TARTU ÜLIKOOL

MATEMAATIKA-INFORMAATIKATEADUSKOND

Matemaatika instituut

Matemaatika eriala

Merili Liivoja

Otsesuunatud tehisnärvivõrgud paketis R

Bakalaureusetöö

Juhendaja: dots. Peep Miidla

Tartu 2013

Sisukord

Sissejuhatus

Inimese aju on keeruline ja võimas süsteem, mis on võimeline lahendama väga erinevaid ülesandeid. Paljude teadlaste eesmärgiks ongi arendada välja selline arvutirakendus, mis jäljendab aju funktsioneerimist ja lahendab ülesandeid sarnaselt inimese ajuga. Bioloogilise närvivõrgu lihtsustatud mudeliks on tehisnärvivõrgud. Tehisnärvivõrgud jagunevad otsesuunatuteks ja rekurrentseteks. Antud bakalaureusetöö ülesandeks on anda ülevaade otsesuunatud närvivõrkudest ja nende töö põhimõtetest. Töö sisaldab mitmeid illustreerivaid näiteid ning peaks olema heaks õppevahendiks inimesele, kes soovib antud teemaga lähemalt tutvust teha.

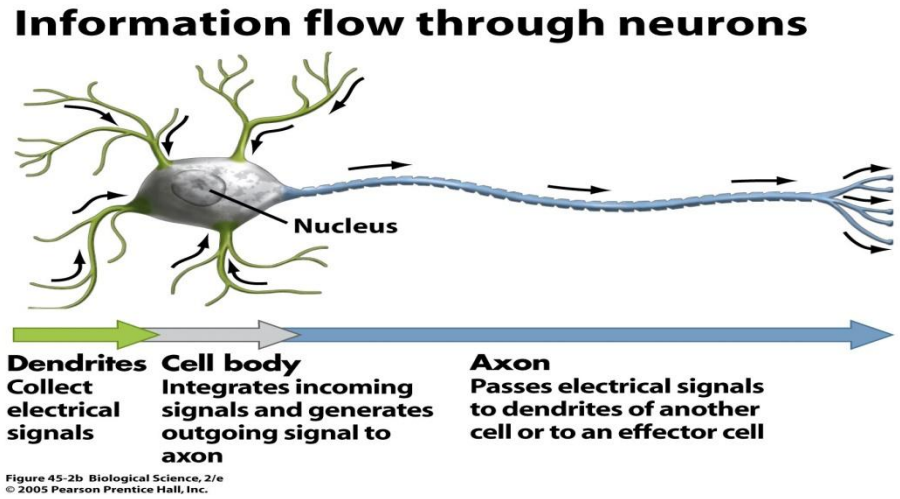
Töö koosneb kahest osast. Esimene osa annab teoreetilise ülevaate otsesuunatud närvivõrkudest ning teises osas tuuakse näide tehisnärvivõrkude rakendusest ja selle realiseerimisest paketi **R** baasil. Teoreetilises ülevaates kirjeldatakse lühidalt bioloogilist neuronit ja bioloogilist närvivõrku. Antakse ülevaade tehisneuroni arengust, kirjeldatakse tehisneuroni ehitust ning nende erinevaid tüüpe. Näidatakse kuidas tehisneuronitest koostada närvivõrke ning kuidas närvivõrk tulemusi arvutab.

Tehisnärvivõrkude õpetamise peatükis kirjeldatakse kahte enamlevinud õpetamis-algoritmi. Üks nendest on mõeldud üksiku neuroni õpetamiseks ning teine otsesuunatud närvivõrgu õpetamiseks.

Töö praktilises osas näidatakse, kuidas on võimalik konstrueerida närvivõrk, mis leiab ruutjuure etteantud arvust. Praktiline osa realiseeritakse paketis **R.** Samuti antakse ülevaade paketi **R** alampaketist **neuralnet**, mis võimaldab koostada erinevaid otsesuunatud närvivõrke.

1. Bioloogiline neuron ja bioloogiline närvivõrk

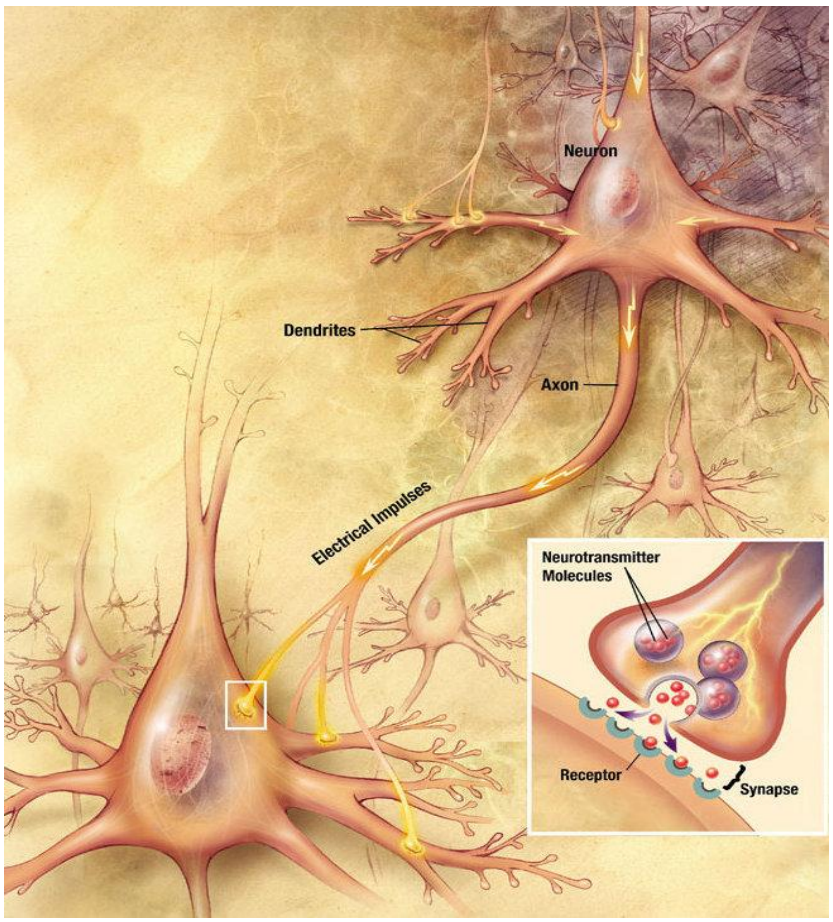
Inimese aju koos seljaaju ja närvidega koosneb rohkem kui 100 miljardist närvirakust (vt [3], lk 2). Närvirakk ehk neuron on andmeid töötlev süsteem, mis saab informatsiooni närviraku jätkete ehk dendriitide (vt joonis 1, *Dendrites*) kaudu. Dendriidid on bioloogilise närvivõrgu sisendkanalid. Närvirakk kannab edasi elektrilisi signaale ehk närviimpulsse. Neuron võtab sisendsignaalid vastu ja kui signaal on piisavalt tugev, siis toimud neuroni kehas info töötlemine. Signaalid juhitakse neuronist välja aksoni (vt joonis 1, *Axon*) kaudu, mis on bioloogilise neuroni väljundkanal. Neuronil võib olla palju dendriite aga ainult üks akson.



Joonis 1. Bioloogiline neuron [5]

Närvirakud on omavahel ühendatud võrku sünaptiliste ühenduste (vt joonis 2, *Synapse*) kaudu. Sünaptiline ühendus on neuronitevaheline ühendus, mis võimaldab närviimpulsi ülemineku ühelt neuronilt teisele. See tähendab, et ühe neuroni akson on ühendatud järgmiste neuronite dendriitidega või rakukehadega ning signaal ühe neuroni väljundist antakse edasi teiste neuronite sisenditele. Sünapsid jagunevad elektrilisteks ja keemilisteks. Elektrilise sünapsi korral antakse signaal muutmata kujul edasi järgmisele

rakule. Keemiliste sünapside korral on kaks võimalust. Keemiline sünaps võib olla kas pidurdus- või erutussünaps. Vastavalt sellele võib ta nõrgendada või tugevdada elektrilise impulsi tugevust. Tänu neuronitele toimuvad meie kehas näiteks lihaste kokkutõmbed ja lõdvenemised.



Joonis 2. Bioloogiline närvivõrk [6]

2. Tehisneuron

Esimesed sammud tehisnärvivõrkude arengus tegid Warren McCulloch ja Walter Pitts, kes pakkusid 1943. aastal välja neuronite tööd kirjeldava mudeli [8]. See oli väike kuid oluline samm üksiku neuroni sisendi-väljundipõhise käitumise genereerimise suunas.

McCulloch-Pitts'i neuroni idee on järgmine.

Kui neuroni sisendsignaalide summa on piisavalt tugev, siis neuron aktiveerub, vastasel juhul tegevust ei toimu. Matemaatiliselt, kui neuroni aktiveerimisfunktsioon F on treppfunktsioon, β on sisendsignaalide kogusumma ja θ on neuroni lävend, siis

$$F(\beta) = \begin{cases} 1, \text{kui } \beta \geq \theta, \\ 0, \text{kui } \beta < \theta. \end{cases}$$

Kui sisendsignaalide summa on suurem kui θ, siis neuron aktiveerub ja väljastab arvu 1, vastasel juhul tegevust ei toimu, s.t. neuron väljastab arvu 0.

Mõistmaks paremini antud neuroni mudelit, vaatleme üht näidet. Järgnevat näidet on kirjeldanud Michael Marsalli oma töös „*McCulloch Pitts Neurons*“” (vt [7], lk 1).

Näide 1. Oletame, et linnu ajus on neuron, millel on kaks vastuvõtjat. Mõlemad vastuvõtjad on ühenduses linnu silmadega. Kui lind näeb ümmargust objekti, saadetakse signaal esimesse vastuvõtjasse. Kui objekt on mingi muu kujuga, siis signaali ei saadeta. Seega esimene vastuvõtja vastab küsimusele „Kas objekt on ümmargune?“”. Kui lind näeb sinakas-lillat objekti, saadetakse signaal teise vastuvõtjasse. Kui aga objekt on muud värvi, siis signaali ei saadeta. Seega teine vastuvõtja vastab küsimusele „Kas objekt on sinakas-lillat värvi?“”. Paneme tähele, et mõlemale küsimusele saab vastuseks olla kas „jah“” või „ei“” ning signaal saadetakse ainult siis, kui vastus on „jah“”. Tahame koostada sellise neuroni mudeli, mis aitaks linnul üles leida mustika, kuid

vältida punaste marjade söömist. Teiste sõnadega tahame, et neuron saadaks signaali „söödav“, kui objekt on ümmargune ja sinakas-lilla. Kui kasvõi üks tingimustest pole täidetud, siis signaali ei saadeta ning lind sööb vaid siis, kui vastav signaal on saadetud. Antud olukorda kirjeldab tabel 1.

Objekt	Sinakas-lilla?	Ümmargune?	Söödav?
Mustikas	jah	jah	jah
Golfipall	ei	jah	ei
Sinilill	jah	ei	ei
Hot Dog	ei	ei	ei

Tabel 1. Näites 1 koostatud neuroni mudeli kirjeldus

McCulloch-Pitts'i neuron liidab vastuvõtjatest saadud signaalid ning saadud signaalide kogusumma põhjal otsustab, kas väljastada „jah“ või „ei“. Kui saadud signaalide kogusumma on piisavalt tugev, siis neuron aktiveerub ja väljastab „jah“ signaali, vastasel korral tegevust ei toimu, s.t. et neuron väljastab „ei“ signaali. Et saaksime ka oma neuroni mudelis signaale liita, kirjutame 1, kui vastus on „jah“, ning 0, kui vastus on „ei“. Tabel 1 näeb siis välja järgmine.

Objekt	Sinakas-lilla?	Ümmargune?	Söödav?
Mustikas	1	1	1
Golfipall	0	1	0
Sinilill	1	0	0
Hot Dog	0	0	0

Tabel 2. Näites 1 koostatud neuroni mudeli kirjeldus kodeeritud väärtustega

Nüüd peame otsustama, millal on sisendsignaalide summa piisavalt suur. McCulloch ja Pitts kasutasid selleks lävendi mõistet. Igal neuronil on lävend, mida võrreldakse signaalide kogusummaga. Kui signaalide kogusumma on suurem või võrdne antud lävendiga, siis neuron väljastab 1 (ehk „jah“ signaali). Kui aga signaalide kogusumma on lävendist väiksem, siis väljastatakse 0 (ehk „ei“ signaal). Antud ülesande puhul peame valime sellise lävendi, mille korral mõlemad kahest tingimusest oleksid täidetud.

Seega valime lävendiks 2. Kui lind näeb mustikat, siis mõlemad vastuvõtjad väljastavad 1. Neuron summeerib sisendsignaalid ning võrdleb seda lävendiga. Kuna sisendsignaalide kogusumma ($= 2$) on võrdne lävendiga ($= 2$), siis neuron väljastab 1 (mis tähendab „söödav“”). Kõiki olukordi kirjeldab tabel 3.

Objekt	Sinakas-lilla?	Ümmargune?	Sisendsignaalide kogusumma	Suurem või võrdne lävendiga?	Söödav?
Mustikas	1	1	2	jah	1
Golfipall	0	1	1	ei	0
Sinilill	1	0	1	ei	0
Hot Dog	0	0	0	ei	0

Tabel 3. Näites 1 koostatud neuroni töö kirjeldus

Muutes lävendi väärtust muutub ka linnu käitumine, seega on oluline valida lävendi väärtus vastavalt ülesandele.

Näide 1 kirjeldas olukorda, kus vastuvõtjad saadavad neuronile ainult ergutavaid signaale. Mida rohkem ergutavaid signaale neuron saab, seda lähemal on ta lävendi ületamisele ning seeläbi ka aktiveerumisele. Vastuvõtjad võivad aga neuronile saata ka selliseid signaale, mis pärsivad neuroni aktiveerumist. Selliseid sisendsignaale nimetatakse pärssivateks. Mida rohkem pärssivaid ning mida vähem ergutavaid signaale neuron saab, seda kaugemal on ta lävendi ületamisele ja aktiveerumisele. McCulloch-Pitts’i neuronit, kus on esindatud nii ergutavad kui ka pärssivad sisendsignaalid, on võimalik konstrueerida järgmiselt.

Sarnaselt näitega 1 tähistada ergutav sisendsignaal arvuga 1, ning kui ergutavat signaali ei saadeta, siis arvuga 0. Nüüd juhul, kui neuronile saadetakse pärssiv signaal, siis see tähistada arvuga -1, ning kui pärssivat signaali ei saadeta, siis arvuga 0. Sellisel viisil konstrueerib pärssivate sisendsignaalidega McCulloch-Pitts’i neuronit Micheal Marsalli (vt [7], lk 2). See käsitlus on lihtsustus sellest, kuidas McCulloch ja Pitts ise pärssivaid sisendsignaale konstrueerisid. Rohkem näiteid ning selgitusi McCulloch-Pitts’i

neuronist leiab Michael Marsalli töös „*McCulloch Pitts Neurons*“” [7] ning artiklist „*McCulloch-Pitts Neurons*“”[9].

Võtame McCulloch-Pitts’i neuroni idee kokku järgmiselt.

Neuroni sisendväärtusteks võivad olla arvud 1 või 0 ning sisendeid võib olla n tükki. Neuroni väljundväärtuseks võib samuti olla arv 1 või 0. Iga sisendsignaal võib olla kas pärssiv või ergutav. Neuron summeerib sisendsignaalid järgmiselt.

- Kui sisendsignaali väärtuseks on 1 ning signaal on ergutav, siis liidetakse summale 1.
- Kui sisendsignaali väärtuseks on 1 ning signaal on pärssiv, siis lahutatakse summast 1.
- Kui sisendsignaali väärtuseks on 0, siis summa ei muutu.

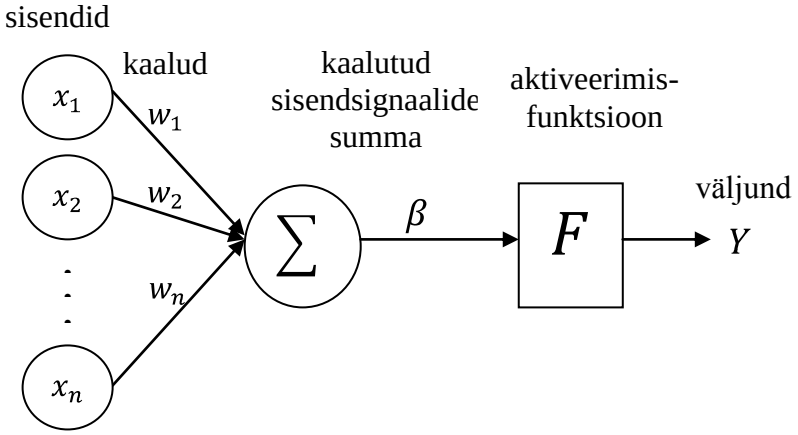
Sedasi toimetatakse kõikide sisendsignaalide väärtuste korral ning saadakse signaalide kogusumma. Kui saadud signaalide kogusumma on suurem või võrdne lävendiga, siis neuron aktiveerub ning väljastab arvu 1, vastasel juhul tegevust ei toimu, s.t. neuron väljastab arvu 0.

McCulloch-Pitts’i neuronil on mitmeid piiranguid.

- Lubab kasutada ainult binaarseid sisendeid ja väljundeid.
- Aktiveerimisfunktsiooniks on treppfunktsioon (*step function*).
- Ei võimalda sisendite korrutamist kaaludega. Kaal on koefitsient, mis reguleerib sisendsignaali tugevust.

Näiteks ei suuda McCulloch-Pitts’i neuron lahendada välistavat disjunktsiooni (XOR) ega ka selle eitust (XNOR).

Sisendsignaalide kogusummat võib teisiti kirjeldada ka kui eelmiste neuronite poolt saadetud väljundite kaalutud summat. Selle pakkus 1958. aastal välja Frank Rosenblatt oma pertseptroni mudelis [11]. Rosenblatti pertseptroni mudel on edasiarendus McCulloch-Pitts’i neuronist, millele on juurde toodud sisendite kaalude mõiste ning erinevalt McCulloch-Pitts’i neuronist võivad Rosenblatti pertseptronil olla reaalarvulised sisendid. Kuna pertseptron on tehisneuroni lihtsaim vorm, siis vaatleme kõigepealt tehisneuroni üldist struktuuri. Seda kirjeldab joonis 3.



Joonis 3. Tehisneuroni mudel

Olgu vaadeldaval neuronil n sisendit $x_1, x_2, \dots, x_n$. Tavaliselt on üheks sisendiks konstant 1 või -1. Sellele sisendile vastavaid kaalukoefitsiente võime nimetada sõltuvalt aktiveerimisfunktsiooni valikust kas lävendiks (kui aktiveerimisfunktsiooniks on treppfunktsioon) või konstantseks sisendiks (kui aktiveerimisfunktsiooniks on mõni muu funktsioon). Iga sisend korrutatakse läbi kaalukoefitsiendiga, vastavalt $w_1, w_2, \dots, w_n$. Kaalukoefitsiendid võival olla nii positiivsed kui negatiivsed. Vastavalt märgile nad kas suurendavad või vähendavad impulsi tugevust. Neuron genereerib summaarse funktsiooni

$$\beta = \sum_{i=1}^{n} x_i w_i.$$

Kaalutud sisendsignaalide summast arvutab aktiveerimisfunktsioon neuroni väljundi

$$Y = F(\beta).$$

Aktiveerimisfunktsiooni valik sõltub konkreetsest ülesandest. Kõige levinumad aktiveerimisfunktsioonid on logistiline funktsioon ja hüperboolne tangens. Nende funktsioonide peamiseks eesmärgiks on hoida neuronite väljundid mõistlikes piirides. Mõnikord kasutatakse ka treppfunktsiooni, lineaarset funktsiooni või Gaussi funktsiooni.

Sarnaselt McCulloch-Pitts'i neuronile on Rosenblatti pertseptroni aktiveerimisfunktsiooniks treppfunktsioon ning väljund võib olla kas „õige“” (1) või „väär“” (0) (Rosenblatti pertseptroni puhul kasutatakse kirjanduses väljundi 1 ja 0

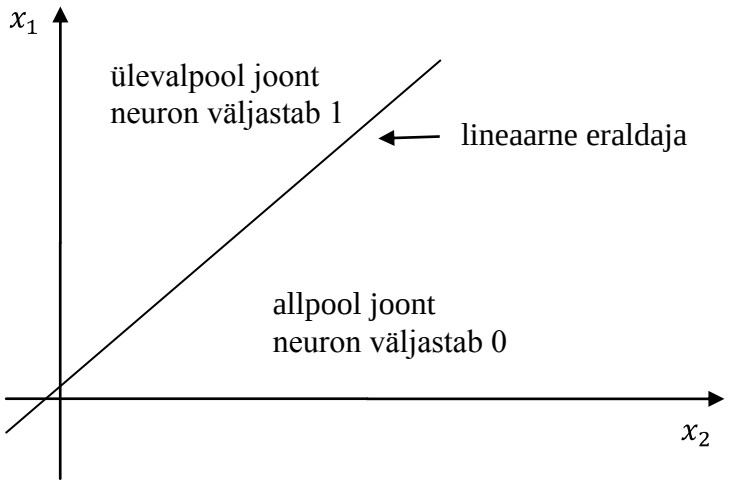
asemel sageli ka 1 ja -1). Kuigi Rosenblatt kirjeldas pertseptroni aktiveerimisfunktsioonina treppfunktsiooni, siis tänapäeval on kirjanduses pertseptroni aktiveerimisfunktsioonina levinud ka teised funktsioonid, näiteks hüperboolne tangens ja logistiline funktsioon. Eristamaks neid tehisneuroni tüüpe, nimetame pertseptrinit, mille aktiveerimisfunktsiooniks on treppfunktsioon, Rosenblatti pertseptroniks. Pertseptron on tehisneuroni vorm, mis on võimeline õppima vastavalt kontekstile ning õpetamine käib näidete alusel. Pertseptroni õpetamisalgoritmiga tutvume lähemalt peatükis 4.

Rosenblatti pertseptronit kasutatakse peamiselt klassifitseerimisülesannete korral. Sellisel juhul on neuroni eesmärgiks sisenditele vastavate punktide korrektne jaotamine klassidesse φ_1 ja φ_2 . Otsustamisreegel on, et kui pertseptroni väljund on 1, siis määrata punkt klassi φ_1, ning kui väljund on 0, siis määrata punkt klassi φ_2. Suurendades väljundkihi neuronite arvu, on võimalik teostada jaotust rohkem kui kahe klassi vahel. Eelduseks aga on, et need klassid peavad olema lineaarselt eraldatavad. Ühte näidet sellisest närvivõrgust, mis jaotab andmeid rohkem kui kahe klassi vahel, vaatleme lähemalt peatükis 3.

Selgitame lineaarse eraldatavuse mõistet järgmiselt. Olgu meil kahe sisendiga pertseptron, mille aktiveerimisfunktsiooniks on treppfunktsioon. Tähistame sisendid vastavalt x_1 ja x_2. Kaalud olgu vastavalt w_1 ja w_2 ning neuroni lävend θ. Siis saame lineaarse eraldaja:

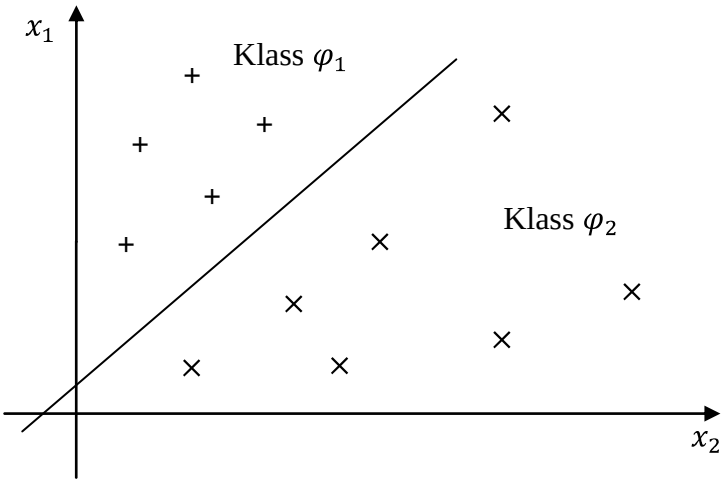
$$w_1 x_1 + w_2 x_2 + \theta = 0$$

Kui punkt (x_1, x_2) asub ülevalpool lineaarset eraldajat, väljastab neuron väärtuse 1 ning punkt määratakse klassi φ_1. Vastasel juhul väljastab see väärtuse 0 ning punkt määratakse klassi φ_2. Antud olukorda kirjeldab ka joonis 4. Antud näite puhul on tegemist olukorraga, kus andmeid klassifitseeritakse kahe tunnuse alusel, millele vastavad etteantud sisendid. Seega on tegu kahemõõtmelise ruumiga ning sellisel juhul on lineaarseks eraldajaks sirgjoon. Juhul aga kui tahame klassifitseerida andmeid n tunnuse alusel, siis on tegemist n-mõõtmelise ruumiga ja lineaarseteks eraldajateks on $n - 1$ mõõtmeline hüpertasand.



Joonis 4. Lineaarne eraldaja

Näide 2. Olgu meil antud andmed, mida kirjeldab joonis 5. Plussid tähistavad sisendeid, mis kuuluvad klassi φ_1, ning ristid tähistavad sisendeid, mis kuuluvad klassi φ_2. Antud näite korral leidub sirge, mis eraldab kõik plussid ristidest. Seega on võimalik konstrueerida pertseptron, mis antud andmed õigesti klassidesse jaotab. Lineaarne eraldaja leitakse kasutades pertseptroni õpetamisalgoritmi. Kui sellist eraldajat ei leidu, siis õpetamine ei jõua kunagi nii kaugele, et kõik andmed oleksid õigesti klassifitseeritud. Pertseptroni õpetamisalgoritmiga tutvume lähemalt 4. peatükis.



Joonis 5. Klassifitseerimisülesanne kahe klassi vahel

Märgime siinkohal, et kui kasutada aktiveerimisfunktsioonina treppfunktsiooni asemel nt logistilist funktsiooni, siis saame iga punkti korral väljundväärtuseks arvu vahemikust (0, 1). Antud arvu saab vaadelda kui tõenäosust, et antud punkt kuulub klassi φ_1.

Paljusid ülesandeid aga ei ole võimalik lineaarse eraldaja abil lahendada. Sarnaselt McCulloch-Pitts'i neuronile ei suuda ka pertseptron lahendada välistavat disjunktsiooni (XOR) ega selle eitust (XNOR). Sellise klassifitseerimisprobleemi lahendas mitmekihiline pertseptron, ning selle õpetamiseks kasutatav hälbe pöördlevi meetod (*backpropagation algorithm*). Mitmekihilisest pertseptronist räägime lähemalt 3. peatükis ning hälbe pöördlevi meetodist anname ülevaate 4. peatükis.

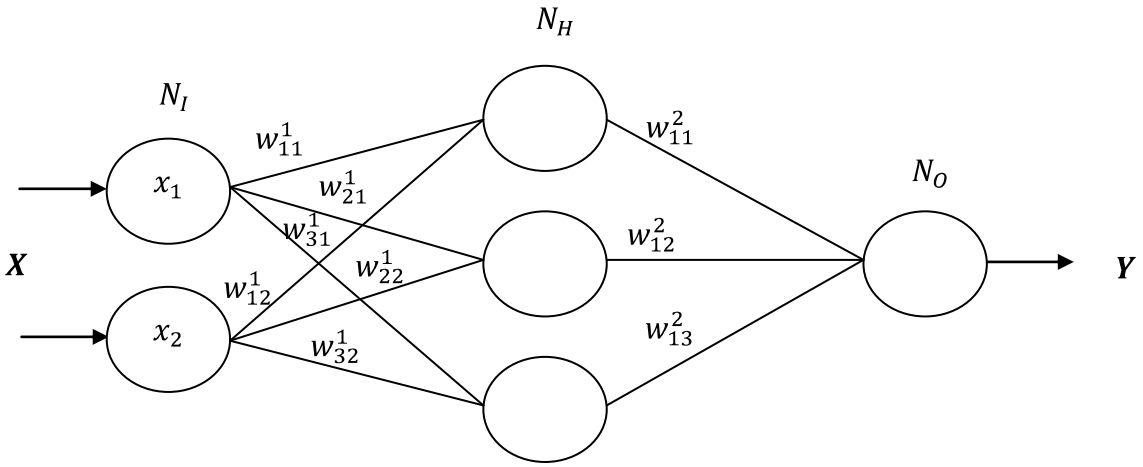
3. Otsesuunatud närvivõrgud

Tehisneuronitest on võimalik koostada väga keerulisi närvivõrke. Tehisnärvivõrkude definitsioone on mitmeid. Üks neist on järgmine.

Tehisnärvivõrk on andmetöötluse süsteem, mis koosneb suurest arvust omavahel püsivalt seotud tehisneuronitest. Tehisneuronid on ühendatud struktuuri, mille eeskujuks on võetud inimese ajukoore funktsioneeriv hierarhia. (vt [10], lk 7)

Reeglina paiknevad neuronid tehisnärvivõrgus kihiti. Struktuuri poolest jagunevad närvivõrgud kaheks: otsesuunatud (*feed forward*) ja rekurrentsed ehk tagasisidega. Otsesuunatud närvivõrgu korral võib neuroni väljund olla seotud ainult järgmisel kihil paikneva neuroni sisendiga. Tagasisidega närvivõrgu korral võib aga neuroni väljund olla seotud nii järgmise kihi neuroni sisenditega kui ka eelmiste kihtide neuronite sisenditega.

Kõige sagedasemini kasutatav otsesuunatud närvivõrk on mitmekihiline pertseptron. Mitmekihiline pertseptron on selline otsesuunatud närvivõrk, milles iga kihi iga neuroni väljund on seotud järgmise kihi iga neuroni ühe sisendiga. Kihti, kus asuvad närvivõrgu sisendid, nimetatakse sisendkihiks. Näiteks joonisel 6 on sisendkiht tähistatud N_I. Kihti, kus asuvad närvivõrgu väljundid, nimetatakse väljundkihiks. Joonisel 6 on väljundkiht tähistatud N_O. Ülejäänud kihte nimetatakse peidetud kihtideks. Joonisel 6 on üks peidetud kiht, mis on tähistatud N_H. Peidetud kihi neuronid saavad informatsiooni eelmise kihi neuronite väljunditest, teisendavad seda ja edastavad info järgmise kihi neuronitele. Sisendkihis info teisendamist ei toimu. See on ka põhjuseks, miks tavaliselt kihtide kokkulugemisel sisendkihti ei arvestata. Seega joonisel 6 toodud pertseptron on kahekihiline. Antud pertseprtonil on ainult üks väljund, kuid neid võib olla ka rohkem. Väljundkihi neuronite arv näitab meile antud närvivõrgu väljundite arvu.



Joonis 6. Kahekihiline pertseptron

Iga neuroni sisend korrutatakse läbi vastava kaalukoefitsiendiga. Neuronid summeerivad kaalutud sisendid ning liidavad saadud summale konstantsed sisendid. Nendest summadest arvutatakse vaadeldud kihi neuronite väljundid. Arvutatud väljundid on järgmise kihi neuronite sisenditeks. Kahekihilise pertseptroni korral, millel on n sisendit, m väljundit ja k neuronit peidetud kihil, kirjeldab antud arvutuskäiku võrrand

$$Y = F^2(W^2F^1(W^1X + \Theta^1) + \Theta^2),$$

kus $X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ on närvivõrgu sisendvektor; $Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$ on närvivõrgu väljundvektor;

$W^1 = \begin{bmatrix} w_{11}^1 & w_{12}^1 & \cdots & w_{1n}^1 \\ w_{21}^1 & w_{22}^1 & \cdots & w_{2n}^1 \\ \vdots & \vdots & \ddots & \vdots \\ w_{k1}^1 & w_{k2}^1 & \cdots & w_{kn}^1 \end{bmatrix}$ on peidetud kihi neuronite kaalukoefitsientide maatriks;

$W^2 = \begin{bmatrix} w_{12}^2 & w_{12}^2 & \cdots & w_{1k}^2 \\ w_{21}^2 & w_{22}^2 & \cdots & w_{2k}^2 \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1}^2 & w_{m2}^2 & \cdots & w_{mk}^2 \end{bmatrix}$ on väljundkihi neuronite kaalukoefitsientide maatriks;

$\Theta^1 = \begin{bmatrix} \theta_1^1 \\ \theta_2^1 \\ \vdots \\ \theta_k^1 \end{bmatrix}$ on peidetud kihi konstantsete sisendite vektor;

$\Theta^2 = \begin{bmatrix} \theta_1^2 \\ \theta_2^2 \\ \vdots \\ \theta_m^2 \end{bmatrix}$ on väljundkihi konstantsete sisendite vektor;

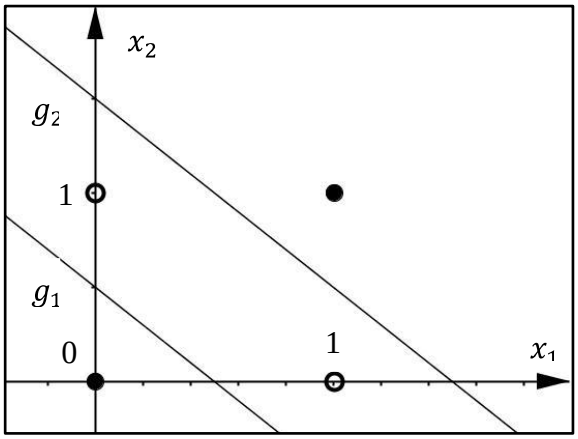
F^1 on peidetud kihi neuronite aktiveerimisfunktsioon;

F^2 on väljundkihi neuronite aktiveerimisfunktsioon.

Näide 3. Mitmekihiline pertseptron võimaldab klassifitseerida andmeid, mis pole ühe lineaarse joonega eraldatavad. Toome näite mitmekihilisest pertseptronist, mis lahendab välistava disjunktsiooni. Välistav disjunktsioon on loogikatehe, mis loetakse vääraks parajasti siis, kui mõlemad sisendid on väärad või mõlemad on tõesed. Välistavat disjunktsiooni kirjeldab tabel 4.

x_1	x_2	XOR	Klass
0	0	0	φ_2
0	1	1	φ_1
1	0	1	φ_1
1	1	0	φ_2

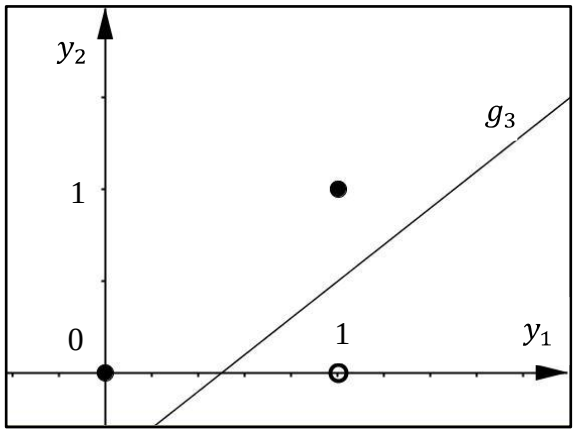
Tabel 4. Välistava disjunktsiooni kirjeldus



Joonis 7. Välistava disjunktsiooni graafiline kirjeldus

Joonisel 7 on esitatud välistava disjunktsiooni graafiline esitlus. Jooniselt on näha, et antud andmed ei ole ühe sirgega eraldatavad. Antud ülesande on võimeline lahendama pertseptron, mille peidetud kihis on kaks neuronit ja väljundkihis 1 neuron. Peidetud

kihi neuronitele vastavad lineaarsed eraldajad g_1 ja g_2, kus üheks võimalikuks lineaarsete eraldajate valikuks on $g_1 = x_1 + x_2 - 0{,}5$ ja $g_2 = x_1 + x_2 - 1{,}5$. Aktiveerimisfunktsioonina kasutame treppfunktsiooni. Seega peidetud kihi neuroni väljundiks on arv 1, kui punkt asub ülevalpool sellele neuronile vastavat lineaarset eraldajat, ja arv 0 vastasel juhul. Peidetud kihi neuronite võimalikke väljundeid kirjeldab graafiliselt joonis 8. Saadus andmed on võimalik eraldada ühe lineaarse eraldajaga. Sellele eraldajale vastab üks neuron väljundkihis. Üheks võimalikuks lineaarseks eraldajaks on sirge $g_3 = y_1 - y_2 - 0{,}5$, kus y_1 ja y_2 on peidetud kihi neuronite väljundid. Väljundkihi neuroni väljundiks määrame arvu 1, kui $y_1 + y_2 - 0{,}5 \geq 0$ (s.t. et punkt kuulub klassi φ_1), ja arvu 0, kui kui $y_1 + y_2 - 0{,}5 < 0$ (s.t. et punkt kuulub klassi φ_2).



Joonis 8. Peidetud kihi neuronite võimalike väljundite graafiline esitlus

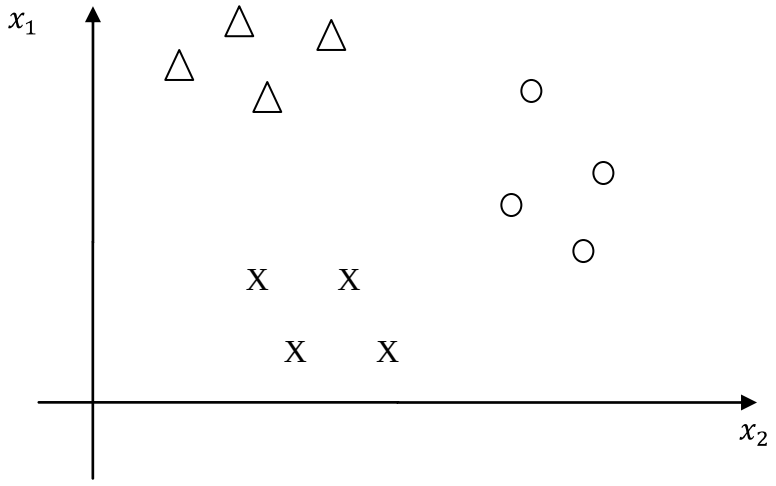
Olemegi koostanud kahekihilise pertseptroni, mis lahendab välistavat disjunktsiooni. Võtame selle närvivõrgu töö kokku järgmise tabelina.

x_1	x_2	y_1	y_2	Närvivõrgu väljund
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

Tabel 5. Näites 3 koostatud närvivõrgu töö kirjeldus

Mitmekihilise pertseptroni erijuhuks on ühekihiline pertseptron. Ühekihiline pertseptron on selline tehisnärvivõrk, mis koosneb ainult sisendkihist ja väljundkihist. Kui üksik pertseptron suudab klassijaotust teostada ainult kahe klassi vahel (vt näide 2), siis ühekihiline pertseptron suudab klassifitseerida andmeid N klassi vahel. Nagu eelnevaltki mainitud sai, on eelduseks, et need klassid peavad olema üksteisest lineaarselt eraldatavad. Klassijaotust enama kui kahe klassi vahel saab teostada meetodi alusel, mida nimetatakse „*one vs all*“” meetodiks. Vaatleme antud meetodi kohta järgmist näidet.

Näide 4. Olgu vaja klassifitseerida andmed kolme klassi vahel kahe tunnuse alusel. Andmeid kolme klassi esindajatega kirjeldab joonis 8. Märgime siinkohal, et need klassid on üksteisest lineaarselt eraldatavad. Kolmnurgad tähistavad klassi φ_1 esindajaid, ristid tähistavad klassi φ_2 esindajaid ning ringid tähistavad klassi φ_3 esindajaid.

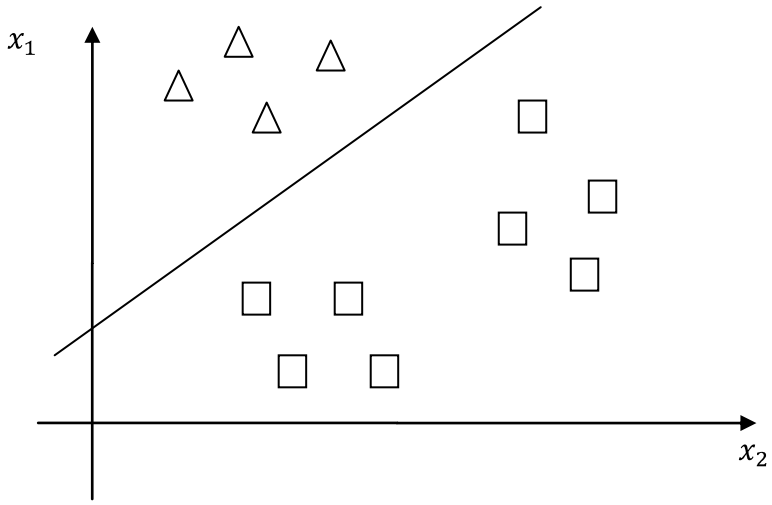


Joonis 9. Klassifitseerimisülesanne kolme klassi vahel

„*One vs all*“” meetodi idee on järgmine.

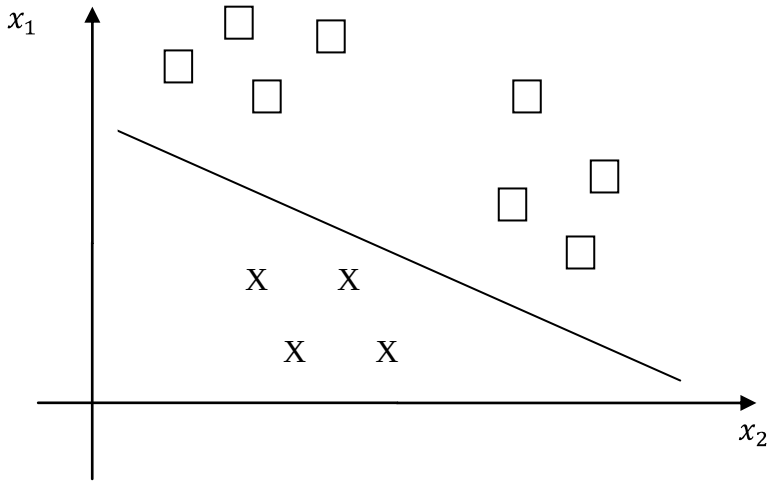
Kuna meil on andmed vaja jaotada kolme klassi vahel, siis võime ülesande jaotada kolmeks osaks, mille iga osa viib läbi klassijaotust kahe klassi vahel (vt näide 2). Esimene osaülesanne selgitab välja, kas punkt kuulub klassi φ_1. Selle osaülesande teostamiseks vaatleme andmestikku selliselt, kus klasside φ_2 ja φ_3 esindajad kuuluvad ühte klassi (vt joonis 9). Nüüd on võimalik leida lineaarne eraldaja, mis antud klasside

esindajad õigesti klassifitseeriks. Sellele lineaarsele eraldajale vastab närvivõrgus üks neuron. Antud neuroni väljundiks määrame arvu 1, kui punkt kuulub klassi φ_1, ning arvu null vastasel korral.



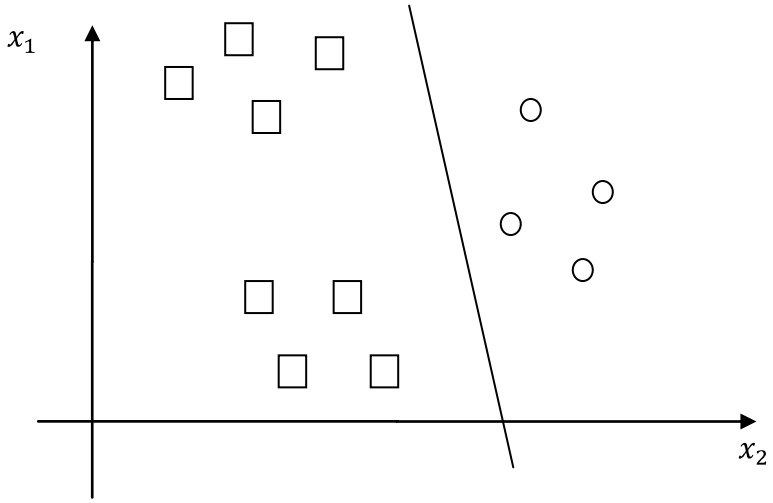
Joonis 10. Klassijaotus klasside φ_1 ja uue klassi vahel

Teine osaülesanne selgitab välja, kas punkt kuulub klassi φ_2. Vaatleme nüüd andmestikku selliselt, kus klasside φ_1 ja φ_3 esindajad kuuluvad ühte klassi (vt joonis 10). Nüüd on meil kaks klassi, mille vahel oskame klassijaotust teha. Sellele osaülesandele vastab samuti üks neuron. Antud neuroni väärtuseks määrame samuti arvu 1, kui punkt kuulub klassi φ_2, ning arvu 0 vastasel juhul.



Joonis 11. Klassijaotus klasside φ_2 ja uue klassi vahel.

Kolmas osaülesanne selgitab välja, kas punkt kuulub klassi φ_3. Kolmanda osaülesande teostamiseks vaatleme andmestikku selliselt, kus klasside φ_1 ja φ_2 esindajad kuuluvad ühte klassi (vt joonis 11) ning viime läbi klassijaotuse klasside φ_3 ja uue klassi vahel. Sellele osaülesandele vastab samuti üks neuron, mis väljastab arvu 1, kui punkt kuulub klassi φ_3, ning arvu 0 vastasel juhul.



Joonis 12. Klassijaotus klasside φ_2 ja uue klassi vahel

Paneme nüüd tähele, et kui punkt kuulub klassi φ_1, siis esimene neuron väljastab arvu 1 ning teine ja kolmas neuron väljastavad arvu 0. Seega närvivõrgu väljundvektoriks on $y = \begin{bmatrix}1\\0\\0\end{bmatrix}$. Kui punkt kuulub aga klassi φ_2, siis esimene ja kolmas neuron väljastavad arvu 0 ning teine neuron väljastab arvu 1. Seega närvivõrgu väljundvektor antud juhul on $y = \begin{bmatrix}0\\1\\0\end{bmatrix}$. Kui aga punkt kuulub klassi φ_3, siis väljastavad esimene ja teine neuron arvu 0 ning kolmas neuron väljastab arvu 1. Sellisel juhul on närvivõrgu väljundvektor kujul $y = \begin{bmatrix}0\\0\\1\end{bmatrix}$. Oleme koostanud närvivõrgu, mille sisendkihis on kaks neuronit ja väljundkihis kolm neuronit ning kõigi selle närvivõrgu neuronite aktiveerimis-funktsiooniks on treppfunktsioon. Konstrueeritud närvivõrk väljastab vastavalt $\begin{bmatrix}1\\0\\0\end{bmatrix}$, kui

punkt kuulub klassi φ_1, $\begin{bmatrix}0\\1\\0\end{bmatrix}$, kui punkt kuulub klassi φ_2, ning $\begin{bmatrix}0\\0\\1\end{bmatrix}$, kui punkt kuulub klassi φ_3.

Närvivõrgu sisendite arvu määrab sisendikomplekti dimensioon. Näiteks kui sisendikomplektiks on kolmeelemendiline vektor, siis sisendite arv on 3. Klassifitseerimisülesannete korral tähistavad sisendid neid tunnuseid, mille alusel andmeid on vaja klassidesse jaotada. Närvivõrgu väljundite arv on aga erinevate klasside arv. Välja arvutud juhul kui klassijaotust viiakse läbi kahe klassi vahel. Siis piisab ühest neuronist väljundkihis. Otsesuunatud närvivõrkude puhul kasutatakse tavaliselt ühte või kahte peidetud kihti. Neuronite arv peidetud kihis sõltub aga konkreetsest ülesandest. Tavaliselt leitakse neuronite arv eksperimentaalselt või vaadeldakse närvivõrgu väljundi hälvet. Võib oletada, et mida keerulisem on ülesanne, seda rohkem neuroneid peaks peidetud kiht sisaldama.

Stanfordi ülikooli Tehisintellekti labori direktor Andrew Ng on andnud närvivõrgu kihtide ja neuronite arvu valiku koha pealt järgmised soovitused [2].

- On mõistlik valik proovida ühe peidetud kihiga närvivõrku (mis on tõenäoliselt ka kõige tavalisele närvivõrgu arhitektuur) või kui peidetud kihte on vaja rohkem kui üks, siis kasutada sama arvu neuroneid kõigis peidetud kihtides.
- Neuronite arv peidetud kihis võiks olla võrreldav sisendkihis paiknevate neuronite arvuga. Neuronite arv peidetud kihis võiks olla näiteks kaks kuni neli korda suurem kui neuronite arv sisendkihis.

4. Tehisnärvivõrkude õpetamine

Tehisnärvivõrgu treenimiseks nimetatakse sellele närvivõrgule sobivate parameetrite valiku protsessi. Närvivõrgu parameetriteks on kaalud ja konstantsed sisendid. Järelikult on treenimise eesmärgiks leida selline närvivõrgu kaalude ja konstantsete sisendite kombinatsioon, mille korral närvivõrk saab püstitatud ülesandega kõige paremini hakkama. Võib eristada kahte treenimise meetodit: õpetamine *(supervised learning)* ja iseõppimine *(unsupervised learning)*. Närvivõrgu õpetamiseks nimetatakse meetodit, mille korral antakse närvivõrgule ette teadaolevad sisendite ja väljundite komplektid *(the training set)*, mis reeglina saadakse need eksperimentaalselt. Õpetamise käigus muudetakse närvivõrgu parameetreid selliselt, et närvivõrgu väljundi ja teadaoleva õige vastuse vaheline erinevus oleks minimaalne. Olgu X sisendväärtuste vektor, Y_p nendele sisendväärtustele vastavate etteantud väljundväärtuste vektor, Y selle närvivõrgu poolt arvutatud väljundväärtuste vektor ja NN närvivõrgu funktsioon, see tähendab, et $Y = NN(X, W, \Theta)$, kus W ja Θ on fikseeritud parameetrid. Siis matemaatiliselt võib õpetamise ülesande kirja panna järgmiselt.

$$|Y_p - NN(X)| = |Y_p - Y| \to 0.$$

Õpetamise tulemusena õpib närvivõrk andma õigeid tulemusi eelnevalt fikseeritud väärtuste korral ning tänu üldistamisvõimele annab ta suhteliselt õigeid tulemusi ka tundmatu sisendväärtuse korra.

Õpetamisalgoritmi valik sõltub konkreetsest ülesandest ning kasutatavast närvivõrgu struktuurist. Järgnevalt vaatleme kahte õpetamisalgoritmi, millest esimene on mõeldud üksiku pertseptroni õpetamiseks ning teine mitmekihilise pertseptroni õpetamiseks.

4.1 Pertseptroni õpetamisalgoritm

Olgu meil n sisendiga pertseptron, mille aktiveerimisfunktsiooniks on treppfunktsioon. Olgu treenimiseks kasutatavad sisendväärtused $X_i = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ ning neile vastav teadaolevad väljundväärtus Y_i, kus i tähistab treenimiseks kasutatava sisendikomplekti ja väljundväärtuse järjekorranumbrit. Kaalukoefitsiendid tähistame $W = [w_1, w_2, \dots, w_n]$. Paneme tähele, et võime lävendit vaadelda kui ühte sisendit, mille väärtus on 1 või -1 ning mille kaal on θ. Kui tähistame lävendile vastava sisendi x_{n+1} ja $\theta = w_{n+1}$, siis saame sisendid kirja panna kujul $X_i = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ x_{n+1} \end{bmatrix}$ ning neile vastavad kaalud $W = [w_1, w_2, \dots, w_n, w_{n+1}]$. Saame lineaarse eraldaja kujul

$$w_1x_1 + w_2x_2 + \dots + w_{n+1}x_{n+1} \geq 0.$$

Pertseptroni õpialgoritm koosneb järgmistest punktidest.

1. Algselt määratakse kõik kaalud nullideks või valitakse suvalised väärtused.
2. Võetakse sisendikomplekt i ning vaadeldakse, kas mõlemad tingimustest $WX_i \geq 0$ ja Y_i=1 (ehk X_i on klassist φ_1) on täidetud.
3. Kui mõlemad tingimused on täidetud, siis kaalusid ei muudeta ning võetakse uus sisendikomplekt.
4. Vea korral uuendatakse kaalusid järgmise reegli alusel.

$W \leftarrow W - X_i^T$, kui $WX_i \geq 0$ ja Y_i=0 (ehk X_i on klassist φ_2), (1)

$W \leftarrow W + X_i^T$, kui $WX_i < 0$ ja Y_i=1 (ehk X_i on klassist φ_1). (2)

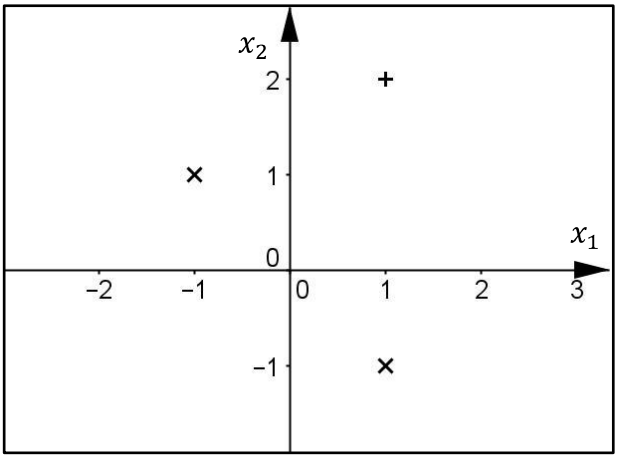
Punkte 2. - 4. korratakse kõikide sisendkomplektide X_i ning neile vastavate etteantud väljundväärtuste Y_i korral.

Kui pertseptroni aktiveerimisfunktsiooniks on näiteks logistiline funktsioon, siis tuleks punktides 1. – 4. lugeda X_i klassi φ_1, kui $Y_i \geq 0{,}5$ ning klassi φ_2, kui $Y_i < 0{,}5$.

Näide 5. Vaatleme järgmisi õpetamiseks kasutatavaid sisendite ja neile vastavate väljundite paare.

$$X_1 = \begin{bmatrix}1\\2\end{bmatrix}, Y_1 = 1;\ X_2 = \begin{bmatrix}-1\\1\end{bmatrix}, Y_2 = 0;\ X_3 = \begin{bmatrix}1\\-1\end{bmatrix}, Y_3 = 0$$

Graafiliselt väljendab antud olukorda joonis 13. Pluss tähistab klassi φ_1 esindajat ning ristid tähistavad klassi φ_2 esindajaid.



Joonis 13. Sisendkomplektide graafiline esitus

Tegemist on Rosenblatti pertseptroniga, mille sisendkihis on 2 neuronit ja väljundkihis 1 neuron. Vaadeldes lävendit kui üht sisendit, mille väärtus on alati 1 ja kaal θ, saame pertseptroni, mille sisendkihis on 3 neuronit ja väljundkihis 1 neuron. Sisendid ja väljundid võime siis kirja panna järgmiselt.

$$X_1 = \begin{bmatrix}1\\2\\1\end{bmatrix}, Y_1 = 1;\ X_2 = \begin{bmatrix}-1\\1\\1\end{bmatrix}, Y_2 = 0;\ X_3 = \begin{bmatrix}1\\-1\\1\end{bmatrix}, Y_3 = 0.$$

Pertseptroni õpetamine algab sellega, et määratakse kaalukoefitsientide väärtused. Valime kaalukoefitsientide väärtusteks juhuslikud arvud.

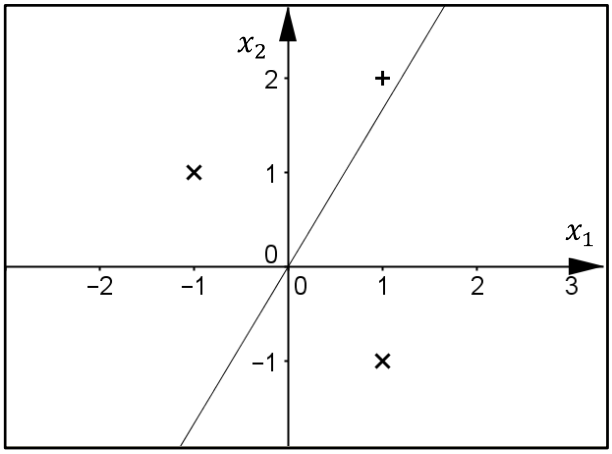
$$W = [1, -0.8, 0].$$

Anname pertseptronile ette esimese sisendväärtuste komplekti X_1 ning arvutame väljundi.

$$Y = WX_1 = [1, -0.6, 0]\begin{bmatrix}1\\2\\1\end{bmatrix} = -0.2 < 0$$

See tähendab, et pertseptroni väljund sisendkomplekti X_1 korral on 0 ja X_1 kuulub klassi φ_2. See aga ei ühti etteantud väljundväärtusega, mis sisendväärtuste

komplekti X_1 korral on 1. Graafiliselt kirjeldab antud olukorda joonis 14. Jooniselt on näha, et antud lineaarne eraldaja ei suuda andmeid õigesti klassifitseerida. Tuletame meelde, et lineaarse eraldaja võrrand on $x_1 w_1 + x_2 w_2 + \theta = 0$.

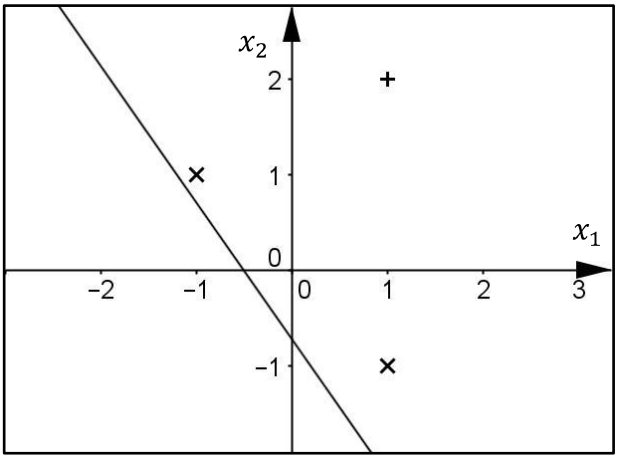


Joonis 14. Lineaarne eraldaja kaalukoefitsientide algväärtuste korral

Arvutame uued kaalud valemi (2) järgi.

$$W \leftarrow W + X_1^T = [1, -0.6, 0][1,2,1] = [2, 1.4, 1]$$

Saame uue lineaarse eraldaja, mida kirjeldab joonis 15.



Joonis 15. Lineaarne eraldaja esimesel iteratsioonisammul

Nüüd anname pertseptronile ette sisendikomplekti X_2 ning saame

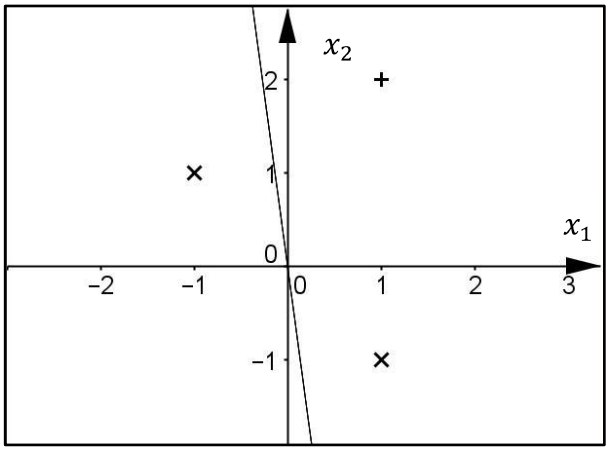
$$Y = WX_2 = [2,1.4,1]\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} = 0.4 \geq 0.$$

See tähendab, et pertseptroni väljund sisendkomplekti X_2 korral on 1 ja X_2 kuulub klassi φ_1. See aga ei ühti etteantud väljundväärtusega, mis sisendväärtuste komplekti X_2 korral on 0. Antud olukord on näha ka jooniselt 15.

Arvutame uued kaalud valemi (1) järgi.

$$W \leftarrow W - X_2^T = [2, 1.4, 1][-1, 1, 1] = [3, 0.4, 0]$$

Saame uue lineaarse eraldaja, mida kirjeldab joonis16.



Joonis 16. Lineaarne eraldaja teisel iteratsioonisammul

Nüüd anname pertseptronile ette sisendikomplekti X_3 ning saame

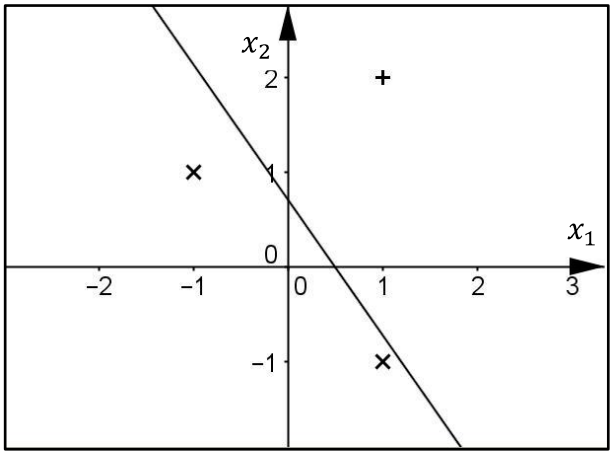
$$Y = WX_3 = [3, 0.4, 0]\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} = 3 \geq 0.$$

See tähendab, et pertseptroni väljund sisendkomplekti X_3 korral on 1 ja X_3 kuulub klassi φ_1. See aga ei ühti etteantud väljundväärtusega, mis sisendväärtuste komplekti X_3 korral on 0. See olukord on näha ka jooniselt 16.

Arvutame uued kaalud valemi (1) järgi.

$$W \leftarrow W - X_3^T = [3, 0.4, 0][1, -1, 1] = [2, 1.4, -1]$$

Saame uue lineaarse eraldaja, mida kirjeldab joonis 17. Jooniselt on näha, et oleme leidnud lineaarse eraldaja, mis suudab etteantud andmed õigesti klassidesse jaotada.

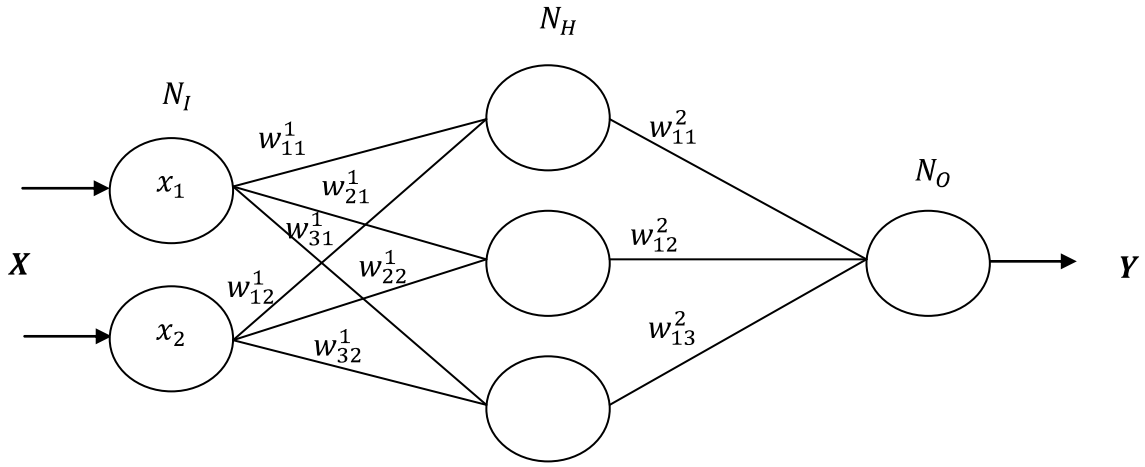


Joonis 17. Lineaarne eraldaja kolmandal iteratsioonisammul

4.2 Hälbe pöördlevi meetod (*backpropagation algorithm*)

Hälbe pöördlevi meetod seisneb tehisnärvivõrgu õpetamises väljunditest sisendite suunas ja seda kasutatakse peamiselt otsesuunatud tehisnärvivõrkude korral. Sarnaselt pertseptroni õpetamisalgoritmile teostatakse ka antud meetodit treeningkomplekte kasutades. Treeningkomplekt tehisnärvivõrgu ülesannete puhul tähendab teadaolevate sisendite ja väljundite väärtustega ülesannet. Selle puhul treenitakse närvivõrk, ehk leitakse kaalude uusi väärtusi seni, kuni närvivõrk töötab kõigi treeningkomplekti ülesannete korral korrektselt ja õigesti. Antud meetodi idee on järgmine. Kõigepealt arvutatakse närvivõrgu väljundid. Arvutatud väljundeid võrreldakse tegelike väljundväärtustega ning leitakse väljundite hälbed. Leitud hälvete abil arvutatakse uued kaalukoefitsiendid. Hälvete ja kaalukoefitsientide arvutamine käib väljundkihist sisendkihi suunas.

Vaatleme hälbe pöördlevi meetodi järgi kaalukoefitsientide arvutamist kahekihilise pertseptroni puhul.



Joonis 6. Kahekihiline pertseptron

Esialgselt omistatakse võrgu parameetrite väärtusteks juhuslikult valitud arvud. Peidetud kihi neuronite väljundväärtused arvutatakse järgmiselt

$$y_j^h = F^1\left(\sum_i w_{ji}^h x_i + \theta_{\mathrm{j}}^{\mathrm{h}}\right) y_j^h$$

ning väljundkihi neuronite väljundid järgmiselt

$$y_k = F^2\left(\sum_j w_{kj}^v y_j^h + \theta_{\mathrm{k}}^{\mathrm{v}}\right).$$

Siin i tähendab neuroni sisendi järjekorranumbrit; j tähendab neuroni järjekorranumbrit peidetud kihis; k tähendab neuroni järjekorranumbrit väljundkihis; x_i on selle närvivõrgu sisendid; w_{ji}^h on peidetud kihi neuronite kaalukoefitsientide väärtused; $\theta_{\mathrm{j}}^{\mathrm{h}}$ on peidetud kihi neuronite konstantsed sisendid; F^1 on peidetud kihi neuronite aktiveerimisfunktsioon; y_j^h on peidetud kihi neuronite väljundid; w_{kj}^v on väljundkihi neuronite kaalukoefitsientide väärtused; $\theta_{\mathrm{k}}^{\mathrm{v}}$ on väljundkihi neuronite konstantsed sisendid; F^2on väljundkihi neuronite aktiveerimisfunktsioon ja y_k on selle närvivõrgu väljundid. Et vältida üleliigseid tähistusi, on indeksite muutumisvahemikud jäetud märkimata.

Väljundkihi neuronite hälbed leitakse järgmise valemi abil.

$$\delta_k = y_k(1 - y_k)(y_k^p - y_k)$$

Peidetud kihi neuronite hälbed arvutatakse järgmiselt.

$$\delta_j^h = y_j(1 - y_j)\sum_k w_{kj}^v \delta_k$$

Suurused y_k^p tähistavad soovitud väljundväärtuseid. Sisendkihi neuronite hälbeid ei leita, sest sisendkihi neuroniteks on etteantud sisendväärtused ning seal arvutamist ei toimu.

Kaalukoefitsientide uued väärtused leitakse järgmiselt.

$$w_{kj}^v \leftarrow w_{kj}^v + \eta \delta_k y_j^h,$$

$$w_{ji}^h \leftarrow w_{ji}^h + \eta \delta_j^h x_i,$$

kus η on õppimise kiirust iseloomustav koefitsient. Mida suurem on õppimise kiirust iseloomustav koefitsient, seda rohkem muutuvad närvivõrgu parameetrid igakordsel ümberarvutamisel. Peale uute parameetrite leidmist arvutatakse uued närvivõrgu väljundid ning kui väljundi hälve on väiksem nõutavast, siis lõpetatakse õpetamine. Vastasel korral õpetamine jätkub. Sama algoritmi järgi leitakse ka neuronite konstantsete sisendite optimaalsed väärtused.

Õppimiskordaja ehk õppimise kiirust iseloomustav koefitsient η on nn metaparameeter, mille väärtuse määrab kasutaja. Kui õppimise kiirust iseloomustav koefitsient η on liiga väike, siis õpetamise võib olla väga aeglane ning õpetamiseks kuluv sammude arv väga suur. Kui aga η on liiga suur, siis ei pruugi algoritm koonduda. Selle parameetri sobiva väärtuse peab kasutaja ise välja selgitama.

Hälbe pöördlevi meetod tagab funktsiooni lokaalse miinimumi leidmise ning seepärast sõltub õpetamise tulemus parameetrite algväärtuste valikust. Kuna parameetrite algväärtused on juhuslikud arvud, siis erinevatel katsetel koondub algoritm erinevalt. Parima tulemuse saavutamiseks tuleb õpetamisprotsessi korrata mitu korda. Stanfordi Ülikooli tehisintellekti labori direktor Andrew Ng on öelnud, et fakt, et hälbe pöördlevi meetod tagab lokaalse miinimumi, pole praktikas suur probleem [2]. Nimelt, kuigi antud meetod ei pruugi tagada globaalset miinimumi, siis enamus kordadel leiab ta siiski hea lokaalse miinimumi.

5. Tehisnärvivõrgud paketis R

R on programmeerimiskeel, mis on eeskätt koostatud statistikavajadusi silmas pidades. **R** on vaba tarkvara ja seda on võimalik alla laadida **R** koduleheküljelt [12]. Samuti leiab sellelt leheküljelt mitmeid juhendeid antud programmeerimiskeele kasutamiseks. **R** koosneb põhiosast ja erinevatest lisapakettidest. Närvivõrkude jaoks on mitmeid lisapakette: **neuralnet**, **nnet**, **AMORE** jne. Antud töös keskendume paketile **neuralnet**, millega on võimalik koostada mitmekihilisi pertseptroneid. Närvivõrgu treenimiseks lubab antud pakett kasutada järgmisi õpetamismeetodeid: hälbe pöördlevi meetod, individualiseeritud hälbe pöördlevi meetod kaalude ülevaatamisega või kaalude ülevaatamiseta ning modifitseeritud versioon.

Närvivõrgu õpetamine paketis **neuralnet** näeb välja järgmine.

1. Närvivõrk arvutab väljundi antud sisendikomplekti ja kaalude korral.
2. Närvivõrgu hälbefunktsiooni E arvutamiseks on antud paketis kaks võimalust.

$$E = \frac{1}{2}\sum_{l}\sum_{k}(y_{lk}^{p} - y_{lk})^2 \tag{3}$$

või

$$E = -\sum_{l}\sum_{k}(y_{lk}^{p}\ln(y_{lk}) + (1 - y_{lk}^{p})\ln(1 - y_{lk})), \tag{4}$$

kus l tähendab sisendikomplekti ning k väljundkihi neuronite järjekorranumbrit. Närvivõrgu hälbefunktsiooni ülesandeks on mõõta erinevust oodatavate ja närvivõrgu poolt arvutatud väljundite vahel.

3. Kui närvivõrgu poolt arvutatud väljund erineb oodatust, siis korrigeeritakse kõik kaalud vastavalt kasutatavale õpetamisalgoritmile.

Protsess lõpetatakse, kui peatumiskriteerium on täidetud, see tähendab, kui hälvete muutused on väiksemad fikseeritud tõkkest.

Sagedasti kasutatav õpetamisalgoritm on individualiseeritud hälbe pöördlevi meetod (*resilient backpropagation*), mis erineb tavalisest hälbe pöördlevi meetodist selle poolest, et igal kaalul on erinev õppimise kiirust iseloomustav koefitsient η_m, kus m tähistab vastavat kaalukoefitsienti. Seega ei pea määrama üleüldist õppimise kiirust iseloomustavat koefitsienti η, mis kehtiks terve õpetamisprotsessi ning iga kaalu korral. Uued kaalud arvutatakse kahekihilise pertseptroni puhul järgmiselt.

$$w_{kj}^{o} \leftarrow w_{kj}^{o} - \eta_{kj} \cdot sign(\delta_k y_j^h),$$

$$w_{ji}^{h} \leftarrow w_{ji}^{h} - \eta_{ji}^{h} \cdot sign(\delta_j^h x_i).$$

Tavalise hälbe pöördlevi meetodi puhul, nagu nägime ka juba 4. peatükis, arvutatakse kahekihilise pertseptroni uued kaalud järgmiselt.

$$w_{kj}^{o} \leftarrow w_{kj}^{o} + \eta\delta_k y_j^h;$$

$$w_{ji}^{h} \leftarrow w_{ji}^{h} + \eta\delta_j^h x_i.$$

Viimases neljas valemis tähendab i neuroni sisendi järjekorranumbrit; j tähendab neuroni järjekorranumbrit peidetud kihis; k tähendab neuroni järjekorranumbrit väljundkihis; x_i tähistab närvivõrgu sisendeid; y_{ji}^{h} tähistab peidetud kihi neuronite väljundeid; δ_j^hja δ_k tähendavad vastavalt peidetud kihi neuronite hälbeid ja väljundkihi neuronite hälbeid; η_{ji}^{h}ja η_{kj} tähistavad vastava kaalukoefitsiendi õppimise kiirust iseloomustavat koefitsienti; w_{kj}^{o} tähistab väljundkihi neuroni kaalukoefitsientide väärtusi ning w_{ji}^{h} peidetud kihi neuronite kaalukoefitsientide väärtusi.

Kaalude ülevaatamise mõiste tähendab juba leitud kaalude muutude vähendamist enne järgmist iteratsioonisammu, eesmärgiga vältida võimalike paremate lahendite vahelejätmist.

Märgime, et paketis **R** on veel üks meetod, mida me käesolevas bakalaureusetöös ei käsitle. Modifitseeritud meetodi kohta saab lähemalt lugeda Anastasiadis'i, Magoulas'i ja Vrahatis'i artiklist „*New globally convergent ...*"" [1].

Närvivõrgu õpetamiseks paketis **R** peab meil olema andmetabel, mis sisaldab õpetamiseks kasutatavaid sisendikomplekte ja neile sisendikomplektidele vastavaid oodatavaid väljundväärtuseid. Närvivõrgu õpetamiseks kasutatav funktsioon on $neuralnet$. Märgime, et paketi **R** närvivõrkude õpetamiseks kasutatav alampakett on sama nimega nagu selle paketi põhifunktsioon. Antud funktsioon võimaldab määrata peidetud kihtide arvu ning neuronite arvu igas kihis. Vaikimisi väärtuseks on üks peidetud kiht ühe neuroniga. Teoreetiliselt lubab pakett **neuralnet** kasutada suvalist arvu neuroneid nii sisend- kui väljundkihis. Praktikas aga võib väga suurte arvude kasutamine tekitada raskusi. Funktsiooni $neuralnet$ kasutamise süntaks on järgmine.

$$nn \leftarrow neuralnet(formula, data, hidden, \dots)$$

Tähtsamad sisendmuutujad on:

- $formula$ – kirjeldus mudeli kohta. See esitatakse kujul: $v\ddot{a}ljundid \sim sisendite\ summa$. Kasutatav andmestik, mis vajaminevaid muutujaid sisaldab, peab olema ette antud.

- $data$ – andmetabel, mis sisaldab eelmises punktis kirjeldatud valemi muutujaid.

- $hidden$ – vektor, mis määrab peidetud kihtide ning neis kihtides paiknevate neuronite arvud. Näiteks vektor (3, 2, 1) kirjeldab närvivõrku, milles on kolm peidetud kihti. Esimeses peidetud kihis on 3 neuronit, teises 2 ja kolmandas 1. Vaikimisi väärtus on 1.

- $treshhold$ – arv, mis kirjeldab fikseeritud tõket, millest väiksemate hälvete muutude korral õpetamisprotsess lõpetatakse. Vaikimisi väärtus on 0,01.

- rep – korduste arv õpetamisprotsessis. Vaikimisi väärtus on 1.

- $startweights$ – vektor, mis sisaldab kaalude algväärtusi. Vaikimisi väärtuseks on juhuslikud arvud normaaljaotusest.

- $algorithm$ – string, mis sisaldab õpetamisalgoritmi tüüpi. Võimalikud väärtused on „backprop“”, „rprop+“”, „rprop-“” , „saq“” või „slr“”. String „backprop“” tähendab tavalist hälbe pöördlevi meetodit, „rprop+“” ja „rprop-“” tähendavad vastavalt kaalude ülevaatamisega ja kaalude ülevaatamiseta individualiseeritud hälbe pöördlevi meetodit. Stringid „saq“” ja „slr“” tähendavad modifitseeritud meetodit. Vaikimisi väärtuseks on „rprop+“”.

- $err.fct$ – string, mis sisaldab hälbefunktsioon. Võimalikud väärtused on „sse“” (vt lk 30, valem (3)) ja „ce“” (vt lk30, valem (4)). Vaikimisi väärtus on „sse“”.

- $act.fct$ – string, mis sisaldab aktiviseerimisfunktsiooni. Võimalikud väärtused „logistic“” ja „tanh“”. String „logistic“” tähendab logistilist aktiviseerumis-funktsiooni ja „tanh“” tähendab hüperpoolset tangensit. Vaikimisi väärtuseks on „logistic“”.

- $linear.output$ – loogilist tüüpi muutuja. Kui me ei taha väljundkihi neuronite aktiveerimisfunktsiooniks logistilist ega hüperboolset tangensit, siis peab antud sisendmuutuja väärtus olema TRUE. Sellisel juhul väljundkihi neuronite väljunditeks on sisendite kaalutud summa. Vaikimisi väärtuseks on TRUE.

Informatsioon õpetamisprotsessi kohta salvestatakse muutujas nn. Muutujanimi nn omistatakse närvivõrgule kasutaja poolt. Mõned olulised väärtused:

- $net.result$ – väljastab üleüldised tulemused, see tähendab närvivõrgu väljundid igal iteratsioonisammul.

- $weights$ – väljastab treenitud närvivõrgu kaalukoefitsientide väärtused.

- $result.matrix$ – andmemaatriks, mis sisaldab närvivõrgu hälvet, saavutatud lävendit, õpetamisprotsessiks kulunud samme ning leitud kaalukoefitsiente.

- $startweights$ – väljastab esialgsed kaalukoefitsientide väärtused.

Tulemuste visuaalset vaatlemist võimaldab käsk $plot(nn)$, mis koostab treenitud närvivõrgu joonise. Vaikimisi märgitakse joonisele õpetamise käigus leitud kaalud, samuti põhiline informatsiooni õpetamise protsessi kohta nagu näiteks närvivõrgu hälve ja sammude arv, mis õpetamiseks kulus.

Treenitud närvivõrgu kasutamiseks on käsk $compute$. Seda kasutatakse väljundite arvutamiseks tundmatute sisendkomplektide korral. Antud käsk väljastab kõigi närvivõrku kuuluvate neuronite väljundid.

5.1 Närvivõrk, mis väljastab ruutjuure etteantud arvust

Järgnevalt anname juhise, kuidas konstrueerida paketis **R** selline närvivõrk, mis väljastab ruutjuure etteantud arvust. Programmikood on esitatud ka lisas 1 kuid järgnevalt seletame lahti, kuidas antud programm on koostatud ning mida see sisuliselt teeb (lähtekoodi vt [4]). Selleks, et saaksime kasutada paketi **R** alampaketi **neuralnet** närvivõrgu õpetamise funktsiooni $neuralnet$, tuleb antud pakett installeerida. Seda saab teha kasutades järgnevaid käske.

$install.packages('neuralnet')$
$library("neuralnet")$

Treeningkomplektide jaoks genereerime 50 suvalist arvu, mis on ühtlaselt jaotunud vahemikus (0, 100). Selle jaoks on paketis **R** funktsioon $runif(n, min, max)$, kus n tähendab numbrite arvu, mida saada tahame, min ja max tähendavad vastavalt alumist ja ülemist tõket. Hoiustame genereeritud arvud andmetabelina. Andmetabeli moodustamiseks on funktsioon $as.data.frame(x)$, mis moodustab andmetabeli elementidest x. Nimetame selle andmetabeli $traininginput$. Andmetabel $traininginput$ sisaldab närvivõrgu treenimiseks kasutatavaid sisendeid. Leiame neile sisenditele ka vastavad oodatavad väljundid. Kuna tahame koostada sellise närvivõrgu, mis väljastaks ruutjuure, siis tuleb leida andmetabelis $traininginput$ paiknevatest arvudest ruutjuured. Ruutjuure leidmiseks on funktsioon $sqrt(x)$, mis väljastab arvu x ruutjuure. Kusjuures x võib olla ka andmetabel. Hoiustame saadud ruutjuured tabelis,

mille muutujanimeks omistame $trainingoutput$. Kirjeldatud tegevuse saab paketis **R** kirja panna järgmiste käskudega.

$$traininginput \leftarrow as.data.frame(runif(50, min = 0, max = 100))$$
$$trainingoutput \leftarrow sqrt(traininginput)$$

Ühendame andmetabelid $traininginput$ ja $trainingoutput$ üheks andmetabeliks. Andmetabeleid saab omavahel ühendada kasutades funktsiooni $cbind(x, y)$, kus x ja y on andmetabelid, mida tahetakse ühendada. Nimetame saadud andmetabeli $trainingdata$. See andmetabel sisaldab närvivõrgu treeningkomplekte, mille esimeses veerus paiknevad treenimiseks kasutatavad sisendid ning teises veerus neile sisenditele vastavad oodatavad väljundid. Nimetamegi andmetabeli veerud vastavalt „Sisend“” ja „Väljund“”. Seda võimaldab teha funktsioon $colnames(x)$ kus x on andmetabel, mille veergudele tahame nimetusi anda. Antud tegevus näeb programmis R välja järgmine.

$$trainingdata \leftarrow cbind(traininginput, trainingoutput)$$
$$colnames(trainingdata) \leftarrow c("Sisend", "Väljund")$$

Järgnevalt kirjeldame närvivõrgu treenimisprotsessi. Konstrueerime kahekihilise närvivõrgu, millel on üks sisend ja üks väljund. Sisendiks on arv, millest tahame ruutjuurt saada ning väljundiks närvivõrgu poolt arvutatud ruutjuur. Peidetud kihi neuronite arvuks määrame 10 neuronit. Nagu eelnevast juba teame on närvivõrgu õpetamiseks kasutatav funktsioon $neuralnet(formula, data, hidden, \dots)$. Treening-komplektiks on eelnevalt konstrueeritud andmetabel $trainingdata$ ning fikseeritud tõkkeks, millest väiksemate hälvete muutude korral õpetamisprotsess lõpetatakse, määrame 0,01. Õpetamisalgoritmina kasutame individualiseeritud hälbe pöördlevi meetodit, mille korral pole vaja määrata üldist õppimise kiirust määravat koefitsienti. Närvivõrgu muutujanimeks määrame $net.sqrt$. Selles muutujanimes salvestub info treeningprotsessi kohta. Üldist informatsiooni, s.o. treenitud närvivõrgu hälve (vt joonis 18, *Error*), õpetamiseks kulunud sammude arv (vt joonis 18, *Steps*) ja saavutatud lävend (vt joonis 18, *Reached treshold*), väljastab treenitud närvivõrgu kohta käsk $print(nn)$, kus nn on närvivõrgule omistatud muutujanimi. Paketis **R** näevad antud käsud välja järgmised.

$net.sqrt \leftarrow neuralnet(Väljund \sim Sisend, trainingdata, hidden = 10, threshold = 0.01)$

$print(net.sqrt)$

Viimane käsk väljastab tulemuseks andmetabeli, mida kirjeldab joonis 18. Märgime, et antud tulemused on saadud kasutades fikseeritud treeningkomplekte, mille sisendväärtused leidsime juhuslikult kasutades funktsiooni $runif$ ning oodatavad väljundväärtused arvutasime kasutades funktsiooni $sqrt$. Et saada sama tulemus, mida kirjeldab joonis 18, peavad nii treeningkomplektid kui ka kaalukoefitsientide algväärtused olema identsed nende väärtustega, mis antud näites genereeriti juhuslikult. Näites kasutatud juhuslikult genereeritud treeningkomplektide sisendid ning neile vastavad oodatavad väljundid on toodud lisas 1. Juhuslikult genereeritud kaalu-koefitsientide algväärtused on välja toodud lisas 2.

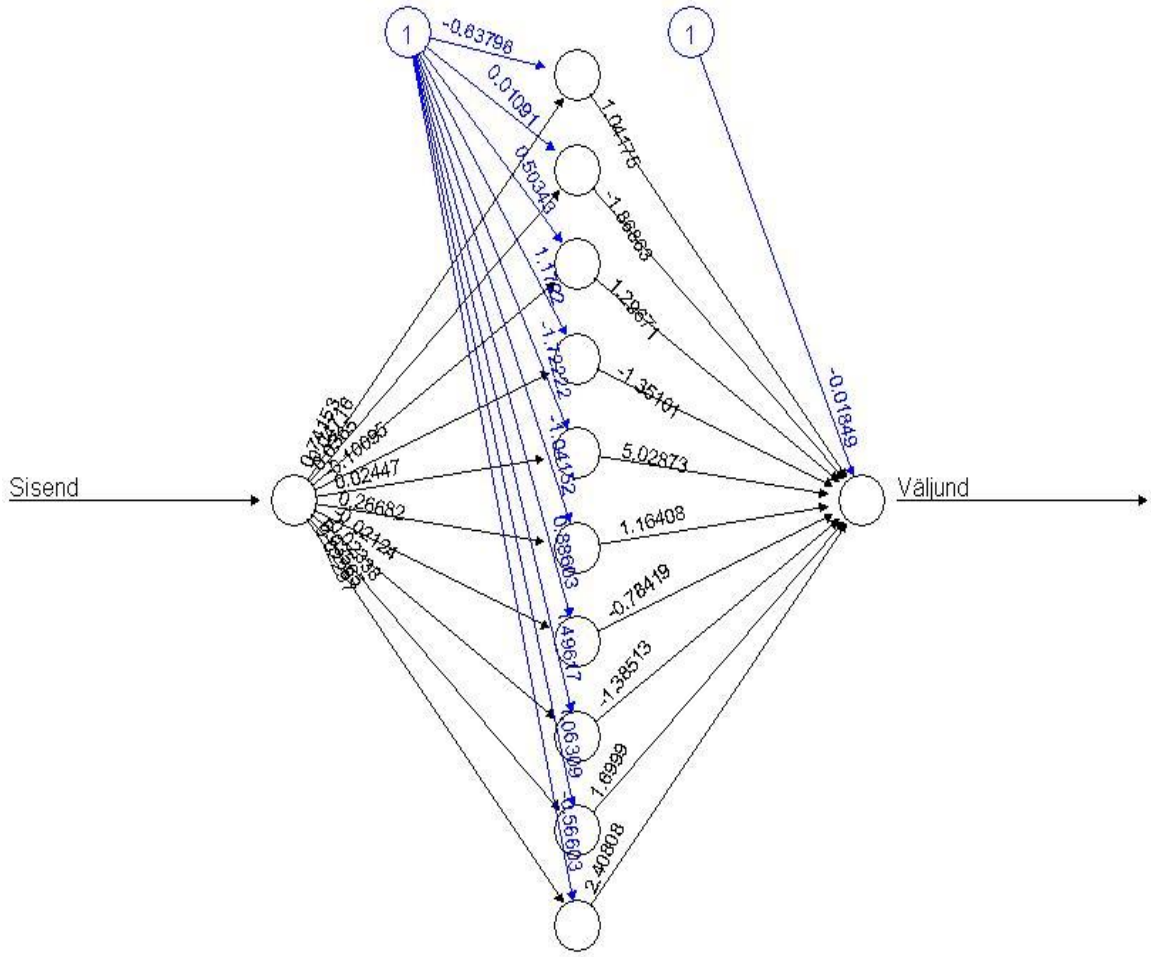
```
1 repetition was calculated.

            Error Reached Threshold Steps
1 0.0003829372519    0.009716605558  5474
```

Joonis 18. Üldine informatsioon närvivõrgu kohta

Treenitud närvivõrgust saab koostada joonise kasutades funktsiooni $plot(nn)$, kus nn on närvivõrgule omistatud muutujanimi. Joonisele märgitakse õpetamise käigus leitud kaalud, samuti põhiline informatsiooni õpetamise protsessi kohta nagu närvivõrgu hälve ja sammude arv, mis õpetamiseks kulus. Meie konstrueeritud närvivõrgust joonise koostav käsk näeb välja järgmine.

$plot(net.sqrt)$

Antud käsk väljastab närvivõrgust järgmise joonise. Treenitud närvivõrgu kaalukoefitsientide väärtused on välja toodud lisas 3.



Error: 0.000383 Steps: 5474

Joonis 19. Treenitud närvivõrk

Oleme koostanud närvivõrgu $net.sqrt$, mis väljastab ruutjuure etteantud arvust. Antud närvivõrgu hälve on 0,000383 ning selle närvivõrgu treenimiseks tehti 5474 sammu. Vaatleme järgmiselt, kuidas käitub treenitud närvivõrk tundmatute sisendväärtuste korral. Anname närvivõrgule $net.sqrt$ ette arvude 1 kuni 10 ruudud ning vaatame, mis väljundid närvivõrk antud sisendite korral arvutab. Arvude 1 kuni 10 ruudud hoiustame andmetabelina, mille nimetame $testdata$. Antud andmetabeli edastame närvivõrgule kasutades käsku $compute$. Närvivõrgu poolt arvutatud väljundid väljastab käsk $nn\$net.result$, kus nn tähendab närvivõrgule omistatud muutujanime. Kirjeldatud tegevus näeb programmis R välja järgmine.

$$testdata \leftarrow as.data.frame((1:10)\text{^}2)$$

$net.sqrt.results \leftarrow compute(net.sqrt, testdata)$
$print(net.sqrt.results\$net.result)$

Muutuja $net.sqrt.results$ sisaldab informatsiooni antud närvivõrgu iga neuroni väljundi kohta iga sisendi korral. Käsk $print(net.sqrt.results\$net.result)$ väljastab aga ainult väljundkihi neuroni väärtused ehk selle närvivõrgu väljundväärtused iga sisendväärtuse korral. Väljastatud tulemust kirjeldab joonis 20.

```
[1,] 1.051690764
[2,] 2.004334490
[3,] 2.999651314
[4,] 3.999261622
[5,] 5.000600326
[6,] 6.003965879
[7,] 6.996882397
[8,] 7.998709473
[9,] 9.006016246
10,] 9.970623687
```

Joonis 20. Arvutatud väljundväärtused

Esitame ülevaatlikuma versiooni tulemustest. Selleks koostame andmetabeli, kuhu lisame närvivõrgule etteantud sisendväärtused, millest närvivõrk peab ruutjuurte leidma. Samuti lisame sinna oodatavad väljundväärtused antud sisendväärtuste korral ning viimasena närvivõrgu poolt arvutatud väljundid, mis on eraldi välja toodud ka joonisel 20. Nimetame selle andmetabeli $cleanoutput$, ning anname tabeli vastavatele veergudele nimetuseks „Sisend“”, „Oodatav väljund“” ja „Närvivõrgu arvutatud väljund“”. Antud tegevus näeb paketis **R** välja järgmine.

$cleanoutput \leftarrow cbind(testdata, sqrt(testdata),$
$as.data.frame(net.sqrt.results\$net.result))$
$colnames(cleanoutput)$
$\leftarrow c("Sisend", "Oodatav\ väljund", "Närvivõrgu\ arvutatud\ väljund")$
$print(cleanoutput)$

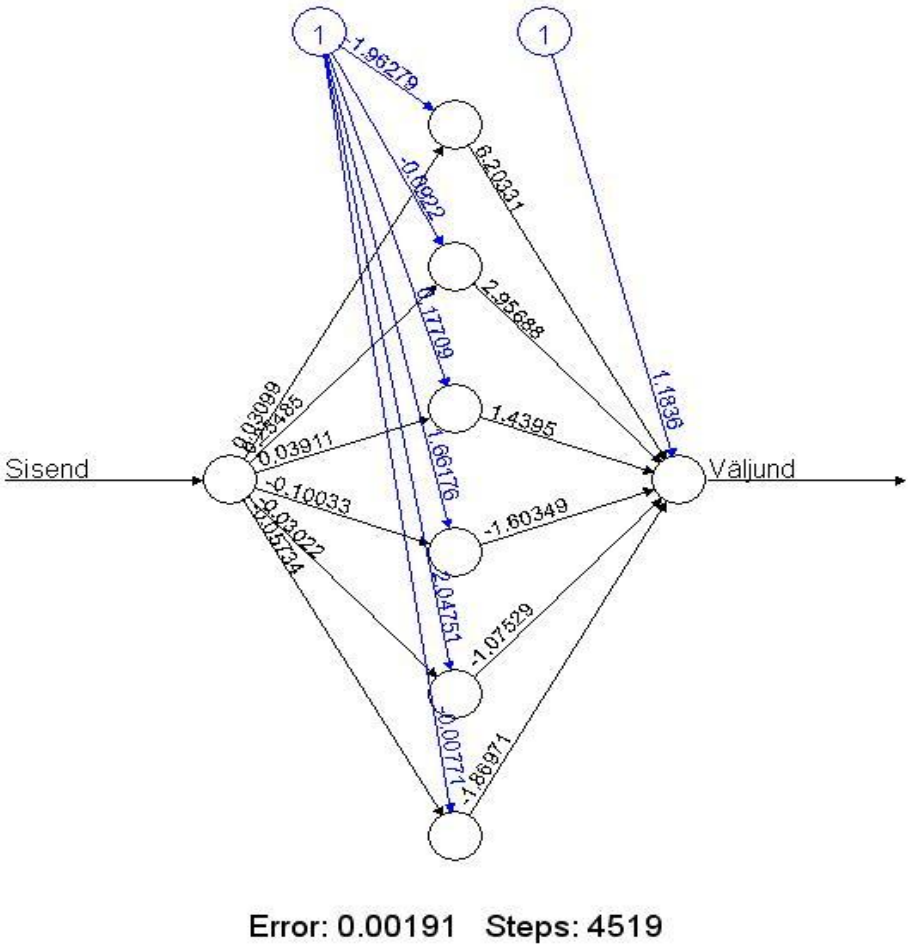
Tulemuseks väljastatakse andmetabel, mis on toodud joonisel 21.

Sisend	Oodatav väljund	Närvivõrgu arvutatud väljund
1	1	1.051690764
4	2	2.004334490
9	3	2.999651314
16	4	3.999261622
25	5	5.000600326
36	6	6.003965879
49	7	6.996882397
64	8	7.998709473
81	9	9.006016246
100	10	9.970623687

Joonis 21. Arvutatud väljundid tundmatute sisendväärtuste korral

Märgime, et kui kasutame närvivõrgu treenimiseks juhuslikult genereeritud treeningkomplekte ja kaalukoefitsientide algväärtusi, siis igakordsel programmi käivitamisel saame eelmisest erineva tulemuse. Lisas 5 on toodud selline programmikood, mis genereerib samasuguse närvivõrgu, mida kirjeldavad joonised 18 kuni 21. Antud programmikoodi korral on treeningkomplektidena kasutatud samu sisendite ja väljundite paare, mida kasutati eelnevalt kirjeldatud närvivõrgu treenimiseks ning mis on eraldi välja toodud lisas 2. Samuti on selles programmikoodis fikseeritud närvivõrgu kaalukoefitsientide algväärtused. Fikseeritud kaalukoefitsientide algväärtused on võrdsed nendega, mida eelmises näites kirjeldatud närvivõrgu korral genereeriti juhuslikult ning mis on eraldi välja toodud lisas 3.

Antud näite puhul kasutasime fikseeritud tõkke väärtuseks arvu 0,01. Nagu eelnevast teame, tähendab selle tõkke väärtus arvu, millest väiksemate hälvete muutude korral õpetamisprotsess lõpetatakse. Muutes fikseeritud tõkke väärtust väiksemaks, väheneb ka närvivõrgu hälve, kuid tuleb arvestada sellega, et mida väiksem fikseeritud tõkke väärtus on, seda kauem aega kulub närvivõrgu treenimiseks. Peidetud kihis paiknevate neuronite sobilik arv leitakse tavaliselt eksperimentaalselt. Katsetades antud ülesande korral väiksemat arvu neuroneid peidetud kihis, on võimalik leida närvivõrk, mis saab eelneva ülesandega samahästi või isegi paremini hakkama (s.t. et närvivõrgu hälve ning sammude arv ei suurene). Näiteks joonis 22 kirjeldab närvivõrku, mis väljastab samuti ruutjuure etteantud arvust, kuid mille peidetud kihis on kõigest 6 neuronit. Joonisel 22 kirjeldatud närvivõrgu treeningkomplektid on toodud lisas 6, kaalukoefitsientide algväärtused lisas 7 ja treenitud närvivõrgu kaalukoefitsiendid lisas 8.



Joonis 22. Närvivõrk kuue neuroniga peidetud kihis

Liiga väikese neuronite arvu korral aga muutub närvivõrgu hälve suuremaks ning seega ei saa närvivõrk enam antud ülesandega nii hästi hakkama. Vastupidiselt, suurendades neuronite arvu peidetud kihis, on võimalik erinevaid treeningkomplekte ja kaalukoefitsientide algväärtusi katsetades leida närvivõrk, mis saab antud ülesandega samahästi hakkama. Selline teguviis aga pole kuigi efektiivne, sest mida rohkem neuroneid peidetud kiht sisaldab, seda keerulisem ta on ning tavaliselt suurendades neuronite arvu suureneb ka treenimiseks kuluv aeg.

Oluline on ka treeningkomplektide valik. Kui me tahame treenitud närvivõrku kasutada ruutjuure leidmiseks arvudest vahemikus 0 kuni 100, siis peaksid ka treeningkomplektid sisaldama näiteid sellest vahemikust. Vastasel juhul, kui me treenime närvivõrku võtma ruutjuurt arvudest vahemikus (0, 100), ning seejärel tahame, et treenitud närvivõrk väljastaks meile ruutjuure arvust 225, siis antud närvivõrk sellega kuigi hästi toime ei tule. Joonisel 19 kirjeldatud närvivõrk väljastab arvu 225 korral 12,43756 (õige vastus

on 15). Tähelepanelik peab olema ka treeningkomplektide arvuga, sest mida rohkem treeningkomplekte me närvivõrgu õpetamiseks kasutame, seda rohkem aega õpetamine võtab.

Nagu eelnevast arutelust näeme, siis närvivõrkude konstrueerimine on ülesandepõhine ning parima tulemuse saamiseks tuleks katsetada erinevaid treeningkomplekte ning muid algväärtuseid nagu näiteks neuronite arv peidetud kihis ja kaalukoefitsientide algväärtused.

Feed-forward neural networks with R

Bachelor's thesis

Summary

Merili Liivoja

Human brain is a complex and powerful system that is able to solve a wide variety of tasks. The aim of many scientists is to develop a computer simulation that mimics the brain functions and solves problems the way our brains do. Very simplified models of biological neural networks are artificial neural networks. There are two different types of artificial neural networks – feed forward neural networks and recurrent neural networks. This thesis gives an overview of feed-forward neural networks and their working principles.

The thesis is divided into two main parts. The first part is the theory of feed-forward neural networks and the second part is a practical example of neural network with software **R**. The first part gives an overview of the artificial neuron and its history. Also different types of artificial neurons are introduced. The first part includes instructions of how feed-forward neural networks are composed and explains how they calculate the results.

Separate chapter is devoted to training artificial neural networks. The chapter gives an overview of two main training algorithms – perceptron training algorithm and back-propagation algorithm. The first is designed to train perceptrons and the second is often used in training multi-layer feed-forward neural networks.

The last topic explains how to construct feed-forward neural networks with software **R**. It includes a tutorial of how to build a neural network that calculates the square root. The tutorial will produce a neural network which takes a single input and produces a single output. Input is the number that we want square rooting and the output is the square root of the input.

Viidatud allikad

[1] A. Anastasiadis, G. Magoulas, M. Vrahatis, *New globally convergent training scheme vased on the resilient propagation algoritm,* Elsevier, 2005. Saadaval veebilehelt:
<http://www.google.ee/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCgQFjAA&url=http%3A%2F%2Fciteseerx.ist.psu.edu%2Fviewdoc%2Fdownload%3Fdoi%3D10.1.1.144.4759%26rep%3Drep1%26type%3Dpdf&ei=CIasUcqvJcar0gXV24CADg&usg=AFQjCNEC5nC__WywjeX3X7ngwp4qNXJEPw&bvm=bv.47244034,d.d2k>
[1. juuni 2013]

[2] A. Ng *Machine Learning: Putting It Together,* Stanfordi ülikool. Saadaval veebilehelt: < https://class.coursera.org/ml/lecture/preview> [12. mai 2013]

[3] C. Freudenrich, *How Nerves Work.* Saadaval veebilehelt:
<http://science.howstuffworks.com/life/human-biology/nerve3.htm> [10. mai 2013]

[4] *Neural Networks with R – A Simple Example.* Saadaval veebilehelt:
<http://gekkoquant.com/2012/05/26/neural-networks-with-r-simple-example/>
[4. märts]

[5] *Information flow through neurons,* 2005. Saadaval veebilehelt:
<http://www.uic.edu/classes/bios/bios100/lectures/nervous.htm> [1 aprill 2013]

[6] E. Ling, *Mapping the Brain with Connectomics,* 2011. Saadaval veebilehelt: <http://dmsbulletin.hms.harvard.edu/?p=965> [10. mai 2013]

[7] M. Marsalli, 2006 *McCulloch Pitts Neurons.* Saadaval veebilehel:
<http://www.mind.ilstu.edu/curriculum/mcp_neurons/mcp_neuron_1.php>
[8. aprill 2013]

[8] W. S. McCulloch, W. H. Pitts, *A logical calculus of the ideas immanent in nervous activity,* 1943. Saadaval veebilehel:
<http://www.google.ee/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0CC8QFjAB&url=http%3A%2F%2Fwww.cse.chalmers.se%2F~coquand%2FAUTOMATA%2Fmcp.pdf&ei=oKSsUd76IaOo0wXylIGwAQ&usg=AFQjCNGHHJH8L6--e0M8d8B4ity82ErG8g&bvm=bv.47244034,d.d2k > [28. mai 2013]

[9] *McCulloch-Pitts Neurons*. Saadaval veebilehelt: <http://www.i-programmer.info/babbages-bag/325-mcculloch-pitts-neural-networks.html> [12. Mai 2013]

[10] E. Petlenkov, *Tehisnärvivõrgud ja nende rakendused,* Tallinn, 2004. Saadaval veebilehel: <http://www.dcc.ttu.ee/las/ISS0010/Tehisnarvivorgud-Eduard2004.pdf> [5 märts 2013]

[11] F. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain,* American Psychological Association, 1958. Saadaval veebilehelt: <http://www.google.ee/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCgQFjAA&url=http%3A%2F%2Fblogimg.chinaunix.net%2Fblog%2Fupfile2%2F090505101133.pdf&ei=j6OsUf70A9TJ0AWX-YCQDw&usg=AFQjCNE2SDp8X_DTvyh3IR74h-nw8s_b8g&bvm=bv.47244034,d.d2k> [28. mai 2013]

[12] The R Project for Statistical Computing. Veebilehe aadress: <http://www.r-project.org/>

Lisa 1. Programmikood närvivõrgu *net.sqrt* koostamiseks

```
install.packages('neuralnet')
library("neuralnet")
traininginput <-  as.data.frame(runif(50, min=0, max=100))
trainingoutput <- sqrt(traininginput)
trainingdata <- cbind(traininginput, trainingoutput)
colnames(trainingdata) <- c("Sisend", "Väljund")
net.sqrt <- neuralnet(Väljund~Sisend, trainingdata, hidden=10, threshold=0.01)
print(net.sqrt)
plot(net.sqrt)
testdata <- as.data.frame((1:10)^2)
net.sqrt.results <- compute(net.sqrt, testdata)
print(net.sqrt.results$net.result)
 cleanoutput <- cbind(testdata,sqrt(testdata),
                      as.data.frame(net.sqrt.results$net.result))
colnames(cleanoutput) <- c("Sisend","Oodatav väljund","Närvivõrgu arvutatud
väljund")
print(cleanoutput)
```

Lisa 2. Närvivõrgu *net.sqrt* õpetamiseks kasutatud treeningkomplektid

Sisendväärtus	Oodatav väljundväärtus
70,50177329	8,396533409
55,93833795	7,47919367
5,929945363	2,435147914
76,74725598	8,760551123
68,00466331	8,246494001
1,677551656	1,295203326
55,72158482	7,464689198
19,94779098	4,466294995
28,49603607	5,338167857
58,94564723	7,677606869
48,73751523	6,981225912
73,55340158	8,576327978
46,21821195	6,798397749
16,03209765	4,004010196
40,62718691	6,373945945
57,38872048	7,575534336
36,43447456	6,036097627
14,51573139	3,809951626
65,75318624	8,10883384
74,42538082	8,627014595
67,99820622	8,246102487
17,04326419	4,128348845
14,72845338	3,837766718
24,10336316	4,90951761
29,71390095	5,451045858
97,05557977	9,851679033
12,02125046	3,467167499
31,29204924	5,593929678
64,52920635	8,033007304
46,1455805	6,793053842
36,38973616	6,032390584
23,73288919	4,871641324
44,68857842	6,684951639
61,52430857	7,843743275
53,12598459	7,288757411
68,40757299	8,270887074
1,977226743	1,406138949
8,960301545	2,993376279
19,81857859	4,451806217

51,23013384	7,157522884
22,38905553	4,731707465
8,099336154	2,845933266
21,57369084	4,644748738
82,53721455	9,084999425
44,6527449	6,682270938
70,20672073	8,378945085
24,13005682	4,91223542
83,10746073	9,116329345
10,09515366	3,177287154
77,05153385	8,77790031

Lisa 3 Närvivõrgu *net.sqrt* kaalukoefitsientide algväärtused

Peidetud kihi kaalukoefitsiendid:	Peidetud kihi neuronite konstantsed sisendid (joonisel sinisega):	Väljundkihi kaalukoefitsiendid:	Väljundkihi neuroni konstantne sisend (joonisel sinisega):
1,259363941 −0,4425448595 −0,06036693942 −1,142154692 0,1080250761 0,5091724843 −0,01715897811 −0,7979894747 −0,105438413 −0,1005217830	0,324838465 0,0277453408 0,40958149897 1,186747411 −1,5062423203 0,5753932528 0,88047626774 1,0579548391 0,964136932 −0,6912302557	0,6447007902 −1,5851473587 0,8881129737 −1,2428234211 0,3219118310 0,7585700676 −0,7858889148 −0,9592540018 1,2901681895 0,8573741759	−0,4155393324

Lisa 4. Treenitud närvivõrgu kaalukoefitsientide väärtused

Peidetud kihi kaalukoefitsiendid:	Peidetud kihi neuronite konstantsed sisendid (joonisel sinisega):	Väljundkihi kaalukoefitsiendid:	Väljundkihi neuroni konstantne sisend (joonisel sinisega):
0,74153 −0,04716 0,03650 −0,10095 0,02447 0,26682 −0,02124 −0,02338 0,03565 0,02297	−0,63796 0,01091 0,50343 1,17220 −1,72222 −1,04152 0,88603 1,49617 1,06309 −0,56603	1,04175 −1,86863 1,29671 −1,35101 5,02873 1,16408 −0,78419 −1,38513 1,69990 2,40808	−0,01849

Lisa 5. Programmikood fikseeritud treeningkomplektide ja kaalukoefitsientide algväärtustega

```
library("neuralnet")
traininginput <- rbind(70.50177329, 55.93833795, 5.929945363, 76.74725598,
68.00466331, 1.677551656, 55.72158482, 19.94779098, 28.49603607, 58.94564723,
48.73751523, 73.55340158, 46.21821195, 16.03209765, 40.62718691, 57.38872048,
36.43447456, 14.51573139, 65.75318624, 74.42538082, 67.99820622, 17.04326419,
14.72845338, 24.10336316, 29.71390095, 97.05557977, 12.02125046, 31.29204924,
64.52920635, 46.1455805, 36.38973616, 23.73288919, 44.68857842, 61.52430857,
53.12598459, 68.40757299, 1.977226743, 8.960301545, 19.81857859, 51.23013384,
22.38905553, 8.099336154, 21.57369084, 82.53721455, 44.6527449, 70.20672073,
24.13005682, 83.10746073, 10.09515366, 77.05153385)
trainingoutput <- sqrt(traininginput)
trainingdata <- cbind(traininginput, trainingoutput)
colnames(trainingdata) <- c("Sisend", "Väljund")
net.sqrt.startweights <- c(0.324838465, 1.259363941, 0.0277453408, -0.4425448595,
0.40958149897, -0.06036693942, 1.186747411, -1.142154692, -1.5062423203,
0.1080250761, 0.5753932528, 0.5091724843, 0.88047626774, -0.01715897811,
1.0579548391, -0.7979894747, 0.964136932, -0.105438413, -0.6912302557,
-0.1005217830, -0.4155393324, 0.6447007902, -1.5851473587, 0.8881129737,
-1.2428234211, 0.3219118310, 0.7585700676, -0.7858889148, -0.9592540018,
1.2901681895, 0.8573741759)
net.sqrt <- neuralnet(Väljund~Sisend, trainingdata, hidden=10, threshold=0.01)
print(net.sqrt)
plot(net.sqrt)
testdata <- as.data.frame((1:10)^2)
net.sqrt.results <- compute(net.sqrt, testdata)
print(net.sqrt.results$net.result)
cleanoutput <- cbind(testdata,sqrt(testdata),
                     as.data.frame(net.sqrt.results$net.result))
colnames(cleanoutput) <- c("Sisend", "Oodatav väljund", "Närvivõrgu arvutatud
väljund")
print(cleanoutput)
```

Lisa 6. Joonisel 22 kirjeldatud närvivõrgu treeningkomplektid

Sisendväärtus	Oodatav väljundväätus
85,90048717	9,26825157
91,91997196	9,587490389
52,45971382	7,242907829
43,54811804	6,59909979
41,97624957	6,478908054
39,10394236	6,25331451
76,04038189	8,72011364
87,83729218	9,372155152
88,65255937	9,415548809
63,95595474	7,997246698
13,77803776	3,71187793
75,80131867	8,706395274
75,80131867	6,742783112
17,13653144	4,139629384
17,49320512	4,182487911
13,6843212	3,699232514
71,88119539	8,478277855
20,26970999	4,502189466
37,44489297	6,119223232
17,4194451	4,173660875
52,24590777	7,228133076
88,50430043	9,407672424
69,89357162	8,360237534
78,734057	8,873221343
23,72732544	4,871070256
67,04064871	8,18783541
44,68500351	6,684684249
36,69375933	6,057537398
47,7861994	6,912756281
96,9924968	9,848476877
91,03753748	9,54135931
27,26837725	5,221913179
81,06411859	9,003561439
62,79246179	7,92416947
38,13240381	6,175144032
57,90781041	7,609718156
41,58102705	6,448335215
56,7733953	7,534812227
69,26344801	8,322466462

2,466944722	1,57065105
63,25918834	7,953564505
4,145240039	2,035986257
77,15916277	8,784028846
53,75682549	7,331904629
79,09863531	8,893741356
18,06806086	4,250654169
84,31644475	9,182398638
11,80740423	3,436190365
71,77943804	8,472274667
75,412046	8,68401094

Lisa 7. Joonisel 22 kirjeldatud närvivõrgu kaalukoefitsientide algväärtused

Peidetud kihi kaalukoefitsiendid:	Peidetud kihi neuronite konstantsed sisendid (joonisel sinisega):	Väljundkihi kaalukoefitsiendid:	Väljundkihi neuroni konstantne sisend (joonisel sinisega):
0,2615023221 1,775692955 −0,03450896731 −1,635231433 −0,6228421547 0,2431074410	−0,4574101417 1,065687197 0,69419960762 1,559215842 1,3639112877 0,9409337212	1,55798653111 1,57335914944 −0,08533763453 −1,58470541249 −0,09814890700 −2,00516446993	−0,19991288643

Lisa 8. Joonisel 22 kirjeldatud treenitud närvivõrgu kaalukoefitsiendid

Peidetud kihi kaalukoefitsiendid:	Peidetud kihi neuronite konstantsed sisendid (joonisel sinisega):	Väljundkihi kaalukoefitsiendid:	Väljundkihi neuroni konstantne sisend (joonisel sinisega):
0,03099 0,25485 0,03911 −0,10033 −0,03022 −0,05734	−1,96279 −0,09220 0,17709 1,66176 2,04751 −0,00771	6,20331 2,95688 1,43950 −1,60349 −1,07529 −1,86971	−1,18360

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina Merili Liivoja (sünnikuupäev: 19.02.1991)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Otsesuunatud tehisnärvivõrgud paketis R", mille juhendaja on dots. Peep Miidla,

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **4.06.2013**