ij})^2}$$

Kaava 6 [Picton, 2000]. Neuronin sijainnin selvittäminen Kohosen verkossa.

$$D_j = \sqrt{\Sigma_{i=1}^{2}(x_i - w_{ij})^2} = \sqrt{(x_1 - w_{1j})^2 + (x_2 - w_{2j})^2}$$

Kaava 7 [Picton, 2000]. Neuronin sijainnin selvittäminen kaksiulotteisen ongelman tapauksessa Kohosen verkossa.

Verkkoon tuotavaa syötearvoa verrataan kaikkiin verkon elementteihin samalla tavalla kuin edellä kuvatun alkutilan järjestyksen luomisen yhteydessä verrataan neuroneja toisiinsa. Pienimmän D:n arvon mukaan verkosta valitaan tietty elementti c, ja sen ympäriltä valitaan ennalta määritellyllä etäisyydellä N_c olevat elementit. Tätä valintaa, jota kutsutaan tässä lähijoukoksi N_c (neighbourhood of element c) on havainnollistettu kuvassa 4. [Picton, 2000]



Kuva 4 [Picton, 2000]. Lähijoukko N_c.

Kun lähijoukko N_c on tunnistettu, kaikki ko. joukon elementtien painot muutetaan Kohosen oppimisen mukaisesti (kaava 8). Kaikki muut painot verkossa jätetään ennalleen.

$$\Delta w_{ij} = k(x_i - w_{ij})y_j$$

Kaava 8 [Picton, 2000]. Painoarvojen muutos Kohosen verkossa.

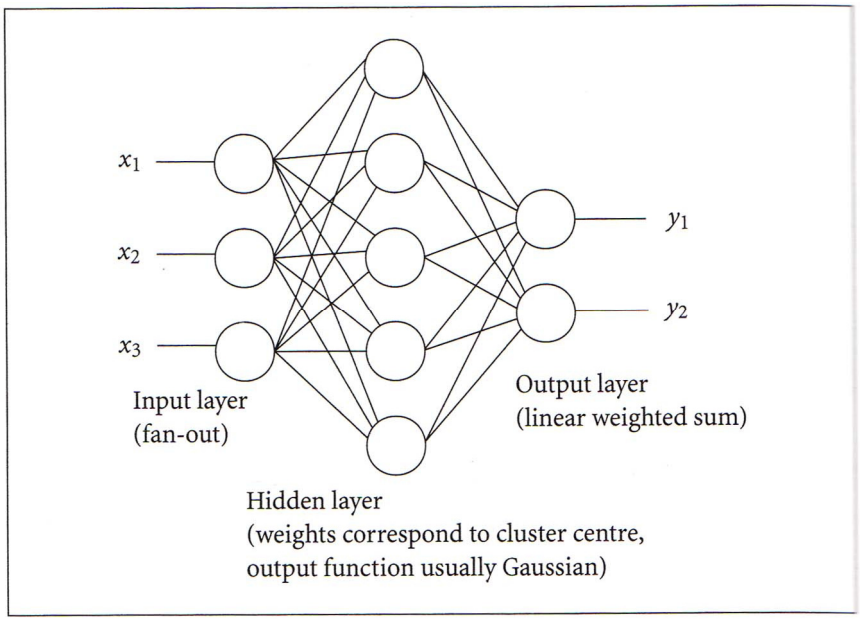
N_c:n koko ja kaavassa 8 esiintyvän vakion k suuruus ovat merkittäviä Kohosen verkon käytön kannalta. On myös tärkeää, että ne molemmat pienenevät opetuksen edetessä. [Picton, 2000]

Kohosen neuroverkon tulosarvoa ei saada neuroverkosta samaan tapaan kuin esimerkiksi monikerroksisen perceptronin kohdalla. Neuroverkon kouluttaminen sisältää samankaltaisten mallien ryhmittelyn malliavaruudessa, joten samankaltaisten mallien osiot saavat ne neuronit laukeamaan, jotka ovat fyysisesti lähellä toisiaan. Yksinkertaisimmillaan tulosarvo on syötteiden painotettu summa, mahdollisesti sigmoidifunktion läpi ajettuna. Tällöin tulosarvo on 1 tietyllä koordinaattiavaruuden alueella, joka vastaa tiettyä mallien luokkaa. Tällöin pitäisi olla mahdollista tunnistaa, mitkä alueet kuuluvat mihinkin luokkaan näyttämällä verkolle tunnettuja malleja ja katsomalla, mitkä osiot aktivoituvat. [Picton, 2000]

3.6.2. Radiaalikantaverkot

Radiaalikantaverkko (radial basis function network, myöhemmin RBF), joka on tyypiltään ns. eteenpäin syöttävä verkko (feedforward network), on suurin haastaja monikerroksiselle perceptronille. RBF on syntynyt pohjautuen pääosin perinteisiin

tilastollisiin mallinluokitustekniikoihin, jotka ovat saavuttaneet suuren yleisön tietoisuuden vasta kun niistä on muodostettu oma neuroverkkomallinsa. [Picton, 2000]



Kuva 5 [Picton, 2000]. Radiaalikantaverkko.

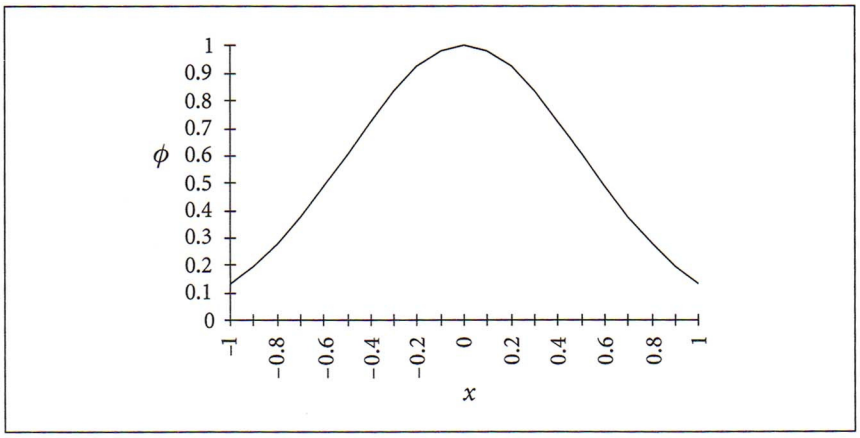
RBF:n kolmikerroksinen perusarkkitehtuuri on esitetty kuvassa 5. Syötekerros ei prosessoi mitään. Toinen kerros eli piilokerros esittää epälineaarisen kuvauksen (mapping) syöteavaruudesta moniulotteisessa avaruudessa, jossa mallit ovat lineaarisesti erottuvia. Kolmas kerros, eli tuloskerros, esittää yksinkertaisen painotetun summan lineaarisena tuloksena. Jos RBF:llä ollaan tekemässä malliluokitusta, saattaa esimerkiksi sigmoidifunktion käyttö tulosneuroneille olla tarpeen. [Picton, 2000]

RBF:n toiminta perustuu piilokerroksessa tapahtuvaan prosessointiin. Syöteavaruuden mallit muodostavat osioita (cluster), ja jos näiden osioiden keskipisteet ovat tunnettuja, etäisyys osioiden keskipisteistä voidaan mitata. Tämä etäisyyden mittaus tehdään epälineaarisesti. Jos malli on alueella, joka sijaitsee lähellä osion keskipistettä, saadaan arvo, joka on lähellä arvoa 1. Mitä etäämmäksi tästä pisteestä mennään, sitä pienemmäksi arvo laskee. [Picton, 2000]

$$\phi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right)$$

Kaava 9 [Picton, 2000]. Yleisimmin käytetty radiaalikantafunktio.

Useimmiten käytetty radiaalikantafunktio on esitetty kaavassa 9. Kaavassa r on etäisyys osion keskipisteestä, ja sen yhtälö piirtää käyrän, joka on nähtävissä kuvassa 6. [Picton, 2000]



Kuva 6 [Picton, 2000]. Yleisimmin käytetyn radiaalifunktion kuvaaja.

Etäisyys osion keskipisteestä lasketaan euklidisena etäisyytenä kaavan 10 esittämällä tavalla, jolloin piilokerroksen j:nnen neuronin tulosarvo saadaan kaavalla 11.

$$r_j = \sqrt{\sum_{i=1}^{n} (x_i - w_{ij})^2}$$

Kaava 10 [Picton, 2000]. Etäisyys osion keskipisteestä.

$$\phi_j = \exp\left(-\frac{\Sigma_{i=1}^{n} (x_i - w_{ij})^2}{2\sigma^2} \right)$$

Kaava 11 [Picton, 2000]. Piilokerroksen j:nnen neuronin tulosarvo.

Muuttuja σ määrittelee kuvassa 6 nähtävän kellokuvaajan leveyden eli säteen, ja se pitää asettaa empiirisesti [Picton, 2000].

RBF:n piilokerroksessa on yksiköitä, joiden painoarvot vastaavat osion keskipisteen vektoriesitystä. Nämä painot voidaan löytää käyttämällä perinteistä osiointialgoritmia, kuten k-keskiarvoalgoritmia (k-means algorithm) tai esimerkiksi Kohosen algoritmia. [Picton, 2000]

K-algoritmi toimii siten, että asetetaan ensin k satunnaista pistettä malliavaruuteen. Tämän jälkeen jokainen opetusdatan esiintymä liitetään lähimpään edellä asetetuista pisteistä. Tämän jälkeen lasketaan jokaisessa pisteessä siihen liitettyjen esiintymien koordinaattien keskiarvot, jotka määräävät ko. pisteen uuden paikan. Tätä prosessia jatketaan, kunnes muutosta ei enää tapahdu. [Picton, 2000]

Kun osioiden keskipisteet on löydetty, pitää seuraavaksi määritellä Gaussin käyrän (kuva 6) säde. Tähän käytetään useimmiten P lähintä -algoritmia (P-nearest neighbour),

jolloin jokaiselle osion keskipisteelle haetaan P lähintä osion keskipistettä ja lasketaan niiden välisten etäisyyksien neliöiden summan neliöjuuri (kaava 12). [Picton, 2000]

$$\sigma_j = \sqrt{\frac{1}{P}\sum_{i=1}^{P}(c_k - c_i)^2}$$

Kaava 12 [Picton, 2000]. P lähintä -algoritmin kaava.

Edellä annettua metodia käytetään piilokerroksen opetuksessa, jonka jälkeen tuloskerros opetetaan käyttäen jotain perinteistä gradientinvähennystekniikkaa (gradient descent technique).

3.6.3. Probabilistiset verkot

Probabilistinen verkko (probabilistic neural network, myöhemmin PNN) on alun perin syntynyt rinnakkaiseksi toteutukseksi vanhalle tilastolliselle tekniikalle, jota on käytetty mallien luokitteluun. [Callan, 1999]

PNN:ssä malli luokitellaan sen perusteella, mikä sen etäisyys on muista malleista. PNN perustuu Bayesin luokittelutekniikkaan. Tietyn esimerkkimallin perusteella voidaan tehdä päätelmiä siitä, mistä luokasta esimerkkimalli on peräisin. Päätelmän tueksi pitää arvioida todennäköisyyden tiheysfunktio (probability density function, PDF) jokaiselle luokalle. Tämä arvio luodaan opetusdatan pohjalta. Luokka, jossa on hyvin tiheä populaatio tuntemattoman näytteen (esimerkkimallin) alueella, on ensisijaisempi valinta kuin muut luokat. Myös luokat, joiden aikaisempi todennäköisyys tai väärinluokittelun aiheuttama virhe ovat suuria, ovat ensisijaisia valintoja. Kahdesta luokasta, A ja B, valitaan luokka A, jos kaavassa 13 esitetty epäyhtälö toteutuu. [Callan, 1999]

$$h_A c_A f_A(\mathbf{x}) > h_B c_B f_B(\mathbf{x})$$

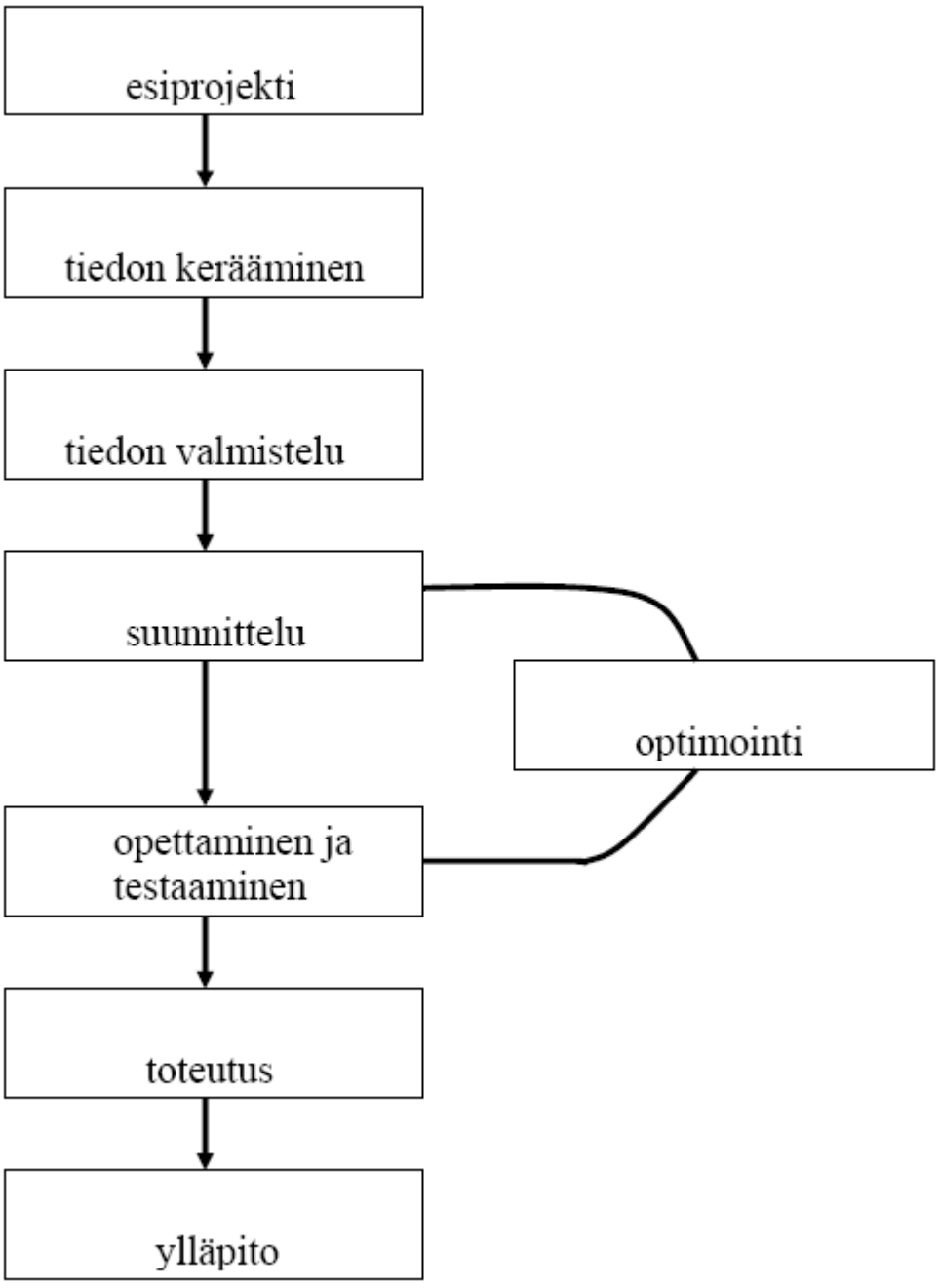
Kaava 13 [Callan, 1999]. Luokan A valintaan johtava epäyhtälö.

Kaavassa 13 h on aiempi todennäköisyys, c on väärinluokittelun aiheuttama virhe (cost of misclassification) ja f(x) on tiheysfunktio. Väärinluokittelun aiheuttaman virheen suuruuden arvioiminen edellyttää tietoa sovellusalasta, mutta monilla sovellusaloilla sekä väärinluokittelun aiheuttama virhe että aiemmat todennäköisyydet käsitellään samoin jokaisen luokan kohdalla. [Callan, 1999]

PNN:ssä ei tapahdu opetusta samassa mielessä kuin esimerkiksi monikerroksisessa takaisinlevitysalgoritmia käyttävässä verkossa, koska kaikki PNN-verkon parametrit (elementtien ja painojen määrä) määritellään suoraan opetusdatan perusteella. Ensimmäisen kerroksen painot on määritelty suoraan opetusmalleista, joita yhdistetään toisessa kerroksessa. Viimeinen kerros tuottaa signaalien summien perusteella tuloksen eli käytännössä kertoo, mihin luokkaan annettu syöte todennäköisimmin kuuluu.

4. Neuroverkkosysteemin toteutuksesta

Jotta päästään tutkimaan oppivaa neuroverkkoa käytännössä, tulee työskentelyn aikana käydä läpi tiettyjä vaiheita. Tässä työssä seuraan pääpiirteittäin Juholan [2000] esittämää vaihejakoa neuroverkkosysteemin käytännön toteutuksessa (kuva 7).



Kuva 7 [Juhola, 2000]. Neuroverkkosysteemin tyypillinen kehittämiselinkaari.

Esiprojektin aikana määritellään neuroverkkosysteemille asetetut tavoitteet. Paras tapa edetä tässä on rajata käsiteltävä ongelma hyvin ja määritellä realistiset odotukset tavoiteltavasta tuloksesta. Ongelma-alueen hyvä hallinta ja historiatiedon hankinta ovat myös keskeisessä asemassa projektia käynnistettäessä. [Juhola, 2000]

Tiedon keräämisessä tulee huomioida kerättävän tiedon edustavuus suhteessa mallinnettavaan ongelmaan, sillä jos kerätty tieto vastaa heikosti tutkittavaa asiaa, ei saavuteta totuudenmukaisia tuloksia. Tämän lisäksi tieto täytyy esiprosessoida. Tiedon esiprosessointi on kriittinen ja paljon aikaa vievä vaihe neuroverkkosysteemin toteutuksessa, mutta se on syytä suorittaa huolella, jotta esiprojektissa asetetut tavoitteet saavutettaisiin. Mikäli neuroverkon suorituskyky jää alle asetettujen tavoitteiden, tiedon esiprosessointi saatetaan joutua uusimaan. [Juhola, 2000]

Suunnitteluvaiheessa määritetään neurolaskennan välineet, mallin rakenne ja alkuehdot. Kun neurolaskentamalli on laadittu, sen toimivuus tulee testata. Tähän tehtävään kuuluu mallin opettaminen ja systeemin suorituskyvyn testaaminen tiedolla, jota ei ole käytetty opettamisessa. [Juhola, 2000]

Neurolaskentamallin optimointi tapahtuu myös testaamisvaiheen yhteydessä. Mikäli tulokset osoittavat systeemin suorituskyvyn huonoksi, voidaan mallin kehittämistä jatkaa esimerkiksi muuttamalla neuroverkon rakennetta. Joskus voi olla tarpeellista palata uudestaan tiedon keräämiseen ja esiprosessointiin. On myös mahdollista, että käytettävissä olevassa tiedossa ei ole riittävästi ennustettavaa informaatiota. Tässä tapauksessa on mahdollista pyrkiä parempiin tuloksiin jakamalla ongelma pienempiin osaongelmiin, joita tarkastellaan erikseen. [Juhola, 2000]

Neurolaskentamallin kehittämisen ja testaamisen jälkeen sitä on yhä syytä kehittää ja ylläpitää. Mallin toimintaa ja sen tekemiä päätöksiä on seurattava, jotta verkko ei aiheuta ongelmia niissä käytännön tilanteissa, joita se on asetettu ratkomaan. [Juhola, 2000]

5. Datan käsittely

Tässä luvussa esittelen käyttämäni datan keräämisen ja käsittelyn vaiheittain seuraten Juholan [2000] esittämää mallia, jonka esittelin luvussa 4.

5.1. Esiprojekti

Esiprojektiin liittyvät tavoitteet systeemin suorituskyvylle ovat tässä työssä suhteellisen väljät. Ihminen, varsinkin sellainen jolla on vähänkään kokemusta pokerista, pystyy ennakoimaan jonkin verran vastapelaajan toimia. Tämän työn tavoitteena on luoda neuroverkkosysteemi, joka pystyisi vähintään samantasoiseen suoritukseen kuin jonkin verran kokenut pokerinpelaaja. Aiheesta on aiemmin tehty tutkimuksia, joilla on päästy jopa 75–90 % tarkkuuteen pelaajien toimintojen ennustamisessa [Davidson, 1999]. Käytän pohjana Davidsonin [1999] tutkimusta ja yritän kehittää eteenpäin muuttujien valintaa ja neuroverkon rakennetta siten, että pääsisin mahdollisimman hyvään tulokseen. Tuloksen minimiedellytyksenä pidän 50 % tarkkuutta, koska tulosvaihtoehtoja on kolme, ja jo arvaamalla voi päästä keskimäärin n. 33,3 % tarkkuuteen.

Pokerissa pitkäjännitteinen vastapelaajien toimintatapojen opiskelu on tärkeässä asemassa, minkä vuoksi käytettävän datan tulee sisältää useita pelattuja käsiä samalta pelaajalta. Lisäksi parhaat pelaajat vaihtelevat pelityyliään, joten kovin pienellä otoksella ei voitaisi tehdä juurikaan johtopäätöksiä yksittäisen pelaajan toiminnoista. Jotta työ säilyttäisi yleisen luonteensa, tutkin useita pelaajia. Yhtä pelaajaa tutkittaessa voitaisiin käytännössä muodostaa malli vain kyseisen pelaajan toiminnoista.

5.2. Tiedon kerääminen ja syötteiden valinta

Syötetietona käytetään Pokerhand-palvelusta [Pokerhand.org, 2007] kerättyä dataa pokerikäsistä. Olen kerännyt em. palvelusta Texas Hold'em limit -pelien tietoja syöttämällä dataa käsin itse tekemäni PHP-sovelluksen kautta tietokantaan. Olen kerännyt jokaiseen peliin liittyvät yksittäiset pelaajien toiminnot sekä niihin liittyvät oleelliset muuttujat. Jokaisesta Pokerhand-palvelusta kerätystä kädestä on saatu tietokantaan useita rivejä dataa, koska jokainen yksittäinen toiminto on tallennettu omana tietokantarivinään.

Tietokantaan kerättyjen tietojen perusteella tarkoitukseni on opettaa neuroverkko ennakoimaan pokerinpelaajan toimintoja tietynlaisissa tilanteissa. Syötetiedoista tuloksiin päästään parhaiten suorittamalla tietyt toimenpiteet käytännön toteutuksen aikana. Juholan [2000] esittämä neuroverkkosysteemin elinkaari on esitetty kuvassa 7. Seuraan pääpiirteittäin tätä mallia neuroverkkosysteemin käytännön toteutuksessa.

Taulukossa 2 on esitetty syötearvot, joita kerätystä datasta käytetään neuroverkon syöteneuroneissa. Syötteiden valinnassa on seurattu pitkälti Davidsonin [1999] työssään käyttämien syötteiden listaa jonkinasteisin muutoksin.

#	**Syöte**	**Kuvaus**
1	Pottikerroin (pot odds)	Pyydetyn panostuksen suhde pottiin
2	Panostussuhde (bet ratio)	Omien panostusten suhde kaikkiin panostuksiin
3	Sitoutuminen kierrokseen	Aikaisempi panostus tai maksu kierroksella
4	Pelaajien määrä	Jaossa mukana olevien pelaajien määrä
5	Asema (position)	Pelaajan paikka pelipöydässä
6	Kierroksen panostukset	Panostusten ja korotusten määrä ennen pelaajan vuoroa
7	Kierros	Onko kyseessä flop-, turn- vai river-kierros
8	Edellinen toiminto	Oliko pelaajan edellinen toiminto panostus tai korotus
9	Ässä pöydässä	Onko pöytään jaettu ässä
10	Kuningas pöydässä	Onko pöytään jaettu kuningas
11	Värin veto	Onko pöydässä värin veto

Taulukko 2. Neuroverkon syötteet.

Käyn tässä vielä läpi syötteet yksitellen kertoen täsmällisemmin niiden merkityksen sekä perustellen niiden valintaa. Pyrin myös perustelemaan mahdollisimman hyvin ne muutokset, joita olen tehnyt Davidsonin [1999] esittämään syötejoukkoon.

Pottikerroin tarkoittaa täsmällisesti ilmaistuna suhdelukua pelaajalta pyydetyn panostuksen ja potissa jo olevien pelimerkkien välillä (tutkimuksessa on tutkittu pääasiassa käteispelejä, mutta tässä puhutaan selkeyden vuoksi pelimerkeistä). Oletetaan, että potissa on ennen panostuskierroksen alkua esimerkiksi 800 pelimerkkiä, ja joku edellisistä pelaajista panostaa 100 pelimerkkiä. Toinen edeltävä pelaaja maksaa, minkä jälkeen tulee tarkasteltavan pelaajan vuoro päättää, mitä hän tekee. Pelaajalta pyydetään 100 pelimerkkiä ja potissa on nyt 800 + 100 + 100, eli yhteensä 1000 pelimerkkiä. Pottikerroin on tällöin 100:1000 eli 1:10. Pottikerroin on tärkeä käsite tehtäessä päätöstä toiminnasta. Pyysing [2005] esittää pottikertoimeen liittyvän peruskysymyksen seuraavasti: ”Kannattaako minun maksaa vastustajani pyytämä hinta tuon kokoisesta potista, kun käteni todennäköisyys parantua voittavaksi kädeksi on tämä?”. Toiminnon ennustamiseen liittyy siis vahvasti pottikertoimen ja käden mahdollisuuksien yhdessä muodostama odotusarvo voitolle.

Panostussuhde on omien panostusten määrän suhde kaikkien jaon aikana tehtyjen panostusten määrään. Pelaaja, joka panostaa kerran, on taipuvainen panostamaan

uudestaan, koska panostaminen ja sen jälkeinen sököttäminen on usein merkki käden heikkoudesta. Tehokas panostaminen kertoo yleensä vahvasta kädestä, mutta voi olla myös bluffausta. Oli tapaus kumpi hyvänsä, omien panostusten suhde koko pöydän panostuksiin jaon aikana kertoo paljon siitä, kuinka pitkälle pelaaja on valmis etenemään ja millä hinnalla.

Sitoutuminen kierrokseen esitetään totuusarvona, ja se saa arvon 1 silloin, kun pelaaja on joko panostanut tai maksanut panostuksen kuluvan panostuskierroksen aikana. Muussa tapauksessa muuttuja saa arvon 0.

Jaossa mukana olevien pelaajien määrä on hyvin merkittävä tekijä, sillä mitä enemmän vastassa on käsiä, sitä todennäköisempää on, että oma käsi ei ole voittava käsi. Lisäksi päätöksiä vaikeuttaa usean pelaajan tarkkaileminen yhtä aikaa sekä vastustajien mahdollisuudet vaikuttaa peliin. Koska nettipokeripöydässä voidaan pelata myös vajailla pöydällisillä, en jaa tätä lukua kymmenellä, kuten Davidson [1999] on tehnyt. Lisäksi jätän huomiotta alkuperäisen pelaajien määrän: Mielestäni on järkevämpää keskittyä tarkastelemaan kulloinkin voimassa olevaa tilannetta ja siihen liittyviä muuttujia kuin aikaisempia tilanteita, koska täydellistä informaatiota aiemmista tilanteista ei kuitenkaan ole käytettävissä. Aikaisemmat toiminnot vaikuttavat kylläkin tarkasteltavana olevaan tilanteeseen, mutta jaossa luovuttaneiden pelaajien määrällä ei ole mielestäni riittävän selvää yhteyttä kanssapelaajien toimintaan. Oleellista on se, montaako kättä vastaan pelataan.

Asema tarkoittaa pelaajan asemaa suhteessa jakajan nappiin eli toimintavuoroa panostuskierroksen aikana. Asema on hyvin merkittävä tekijä pokeripöydässä, sillä varhaisessa asemassa oleva pelaaja joutuu tekemään päätöksensä ennen useita muita pelaajia, kun taas esimerkiksi viimeisestä asemasta näkee, miten kaikki muut pelaajat ovat toimineet, mikä helpottaa omaa peliä huomattavasti. Koska pelaajien määrä vaihtelee pelin kuluessa, myös asema voi muuttua. Tämän vuoksi asema esitetään suhdelukuna toimintavuoron ja pelaajien määrän suhteen. Esimerkiksi tilanteessa, jossa jaossa on mukana kuusi pelaajaa ja tarkasteltava pelaaja toimii toisena heistä, asema-muuttuja saa arvokseen 2 / 6 eli likiarvon 0,33.

Kierroksen panostusmäärä kertoo, kuinka monta pelaajaa on panostanut tai korottanut kierroksen aikana ennen tarkasteltavan pelaajan päätöstä. Jos esimerkiksi kierroksella ensimmäisenä toimiva pelaaja panostaa, toinen korottaa ja kolmas korottaa, on kierroksen panostusmäärä neljännen pelaajan kohdalla 3. Jokainen korotus kertoo vahvuudesta tai ainakin pitää yllä mahdollisuutta, että korotukset jatkuvat. Texas Hold'em limit -pokerissa korotusten määrä on rajoitettu yleensä kolmeen korotukseen pelikierroksessa (eli yksi panostus ja kolme korotusta), joten kierroksen panostusmäärä saa kokonaislukuarvoja väliltä [0,4].

Jokainen pelikierros pelataan yleensä hieman eri tavalla muihin pelikierroksiin verrattuna. Davidson [1999] on jättänyt tutkimuksessaan pre-flop-kierroksella eli ennen

flopin jakoa tapahtuvien toimintojen tarkastelun pois ja parantanut näin neuroverkkonsa tehokkuutta keskimäärin 55–70%:sta 75–90%:iin. Davidson [1999] teki kaksi olettamusta näin suuren muutoksen aiheuttajiksi: ensinnäkin ennen floppia pelattava peli eroaa suuresti flopin jälkeen pelattavasta pelistä, ja lisäksi ennen floppia tehtävien toimintojen ennustaminen on paljon vaikeampaa, koska käytettävissä on vähemmän informaatiota. Näitä ajatuksia tukee myös Pyysingin [2005] kirjassaan esittämä floppipelistä kertovan luvun aloitus: ”Lähtökorttien pelaaminen on melko mekaanista toimintaa. Flopissa peli muuttuu, koska kortteja tulee kerralla kolme lisää ja ne ovat kaikkien pelaajien yhteisiä. Floppi muuttaa kaiken.” Edellä esitetyn vuoksi pidän hyvin perusteltuna pitäytyä itsekin tarkastelemaan kierroksia flop, turn ja river, joita vastaavasti kierros-muuttuja saa arvot 1, 2 ja 3.

Edellinen toiminto on muuttuja, joka saa totuusarvon 1, mikäli pelaajan edellinen toiminto on ollut panostus tai korotus. Jos edellinen toiminto on ollut maksu, muuttuja saa arvon 0. Voi tuntua irrelevantilta tarkastella vain yhtä edellistä toimintoa, koska jokainen pelaajan aikaisempi toiminto vaikuttaa tulevan pelin arviointiin, mutta panostussuhde, sitoutuminen kierrokseen ja muut pelaajan toimintoihin liittyvät muuttujat antavat uskoakseni riittävästi lisäinformaatiota edellinen toiminto -muuttujan tueksi, jotta johtopäätöksiä voidaan tehdä.

Loput syötemuuttujat koskevat pöydässä näkyviä kortteja ja niihin liittyviä mahdollisuuksia. Texas Hold’em -pelissä isot kortit ovat vahvoja ja ne pelataan usein pidemmälle, minkä vuoksi tarkastellaan ässien ja kuninkaiden esiintymistä pöydässä. Koska molempia tarkastellaan omalla muuttujallaan, riittää todeta, onko ässää tai kuningasta pöydässä, eikä tarvitse tarkastella niiden määriä. Periaatteessa tilanne, jossa pöydässä on useampia kuninkaita kuin ässiä, antaa tällöin hieman vääristynyttä dataa. Tapaus on kuitenkin marginaalinen. Molemmat tapaukset käsitellään totuusarvoina siten, että muuttuja saa arvon 1, jos kyseessä oleva kortti esiintyy pöydässä. Muulloin muuttujan arvo on 0.

Kolmas pöydässä näkyviin kortteihin liittyvä muuttuja on värin mahdollisuus. Davidsonin [1999] tutkimuksessa on uskoakseni tarkasteltu tilannetta, jossa pöydässä on vähintään kolme korttia samaa maata, eli kahdella kyseistä maata edustavalla taskukortilla pelaajalla on väri. Samoin tehdään myös tässä työssä, eli värin veto saa arvon 1, kun pöydässä on 3 tai enemmän samaa maata. Tässä kohtaa voitaisiin ajatella, että kahden – tai jopa yhden, jos vasta floppi on jaettu – samaa maata olevan kortin esiintyminen antaa jo pelaajalle mahdollisuuden väriin, mutta koska mahdollisuus ei ole vielä realisoitunut (pelaajalla ei voi olla väriä tarkasteluhetkellä), en pidä tarpeellisena ottaa tällaisia tilanteita huomioon värin vetoon viittaavassa muuttujassa.

Davidson [1999] käytti joitakin sellaisia syötteitä, jotka olen itse jättänyt pois tavoitteenani tehostaa neuroverkkoa jättämällä syötteiksi vain oleellisimmat. Pottisuhde (pot ratio), joka on suhdeluku pelaajan pottiin laittamien pelimerkkien ja potissa

yhteensä olevien pelimerkkien määrän välillä, ei ole mielestäni kiinnostava siinä mielessä, että pottiin laitettu raha on joka tapauksessa menetetty. Potti voitetaan vasta, kun showdownissa näytetään voittokortit, muussa tapauksessa kaikki pöytään laitettu raha on hävittyä. Jotkut pelaajat laskevat myös tätä suhdelukua pelatessaan ja huomioivat sen toiminnassaan, mutta itse en pidä pottisuhdetta yhtä tärkeänä perusteena pelaajan tekemälle päätökselle kuin syötetaulukossa (taulukko 2) esittämiäni. Lisäksi sitoutuminen kierrokseen kertoo täsmällisemmin siitä, kuinka halukas pelaaja on jatkamaan panostusta tai maksamista aikaisempien toimien valossa. Last-Bets-To-Call -syötemuuttujan merkitys ei käy Davidsonin [1999] työstä täysin selväksi, mutta uskon sen tarkoittavan sitä, onko pelaaja viimeinen toimija kierroksella. Koska sisällytin asema-muuttujan omaan syötelistaani, pidän Last-Bets-To-Call -muuttujan mukaan ottamista nyt tarpeettomana.

Juhola [2000] esittää kaksi tärkeää huomiota kerättyyn dataan liittyen. Ensinnäkin tietoa pitää olla riittävästi. Juhola esittää karkean yleissäännön tarvittavan tiedon määrälle seuraavasti: ”Opetusjoukon tietojen lukumäärän pitäisi olla kymmenen kertaa neuroverkon vapaiden parametrien eli painokertoimien kokonaismäärä.” Rajatakseni aihetta aion käsitellä vain tilanteita, joissa aiempi pelaaja on korottanut, jolloin voin jättää tulosjoukosta pois sököttämis- ja panostustoiminnot. Näiden tulosvaihtoehtojen pois jättäminen uskoakseni parantaa myös neuroverkon toimintaa, koska sököttäminen ei oikeastaan kerro mitään, ja panostus on todennäköisempi bluffaus kuin korotus. Jäljelle jäävät toiminnot ovat siis luovutus, maksu ja korotus, eli käytetään yhteensä kolmea tulosneuronia. Käytettäessä 11 syöteneuronia ja yhtä piilokerrosta saataisiin tarvittavien datarivien minimimääräksi edellä esitetyllä laskentatavalla 10 x (11 x N + N x 3), missä N on piilokerroksen sisältämien neuronien määrä. Piilokerroksen neuronien määrä tarkentuu neuroverkon opetuksen yhteydessä, mutta se tuskin tulee olemaan suurempi kuin syöteneuronien määrä. Edellä esitetyn laskentatavan mukaan tarvittava syötedatan määrä työssäni on 1540 opetusesimerkkiä, jos käytetään arvoa 11 sekä syöte- että piiloneuronien lukumääränä. Tutkimuksen luonteesta johtuen syötedataan on hyvä saada erityyppisiä pelaajia, jotta saadaan muodostettua mahdollisimman kattava opetusjoukko, mutta toisaalta yhden pelaajan toimien seuraaminen useamman jaon ajalta on myös hyödyllistä. Käytetyn datan määrä lopullisessa työssä on 1134 riviä, joka edellä annetun kaavan mukaan sallisi karkeasti arvioiden maksimissaan 8 piiloneuronin käyttämisen, mikäli syötekerroksessa on 11 neuronia.

Toinen Juholan [2000] esittämä vaatimus on esimerkkien suhde tulosluokissa. Käytännössä on tärkeää, että opetusdatassa on jokaista kolmea tulosluokkaa (luovutus, maksu, korotus) suunnilleen yhtä paljon, eli tässä tapauksessa n. 33 % kutakin. Tarvittaessa opetusjoukon tietoja voidaan kahdentaa tämän tavoitteen saavuttamiseksi, mutta tällöin opetusjoukon pitäisi olla erotettu testijoukosta.

Olen kerännyt dataa yhteensä 1134 riviä. Dataa on kerätty yhteensä 252 jaetusta pokerikädestä – pelaajat voivat tehdä useita toimintoja jokaisessa jaossa. Osasta jakoja on kerätty toimintoja kaikkiin kolmeen tulostyyppiin, ja jotta check- ja bet-tyyppiset toiminnot saatiin jätettyä datajoukon ulkopuolelle, keskityin tarkastelemaan vain tilanteita, joissa vähintään yksi pelaaja on korottanut panostuskierroksen aikana. Koska (ainakin) limit-peleissä suurin osa luovutuksista tapahtuu ennen floppia tai heti flopin jaon jälkeen, luovutusten määrä tarkasteltavissa panostuskierroksissa oli selkeästi vähäisempi kuin korotusten ja erityisesti maksujen määrä, joka oli näistä kolmesta suurin. Tämän vuoksi osasta jakoja käsiteltiin vain korotukset ja luovutukset, ja osasta ainoastaan luovutukset. Lopulta 1134 rivissä on jokaista toimintoa yhtä paljon, eli 378 kappaletta. Näillä lukemilla täytetään Juholan [2000] esittämä vaatimus tulosluokkien keskinäisestä suhteesta.

Kerätty data sisältää toimintoja 477 eri pelaajalta. Osalta pelaajista on mukana vain yksi rivi, tarkastelluimmalta pelaajalta 32 riviä. Vaikka vastustajan toiminnan ennakoiminen pokerissa perustuukin yleensä pidempiaikaiseen saman pelaajan tarkkailuun, tämän työn yhteydessä mahdollisimman yleinen datajoukko antanee parhaan tuloksen, sillä ollaan mallintamassa yleistä, keskimääräistä tapaa toimia, ei tiettyjen pelaajien pelityyliä. Syötemuuttujat on pyritty valitsemaan siten, että ne kuvaisivat pelitilanteen mahdollisimman hyvin ja niiden avulla voitaisiin perustella pelaajien tekemiä päätöksiä. Korostettava kuitenkin on, että jokainen pelaaja toimii yksilöllisesti ja parhaat pelaajat hyvinkin vaihtelevalla pelityylillä, joten neuroverkon tapa ennustaa toimintoja antaa vain eräänlaisen keskimääräisen, tiettyihin muuttujiin perustuvan mallin.

Tiedon auditointiin liittyen riittää todeta, että koska lopullinen tietojen keruu ja tarkistus tapahtui melko pitkälti käsin, puuttuvia arvoja ei ole. Asia on myös tarkistettu tarkastelemalla tietokantaa, joka kerätystä datasta muodostettiin.

5.3. Tiedon esiprosessointi

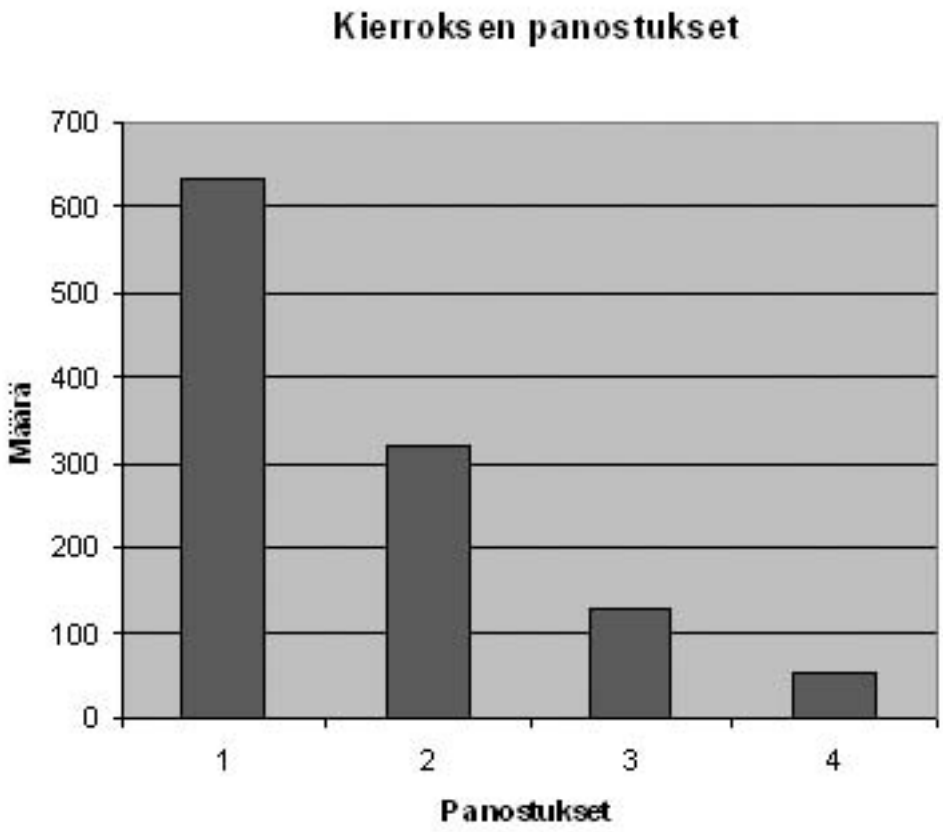
Tiedon keräämisen jälkeen tietoa pitää esiprosessoida, jotta se toimisi mahdollisimman luotettavasti ja tehokkaasti neuroverkossa. Käsittelen tässä Juholan [2000] esittämiä tiedon esiprosessointiin liittyviä tarkasteluja kerätyn pokeridatan suhteen.

Luvussa 6 kerron tarkemmin eri tavoin esiprosessoiduilla syötemuuttujilla opettamistani neuroverkoista. Neuroverkkoa on testattu kaiken kaikkiaan 87 erilaisella datajoukolla, ja vain 4 joukolle näistä on tehty täydellinen, jokaista syötearvoa koskeva esiprosessointi. Lopuissa tapauksissa vain osa syötemuuttujista – tai ei yhtäkään niistä – on esiprosessoitu. Käyn tässä kuitenkin läpi esiprosessoinnin jokaisen muuttujan osalta, koska jokainen yksittäinen muuttuja on esiprosessoitu useassa datajoukossa.

Sellainen parametri, jonka arvoista valtaosa edustaa samoja arvoja, on neuroverkossa tarpeeton. Tutkin jokaisen syötemuuttujan jakaumaa selvittääkseni näiden arvojen vaihtelevuuden tason. Valitsemissani syötemuuttujissa keräämäni

aineisto sisältää kuitenkin jokseenkin vaihtelevia parametrien arvoja. Vähiten vaihtelua esiintyy 2-arvoisissa parametreissa eli totuusarvoissa, ja näissäkin vähiten vaihteleva arvo on värin mahdollisuus, joka saa arvon 1 n. 19,4 %:ssa tapauksista. Lopuissa n. 80,6 %:ssa tapauksista värin mahdollisuus saa arvon 0, joten tämänkään muuttujan kohdalla ei voida mielestäni puhua vaihtelemattomasta parametrista. Näin ollen yhtäkään valituista muuttujista ei tarvitse poistaa vaihtelemattomuuden takia.

Tarkasteltaessa muuttujien jakaumia pidemmälle niissä havaitaan vinoutta. Vinoudesta puhuttaessa muuttujan jakauma ei ole normaalijakauman mukainen. Esimerkkitapaus vinosta muuttujasta nähdään kierroksen panostukset
-muuttujan kohdalla, jonka jakaumaa kuvaava histogrammi on esitetty kuvassa 8.



Kuva 8. Kierroksen panostukset -muuttujan jakauma on esimerkki vasemmalle vinosta muuttujasta.

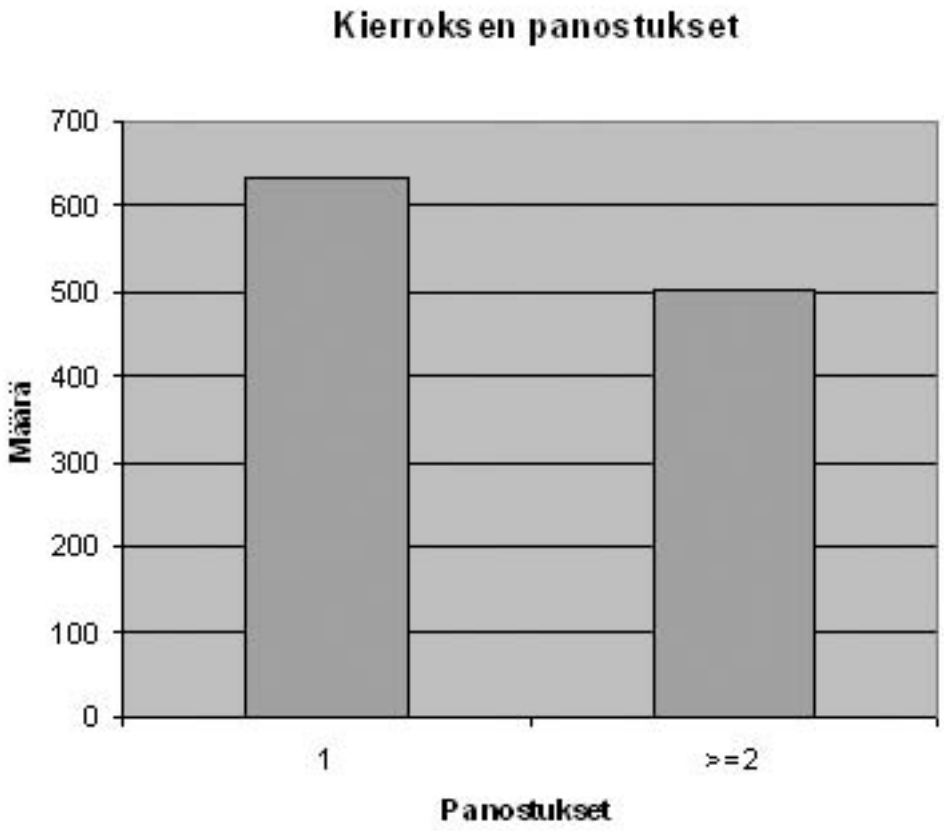
Juhola [2000] on esittänyt vinouden poistoon sovellettavan vinon funktion matemaattista käänteisfunktiota, mutta muuttujan arvoalueen ollessa pieni, kuten useiden aineistoni muuttujien kohdalla on, vinouden poisto onnistuu usein silmämääräisesti arvoalueita yhdistämällä. Kierroksen panostukset -muuttujan kohdalla vinoutta voidaan poistaa rajusti yhdistämällä arvot 2-4. Näin ollen tapauksia, joissa kierroksella on tehty 1 panostus, on 633 kpl, ja tapauksia joissa panostuksia on 2-4, on

501 kpl. Histogrammi tilanteesta, jossa kierroksen panostukset -muuttujan vinoutta on korjattu, on esitetty kuvassa 9. Jotta kierroksen panostusten määrä pysyisi suhteellisesti lähempänä oikeaa arvoaan silloin kun niitä on enemmän kuin 1, lasketaan jälkimmäinen korjatulle muuttujalle annettava arvo arvojen 2, 3 ja 4 suhteellisena keskiarvona kaavan 14 mukaisesti. Kaavassa m on arvon 2 esiintymien lukumäärä, n on arvon 3 esiintymien lukumäärä ja p on arvon 4 esiintymien lukumäärä.

$$\frac{2m+3n+4p}{(m+n+p)}$$

Kaava 14. Suhteellinen keskiarvo.

Näin saadaan tulokseksi 2,465. Kuvassa 9 esitetyn jakauman arvo >=2 (eli muuttujan arvot 2, 3 ja 4) voidaan siis korvata arvolla 2,465 tietokantataulussa, johon tallennetaan esiprosessoitu data. Arvon 1 tulee pysyä ennallaan.

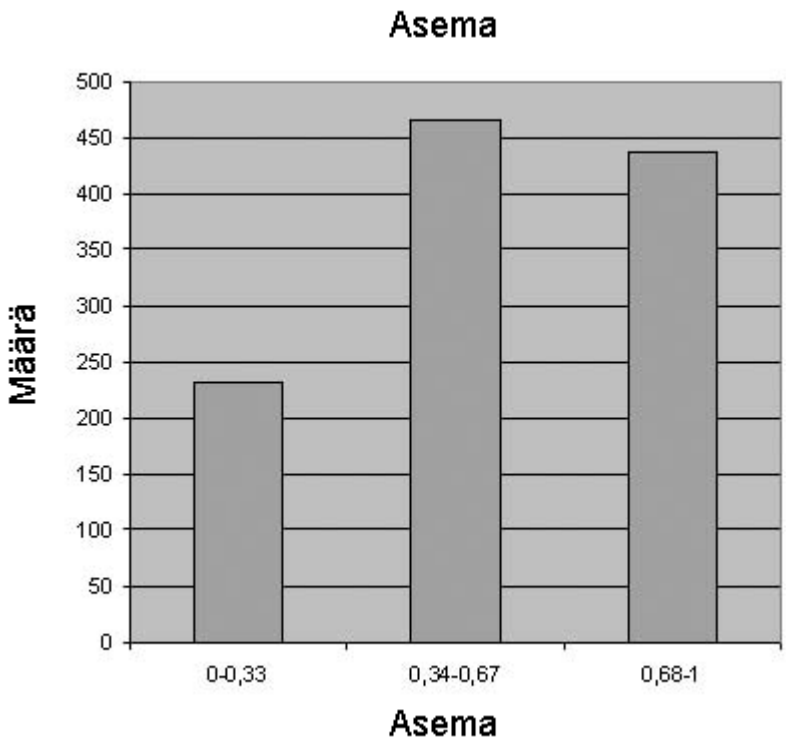


Kuva 9. Kierroksen panostukset -muuttujan jakauma muunnoksen jälkeen on lähempänä normaalijakaumaa kuin alkuperäinen.

Totuusarvomuuttujille, joita ovat sitoutuminen kierrokseen, edellinen toiminto, ässä pöydässä, kuningas pöydässä ja värin veto pöydässä, ei voida suorittaa merkittävää

esiprosessointia, koska mahdollisia arvoja on vain kaksi, eikä kumpikaan arvo ole mielestäni liian vallitseva toiseen nähden, kuten jo edellä totesin. Pelaajien määrä -muuttujan vinous on pyritty korjaamaan yhdistämällä arvot, joissa pelaajia on kierroksella mukana 4 tai enemmän. Alkuperäisessä datassa arvot välillä 4-9 muunnetaan arvoksi 4,49, joka on näistä arvoista laskettu suhteellinen keskiarvo. Suhteellinen keskiarvo lasketaan vastaavasti kuin kierroksen panostukset -muuttujan yhteydessä.

Koska asema-muuttuja on merkittävä ennen kaikkea suhteessa pelaajien määrään, muutetaan asema-muuttujan sisältö suhteeksi asema/pelaajamäärä ennen edellä kuvattua pelaaja-arvojen yhdistämistä. Kun tämän jälkeen yhdistellään arvoalueita kuvan 10 osoittamalla tavalla, saadaan kuvattua arvoalueiden jakauma.



Kuva 10. Muokatun asema-muuttujan histogrammi.

Pyysing [2005] jakaa positiot 10 hengen pöydässä taulukossa 3 esitetyllä tavalla. Koska useimmissa testiaineistossani käsitellyistä peleistä ei ole kymmentä osallistujaa, joudutaan ryhmittely erityyppisiin asemiin tekemään suhteellisesti. Asema-muuttujan arvot ovat väliltä [0, 1] silloin, kun se on laskettu suhteena asema / pelaajien määrä.

Paikka suhteessa jakajan nappiin, napista laskien	**Paikan kuvaus**
Paikat 1-2	pieni ja iso sokkopanos
Paikat 3-5	varhainen positio
Paikat 6-8	keskipositio
Paikat 9-10	myöhäinen positio

Taulukko 3 [Pyysing, 2005]. Peliasemat.

Jotta tulosten jakauma saataisiin mahdollisimman tasaiseksi, määritellään arvoalueet siten, että asema-muuttujan arvot 0-0,49 merkitsevät sokkopanosta tai varhaista positiota, 0,5-0,79 merkitsevät keskipositiota ja arvot 0,8-1 merkitsevät myöhäistä positiota.

Edellä mainitulla jaottelulla varhaisen position esiintymiä aineistossa on yhteensä 232 esiintymää, keskipositiota 466 esiintymää, ja myöhäistä positiota 436 esiintymää. Lasketaan näille ryhmille vielä arvot suhteellisina keskiarvoina, jolloin arvoiksi saadaan edellä esitetyssä järjestyksessä 0,3063; 0,5899 ja 0,9876.

Niistä muuttujista, joiden arvoalueita yhdisteltiin käyttäen suhteellisia keskiarvoja, olen tehnyt myös vaihtoehtoisen esiprosessoinnin. Vaihtoehtoisessa esiprosessoinnissa on annettu arvoalueille uudet arvot lineaarisesti. Esiprosessoinnin vaikutukset alkuperäiseen dataan suhteellisia keskiarvoja käyttäen, sekä näitä lineaarisia arvoja käyttäen on esitetty taulukossa 4.

Syötemuuttuja	Suhteellinen keskiarvo	Lineaarinen arvo
kierroksen panostukset	yhdistetään arvot 2-4 arvoksi 2,465	yhdistetään arvot 2-4 arvoksi 2
pelaajien määrä	yhdistetään arvot 4-9 arvoksi 4,490	yhdistetään arvot 4-9 arvoksi 4
asema	arvot 0-0,49 muutetaan arvoksi 0,3063 arvot 0,5-0,79 muutetaan arvoksi 0,5899 arvot 0,8-1 muutetaan arvoksi 0,9876	arvot 0-0,49 muutetaan arvoksi 0,33 arvot 0,5-0,79 muutetaan arvoksi 0,67 arvot 0,8-1 muutetaan arvoksi 1
pottikerroin	arvot 0,03-0,07 muutetaan arvoksi 0,0569 arvot 0,08-0,10 muutetaan arvoksi 0,0902 arvot 0,11-0,14 muutetaan arvoksi 0,1254 arvot 0,15-0,40 muutetaan arvoksi 0,2023	arvot 0,03-0,07 muutetaan arvoksi 0,05 arvot 0,08-0,10 muutetaan arvoksi 0,09 arvot 0,11-0,14 muutetaan arvoksi 0,125 arvot 0,15-0,40 muutetaan arvoksi 0,275
panostussuhde	arvot 0-0,17 muutetaan arvoksi 0,0113 arvot 0,20-0,80 muutetaan arvoksi 0,4078	arvot 0-0,17 muutetaan arvoksi 0,085 arvot 0,20-0,80 muutetaan arvoksi 0,50
sitoutuminen kierrokseen	ei muuteta alkuperäistä arvoa	ei muuteta alkuperäistä arvoa
edellinen toiminto	ei muuteta alkuperäistä arvoa	ei muuteta alkuperäistä arvoa
ässä pöydässä	ei muuteta alkuperäistä arvoa	ei muuteta alkuperäistä arvoa
kuningas pöydässä	ei muuteta alkuperäistä arvoa	ei muuteta alkuperäistä arvoa
värin veto pöydässä	ei muuteta alkuperäistä arvoa	ei muuteta alkuperäistä arvoa

Taulukko 4. Esiprosessoinnin vaikutus muuttujiin.

5.4. Tiedon koodaaminen ja normalisointi

Kierros-muuttuja voidaan koodata kolmella eri totuusarvolla, sillä vain yksi kierros voi olla kerrallaan käynnissä. Käytetään siis kolmea eri muuttujaa, flop, turn ja river, joista kukin saa totuusarvon 1 silloin kun muut saavat arvon 0. Näin kierros-muuttujan arvojen välille ei oleteta minkäänlaista lineaarista riippuvuussuhdetta, mikä on järkevää,

sillä mielestäni ei voida esimerkiksi väittää, että river-kierros olisi kauempana flop-kierroksesta kuin turn-kierroksesta. Tavallaan pelin luonne muuttuu tiettyyn suuntaan koko ajan kun jakoa jatketaan pidemmälle, mutta uskon pääseväni parempiin tuloksiin koodaamalla kierrosmuuttujan edellä kuvaamallani tavalla. Davidson [Davidson, 1999] on käyttänyt kierrosmuuttujan koodauksessa samaa tekniikkaa. Syötteiden määrä kasvaa tässä tapauksessa kahdella, jolloin käytössä on 13 syöteneuronia. 1134 datarivillä ja 3 tulosneuronilla tämä tarkoittaa sitä, että aiemmin esitetyn datarivien määrään liittyvän säännön perusteella sopiva piiloneuronien määrä olisi 7, kun käytetään yhtä piilokerrosta.

Koska jokaisella neuroverkon syöteparametrilla on oma arvoalueensa, neuroverkko voi painottaa suuremman vaihteluvälin syöteparametreja liiaksi ja pienen vaihteluvälin parametreja liian vähän. Tämän vuoksi on tarpeellista normalisoida kaikki syötteet. Juhola [2000] esittää normalisointiin tavan, jossa jokaisesta syötearvosta poistetaan syöteparametrin tietojen keskiarvo, ja tämän jälkeen jaetaan tulos vielä syöteparametrin tietojen keskihajonnalla. Tällöin keskihajonta-arvo kuvautuu arvoksi 1 ja keskiarvo arvoksi 0. Tässä työssä käytän syötearvojen normalisointiin Juholan [2000] esittämää tapaa.

5.5. Testijoukon valinta ja neuroverkon rakentaminen

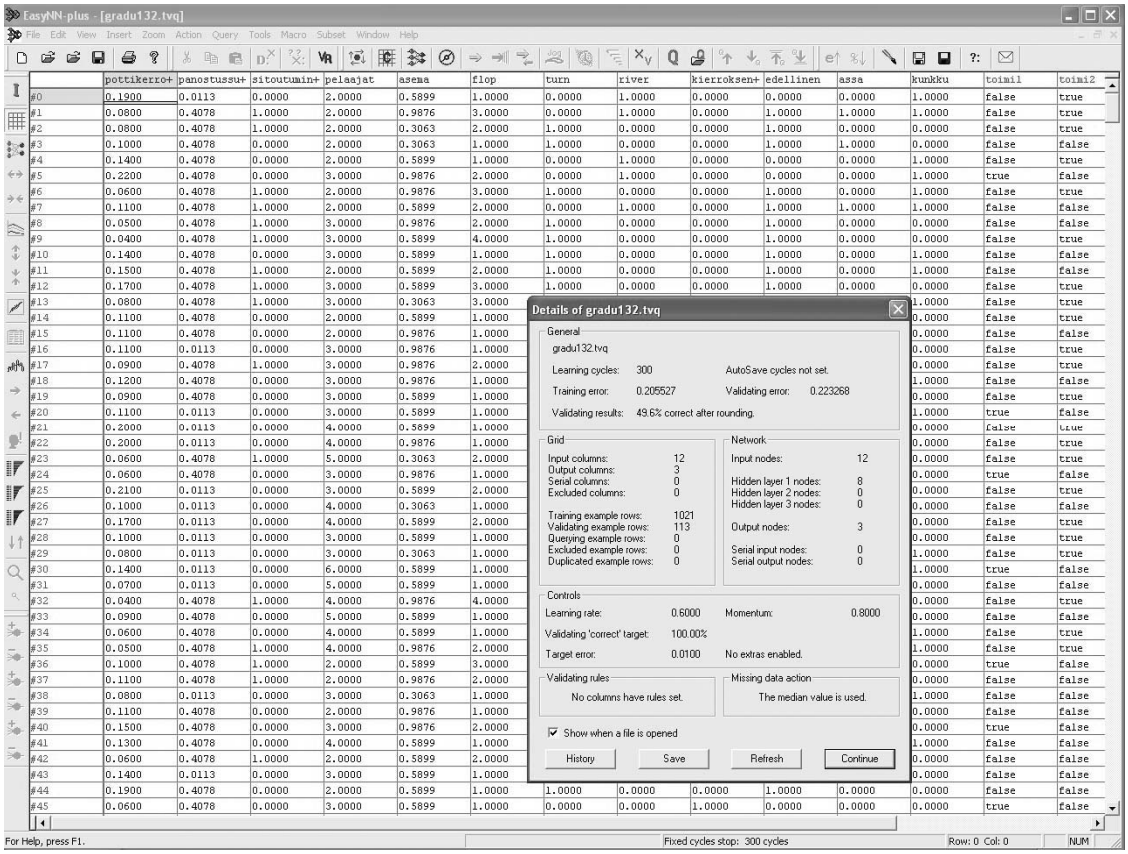
Aineiston pienehkön koon vuoksi käytetään neuroverkon testaamiseen ristiinvalidointia. Ristiinvalidointi toteutetaan kymmenenkertaisena ristiinvalidointina, jolloin satunnaisesti valittu kymmenesosa opetusjoukosta toimii aina kulloinkin testijoukkona lopuille 9 kymmenesosalle, joita käytetään opetusjoukkona. Validointi suoritetaan siis 10 osassa, ja validoinnin tulos on näiden 10 osavalidoinnin keskiarvo.

Tässä työssä käsitellään luokittelu-tyyppistä ongelmaa ja neuroverkko on rakenteeltaan monikerroksinen perceptron, joka on luultavasti suosituin neuroverkkotyyppi.

Käytän neuroverkkoa, jossa on yksi piilokerros, sillä Juholan [2000] mukaan piilokerroksia ei tarvita monta. Kolmella aktiivisella kerroksella, jossa on siis 2 piilokerrosta, voidaan Juholan [2000] mukaan kuvata periaatteessa kaikki mielekkäät kuvaukset. Koska mallinnettava systeemi ei tutkimukseni tapauksessa ole mitenkään erityisen monimutkainen, uskon pärjääväni yhdellä piilokerroksella.

Juholan [2000] mukaan piilosolmujen liiallinen määrän kasvattaminen monimutkaistaa systeemin mallintamaa funktiota, minkä vuoksi voi käydä niin, että verkko ei löydä yleistä ratkaisua, vaan tulee liian spesifiksi tai yliopetetuksi eli ylisovitetuksi. Ylisovitus tarkoittaa sitä, että neuroverkko ei kykene luomaan yleisiä malleja, vaan toimii vain sellaisten satunnaisten mallien varastona, jotka tarjoavat vähän yhtäläisyyksiä joukkoon, josta nämä tapaukset on muodostettu [Hanson et al., 2001]. Tässä työssä aloitan testaamisen 8 piiloneuronilla ja kokeilen käytännön testaamisen loppuvaiheessa piiloneuronien määrän muuttamisen vaikutusta tulokseen.

Käytän neuroverkon muodostamiseen EasyNN Plus -ohjelmaa (kuva 11). Kyseisen ohjelman kokeiluversio on saatavilla ilmaiseksi osoitteesta http://www.easynn.com/, mutta koska kokeiluversiossa on rajoitettu opetusdatan määrä maksimissaan 100 riviin, käytän tämän työn yhteydessä ohjelman maksullista versiota. Ohjelmalle voidaan syöttää opetusdataa käsin tai tiedostosta, minkä jälkeen määritellään halutut rivit opetusjoukoksi ja toiset rivit validointijoukoksi. Tämän jälkeen voidaan muodostaa, opettaa ja validoida neuroverkko halutuilla neuroverkon ominaisuuksilla. On tärkeää huomata, että neuroverkon validointijoukon tulee olla eri testijoukosta eriävä.



Kuva 11 [EasyNN Plus, 2008]. Kuvakaappaus EasyNN Plus -ohjelmasta.

Kerron lisää EasyNN Plus -ohjelman toiminnasta luvun 6 alussa.

6. Neuroverkon opetus, validointi ja testaus

Kuten edellä luvussa 5 mainitsin, käytin neuroverkon opetukseen, validointiin ja testaukseen EasyNN Plus -nimistä ohjelmaa. EasyNN Plus käyttää aktivaatiofunktiona sigmoidifunktiota, jonka kuvaaja on esitetty kuvassa 1.

Neuroverkon toimintaa testattiin 10-kertaisella ristiinvalidoinnilla 87 eri datajoukolla, joissa muuttujien määrää ja esiprosessointia sekä neuroverkon ominaisuuksia on muutettu opetus- ja testauskertojen välillä pyrkien koko ajan parempaan tulokseen.

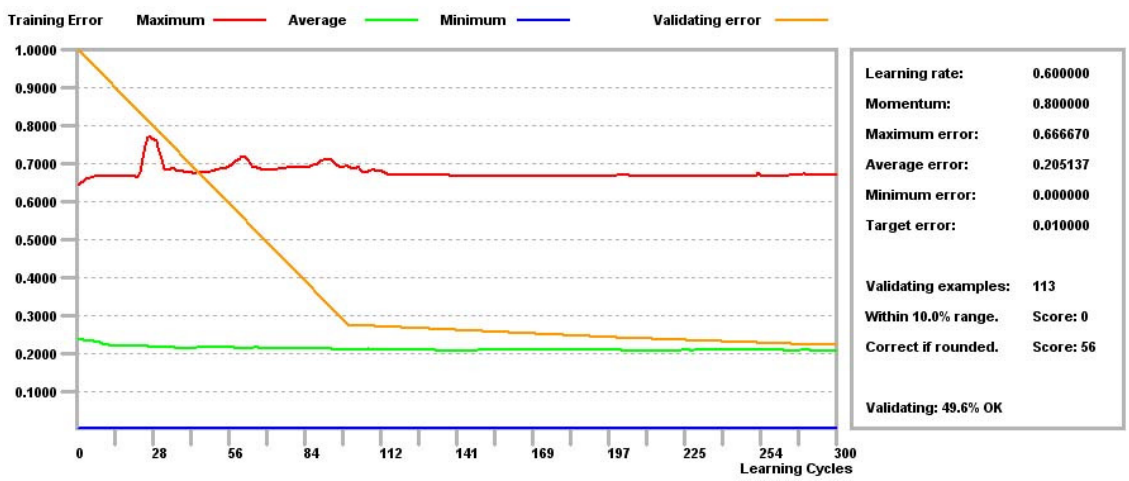
Sain nostettua tuloksen alkuperäisen, täysin muokkaamattoman datan opetuksenjälkeisen validoinnin antamasta arvosta 39,29 arvoon 51,66. Neuroverkon validoinnin tulokset ovat käytännössä prosenttilukuja, jotka kertovat kuinka monta prosenttia testijoukon riveistä sai oikean tuloksen opetettua neuroverkkoa käyttämällä.

Neuroverkon kyky ennustaa vastustajan toiminto nousi siis yli 12 prosenttiyksikköä, mitä voidaan mielestäni pitää kohtuullisena tuloksena. Arvaamalla oikean tuloksen saisi joka kolmas kerta, muokatulla datalla opetettu neuroverkko sen sijaan antaa oikean tuloksen useammin kuin joka toinen kerta.

Dataa ja neuroverkkoa muokkaamalla olen toteuttanut yhteensä 87 erilaista datajoukkoa. Näistä saatavat tulokset on muodostettu 10-kertaisella ristiinvalidoinnilla siten, että kullakin opetuskerralla 1/10 datajoukon riveistä toimii testijoukkona (käytännössä 113 tai 114 riviä) ja loput 9/10 opetusjoukkona. Testijoukot on luotu satunnaisesti käyttäen apuna itse koodattua PHP-scriptiä.

Jokaisen datajoukon kohdalla pyrin luomaan potentiaalisia vaihtoehtoja paremman tuloksen saavuttamiseksi muodostaen uudet datajoukot aina jossain määrin vanhojen pohjalta. Datan esiprosessointi tehtiin pääasiassa suoraan tietokantaan, ja normalisointi toteutettiin omatekoisen PHP-scriptin avulla luomalla tietokantaan uusi taulu normalisoidulle versiolle vanhan taulun datasta. Normalisoinnin operaatiot suoritettiin myös em. scriptin avulla. Datan siirtäminen tietokannasta EasyNN Plus -ohjelmalle toteutettiin tekstitiedostosta, ja em. tekstitiedosto luotiin myös omatekoisella PHP-scriptillä.

Jokaisella opetuskerralla data käytiin läpi 300 kertaa, sillä suurempi läpikäyntien määrä ei olisi merkittävästi vaikuttanut tulokseen. Tämä nähdään myös EasyNN Plus -ohjelman opetuksesta piirtämästä kuvaajasta (kuva 12) erään opetus- ja validointiajon yhteydessä. Validointivirheen (oranssi käyrä) pienentyminen hidastuu ratkaisevasti n. 100 epookin jälkeen.

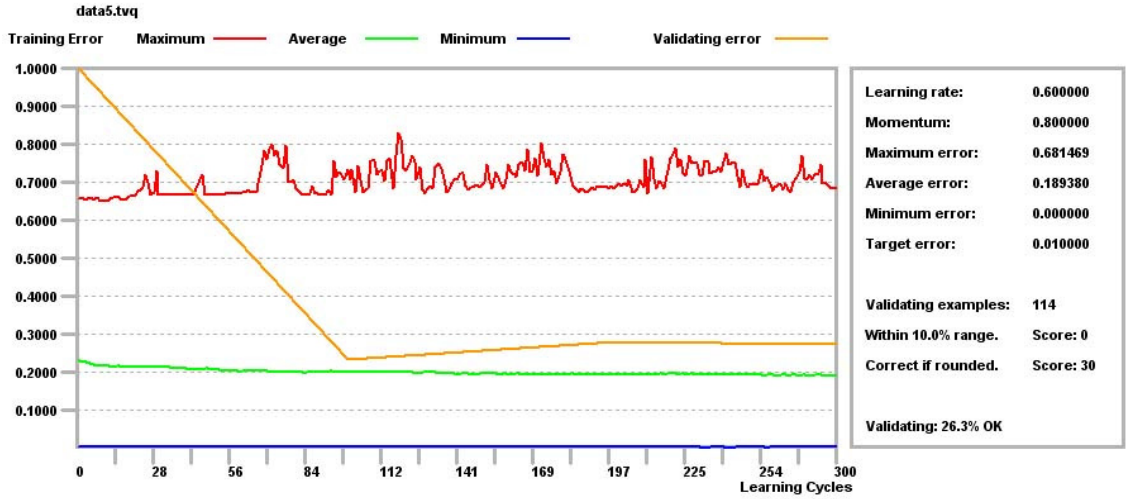


Kuva 12 [EasyNN Plus, 2008]. EasyNN Plus-ohjelman piirtämästä kuvaajasta nähdään, että validointivirheen pieneneminen hidastuu huomattavasti n. 100 epookin jälkeen.

Validointivirhe on neuroverkon opetuksen aikana mitattava virhe neuroverkossa saadun tuloksen ja odotetun tuloksen välillä. Neuroverkon opetuksen aikana tätä validointivirhettä pyritään pienentämään säätämällä neuroverkossa esiintyviä painoarvoja. Mitä pienemmäksi tämä virhe saadaan, sitä paremman tuloksen neuroverkko antaa validointijoukolla testatessa – tähän tulokseen kiteytyy neuroverkon suorituskyky.

Neuroverkon opetus on hyvä lopettaa ajoissa, jotta yliopetusta ei tapahtuisi eikä neuroverkon yleistämiskyky heikkenisi. Kuvan 12 esittämästä kuvaajasta nähdään, ettei validointivirhe eikä näin ollen neuroverkon tuottama validointitulos enää muutu ratkaisevasti 100 epookin jälkeen, minkä vuoksi neuroverkon opetus on järkevää lopettaa suhteellisen pian tämän jälkeen.

Kuvassa 13 nähdään toinen esimerkki EasyNN-ohjelman muodostamasta oppimiskuvaajasta. Tähän kuvaan on tarkoituksella valittu sellaisen validointi, josta on saatu selkeästi huonompi tulos kuin kuvan 12 esittämässä tilanteessa, jossa validointitulos on 49,6 %.



Kuva 13 [EasyNN Plus, 2008]. Toinen esimerkki EasyNN-ohjelman antamista käyristä opetuksen yhteydessä.

Pääpiirteiltään kuvan 13 kuvaajat seurailevat samansuuruisia tasoja kuin kuvan 12 käyrät. Muutamia selkeitä eroja on kuitenkin havaittavissa. Keskiarvovirheen käyrä (vihreä) on melkein samanlainen kuin kuvassa 12. Sen sijaan maksimivirheen käyrän (punainen) muoto on muuttunut radikaalisti. Maksimivirheen kuvaajassa esiintyy paljon enemmän vaihtelua kuvassa 13 kuin kuvassa 12, mikä kertoo todennäköisimmin siitä, että oikeanlaisten painoarvojen määrittely neuroverkolle on ollut jälkimmäisessä tapauksessa hankalampaa. Tämä määrittelyn vaikeus heijastuu myös suoraan validointitulokseen, joka on jälkimmäisessä tapauksessa 26,3 %. Myös validointivirheen käyrässä nähdään selvä ero: validointivirheen vähenemisnopeuden laskiessa rajusti n. 100 epookin jälkeen validointivirhe jatkaa silti vähenemistä aikaisemmassa tapauksessa (kuva 12). Myöhemmässä tapauksessa (kuva 13), validointivirhe lähtee jopa uudelleen kasvuun, mikä ennakoi myös aiempaa huonompaa validointitulosta.

Taulukosta 5 nähdään tulosneuronien esiintyminen eri validointijoukoissa. Tulosneuronit jakautuvat validointijoukoissa mielestäni riittävän tasaisesti. Joissakin neuroverkon opetus- ja validointikerroissa EasyNN Plus -ohjelma halusi muuttaa validointijoukon rivejä opetusjoukon riveiksi. Tämä johtui siitä, että validointijoukko sisälsi jonkin syötemuuttujan kohdalla arvon, jota ei löytynyt opetusjoukosta. Esimerkki ohjelman virheilmoituksesta em. tilanteessa: ”The grid contains validating data outside the training range. Input column 1 ‘panostussuhde’, ‘0.8000’ is not in the training range.” Ratkaisuna tällaisessa tilanteessa muutettiin kyseinen validointijoukon rivi opetusjoukon riviksi. Vaihtoehtona olisi ollut pudottaa rivi pois molemmista joukoista.

Validointijoukko	Luovutus	Maksu	Korotus
#1	31	43	39
#2	31	41	41
#3	42	43	29
#4	41	31	41
#5	35	35	44
#6	43	37	34
#7	36	32	45
#8	38	38	37
#9	45	36	33
#10	36	42	35

Taulukko 5. Tulosneuronien jakautuminen validointijoukoissa.

Ensimmäinen datan testaus neuroverkossa tehtiin täysin käsittelemättömällä datalla ja ohjelman tarjoamilla oletusarvoisilla neuroverkon asetuksilla. Piiloneuronien määräksi määriteltiin kuitenkin 8. Opetuskerroin, jota oletuksena käytettiin, on 0,6 ja momentti 0,8. Validointitulos täysin käsittelemättömällä datalla on 39,29 %. Koska tulosneuroneita on yhteensä 3, tulos voidaan arvata oikein n. 33,3 % tapauksista, joten käsittelemätön data antaa jo hieman paremman tuloksen kuin puhdas arvaus.

Seuraavat viisi testauskertaa sisälsivät kaikki luvussa 5 esitellyn datan esiprosessoinnin (suhteellisilla ja lineaarisilla keskiarvoilla, taulukko 4) ja datan normalisoinnin muodostamat variaatiot. Taulukosta 6 nähdään näiden datajoukkojen kuvaukset sekä 10-kertaisella ristiinvalidoinnilla saadut tulokset. Jokainen taulukon data-numero vastaa yhtä opetus- ja validointikertaa erilaisella datalla neuroverkossa. Numerointi alkaa taulukossa 6 numerosta 2 siksi, että alkuperäisen, käsittelemättömän datan opetus ja testaus toteutettiin numerolla 1.

Data #	Kuvaus	Tulos
2	Esiprosessoimaton ja normalisoitu data.	38,88
3	Esiprosessoitu (suhteellinen ka.) ja normalisoimaton data	38,31
4	Esiprosessoitu (suhteellinen ka.) ja normalisoitu data	37,10
5	Esiprosessoitu (lineaarinen ka.) ja normalisoimaton data	37,09
6	Esiprosessoitu (lineaarinen ka.) ja normalisoitu data	36,11

Taulukko 6. Datajoukot 2–6.

Kuten taulukon 6 tulos-sarakkeesta nähdään, neuroverkon tulosta ei saada parannettua koko datajoukon esiprosessoinnilla ja normalisoinnilla. Muutokset tuloksessa ovat suhteellisen pieniä mutta negatiivisia. Datan läpiprosessointi ikään kuin ”tylsyttää” datan kauttaaltaan, eikä parannusta näin ollen synny. Seuraavaksi pyrinkin

parempiin tuloksiin keskittymällä vain osaan muuttujista kerrallaan yrittäen selvittää, mitkä muutokset vaikuttavat tulokseen parhaalla mahdollisella tavalla.

Seuraavat 10 testausta tein esiprosessoimalla vain yhden muuttujan kerrallaan normalisoimatta dataa missään vaiheessa. Tulokset näistä testeistä ovat nähtävillä taulukossa 7. Totuusarvomuuttujia ei ole esiprosessoitu, koska esiprosessointi on suoritettu kuten luvussa 5 on esitetty. Kokeilin erikseen suhteellisen keskiarvon mukaista esiprosessointitapaa, sekä lineaariseksi nimeämääni esiprosessointia.

Data #	**Kuvaus**	**Tulos**
7	Vain syötemuuttuja pottikerroin esiprosessoitu (suhteellinen ka.)	39,11
8	Vain syötemuuttuja panostussuhde esiprosessoitu (suhteellinen ka.)	42,55
9	Vain syötemuuttuja pelaajien määrä esiprosessoitu (suhteellinen ka.)	38,57
10	Vain syötemuuttuja asema esiprosessoitu (suhteellinen ka.)	39,29
11	Vain syötemuuttuja kierroksen panostukset esiprosessoitu (suhteellinen ka.)	36,30
12	Vain syötemuuttuja pottikerroin esiprosessoitu (lineaarinen ka.)	38,47
13	Vain syötemuuttuja panostussuhde esiprosessoitu (lineaarinen ka.)	42,55
14	Vain syötemuuttuja pelaajien määrä esiprosessoitu (lineaarinen ka.)	37,59
15	Vain syötemuuttuja asema esiprosessoitu (lineaarinen ka.)	40,45
16	Vain syötemuuttuja kierroksen panostukset esiprosessoitu (lineaarinen ka.)	36,30

Taulukko 7. Datajoukot 7–16.

Taulukosta 7 nähdään, että panostussuhteen ja aseman esiprosessoinnilla näyttäisi olevan positiivinen vaikutus tulokseen. Datajoukkojen 7–16 kohdalla nähdään, että vain yhdessä tapauksessa viidestä muuttujien esiprosessointi lineaarisen keskiarvon mukaan on tuottanut paremman tuloksen kuin esiprosessointi suhteellisen keskiarvon mukaan. Kahdessa tapauksessa esiprosessoinnilla ei ole ollut vaikutusta tulokseen. Tämän vuoksi en jatkossa käytä enempää aikaa eri metodeilla esiprosessoitavien datajoukkojen luomiseen, vaan jatkan suhteellista keskiarvoa käyttäen.

Seuraavaksi päätin kokeilla normalisoinnin vaikutusta edellä saatuihin tuloksiin. Normalisoinnin vaikutus on nähtävissä taulukossa 8.

Data #	Kuvaus	Tulos
17	Data 7 normalisoituna	39,90
18	Data 8 normalisoituna	41,75
19	Data 9 normalisoituna	40,18
20	Data 10 normalisoituna	38,88
21	Data 11 normalisoituna	37,01
22	Data 12 normalisoituna	37,95
23	Data 13 normalisoituna	41,75
24	Data 14 normalisoituna	41,07
25	Data 15 normalisoituna	39,29
26	Data 16 normalisoituna	37,01

Taulukko 8. Datajoukot 17–26.

Viidessä tapauksessa kymmenestä normalisointi paransi tulosta normalisoimattomiin esiprosessoituihin datajoukkoihin verrattuna, viidessä heikensi. Alkuperäisen käsittelemättömän datan tulokseen verrattuna tulos on parantunut datajoukoissa 17, 18, 19, 23, ja 24, minkä lisäksi tulos on joukossa 25 sama. Nämä vastaavat datajoukkoja, joissa käsitellyt muuttujat ovat pottikerroin, panostussuhde, pelaajien määrä sekä asema.

Seuraavaksi testasin kaikki variaatiot, joissa useampi kuin yksi edellä mainituista viidestä muuttujasta on esiprosessoitu. Näin saatiin aikaan 11 uutta datajoukkoa, joiden kuvaukset ja tulokset ovat nähtävillä taulukossa 9.

Data #	Kuvaus	Tulos
27	Panostussuhde ja pottikerroin esiprosessoitu (suhteellinen ka.)	36,05
28	Panostussuhde ja pelaajien määrä esiprosessoitu (suhteellinen ka.)	38,88
29	Panostussuhde ja asema esiprosessoitu (suhteellinen ka.)	44,30
30	Pottikerroin ja pelaajien määrä esiprosessoitu (suhteellinen ka.)	39,17
31	Pottikerroin ja asema esiprosessoitu (suhteellinen ka.)	38,75
32	Pelaajien määrä ja asema esiprosessoitu (suhteellinen ka.)	42,13
33	Panostussuhde, pottikerroin ja pelaajien määrä esiprosessoitu (suhteellinen ka.)	39,04
34	Panostussuhde, pottikerroin ja asema esiprosessoitu (suhteellinen ka.)	36,05
35	Panostussuhde, pelaajien määrä ja asema esiprosessoitu (suhteellinen ka.)	39,23
36	Pottikerroin, pelaajien määrä ja asema esiprosessoitu (suhteellinen ka.)	39,17
37	Panostussuhde, pottikerroin, pelaajien määrä ja asema esiprosessoitu (suhteellinen ka.)	39,04

Taulukko 9. Datajoukot 27–37.

Datajoukoissa 38–48 (taulukko 10) testattiin datajoukkojen 27–37 data normalisoituna. Mikään näistä normalisoitujen datajoukkojen tuottamista validointituloksista ei ollut parempi kuin datajoukon 29 tulos, joten yhä paremman tuloksen etsimistä jatketaan datajoukon 29 perusteella. Datajoukossa 29 panostussuhde ja asema on esiprosessoitu käyttäen suhteellista keskiarvoa.

Data #	**Kuvaus**	**Tulos**
38	Data 27 normalisoituna	38,00
39	Data 28 normalisoituna	44,10
40	Data 29 normalisoituna	42,81
41	Data 30 normalisoituna	38,55
42	Data 31 normalisoituna	38,91
43	Data 32 normalisoituna	39,89
44	Data 33 normalisoituna	40,86
45	Data 34 normalisoituna	39,07
46	Data 35 normalisoituna	43,92
47	Data 36 normalisoituna	38,55
48	Data 37 normalisoituna	40,36

Taulukko 10. Datajoukot 38–48.

Seuraavissa 10 datajoukossa poistetaan kussakin 1 muuttuja kerrallaan syötemuuttujista (taulukko 11). Esiprosessointi on suoritettu kuten datajoukossa 29.

Data #	**Kuvaus**	**Tulos**
49	Data 29, pottikerroin poistettu	40,91
50	Data 29, panostussuhde poistettu	37,42
51	Data 29, sitoutuminen kierrokseen poistettu	40,41
52	Data 29, pelaajien määrä poistettu	39,48
53	Data 29, asema poistettu	39,94
54	Data 29, kierroksen panostukset poistettu	43,58
55	Data 29, edellinen toiminto poistettu	39,53
56	Data 29, ässä pöydässä poistettu	41,31
57	Data 29, kuningas pöydässä poistettu	45,88
58	Data 29, värin veto pöydässä poistettu	50,67

Taulukko 11. Datajoukot 49–58.

Syötemuuttujien poistaminen on vaikuttanut tähän asti tehdyistä yksittäisistä toimista eniten tuloksen paranemiseen. Totuusarvomuuttujien kuningas pöydässä ja värin veto pöydässä poistamisella oli positiivinen vaikutus aikaisempaan tulokseen. Jälkikäteen ajateltuna voidaan siis todeta, että vähiten hajontaa tarjoavat 2-arvoiset muuttujat olisi mahdollisesti kannattanut jättää pois alkuperäisestä syötemuuttujajoukosta.

Taulukossa 12 ensimmäinen datajoukko, numero 59, on edellisistä datajoukoista muodostettu yhdistelmä, jossa muuttujat kuningas pöydässä ja väri on poistettu. Tämä ei parantanut tulosta. Datajoukoissa 60–69 on muutettu piiloneuroneiden määrää neuroverkossa käyttäen edellä saaduista tuloksista parasta, eli datajoukkoa 58.

Käsitellään tapaukset, joissa piiloneuronien määrä on välillä 1–11. Autio ja muut [Autio et al., 2006] esittävät omassa samankaltaisen verkon avulla tehdyssä tutkimuksessaan, että useimmiten on parasta valita niin vähän piiloneuroneita kuin mahdollista, jotta vähennetään yliopetuksen riskiä. Tämän sekä aineistoni suhteellisen pienen määrän huomioon ottaen yli 11 piiloneuronin verkkoja ei ole mielestäni järkevää tarkastella.

Data #	**Kuvaus**	**Tulos**
59	Data 29, kuningas ja väri poistettu	42,79
60	Data 58, 1 piiloneuroni	50,76
61	Data 58, 2 piiloneuronia	49,26
62	Data 58, 3 piiloneuronia	50,67
63	Data 58, 4 piiloneuronia	47,75
64	Data 58, 5 piiloneuronia	37,32
65	Data 58, 6 piiloneuronia	49,89
66	Data 58, 7 piiloneuronia	45,45
67	Data 58, 9 piiloneuronia	49,24
68	Data 58, 10 piiloneuronia	46,70
69	Data 58, 11 piiloneuronia	45,55

Taulukko 12. Datajoukot 59–69.

Kuten taulukosta 12 nähdään, 1 piiloneuronilla toteutettu neuroverkko antoi vielä hieman paremman tuloksen kuin alkuperäinen 8 piiloneuronilla toteutettu. 3 piiloneuronilla toteutettu neuroverkko antoi saman tuloksen kuin alkuperäinen. Ero tässä tapauksessa on kuitenkin hyvin pieni, ja kyseessä ovat neuroverkon eivätkä sille annettujen syötearvojen ominaisuudet, minkä vuoksi kokeilin neuroverkkoa vielä sekä 1 että 8 piiloneuronin avulla toteutettuna muuttamalla lisäksi oppimiskerrointa (learning rate) ja momenttia (momentum). Alkuperäisessä testauksessa oppimiskerroin on saanut arvon 0,6 ja momentti arvon 0,8.

Testataan neuroverkkoa eri asetuksilla taulukossa 13 kuvatuilla variaatioilla.

Data #	Kuvaus	Tulos
70	Data 58, 1 piiloneuroni, oppimiskerroin 0,2; momentti 0,2	42,77
71	Data 58, 1 piiloneuroni, oppimiskerroin 0,2; momentti 0,5	45,44
72	Data 58, 1 piiloneuroni, oppimiskerroin 0,2; momentti 0,8	49,87
73	Data 58, 1 piiloneuroni, oppimiskerroin 0,5; momentti 0,2	47,47
74	Data 58, 1 piiloneuroni, oppimiskerroin 0,5; momentti 0,5	50,22
75	Data 58, 1 piiloneuroni, oppimiskerroin 0,5; momentti 0,8	49,06
76	Data 58, 1 piiloneuroni, oppimiskerroin 0,8; momentti 0,2	50,49
77	Data 58, 1 piiloneuroni, oppimiskerroin 0,8; momentti 0,5	51,66
78	Data 58, 1 piiloneuroni, oppimiskerroin 0,8; momentti 0,8	50,31
79	Data 58, 8 piiloneuronia, oppimiskerroin 0,2; momentti 0,2	43,88
80	Data 58, 8 piiloneuronia, oppimiskerroin 0,2; momentti 0,5	43,93
81	Data 58, 8 piiloneuronia, oppimiskerroin 0,2; momentti 0,8	42,88
82	Data 58, 8 piiloneuronia, oppimiskerroin 0,5; momentti 0,2	44,37
83	Data 58, 8 piiloneuronia, oppimiskerroin 0,5; momentti 0,5	44,48
84	Data 58, 8 piiloneuronia, oppimiskerroin 0,5; momentti 0,8	46,52
85	Data 58, 8 piiloneuronia, oppimiskerroin 0,8; momentti 0,2	44,21
86	Data 58, 8 piiloneuronia, oppimiskerroin 0,8; momentti 0,5	46,94
87	Data 58, 8 piiloneuronia, oppimiskerroin 0,8; momentti 0,8	44,59

Taulukko 13. Datajoukot 70–87.

Paras tulos saadaan, kun käytetään yhtä piiloneuronia ja oppimiskertoimeksi asetetaan 0,8 sekä momentiksi 0,5 (datajoukko 77). Tällöin neuroverkon validointitulos on 51,66, joka on myös lopullinen tulokseni.

7. Tulokset ja yhteenveto

Tätä tutkimusta tehdessäni huomasin joissakin vaiheissa vaihtoehtoisia tapoja toimia, ja saatuani oman tutkimukseni päätökseen pystyn esittämään joitakin oletuksia siitä, miten tutkimuksessa olisi voitu päästä parempaan tulokseen.

Suurin yksittäinen muutos neuroverkon validointituloksessa tapahtui, kun pudotettiin yksi totuusarvo-tyyppisistä muuttujista pois. Muuttujien valinnalla on tietysti suuri merkitys validointituloksen kannalta, ja edellinen huomio tukee tätä. Muuttujien valintaa olisi voinut pohtia vielä syvällisemmin, ja neuroverkon toimintaa olisi voinut testata erillisillä muuttujajoukoilla laajemminkin, mikä ei ollut tämän tutkimuksen puitteissa mahdollista.

Eri datajoukkojen muodostamisen olisi voinut tehdä eri järjestyksessä, aloittaen juuri esimerkiksi käytettävien muuttujien valinnasta ja muokaten vasta sen jälkeen niiden arvoja.

Datajoukkojen esiprosessointi olisi myös voitu suunnitella paremmin etukäteen. Joissakin työn toteutuksen vaiheissa huomasin, että eri tavalla esiprosessoidulla muuttujalla olisi voinut olla positiivinen vaikutus lopputuloksiin.

Tulos on parempi kuin arvaamalla saatu, mutta pokeriammattilainen saattaisi voittaa neuroverkon tuloksen. Tässä yhteydessä ei ole ollut mahdollista tutkia pokeriammattilaisen kykyä ennustaa vastaavia tilanteita.

Yksi syy huonompaan tulokseen kuin esimerkiksi Davidsonin tutkimuksessa on se, että olen kerännyt datani peleistä, joissa pelataan oikealla rahalla. Pelattaessa oikealla rahalla pelaajien taso on yleensä korkeampi, ja pelaajat vaihtelevat pelityyliään useammin. Davidson myöntää pelityylin nopean vaihtelun olleen yksi mallintamista vaikeuttava tekijä, mutta uskon, että hänen aineistossaan vaihtelua on vähemmän, sillä hän on kerännyt datansa IRC-pokeri -peleistä, joissa ei liiku rahaa.

Myös se, että omassa aineistossani oli paljon pelaajien välistä hajontaa, heikensi luultavasti tulosta. Vaikka tietyt perustaktiikat ovat keskenään yhteneviä, jokaisella pelaajalla on kuitenkin pitkällä aikavälillä tarkasteltuna omaperäinen tapansa pelata.

Käsittelen lopuksi vielä joitakin jatkotutkimus- ja kehitysehdotuksia. Ensimmäiseksi voin todeta, että paremmin valituilla ja esiprosessoiduilla muuttujilla voi olla mahdollista päästä parempaan tulokseen. Toisenlaista näkökulmaa voisi etsiä myös tutkimalla esimerkiksi tarkemmin rajattua pelaajajoukkoa.

Yksi mielenkiintoinen lisätutkimuksen kohde olisi verrata neuroverkon ennustuskykyä pokeriammattilaisen vastaavaan. Pokeriammattilaiselle voitaisiin antaa esimerkiksi samat syötetiedot kuin neuroverkolle, yksi kerrallaan, ja hän voisi niiden avulla pyrkiä ennustamaan pelaajien toimintoja. Tällaisen koejärjestelyn antamien

tulosten perusteella on helppo verrata neuroverkon paremmuutta tai huonommuutta suhteessa pokeriammattilaiseen.

Tässä työssä on tutkittu pääasiassa pienen rahan käteispelejä, joissa pelimuotona on Texas Hold'em fixed limit. Erikokoisilla panostuksilla voitaisiin helposti saada erityyppisiä tuloksia, sillä yleensä parhaat pelaajat pelaavat isommilla panoksilla. Turnauspelit ovat käytännössä oma lajinsa, jossa riittäisi myös tutkittavaa. Lisäksi tutkimuksen aiheita löytyisi varmasti myös no limit -pelin parista, puhumattakaan täysin muista pokerin varianteista.

Viiteluettelo

[Autio, 2003] Lassi Autio, Neuroverkoilla luokittelu ja tapausten keinotekoinen lisääminen aineistoon, pro gradu -tutkielma, Tietojenkäsittelytieteiden laitos, Tampereen yliopisto, 2003.

[Autio et al, 2006] Lassi Autio, Martti Juhola ja Jorma Laurikkala, On the neural network classification of medical data and an endeavour to balance non-uniform data sets with artificial data extension, *Computers in Biology and Medicine* 37, 3 (March 2007), 388-397.

[Barone and While, 1998] Luigi Barone and Lyndon While, Evolving Computer Opponents to Play a Game of Simplified Poker, 8th University of Western Australia Computer Science Research Conf.

[Callan, 1999] Robert Callan, *The Essence of Neural Networks*, Prentice Hall Europe, 1999.

[Davidson, 1999] Aaron Davidson, Using Artificial Neural Networks to model Opponents in Texas Hold'em, University of Alberta Computer Poker Research Group, 1999. Also available in http://spaz.ca/aaron/poker/nnpoker.pdf (checked at 27.3.2008).

[EasyNN Plus, 2008] EasyNN-ohjelmasta otetut kuvakaappaukset.

[Estebon, 1997] Michele D. Estebon. Perceptrons: An Associative Learning Network, Virginia Tech CS 3604, 1997. Also available in http://ei.cs.vt.edu/~history/Perceptrons.Estebon.html (checked at 27.3.2008).

[Hanson et al., 2001] S. J. Hanson, W. Gause and B. Natelson, Detection of Immunologically Significant Factors for Chronic Fatigue Syndrome Using Neural-Network Classifiers, *Clinical and Diagnostic Laboratory Immunology* 8, 3 (May 2001), 658-662.

[IBM Research, 2008] www site of IBM Research, http://www.research.ibm.com/deepblue/watch/html/c.shtml (checked at 27.3.2008)

[Johanson, 2007] Michael Bradley Johanson, Robust Strategies and Counter-Strategies: Building a Champion Level Computer Poker Player, University of Alberta, 2007.

[Juhola, 2000] Martti Juhola, Neurolaskenta. Kurssimateriaali, Tietojenkäsittelytieteiden laitos, Tampereen yliopisto, 2000.

[Korb et al., 1999] Kevin B. Korb, Ann E. Nicholson and Nathalie Jitnah, Bayesian Poker, School of Computer Science and Software Engineering, Monash University, Clayton, Australia, 1999. Also available in http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.3242 (checked at 27.3.2008).

[Mitchell, 1997] Tom M. Mitchell, *Machine Learning*, McGraw-Hill Companies, 1997.

[Picton, 2000] Phil Picton, *Neural Networks*, Second Edition, Palgrave, 2000.

[Pokerhand.org, 2007] www site that has collected data from thousands of played poker hands. www.pokerhand.org (checked at 27.3.2008).

[Pyysing, 2005] Aki Pyysing ja Marko Erola, *Pokerin Käsikirja*, Like, 2005.

[PokerStars.com, 2008] Pokerstars.com, "The World's largest poker site" http://www.pokerstars.com/poker/games/texas-holdem/ (checked at 27.3.2008).

[Stanley et al., 2005] Kenneth O. Stanley, Bobby D. Bryant and Risto Miikkulainen, Evolving Neural Network Agents in the NERO Video Game, Department of Computer Sciences, The University of Texas at Austin, 2005.