LATVIJAS UNIVERSITĀTE
FIZIKAS UN MATEMĀTIKAS FAKULTĀTE
MATEMĀTIKAS NODAĻA

VALŪTAS KURSU PROGNOZĒŠANA AR ATGRIEZENISKO IZPLATĪŠANĀS ALGORITMU

BAKALAURA DARBS

Autors: **Zane Balode**

Stud. apl. zb10012

Darba vadītājs: prof. Dr.mat. Jānis Buls

RĪGA 2015

Anotācija

Bakalaura darbā ir apskatīti mākslīgie neironi un mākslīgo neironu tīklu arhitektūra (topoloģija). Ir aprakstīts atgriezeniskais izplatīšanās algoritms, kā arī ir izvestas formulas sešiem neironu tīklu modeļiem un vispārīgajam gadījumam. Ar šiem modeļiem ir prognozēti seši valūtu kursi, un prognožu precizitāte tiek salīdzināta ar $ARIMA$ modeļu prognozēšanas precizitāti. Darba ietvaros ir uzprogrammēts tīkls *Matlab* un *R* vidē. Darbā arī tiek veikti eksperimenti, lai noskaidrotu, kādi parametri ir jāizvēlas, lai ar attiecīgo modeli varētu prognozēt attiecīgos valūtu kursus. Darba mērķis ir izveidot mākslīgo neironu tīklu, kas tiek apmācīts ar atgriezenisko izplatīšanās algoritmu, kas prognozētu precīzāk kā $ARIMA$ modelis.

Atslēgas vārdi: mākslīgie neironu tīkli, perceptrons, atgriezeniskais izplatīšanās algoritms, prognozēšana, $ARIMA$, valūtu kurss

Abstract

The themes covered in this bachelor thesis are artificial neurons and artificial neural network topology. The thesis also covers backpropagation algorithm, and calculated formulas for six modules. Six exchange rates are forecast using these modules and the results are compared with the $ARIMA$ module. As a part of this thesis a neural network in both *Matlab* and *R* environments was programmed. Experiments were also carried out to find out what parameters have to be chosen so that a specific module can forecast a specific exchange rate. The goal of this thesis is to create a neural network, which is powered by backpropagation algorithm, which could forecast more precisely than an $ARIMA$ module.

Keywords: artificial neuron network, perceptron, backpropogation algorithm,$ARIMA$, forecasting, exchange rate

Apzīmējumi

MAD - vidējā absolūtā standartkļūda,

$MAPE$ - vidējā absolūtā kļūda,

MSE - vidējā kvadrātiskā kļūda,

$ARIMA$ - autoregresīvais integrētais slīdošā vidējā process,

ACF - autokorelāciju funkcija,

$PACF$ - parciālā autokorelāciju funkcija,

AIC - Akaike informācijas kritērijs,

BIC - Švarca-Beijesa kritērijs.

Saturs

Ievads

Valūtas tirgus *(foreign exchange market)* ir tirgus, kur viena valūta tiek apmainīta pret otru. Tas sevī ietver tirdzniecību starp vislielākajām pasaules bankām, centrālajām bankām, valūtas spekulantiem, starptautiskajiem uzņēmumiem, valdībām, kā arī citiem finansu tirgiem un institūcijām. Tirgošanās apjoms valūtas tirgos visā pasaulē šobrīd ir ap 2 triljoniem dolāru dienā. Individuālie spekulanti šajā tirgū šobrīd aizņem ļoti mazu daļu un var piedalīties tikai netieši - izmantojot brokerus vai bankas. [8]

Daudziem uzņēmuemiem ir interese prognozēt, kā valūtu kursi mainīsies nākotnē, jo tas palīdz pieņemt dažādus biznesa lēmumus un veidot nākotnes biznesa stratēģijas. Spējot paredzēt, kā mainīsies valūtu kurss, ir iespējams samazināt risku un palielināt peļņu, tādēļ ir nepieciešams atrast modeli, kas spētu pēc iespējas precīzāk prognozēt valūtu kursu izmaiņas. [18] Kamēr netiks atrasts modelis, kas spēj prognozēt valūtu kursus pilnīgi precīzi, tiks meklēti jauni modeļi, kas to spēj darīt precīzāk nekā iepriekš izveidotie, tādēļ šī tēma ir aktuāla.

1943. gadā neiropatalogs Varens Makuločs *(Warren McCulloch)* un matemātiķis Valters Pits *(Walter Pitts)* sāka attīstīt teoriju par mākslīgo neironu un mākslīgo neironu tīklu. [13] Mākslīgie neironu tīkli ir datu modelēšanas instruments, kas ir spējīgs iegūt un attēlot sarežģītas ieeju un izeju attiecības. Motivācija neironu tīklu tehnoloģiju attīstīšanai izriet no vēlēšanās radīt mākslīgu sistēmu, kas varētu realizēt inteliģentus uzdevumus līdzīgus tiem, kādus spēj realizēt cilvēka smadzenes. Astoņdesmitajos gados sāka attīstīt ideju par mākslīgo neironu tīklu pielietojumu valūtu kursu prognozēšana. [8]

Vairāk nekā trīs desmitgades ir veikti pētījumi par valūtas kursa prognozējamību, ar mērķi atrast labāku matemātisko modeli valūtas maiņas kursa prognozēšanai nekā patvaļīgās klejošanas modelis. Jauni modeļi tiek meklēti, jo tradicionālie valūtas kursu prognozēšanas modeļi ir nepietiekami. Ņemot vērā, ka valūtas kursa prognozēšanā nozīme ir gan praktiskajai, gan teorētiskajai daļai, liels skaits metožu un paņēmienu (ieskaitot lineārus un nelineārus) ir pārbaudīti. Ņemot vērā mākslīgo tīklu straujo attīstību, pētnieki un investori cer, ka valūtas tirgus noslēpumi var tikt atrisināti ar neironu tīklu modeļiem. Galvenais iemesls izvēloties mākslīgo neironu tīklu kā valūtas kursa prognozēšanas rīku, ir apstāklis, ka dažādas mākslīgo neironu tīklu atšķirības iezīmes padara tos noderīgus prognozēšanā. [8]

Darba mērķis ir izveidot mākslīgo neironu tīklu, kas spēj prognozēt valūtu kursus labāk par $ARIMA$ modeli. Balstoties uz autora [8] veikto pētījumu - publikāciju analīzi, tiek izvirzīta pirmā hipotēze – ar mākslīgo neironu tīklu valūtu kursu prognozes ir precīzākas nekā ar $ARIMAS$ modeli.

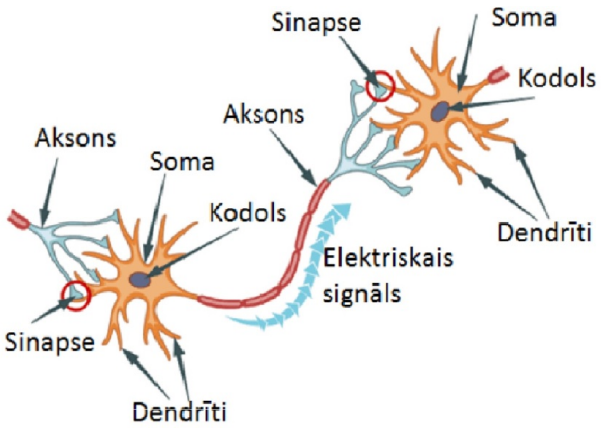
Darba uzdevums ir uzprogrammēt mākslīgo neironu tīklu sešiem modeļiem, kas tiek apmācīts ar atgriezenisko izplatīšanās algoritmu, programmās *Matlab* un *R*, izvest formulas šiem modeļiem, prognozēt valūtu kursus un izdarīt secinājumus. Modeļi tiek izvēlēti gan tādi, kas tiek aprakstīti apskatītajā literatūrā, gan tādi, kas netiek aprakstīti apskatītajā literatūrā. Tātad tiek izvirzīta otrā hipotēze – 1., 2. un 4. modelis spēs prognozēt precīzāk nekā 3., 5. un 6. modelis.

Atgriezeniskā izplatīšanās algoritma teorētiskais apraksts tiek galvenokārt balstīts uz avotiem - [13], [12] un [3]. Valūtu kursu prognozēšana ar mākslīgo neironu tīklu tiek pamatā balstīta uz avotu [8] un ar $ARIMA$ modelim uz avotiem [9] un [4]. Kā arī tiek apskatīts bakalaura darbs, kas ir pētījis līdzīgu problēmu - [15].

Darba sākumā lasītājs tiek iepazīstināts ar bioloģisko neironu, formālo mākslīgo neironu, mākslīgo neironu tīklu un vienkāršiem piemēriem. Tālāk tiek detalizēti aprakstīts atgriezeniskais izplatīšanās algoritms, kā arī tiek izvestas visu apskatīto modeļu formulas. Tiek aprakstīti citu autoru pētījumi saistībā ar valūtu kursu prognozēšanu, lietojot mākslīgo neironu tīklu - parametru izvēle, modeļu izvēle. Nobeigumā ir aprakstīti rezultāti, kas tika iegūti ar makslīgo neironu tīklu, ka arī tie tiek salīdzināti ar $ARIMA$ modeļa prognozes reuzltātiem. Pielikumā ir izvestas vispārīgā modeļa formulas atgriezeniskajam izplatīšanās algoritmam un izveidotie kodi programmās *Matlab* un *R*.

1 Bioloģiskais neirons

Cilvēka smadzenēs ir apmēram 100 miljardi šūnu jeb neironu. Neirozinātnē bioloģiskais neironu tīkls (dažreiz to sauc par nervu ceļu) ir virkne ar savstarpēji cieši saistītiem neironiem, kas kopā risina konkrētu problēmu, piemēram, tendences atpazīšanu un prognozēšanu, optimizāciju vai datu klasifikāciju. [19]



Att. 1.1: **Bioloģiskā neirona uzbūve**

Attēlā 1.1 ir parādīta bioloģiskā neirona uzbūve. Neirons ir nervu sistēmas pamatvienība, kas sastāv no šūnas ķermeņa (somas) un izaugumiem un kas vada nervu impulsus. Tās ir elektriski kairināmas šūnas, kuras apstrādā un nodod tālāk informāciju. No somas iziet neskaitāmi izaugumi – tievie, sazarotie dendrīti un resnākais aksons. Dendrīti nodrošina aktivitāšu uztveršanu no citiem neironiem. Aksons darbojas kā pārvadītājs, kas nosūta aktivitāti citiem neironiem. Sinapses nodrošina signālu apmaiņu starp aksonu un dendrītiem. Ienākošos signālus no citiem neironiem savāc dendrīti. Tālāk neirona ķermenis (soma) apkopo ienākošos signālus. Kad saņemts pietiekams daudzums signālu, t.i., slieksnis ir pārsniegts, neirons rada darbības potenciālu, t.i., ”izšauj”. Šīs darbības potenciāls tiek sūtīts caur aksonu uz citiem neironiem vai uz struktūrām, kas atrodas ārpus nervu sistēmas. Ja netiek sasniegts pietiekams signālu potenciāls, t.i., slieksnis netiek pārsniegts, tad signāls ātri sabojājas un netiek radīts darbības potenciāls. Tādēļ svarīgs ir laiks, ieejas signāliem ir jāienāk reizē, jo spēcīgi ieejas signāli radīs vairākus darbības potenciālus vienā laika vienībā. [19], [20]

Mākslīgie neironu tīkli cenšas simulēt cilvēku smadzeņu spēju mācīties. Neironu tīkls arī

tiek veidots no neironiem un dendrītiem. Atšķirīgi no bioloģiskajiem neironu tīkliem, mākslīgo neironu tīklu struktūra ir nemaināma, tas tiek būvēts no noteikta skaita neironiem un noteikta skaita svariem, kas savieno neironus, kuriem ir noteiktas vērtības. [17]

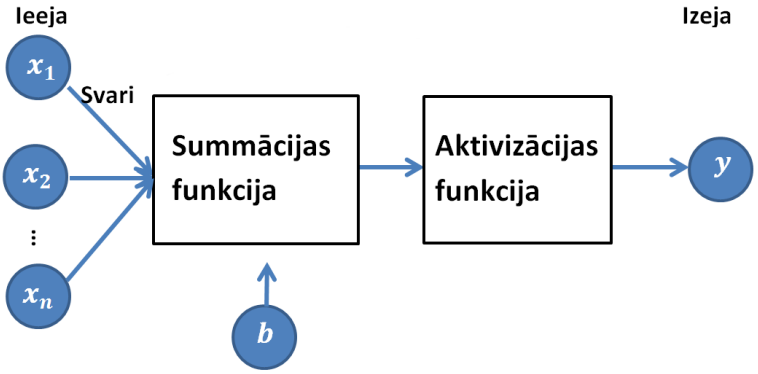
Mākslīgie neironu tīkli tiek uzskatīti kā nākotne skaitļošanā. Tie ir pamācības mehānismi, kam nav nepieciešamas tradicionālās programmētāja prasmes. Daudzos avotos tiek pausti maldīgi uzskati, ka neironu tīkli spēj izdarīt jebko. Šie pārspīlējumi ir radījuši vilšanos potenciālajos lietotājos, kas ir neveiksmīgi mēģinājuši risināt savas problēmas ar neironu tīkliem. Līdz ar to ir radies iespaids, ka neironu tīkli ir sarežģīti un mulsinoši. [19] Ir ļoti daudz rakstu par dažādiem neironu tīkliem, kam ir unikālas prasības un kas ir domāti specifiskiem gadījumiem. Pašlaik tikai daži no šiem neironu tīkliem tiešām tiek izmantoti komerciāli.

2 Mākslīgais neirons

Mākslīgie neironi sastāv no ieejām (bioloģiski – dendrīti), kas, kombinējot ar svariem (bioloģiski – sinapses), iegūst matemātisku funkciju, kuru sauc par summācijas funkciju. Cita funkcija jeb aktivizācijas funkcija aprēķina izeju (bioloģiski – aksons) neironam no summācijas funkcijas izdotās vērtības. Neironu tīkls kombinē neironus, lai apstrādātu informāciju. [3]

Definīcija 1. [13] Formāls neirons ir (X, Y, s, σ), kur $X \subseteq \mathbb{R}^n$, $Y \subseteq \mathbb{R}$ un s, σ ir atēlojumi $s : X \to \mathbb{R}$ un $\sigma : \mathbb{R} \to Y$.

Ar X apzīmē ieejas vērtību kopu, ar Y apzīmē izejas vērtību kopu, ar s apzīmē summācijas funkciju, un ar σ apzīmē aktivizācijas funkciju. Neironā summācijas funkcija un aktivizācijas funkcija tiek izvēlēta atkarībā no modeļa izvēles. [13] Attēlā 2.1 ir parādīta mākslīgā neirona uzbūve.



Att. 2.1: **Mākslīgā neirona uzbūve**

Svari

Svarus apzīmē ar w_i, kur $i = 1, \ldots, m$ un m ir svaru skaits. Svari ir skaitliskas vērtības, kas apmācības procesā tiek uzlaboti, lai beigās neironu tīkls būtu ieguvis spēju risināt konkrētu problēmu. Svaru skaits atbilst ieeju skaitam, bet atsevišķos gadījumos tiek pievienots papildus svars. [7], [13]

Summācijas funkcija

Izmantojot ieejas vērtības (bieži tiek teikts - ieejas signāli) un svarus, tiek iegūta summācijas funkcija. Praksē visbiežāk pielietotā summācijas funkcija:

$$s = \sum_{i=1}^{n} w_i x_i - b, \tag{2.01}$$

kur ar b apzīmē sliekšņa jeb bāzes vērtību, ko bieži vien summēšanas funkcijā pieraksta kā papildus svaru w_0. Iegūtā vērtība tālāk tiek sūtīta uz aktivizācijas funkciju. [3]
Citas summācijas funkcijas:

$$s = \prod_{i=1}^{n} w_i x_1,$$

$$s = max(w_i x_i), i = 1...m,$$

$$s = min(w_i x_i), i = 1...m.$$

Aktivizācijas funkcija

Aktivizācijas funkcija izmanto summācijas funkcijas vērtību s, lai izrēķinātu izejas vērtību y. Aktivizācijas funkcija var būt gan lineāra, gan nelineāra. To izvēlas atkarībā no tā, kādas sarežģītības problēmu neironam ir jāatrisina.

Visbiežāk lietotās aktivizācijas funkcijas ir sliekšņa funkcija (2.02), identitātes funkcija (2.03), loģistiskā sigmoīda funkcija (2.04) un hiperboliskā tangensa funkcija (2.05). [7] Sliekšņa funkcija ir vienkāršākā nelineārā aktivizācijas funkcija. Šo funkciju visbiežāk izmanto Perceptrona neironā.

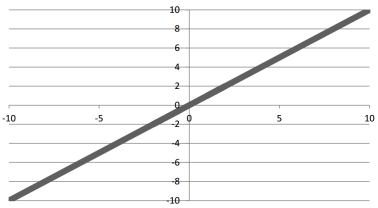
$$\sigma = \begin{cases} 1, & \text{ja} \quad s \geq 0 \\ 0, & \text{ja} \quad s < 0 \end{cases} \tag{2.02}$$

Identitātes funkcija (attēls 2.2) ir triviāla aktivizācijas funkcija. [3] Tā nespēj risināt sarežģītas problēmas, tomēr dod iespēju veikt apstrādi, izmantojot lineārās metodes. Loģistiskā sigmoīda funkcija (attēls 2.3) un hiperboliskā tangensa funkcija (attēls 2.4) ir nelineāra aktivizācijas funkcija, kas veido S-veida grafiku. Šīs aktivizācijas funkcijas ir bioloģiski pamatotas. To priekšrocība ir vienkāršais atvasinājums. [7] Šo funkciju vai tās paveidus visbiežāk izmanto atgriezeniskajā izplatīšanās algoritmā, kas tiks apskatīts vēlāk.

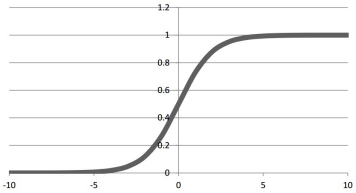
$$\sigma = s \tag{2.03}$$

$$\sigma = \frac{1}{1 + e^{-cs}} \tag{2.04}$$

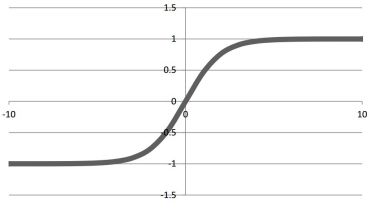
$$\sigma = \frac{e^{s} - e^{-s}}{e^{s} + e^{-s}} \tag{2.05}$$



Att. 2.2: **Identitātes funkcija**



Att. 2.3: **Loģistiskā sigmoīda funkcija**



Att. 2.4: **Hiperboliskā tangensa funkcija**

3 Neironu tīklu arhitektūra (topoloģija)

3.1 Neironu tīklu slāņi

Neironu tīkla arhitektūra ir neironu izvietojums tīklā un to savstarpējās saites, un tā nosaka skaitļošanas spēju. Tīkla arhitektūra tiks aprakstīta ar orientētu grafu.

Definīcija 2. [13] Pāri (V, E) sauc par orientētu grafu G, kur

- $V = \{v_1, v_2, ..., v_n\}$ sauc par galīgu virsotņu kopu un $|V| = n < \infty$;
- $E \subseteq V \times V$ sauc par grafa loku kopu;

$E \ni e = (v, w)$ ir loks grafā G no v uz w.
Kopas V elementus sauc par virsotnēm, bet kopas E elementus sauc par lokiem.

Definīcija 3. [11] Grafs G ir neorientēts grafs $\Leftrightarrow$ ja $\forall(v, w) \in E$ un $\forall(w, v) \in E$.

Definīcija 4. [11] Par grafa G orientētu kontūru sauc tādu kartedžu $(v_1, v_2, ..., v_k) \in V^k$, ka $(v_1, v_2), (v_2, v_3), ..., (v_{k-1}, v_k), (v_k, v_1) \in E$

- Grafu G sauc par grafu bez kontūriem :$\Leftrightarrow$ G nesatur kontūrus;
- $v \in V$ sauc par izteku :$\Leftrightarrow \forall w \in V : (w, v) \notin E$, kas neironu tīklos ir ieeja;
- $v \in V$ sauc par ieteku :$\Leftrightarrow \forall w \in V : (v, w) \notin E$, kas neironu tīklos ir izeja.

Tātad virsotni v sauc par izteku, ja neeksistē neviens loks, kas jebkuru citu virsotni w savienotu ar virsotni v, un virsotni v sauc par ieteku, ja neeksistē neviens loks, kas no šīs virsotnes ved uz jebkuru citu virsotni w. [11]

Definīcija 5. [11] Loku $(v, v) \in E$ sauc par grafa cilpu.

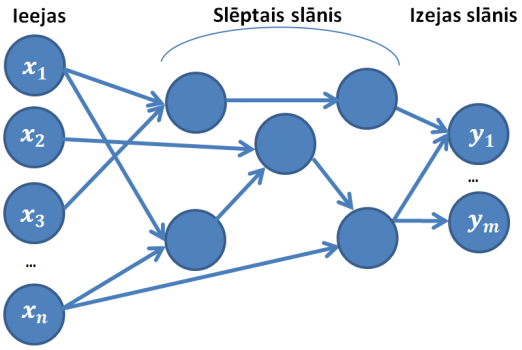
Definīcija 6. [11] $(v_1, v_2), (v_2, v_3), ..., (v_{k-1}, v_k) \in E$ sauc par ceļu L grafam G.
$L = \{(v_1, v_2), (v_2, v_3), ..., (v_{k-1}, v_k)\}$

Tiek izdalīti viena slāņa tīkli un daudzslāņu tīkli, un vienvirziena tīkli, un tīkli ar atgriezeniskajām saitēm.

3.2 Vienvirziena tīkli

Šajā darbā tiks izmantoti tikai tādi grafi, kas ir sakarīgi, bez cilpām, eksistē n grafa iztekas, eksistē m grafa ietekas. Šie tīkli tiks saukti par vienvirziena tīkliem, skatīt attēlu 3.1.

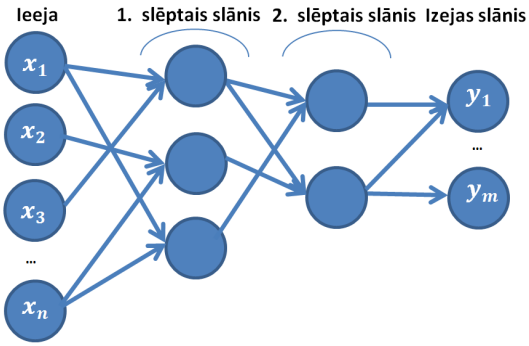
Vienvirziena tīkls ļauj kustēties tikai vienā virzienā - no ieejas uz izeju. Turklāt lielākā daļa



Att. 3.1: **Vinvirziena tīkla uzbūve**

vienvirziena tīklu ir slāņoti, tādā gadījumā jebkura slāņa izeja neietekmēs to pašu slāni. Neironu slānis ir neironu kopums, kas topoloģiski izvietoti kopā un kuru darbināšana un apmācība parasti notiek pēc viena algoritma. Vienvirziena tīkls mēdz būt viena slāņa neironu tīkls, t.i., bez slēptajiem slāņiem. [13]

Definīcija 7. [13] Grafu bez kontūriem sauc par slāņotu $:\Leftrightarrow \exists V_0 \dot{\cup} V_1 \dot{\cup} ... \dot{\cup} V_k = V$ tā, ka $E \subseteq V_0 \times V_1 \cup V_1 \times V_2 \cup ... \cup V_{k-1} \times V_k$ un V_0 satur visas iztekas, un V_k satur visas ietekas.



Att. 3.2: **Trīs slāņu neironu tīkla uzbūve**

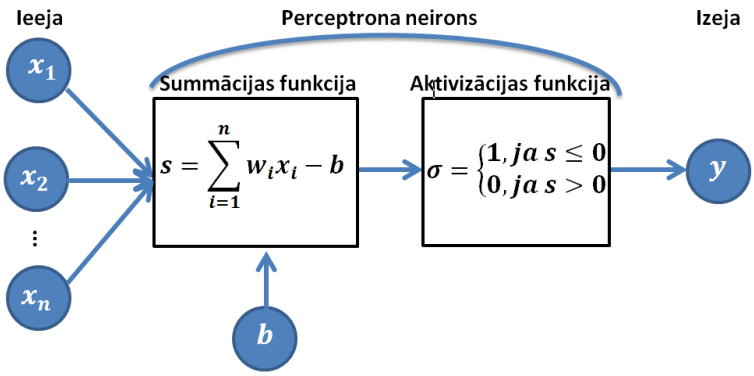
Attēlā 3.2 ir attēlots 3 slāņu neironu tīkls. Ieejas netiek uzskatītas par slāni, līdz ar to slāņu skaits veidojas, saskaitot kopā slēptos slāņus un izejas slāni. Tā arī turpmākajā darbā tiks skaitīti neironu tīklu slāņi.

4 Perceptrons

1943. gadā Varens Makuločš un Valters Pits radīja pirmo mākslīgā neirona modeli. Šis fundamentālais modelis spēj realizēt elementāras loģiskās operācijas – NE (negācija), UN (konjunkcija), VAI (disjunkcija). [13]

Definīcija 8. [3] Ja summācijas funkcija ir 2.01 un aktivizācijas funkcija ir 2.02, tad formālo neironu (X, Y, s, σ) sauc par perceptronu vai Makuloča-Pita neironu.

Perceptrons (attēls 4.1) ir pamata apstrādes elements. Tam ir ieejas, kas var nākt no vides, vai arī



Att. 4.1: **Perceptrona neirona uzbūve**

var būt izejas no citiem perceptroniem, kas nozīmē, ka, ja neironu tīklā ir divi perceptroni, kas savienoti pēc kārtas, tad otrā perceptrona ieeja ir pirmā perceptrona izeja. Savienojošais svars ir saistīts ar katru ieeju, un izeja vienkāršākajā gadījumā ir ieejas svaru summa. [7]

Neirona ieejām tiek padotas vērtības, kas kopā ar attiecīgajiem svariem tiek padotas summācijas funkcijai, rezultāts tiek sūtīts uz aktivizācijas funkciju, kas šajā gadījumā ir sliekšņa funkcija. Aktivizācijas funkcija dod vērtības 0 vai 1. [13]

Konjukcija, disjunkcija, negācija

Algebrisku n-vietīgu operāciju $a : \{0,1\}^n \to \{0,1\}$ sauc par n-vietīgu loģisku operāciju. [5] Vienkāršākās loģiskās operācijas var tikt īstenotas ar Makuloča-Pita neironu. Izejas vērtība 1 ir kā loģiskā vērtība *patiess*, un 0 ir kā *aplams*.

Apskatīsim trīs piemērus UN, VAI un NE, kas konstruēti ar perceptronu. UN un VAI ir divas ieejas, līdz ar to ir divi svari w_1 un w_2, bet NE ir viena ieeja, līdz ar to ir tikai viens svars

w. Summēšanas funkcija ir 2.01, un aktivizācijas funkcija ir 2.02.

Piemērs - UN

Skatīt tabulu 4.1. Ir dotas četru punktu koordinātas (x_1, x_2) un šajā gadījumā izvēlētie svari ir $w^T = (1, 1)$ un slieksnis ir $b = 1.5$, attiecīgi $y = \sigma(s)$, kur $s = x_1 + x_2 - 1.5$.

Piemērs - VAI

Skatīt tabulu 4.2. Tāpat kā iepriekšējā piemērā arī VAI ir dotas četru punktu koordinātas (x_1, x_2) un šajā gadījumā izvēlētie svari ir $w^T = (1, 1)$ un slieksnis ir $b = 0.5$, attiecīgi $y = \sigma(x_1 + x_2 - 0.5)$.

Piemērs - NE

Skatīt tabulu 4.3. Punkts atrodas uz x taisnes, izvēlas svarus $w^T = -1$ un slieksni $b = -0.5$ līdz ar to $y = \sigma(-x + 0.5)$.

Tabula 4.1: **UN**

x_1	x_2	s	$\sigma(s)$
0	0	-1.5	0
0	1	-0.5	0
1	0	-0.5	0
1	1	0.5	1

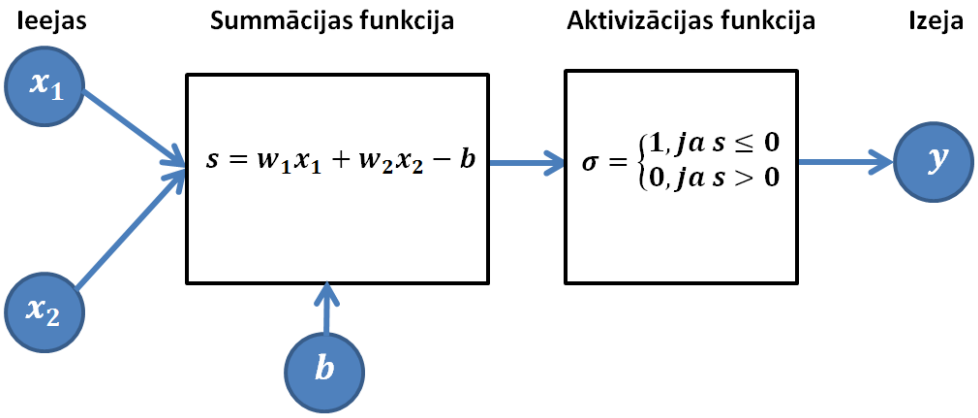
Tabula 4.2: **VAI**

x_1	x_2	s	$\sigma(s)$
0	0	-0.5	0
0	1	0.5	1
1	0	0.5	1
1	1	1.5	1

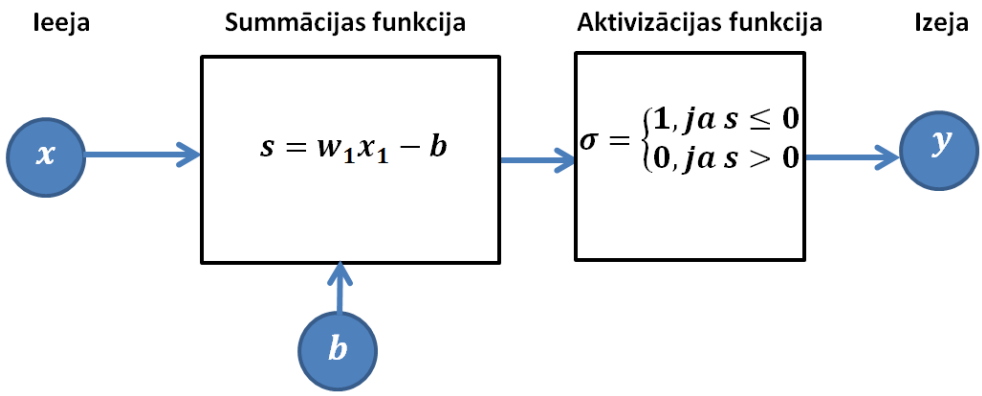
Tabula 4.3: **NE**

x	s	$\sigma(s)$
0	0.5	1
1	-0.5	0

Tālāk tiek parādīts, kā izskatīsies neironu tīkls atbilstošajiem piemēriem UN (attēls 4.2), VAI (attēls 4.2) un NE (attēls 4.3). UN un VAI neironu tīkls ir viens un tas pats, jo to ieeju skaits sakrīt, kā arī izeju skaits sakrīt, bet tiem atšķiras ieejas vērtības, līdz ar to arī rezultāts.



Att. 4.2: **Neironu tīkla uzbūve piemēriem UN un VAI**



Att. 4.3: **Neironu tīkla uzbūve piemēram NE**

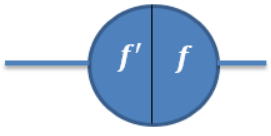
5 Atgriezeniskais izplatīšanās algoritms

Piepūle, kas nepieciešama, lai atrastu pareizās svaru kombinācijas, būtiski palielinās, kad tiek apskatīti neironu tīkli ar daudz sarežģītāku topoloģiju un vairākiem mainīgajiem. Viens no algoritmiem, kas var atrisināt šādas apmācības problēmas, ir atgriezeniskais izplatīšanās algoritms. Atgriezenisko izplatīšanās metodi neironu tīklos lieto daudzslāņu vienvirziena tīklos. Pašlaik tas visbiežāk tiek izmantots kā apmācības algoritms neironu tīkla lietojumprogrammās. 1947. gadā Pauls Verbos (*Paul Werbos*) radīja atgriezenisko izplatīšanās algoritmu. [10]

5.1 Atgriezeniskās izplatīšanās modelis

Šajā apakšnodaļā mērķis ir aprakstīt algoritmu, kas spētu apmācīt daudzslāņu neironu tīklu. Visplašāk izmantotais ir atgriezeniskais izplatīšanās algoritms divslāņu neironu tīklam. Ņemot vērā, ka nepieciešamība pēc diviem vai vairāk slēptajiem slāņiem ir ļoti reta, tiks izvēlēts tieši viens slēptais slānis, un turpmākajā darbā tas arī tiks aprakstīts. [3] Tālāk aprakstītais līdz 5.2 apkšnodaļai ir balstīts uz avotu [10].

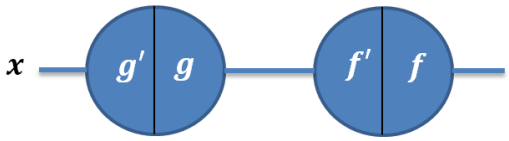
Jau pēc algoritma nosaukuma ir saprotams, ka izmantojamā apmācīšanās metode būs atgriezeniska. Tātad algoritms sastāv no divām daļām. Pirmajā tas virzās uz priekšu, šajā laikā katrā virsotnē tiek aprēķināta funkcija, kas tiek saglabāta virsotnes labajā pusē un funkcijas atvasinājums tiek saglabāts virsotnes kreisajā pusē (attēls 5.1), bet tikai labajā pusē saglabātā informācija tiek virzīta tālāk pa tīklu. Otrā daļa ir atgriezeniskā izplatīšanās pa tīklu, kur tiek lietota saglabātā informācija pa kreisi.



Att. 5.1: **Neirona virsotne sadalīta divās daļās**

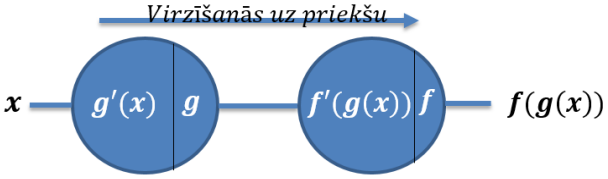
Funkciju kompozīcija

Attēlā 5.2 ir dotas divas virsotnes. Ieeja ir x, ko izmanto, lai aprēķinātu funkciju pirmajā virsotnē un tad tās atvasinājumu. Šeit tiek aprēķinātas divas funkcijas g un f.



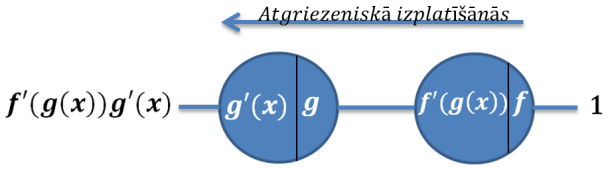
Att. 5.2: **Neironu tīkls ar divu funkciju kompozīciju**

Ņemot vērā, ka otrajai virsotnei ieeja ir aprēķinātā funkcija $g(x)$, tad otrajai virsotnei atvasinātā funkcija ir $f'(g(x))$ (attēls 5.3). Un izeja no šīm virsotnēm ir $f(g(x))$.



Att. 5.3: **Vienvirziena tīkla rezultāts**

Atgriezeniskajā izplatīšanās daļā tīklā tiek ievadīta konstante 1. Ieeja tiek reizināta ar iepriekš saglabāto informāciju kreisajā pusē. Tālāk iegūtais rezultāts tiek padots uz virsotni, kas ir pa kreisi. Ejot atgriezeniski līdz pirmajai virsotnei, iegūst rezultātu $f'(g(x))g'(x)$, (attēls 5.4). Atgriezeniskajā izplatīšanās gaitā ir iegūta diferenciāļa formas invariance.

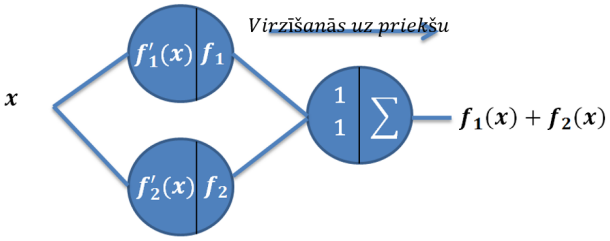


Att. 5.4: **Atgriezeniskās izplatīšanās rezultāts**

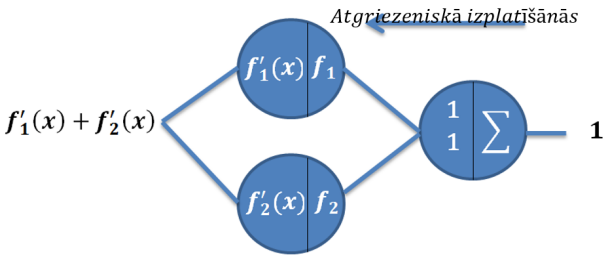
Funkciju saskaitīšana

Nākamais gadījums, kas ir jāapskata, ja ir divas virsotnes, kas nav savienotas savā starpā, (attēls 5.5). Ir dotas divas funkcijas f_1 un f_2. Funkcijas f_1 un f_2 tiek saglabātas labajā pusē, bet to atvasinājumi - kreisajā pusē. Tiek pievienota vēl viena virsotne, kas ļauj saskaitīt divas funkcijas. Parciālais atvasinājums, kas ir pierakstīts kreisajā pusē, ir 1, jo, atvasinot funkciju pēc ieejas, tiek iegūts vienīnieks. Iegūtais rezultāts ir $f_1(x) + f_2(x)$.

Atgriezeniskajā izplatīšanās daļā tīklā tiek ievadīta konstante 1. Ieeja iet uz virsotni pa kreisi, kur šī virsotne sadalās divās daļās (attēls 5.6). Līdz ar to iepriekš saglabātie atvasinājumi funkcijām tiek reizināti ar viens, un tās tiek saskaitītas. Tiek iegūts rezultāts $f_1'(x) + f_2'(x)$, kas ir atvasinājums funkciju summai $f_1 + f_2$.



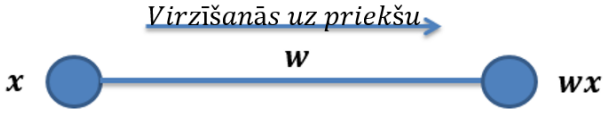
Att. 5.5: **Neironu tīkla funkciju saskaitīšana**



Att. 5.6: **Neironu tīkla atgriezeniskās izplatīšanās rezultāts**

Svari

Tagad pieņem, ka katram lokam ir savs svars w. Vienvirziena tīklā svari w tiek reizināti ar ieeju x, līdz ar to rezultāts ir wx (attēls 5.7).



Att. 5.7: **Vienvirziena tīkla daļa**

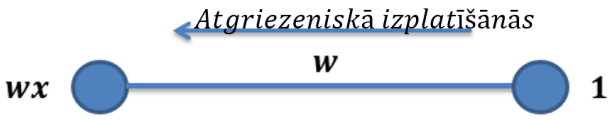
Atgriezeniskajā daļā ieeja ir 1, kas tiek reizināta ar svariem, rezultāts ir w, kas ir atvasinājums wx pēc x (attēls 5.8). Līdz ar to var secināt, ka svari tiek izmantoti vienādi abos virzienos.

5.2 Mācīšanās ar skolotāju

Vissvarīgākā neironu tīklu īpašība ir spēja mācīties no vides un mācīties uzlabot sniegumu. Vide ir informācija, kas ir zināma pirms tiek uzsākta apmācība, t.i., ir zināmas ieejas, zināmas vēlamās atbildes, algoritms, ar kuru tas tiks apmācīts, kā arī var būt zināma cita informācija. Neironu tīkls mācās par tā vidi iteratīva procesa laikā, kad svari un slieksnis tiek uzlaboti. Ar procesu tiek saprasts laika sprīdis no brīža, kad neironiem tiek iedotas ieejas vērtības līdz brīdim, kad tie izdod izejas vērtības. Ideālā variantā tīkls uzzina arvien vairāk par tā vidi pēc katras mācīšanās procesa iterācijas.

Definīcija 9. [12] Apmācība ir neironu tīkla svaru vērtību (un dažreiz arī citu parametru) uzstādīšana, balstoties uz apmācības paraugiem, kas reprezentē noteiktu problēmu.

Definīcija 10. [7] Mācīšanās ir process, ar kuru neirona tīkla brīvie parametri tiek pielāgoti, izmantojot nepārtrauktu simulāciju procesu vidē, kurā tīkls ir nostiprināts.



Att. 5.8: **Atgriezeniskās izplatīšanās daļa**

Ar brīvajiem parametriem tiek saprasti mainīgie lielumi, kas neironu tīklā ir svari un slieksnis. Šī mācīšanās definīcija paredz šādus notikumus [12]:

1. Neirona tīkls tiek simulēts vidē;

2. Neirona tīkls veic izmaiņas;

3. Pārmaiņu dēļ, kas notikušas iekšējā struktūrā, neirona tīkls reaģē jaunā veidā uz vidi.

Tālāk aprakstītais ir balstīts uz avotu [3]. Vienvirziena tīklā visas virsotnes, kas ir slēptajā slānī un izejas slānī, ir neironi, kas aprēķina izejas vērtības, un šis tīkls padod informāciju no virsotnes uz virsotni. Katrs neirons ir spējīgs novērtēt funkciju atkarībā no tā ieejas. Neironu tīkls reprezentē funkciju kompozīciju ķēdi, kas pārveido ieejas vektoru par izejas vektoru, un tas šo funkciju kompozīciju īsteno no ieejas uz izeju. Šī funkcija tālāk tiks uzskatīta par tīkla funkciju. Kopumā ir jāatrod optimālā svaru kombinācija tā, lai tīkla funkcija η tuvojas dotajai funkcijai f, cik tuvu tas ir iespējams. Funkcija f netiek precīzi dota. Iepriekš apskatītajā piemērā (4. nodaļā) par loģisko operāciju UN ir zināmas ieejas un ir zināms vēlamais iznākums, kas attiecīgi ir funkcija f, pati funkcija precīzi nav zināma. Līdz ar to mēs vēlamies atrast funkciju η, kas pēc iespējas tuvāk tiektos uz vēlamo rezultātu.

Pieņem, ka dots ir vienvirziena tīkls ar n ieejām un p izejām. Tīkls var sastāvēt no jebkura daudzuma slēptajām virsotnēm. Ir dots mācīšanās paraugs

$$\{(x_1, d_1), \ldots, (x_m, d_m)\},$$

kas sastāv no p sakārtotiem pāriem, katrs $x_1, ..., x_m$ (ieejas vektori) ir n-dimensionāls vektors un katrs $d_1, ..., d_m$ (vēlamās atbildes vektori) ir p-dimensionāls vektors.
Piemēram, loģiskajā operācija UN mācīšanās paraugs izskatīsies šādi:

$$\{((0,0),0),((0,1),0),((1,0),0),((1,1),1)\}.$$

Pieņem, ka aktivizācijas funkcija katrā neironu tīkla virsotnē ir nepārtraukta un diferencējama. Visi svari tiek izvēlēti patvaļīgi. Kad neironu tīklā tiek ievadīta $x_i, i = 1, ..., n$, tad tīkls izvada vērtību $z_i, i = 1, ..., p$, kas parasti ir atšķirīga no vēlamās atbildes $d_i, i = 1, ..., p$. Tas, ko mēs vēlamies, ir, lai z_i un d_i ir identiski visiem $i = 1, \ldots, p$. Precīzāk, izmantojot mācīšanās algoritmu, mēs vēlamies minimizēt neironu tīkla kļūdas funkciju, kas ir definēta:

$$E = \frac{1}{2}\sum_{i=1}^{p} ||z_i – d_i||^2. \tag{5.21}$$

Kļūdas (5.21) minimizācija notiek, mainot svarus, kas nozīmē, ka ir jāatrod veids, kā šos svarus mainīt. Kad sākas atgriezeniskā izplatīšanās daļa, svaru izmaiņas tiek aprēķinātas pēc formulas:

$$w_i^{new} = w_i^{old} - \varphi\frac{\partial E}{\partial w_i}, \forall i = 1, ..., d, \tag{5.22}$$

kur $0 \leq \varphi \leq 1$ ir mācīšanās koeficients, t.i., proporcionāls parametrs, kas definē soļa garumu katrai iterācijai negatīvā gradienta virzienā.

Kļūda (5.21) tiek atgriezeniski izplatīta, un svari tiek uzlaboti atgriezeniskajā izplatīšanās gaitā katrā slānī. Šīs darbības tiek atkārtotas tik ilgi, kamēr kļūda kļūst mazāka par noteikto kļūdas līmeni ε, t.i., $E \leq \varepsilon$, vai arī apstāšanās notiek pie cita noteikta apstākļa, piemēram, iterāciju skaita. Šajā brīdī var uzskatīt, ka neironu tīkls ir apmācīts.

Gradienta metode

Apmācīšana ar atgriezenisko izplatīšanās algoritmu balstās uz gradienta metodi, kuru pielieto kļūdas funkcijai. Par skalārā lauka u gradientu fiksētā punktā M_0 sauc vektoru, kas nosaka virzienu, kādā skalārā lieluma vērtība palielinās visātrāk. Gradienta koordinātas aprēķina, izmantojot funkcijas u parciālos atvasinājumus punktā M_0. [7] Ja $u = u(x, y, z)$, tad u gradients ir:

$$\nabla u = (\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, ..., \frac{\partial u}{\partial z}).$$

Atgriezeniskā izplatīšanās algoritmā mēs vēlamies minimizēt kļūdu E, tādēļ tā tiek ņemta par skalārā lauka funkciju, un par argumentiem tiek ņemti neironu tīkla svari.

$$\nabla E = (\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, ..., \frac{\partial E}{\partial w_d}).$$

Tā kā kļūda tiek minimizēta, gradients ir ar mīnusa zīmi. [13]

5.3 Algoritma soļu apraksts

Atgriezeniskais izplatīšanās algoritms meklē kļūdas funkcijas minimumu. Svaru kombinācijas, kas minimizē kļūdas funkciju, tiek uzskatītas par atrisinājumu apmācības problēmai. Tā kā šī metode pieprasa aprēķināt kļūdas funkciju katrā iterācijā, ir jāgarantē nepārtrauktība

un diferencējamība kļūdas funkcijai. Skaidrs, ka ir arī nepieciešama cita aktivizācijas funkcija nekā perceptronā, jo perceptrona funkcija ir pārtraukta un līdz ar to arī kļūdas funkcija tāda ir. [3]

Pirms tiek aprakstīti soļi, ir jāsaprot, kādu aktivizācijas funkciju izvēlēties. Ir nepieciešama nelineāra funkcija, kas ir diferencējama visā tās definīcijas apgabalā, jo ir jāmeklē gradients kļūdas funkcijai katrā iterācijā. [3] Šajā darbā tiks izmantota loģistiskā sigmoīda un hiperboliskā tangensa funkcijas $\sigma_c : \mathbb{R} \to (0, 1)$ (attiecīgi 2.04 un 2.05) kā aktivizācijas funkcijas. Parametrs c nosaka funkcijas stāvumu. Abām šīm funkcijām priekšrocība ir to vienkāršais atvasinājums. Tālāk ir iegūti abu funkciju atvasinājumi - loģiskās sigmoīda funkcijas atvasinājums 5.31 un hiperboliskā tangensa atvasinājums 5.32.

$$\sigma_c'(x) = \frac{ce^{-cx}}{(1+e^{-cx})^2} = \frac{1+ce^{-cx}-1}{(1+ce^{-cx})^2} = \frac{1+c^{-cx}}{(1+ce^{-cx})^2} - \frac{1}{1+ce^{-cx})^2} =$$
$$= \frac{1}{1+ce^{-cx}} - \frac{1}{(1+ce^{-cx})^2} = c\sigma_c(x)(1-\sigma_c(x)).$$

Lai vienkāršotu, fiksē $c = 1$ un $\sigma_1(x) = \sigma(x)$. Līdz ar to iegūst formulu:

$$\sigma'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = \sigma(x)(1-\sigma(x)). \tag{5.31}$$

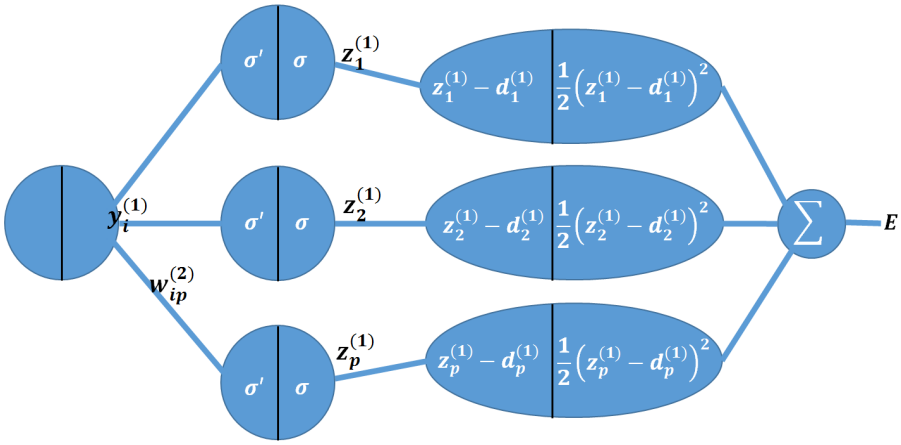
$$\sigma_c'(x) = tanh(cx)' = (\frac{sinh(cx)}{cosh(cx)})' = \frac{cosh(cx)(sinh(cx)-sin(cx))'}{cosh^2(cx)} =$$
$$= \frac{(cosh(cx)cosh(cx)-sinh(cx)sinh(cx))}{cosh^2(cx)} = c(1-tanh^2(cx)) =$$
$$= c(1-\sigma(cx)^2).$$

Arī šeit, lai vienkāršotu, fiksē $c = 1$ un $\sigma_1(x) = \sigma(x)$. Līdz ar to iegūst formulu:

$$\sigma'(x) = (1-\sigma(x)^2). \tag{5.32}$$

Vēl jāapraksta, kā tīklā tiek iegūta kļūdas funkcija (attēls 5.9), jo rezultāti būs nepieciešami, lai aprakstītu soļus. Kļūdas funkcijas iegūšana un soļu apraksts tiek balstīts uz avotu [10]. Lai būtu vieglāk aprakstīt, pieņemsim, ka ir dots tikai viens ieejas vektors, bet pēc tam tiks vispārināts tā, lai ir m ieejas vektori. Tiek iegūtas vērtības $y_i^{(1)}$ un $z_j^{(1)}$. Tiek pievienots papildus viens slānis, kas no neirona tīkla izejām atņem attiecīgo vēlamo vērtību un šo starpību kāpina

kvadrātā un, saskaitot šo starpību kvadrātus, iegūst kļūdu E. Lai tīkls tiktu vispārināts m ieeju vektoriem, vienkārši ir nepieciešami m šādi tīkli.



Att. 5.9: **Kļūdas E (5.21) aprēķināšana neironu tīklā**

Pieņemam, ka ir dotas n ieejas, l ir neironu skaits slēptajā slānī un p ir neironu skaits izejas slānī. Tiek doti ieejas vektori $x^{(i)} \in \mathbb{R}^n, i = 1, ..., N$ (N apzīmē ieejas vektoru skaitu) un attiecīgās vēlamās atbildes $d^{(i)} \in \{0, 1\}^p$

Soļi tiks aprakstīti tikai ar loģistisko sigmoīda aktviziācijas funkciju, jo ar hiperboliskā tangensa aktivizācijas funkciju soļu apraksts ir līdzīgs, un līdz ar to nerodas nepieciešamība to aprakstīt.

Nultais solis

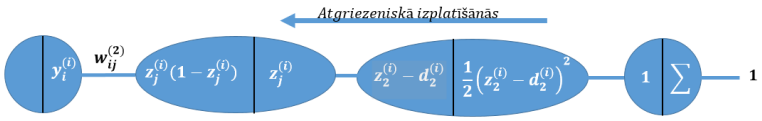
Izvēlas aktivizācijas funkciju (2.04) un sumācijas funkciju (2.01). Tiek fiksēts ieejas vektors $x^{(i)}$ un attiecīgās vēlamās atbildes $d^{(i)}$, mācīšanās ātrums φ, kritērijs ε, pie kura algoritms apstājas. Tiek fiksēts $i = 1$. Tiek patvaļīgi izvēlētas sākotnējās svaru vērtības.

Pirmais solis - vienvirziena tīkls

Neironu tīklā tiek ievadītas ieejas vektors $x^{(i)}$. Tiek aprēķinātas vērtības $y^{(i)} \in \mathbb{R}^l$ un $z^{(i)} \in \mathbb{R}^p$, un tās tiek saglabātas virsotnes labajā pusē. Arī aktivizācijas funkcijas atvasinājums (5.31) tiek saglabāts katras virsotnes kreisajā pusē.

Otrais solis - atgriezeniskā izplatīšanās uz izejas slāni

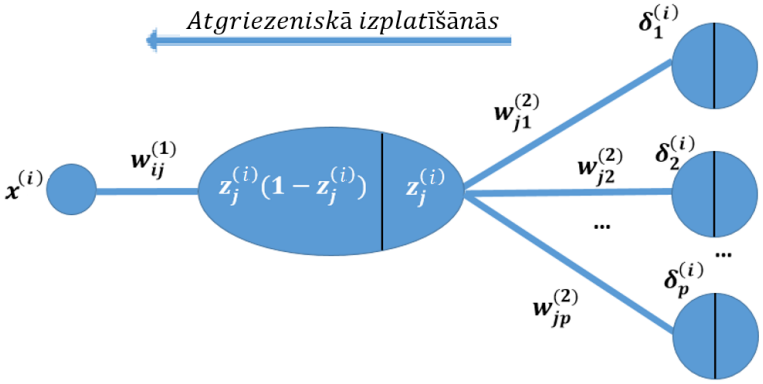
Tiek meklēts parciālais atvasinājums $\frac{\partial E}{\partial w_{ij}^{(2)}}$, (attēls 5.10). Kā jau iepriekš tika aprakstīts, tad tagad ieeja ir 1, kas tiek reizināta ar iepriekš saglabātajām vērtībām, rezultātā $\delta_j^{(i)} = z_j^{(i)}(1 - z_j^{(i)})(z_j^{(i)} - d_j^{(i)}), j = 1, ..., p.$



Att. 5.10: **Atgriezniskā izplatīšanās uz izejas slāni**

Trešais solis - atgriezeniskā izplatīšanās uz slēpto slāni

Tiek meklēts parciālais atvasinājums $\frac{\partial E}{\partial w_{ij}^{(1)}}$. Katra virsotne j slēptajā slānī ir savienota ar katru virsotni q izejas slānī, katram savienojumam ir savs svars $w_{jq}^{(2)}$, kur $q = 1, ..., p$. Atgriezeniskā izplatīšanās kļūda līdz virsotnei j slēptajā slānī ir jāaprēķina, ņemot vērā vis-



Att. 5.11: **Atgriezeniskā izplatīšanās uz slēpto slāni**

us iespējamos atgriezeniskos ceļus, kā parādīts attēlā5.11. Līdz ar to atgriezeniskā kļūda ir $\Delta_j^{(i)} = z_j^{(i)}(1 - z_j^{(i)}) \sum_{q=1}^{p} w_{jq}^{(2)} \delta_q^{(1)}, j = 1, ..., l.$

Palielinām i par viens. Ja $i > N$ dodas uz ceturto soli, citādi dodas uz pirmo soli.

Ceturtais solis

Izeja no neironu tīkla $z^{(i)}$ tiek salīdzināta ar vēlamo atbildi $d^{(i)}$, un kļūdas kopsumma E tiek aprēķināta pēc formulas 5.21. Ja $E < \varepsilon$, tad algoritms apstājas. Citādi uzlabo svarus pēc formulas 5.22.

Fiksē $i = 1$ un izvēlas $\varphi_{t+1} > 0$. Ne vienmēr φ_{t+1} tiek mainīts. Turpmākajā darbā φ tiks atstāts nemainīgs visa algoritma darbības laikā.

Kā izvēlēties parametrus

Šeit rodas jautājums, kā noteikt, cik neironu ir katrā slānī. Kā jau iepriekš tika teikts, tad visbiežāk lietotais tīkls sastāv no ieejām, viena slēptā slāņa un izejas slāņa. Skaidrs, ka ieeju ir tik, cik ir koordinātu ieejas vektorā. Izejas slānī neironus izvēlas tik, cik ir dotas vēlamās atbildes. Piemēram, lai atrisinātu UN ar neironu tīklu, ir nepieciešamas trīs ieejas, jo ieejas vektorā ir divas koordinātas $x_1^{(i)}$, $x_2^{(i)}$ un virsotne, kas vienmēr pieņems vērtību viens, un viens

izejas neirons, jo ieejām ir viena vēlamā atbilde 0 vai 1. Līdz ar to būtībā ir tikai jāapdomā, cik neironu būs slēptajā slānī. Nav tādas formulas vai kādu noteikumu, kā izvēlēties, cik ir jābūt neironiem slēptajā slānī. Līdz ar to vienīgā iespēja, kā noteikt to, cik neironu ir slēptajā slānī, ir eksperimentējot. Līdz ar to tika apskatīti trīs dažādi piemēri UN, VAI un XOR, lai varētu noteikt, kā apmācīšanās procesu ietekmē parametru maiņa. Šis pētījums tika veikts Zinātniskās prakses ietvaros.

Tika izveidots neironu tīkls, kurš tiek apmācīts ar atgriezenisko izplatīšanās algoritmu *Matlab* vidē. Kā ievades datus lieto $x^{(1)} = (1, 0)$, $x^{(2)} = (0, 1)$, $x^{(3)} = (0, 0)$, $x^{(4)} = (1, 1)$ un kā vēlamās izejas lieto vērtības no tabulas 5.1.

Tabula 5.1: **Ieejas vektori un attiecīgās vēlamās atbildes piemēriem XOR, UN un VAI**

$x^{(i)}$	(10)	(01)	(00)	(11)
XOR	1	1	0	0
UN	0	0	0	1
VAI	1	1	0	1

Tika izmantoti trīs dažādi mācīšanās ātrumi $\varphi = 0.1$, $\varphi = 0.5$ un $\varphi = 0.7$. Tika apskatīts, kā slēpto neironu skaits slēptajā slānī ietekmē kļūdu un iterāciju skaitu, kā arī tika mainīts mācīšanās ātrums un pētīta tā ietekme uz tīkla apmācību.

Tika arī noskaidrots, ka sākumā izvēlētajiem svariem ir liela nozīme, jo, izvēloties pārlieku mazus sākuma svarus, t.i., mazākus par 0.001 vai lielākus par -0.001, gandrīz neviens no veiktajiem eksperimentiem nekonverģēja. Izvēloties sākuma svarus mazākus par 0.5, bet lielākus par 0.01 un lielākus par -0.5, bet mazākus par -0.01, tīkls, ja pareizi tika izvēlēts mācīšanās ātrums un slēpto neironu skaits slēptajā slānī, konverģēja gandrīz vienmēr.

Tika arī pārbaudīts, vai nepieciešamo iterāciju daudzums, lai tīkls apmācītos, samazinās, ja tiek palielināts slēpto neironu skaits slēptajā slānī. Tas apstiprinājās daļēji, jo kopumā bija novērojams, ka iterāciju skaits samazinās, bet dažbrīd tas tomēr arī palielinājās. Tas arī parāda, ka ir liela nozīme izvēlētajam neironu skaitam slēptajā slānī. Izvēloties pārlieku lielu skaitu slēpto neironu, tīkls nekonverģēs.

Plašāka šo piemēru analīze veikta zinātniskās prakses izstrādātajā atskaitē.

6 Modeļu izvēle valūtu kursu prognozēšnā

Turpmākais apraksts ir balstīts uz avotu [8]. Valūtu tirgus ir viens no vissarežģītākajiem un dinamiskākajiem tirgiem ar nelineāru svārstīgumu un neregularitāti. Lielu ietekmi uz šiem tirgiem atstāj ekonomiskā izaugsme, tirdzniecības attīstība, akciju cenas un inflācija, līdz ar to ir ļoti grūti prognozēt valūtas kursa izmaiņas.

Kopš Lapedes un Farbers (1987) pirmo reizi ierosināja izmantot daudzslāņu vienvirziena neironu tīklu nelineārā signāla prognozei, daudzi pētījumi, izmantojot mākslīgos neironu tīklus, ir pierādījuši to izmantojamību nelineāru laikrindu prognozēšanai. Tomēr neviens paņēmiens nav bijis pietiekami veiksmīgs, lai pārspētu iepriekšējās metodes un lai prognozētu valūtas tirgū jebkurā situācijā. Tāpēc ir grūti teikt, vai mākslīgie neironu tīkli darbojas labāk nekā citas metodes. Dažas publikācijas liecina, ka mākslīgo neironu tīklu veikumi ir labāki valūtu kursa prognozēšanā, bet citas dod negatīvus secinājumus. Tomēr no publikācijām var spriest, ka nedēļas datiem, mākslīgie neironu tīkli daudz labāk spēj prognozēt valūtu kursus nekā patvaļīgās klejošanas modeļi.

No 45 rakstiem 60%, kas ir publicēti par valūtu kursu prognozēšanu, izmantojot mākslīgo neironu tīklus, norāda, ka mākslīgie neironu tīkli spēj labāk prognozēt valūtu kursus, kamēr tikai $4,4\%$ norāda, ka mākslīgie neironu tīkli spēj prognozēt sliktāk par citām metodēm, un 35.5% norāda, ka dažreiz mākslīgie neironu tīkli, dažreiz citas metodes darbojas labāk. Grāmatā [8] var iepazīties vairāk ar šo 45 darbu analīzi.

Ir izpētīts, ka valūtu kursu prognozēšana ar mākslīgajiem neironu tīkliem, uzrāda vislabākos rezultātus, ja prognozē vienu, trīs vai piecus soļus uz priekšu, bet, ja jāprognozē desmit vai trīsdesmit dienas uz priekšu, tad patvaļīgās kustības modelis uzrāda labākus rezultātus. Kopumā, prognozējot īsam vai vidējam laika posmam, mākslīgie neironu tīkli dod vislabākos rezultātus.

Lai uzzinātu, kādus parametrus vislabāk izvēlēties, lai apmācītu tīklu prognozēt valūtu kursus, tika apskatīti dažādi avoti, lai uzzinātu, vai ir noteikti kādi optimālie parametri.

Apmācības algoritms

Apskatītajās publikācijās visbiežāk lietotais ir atgriezeniskais izplatīšanās algoritms, to izmantoja autori [14], [2] savās publikācijās par valūtu kursu prognozēšanu. Autori [1] testēja trīs dažādus algoritmus - atgriezenisko izplatīšanās, elastīgo izplatīšanās un RPROP, kur visi algoritmi uzrādīja līdzīgus rezultātus. Par elastīgo izplatīšanās algoritmu un RPROP algoritmu plašāk var uzzināt [1] . Arī autora [8] grāmatā ir aprakstīts, ka šobrīd visvairāk lietotais algoritms ir atgriezeniskais izplatīšanās algoritms, kurš balstīts uz gradienta metodi, kļūdas minimizēšanu. Tomēr atgriezeniskajam izplatīšanās algoritmam ir nosliece atrast lokālo minimumu, un, lai uzlabotu šo defektu, piedāvā to labot ar ģenētikas algoritmu izmantošanu [8], bet diemžēl ģenētikas algoritmi ir laikietilpīgi un tiem ir vāja konverģence.

Aktivizācijas funkcija

Visbiežāk lietotā aktivizācijas funkcija apskatītajā literatūrā ir sigmoīda tipa funkcijas [14], piemēram, loģistiskā (2.04) vai hiperboliskā tangensa (2.05) funkcijas. Tomēr var arī izmantot citas funkcijas, ja tās ir diferencējamas. Autori [1] izmantoja hiperbolisko tangensa aktivizācijas funkciju slēptajos neironos slēptajā slānī, un izejā tika izmantota identitātes (2.03) aktivizācijas funkcija. Autors [2] lieto loģistisko sigmoīda funkciju gan slēptajā slānī, gan izejas slānī, tomēr daudzas prognozes bija krietni zemākas nekā patiesās vērtības. Kā apraksta [8], parasti aktivizācijas funkcija tiek izmantota visos slāņos. Tomēr ir izdevīgi izmantot sigmoīda tipa aktivizācijas funkciju slēptajos slāņos un lineāru aktivizācijas funkciju izejas slānī. Autora [8] apskatītajās publikācijās, 34 raksti $(75,56\%)$ izmanto sigmoīda tipa aktivizācijas funkciju (loģistisko vai hiperbolisko tangensu) slēptajā slānī, bet izejas slānī identitātes funkciju. Ņemot vērā, ka valūtas tirgos ir augsta trokšņa aktivitāte, liels svārstīgums un sarežģītība, ir ieteicams izmantot sigmoīda tipa aktivizācijas funkciju, iesaka autors [8].

Mācīšanās koeficients

Gandrīz visos pētījumos par valūtas kursa prognozēšanu ar mākslīgajiem neironu tīkliem nav izmantota iespēja mainīt mācīšanās ātrumu pēc katrām svaru izmaiņām. Šajos pētījumos [8], [1] mācību ātrums tiek fiksēts visā apmācības laikā. Kā jau tika teikts, svarīgi ir noteikt pareizu mācīšanās ātrumu mākslīgajam neironu tīklam. Ja mācīšanās ātrums ir pārāk liels, apmācība var notikt ātri, bet mākslīgais neironu tīkls var arī kļūt nestabils un var vispār neapmācīties. Ja mācīšanās ātrums ir pārāk mazs, tas var novest pie ilga mācīšanās laika un lēnu konverģenci.

Mākslīgo neironu tīkla uzbūve

Apskatītajās publikācijās visbiežāk tiek izmantots divslāņu neironu tīkla modelis, kurš tiek apmācīts ar atgriezenisko izplatīšanās algoritmu, ar identitātes aktivizācijas funkciju izejas slānī un hiperbiliskā tangensa aktivizācijas funkciju slēptā slāņa neironiem.

Autors [8] uzskata, ka gan teorētiskais pierādījums, gan empīriskā izmantošana pierāda, ka divslāņu atgriezeniskais izplatīšanās neironu tīkla modelis ar identitātes aktivizācijas funkciju izejas slānī un hiperboliskā tangensa aktivizācijas funkciju slēptā slāņa neironiem, ir atbilstošs valūtas kursu prognozēšanai. Arī [1],[2] savos eksperimentos izmanto divslāņu mākslīgo neironu tīklu. Autors [14] ir vienīgais no apskatītajiem, kas izmanto trīsslāņu mākslīgo neironu tīklu, bet tā kā savā publikācijā viņš rezultātus nesalīdzina ar divslāņu mākslīgo neironu tīklu, nav iespējams spriest, vai iegūtie rezultāti ir labāki.

Šobrīd paliek atklāts jautājums, kā izvēlēties slēpto neironu skaitu slēptajā slānī. Autors [1] savos eksperimentos izmantoja 40 slēptos neironus slēptajā slānī, tomēr autors nemin, kādēļ tika izvēlēts tieši šāds skaits slēpto neironu, kā arī nav izmantots cits skaits slēpto neironu. Autors [8], izanalizējot visas 45 publikācijas, piedāvā izvēlēties 21 slēptos neironus slēptajā slānī. Tā kā dažādi autori piedāvā dažādu skaitu slēpto neironu slēptajā slānī, neironu skaits slēptajā slānī ir jāizvēlas eksperimentējot.

Autora [8] apskatītajās 30 publikācijā no 45 tiek izmantots vairākslāņu uz priekšu virzīts tīkls, un 15 rakstos no 27 rakstiem, kas uzrādīja, ka mākslīgo neironu tīkls spēj vislabāk prognozēt valūtu kursus, tika izmantots viens slēptais slānis mākslīgajam neironu tīklam. Autors [8] parāda, ka vairākslāņu vienvirziena neironu tīkls ar vienu slēpto slāni, ir pietiekami labs, lai prognozētu valūtu kursus. Tātad divslāņu mākslīgais neironu tīkls ir pietiekami adekvāts, lai prognozētu valūtu kursus.

Dati

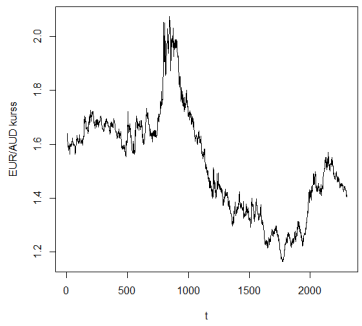
Par ievades datiem tiek izmantoti seši valūtu kursi:

1. EUR/AUD - eiro attiecība pret Austrālijas dolāru,
2. EUR/CAD - eiro attiecība pret Kanādas dolāru,
3. EUR/CZK - eiro attiecība pret Čehijas kronu,
4. EUR/GBP - eiro attiecība pret Lielbritānijas sterliņu mārciņu,
5. EUR/JPY - eiro attiecība pret Japānas jēnu,

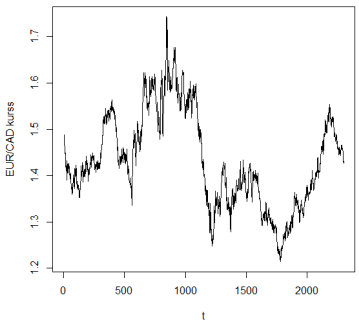
6. EUR/USD - eiro attiecība pret ASV dolāru

par laika periodu 01.09.2004. – 01.09.2014. Tas nozīmē, ka kopā ir 2304 datu punkti. Dati tiek ņemti no Latvijas Bankas mājaslapas [21]. Attēlos 6.1 var redzēt visu valūtu kursu laikrindas.

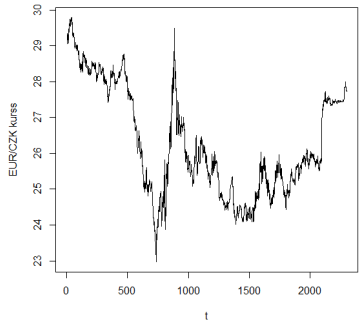
Pieraksts EUR/AUD nozīmē, cik Austrālijas dolāru par vienu eiro ir iespējējams iegādāties.



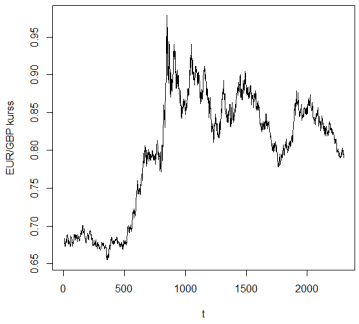
(a) **EUR/AUD**



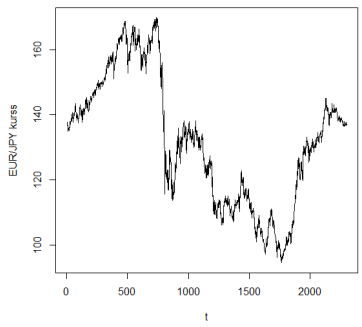
(b) **EUR/CAD**



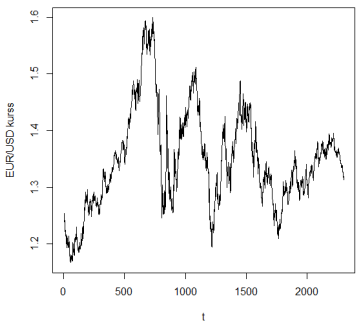
(c) **EUR/CZK**



(d) **EUR/GBP**



(e) **EUR/JPY**



(f) **EUR/USD**

Att. 6.1: **Attēlos (a) - (f) ir uzzīmēti valūtu kursu laikrindu grafiki laika posmā 01.09.2004. - 01.09.2014.**

Datu sagatavošana

Pirms datu ievades tīklā, ir ieteicams tos sagatavot. [8] Tādēļ ieejas dati un vēlamās atbildes tiek normalizētas pēc formulas 6.01, jo izvēlētajām aktivizācijas funkcijām izejas intervāls ir $[0; 1]$ un $[-1 : 1]$, kas var novest pie tā, ka tīkls neapmācās.

$$x'_j = \frac{x_j - min(x)}{max(x) - min(x)}, \tag{6.01}$$

kur $j = 1..G$ un G ir laikrindas garums, kas izvēlētajam laika periodam ir $G = 2304$. Ar x' tiek apzīmēta normalizētā laikrinda un ar x ir apzīmēta patiesā laikrinda. Vēlāk, kad ir nepieciešams aplūkot rezultātus, dati tiek denormalizēti pēc formulas 6.02.

$$x_j = x'_j(max(x) - min(x)) + min(x) \tag{6.02}$$

Tabulā 6.1 ir parādīti visu valūtu kursu maksimālās un minimālās vērtības, kas tiks izmantotas tālākos aprēķinos.

Tabula 6.1: **Valūtu kursu maksimālās un minimālās vērtības**

	EUR/AUD	EUR/CAD	EUR/CZK
min	1.1639	1.2139	22.968
max	2.0735	1.7433	29.788
	EUR/GBP	EUR/JPY	EUR/USD
min	0.65485	94.63	1.1639
max	0.97855	169.75	2.0735

Pēc normalizēšanas visi datu punkti pieņems vērtības intervālā $[0; 1]$. Maksimālā vērtība būs 1, bet minimālā - 0. Ja paņem kursa EUR/JPY 431. datu punktu 1.3527, tad, ievietojot to formulā 6.01, iegūst:

$$x'_j = \frac{1.3527 - 1.1639}{2.0735 - 1.1639} = 0.4303 \tag{6.03}$$

Lai prognozētu, autors [8] izmanto deviņas iepriekšējās vērtības. Tas nozīmē, ka no visiem 2304 datu punktiem prognozēti tiks pēdējie 2295. Tātad tīklam ieejas virsotnes ir desmit, jo tiek pievienota vēl viena virsotne, kas vienmēr pieņem vērtību viens. Visi dati tiek sakārtoti tā, lai tos būtu ērti izmantot ievadīšanai tīklā. 6.04 parādīts, kā izskatās ieejas matrica, un 6.05 tiek parādīts, kā izskatīsies attiecīgās vēlamās izejas.

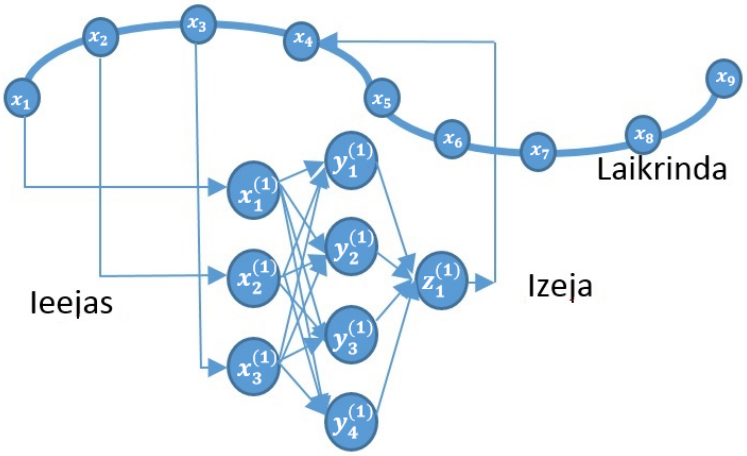
$$\begin{pmatrix} x_1^{(1)} & x_2^{(1)} & ... & x_{10}^{(1)} \\ x_1^{(2)} & x_2^{(2)} & ... & x_{10}^{(2)} \\ ... & ... & ... & ... \\ x_1^{(2295)} & x_2^{(2295)} & ... & x_{10}^{(2295)} \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & ... & x_9 & 1 \\ x_2 & x_3 & ... & x_{10} & 1 \\ ... & ... & ... & ... & \\ x_{2297} & x_{2297} & ... & x_{2304} & 1 \end{pmatrix} \tag{6.04}$$

$$\begin{pmatrix} d^{(1)} \\ d^{(2)} \\ \dots \\ d^{(2)} \end{pmatrix} = \begin{pmatrix} x_{10}^{(1)} \\ x_{11}^{(2)} \\ \dots \\ x_{2305}^{(2)} \end{pmatrix} \tag{6.05}$$

Autors [8], kā arī citi autori iesaka neironu tīklu apmācīšanai lietot 80% jeb 1843 datu punktus no visiem datiem, savukārt testēšanai atlikušos 20% jeb 461 datu punktus no visiem datiem. Tas nozīmē, ka apmācīšanās gadījumā ieejas vektoru skaits ir 1843, bet testēšanai ir 452 ieejas vektors. Ar tīkla apmācību saprot, ka tīklā pēc kārtas tiek laisti ieejas vektori, tiek uzlaboti svari un samazināta kļūda. Ar testēšanu saprot, kad tiek izmantoti tīkla apmācības pēdējie svari, un tie vairs netiek uzlaboti, tiek iegūtas tikai izejas vērtības, netiek izmantotas vēlamās atbildes.

Datu ievade tīklā

Pieņem, ka ir laikrinda x_t, kur mainīgais x mainās laikā un $t = 1, 2...$, un ir jāprognozē vērtība x laikā $t + h$. [9] Attēlā 6.2 ir parādīts princips, kā tiek prognozēta vērtība x_4. Tātad tiek izmantotas iepriekšējās trīs vērtības x_1, x_2 un x_3, kas ir mākslīgā neironu tīkla ieejas un vēlamā izeja tīklam ir x_4. Kad ir iegūta neironu tīkla izejas vērtība $z_1^{(1)}$, tad ievade sākas no jauna un $x_1^{(2)} = x_2$, $x_2^{(2)} = x_3$ un $x_3^{(2)} = x_4$, un vēlamā izeja ir x_5. Šādi tiek apmācīti 80% datu.



Att. 6.2: **Mākslīgo neironu tīkla apmācība un prognozēšana**

Modeļu prognozēšanas precizitātes salīdzināšana

Lai salīdzinātu modeļu prognozēšanas precizitāti tiek izmantoti trīs kļūdu novērtējumi: vidējā absolūtā standartkļūda (MAD, formula 6.06), vidējā kvadrātiskā kļūda (MSE, formula 6.07) un vidējā absolūtā kļūda ($MAPE$, formula 6.08). [16]

$$MAD = \frac{1}{n}\sum_{t=1}^{n}|d_t - z_t| \tag{6.06}$$

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(d_t - z_t)^2 \tag{6.07}$$

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|d_t - z_t|}{d_t} \tag{6.08}$$

Modeļu izvēle

Kopā tiks izvēlēti seši modeļi:

1. aktivizācijas funkcija gan slēptajos neironos, gan izejas neironā ir loģistiskā sigmoīda;
2. aktivizācijas funkcija slēptajos neironos ir loģistiskā sigmoīda un izejas neironā ir identitātes;
3. aktivizācijas funkcija slēptajos neironos ir loģistiskā sigmoīda un izejas neironā ir hiperboliskais tangens;
4. aktivizācijas funkcija slēptajos neironos ir hiperboliskais tangens un izejas neironā ir identitātes;
5. aktivizācijas funkcija gan slēptajos neironos, gan izejas neironā ir hiperboliskais tangens;
6. aktivizācijas funkcija slēptajos neironos ir hiperboliskais tangens un izejas neironā ir loģistiskā sigmoīda.

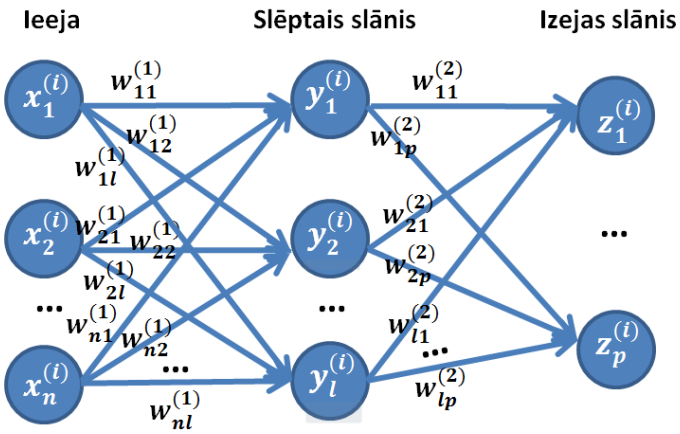
Visbiežāk literatūrā apskatīti ir 1., 2 un 4. modelis, piemēram, [8], [2], [6] un [13], bet 3., 5. un 6. modelis netiek aprakstīti aplūkotajā literatūrā.

7 Modeļu matemātiskais apraksts

Šī nodaļa un 7.1 apakšnodaļa tiek balstīta uz avotiem [13] un [3], jāatzīmē, ka visi formulu izvedumi ir izvērstāki nekā pieejamājā literatūrā. Vienosimies par apzīmējumiem. Turpmāk pieņem, ka ir n ieejas, l ir neironu skaits slēptajā slānī un p ir neironu skaits izejas slānī. Apmācīšanas procesā ir dots mācīšanās paraugs $(x^{(i)}, d^{(i)})$, kur $x^{(i)} \in \mathbb{R}^n, i = 1, ..., N$ ir ieejas vērtību vektors un $d^{(i)} \in \{0, 1\}^p$ ir vēlamā atbilde, kur vektorā būs tik koordinātas, cik ir izejas neironu. $z^{(i)} \in \mathbb{R}^p$ ir izejas vērtība izejas slānī. Visu attēlo attēlā 7.1., lai ir vieglāk saprast apzīmējumus.

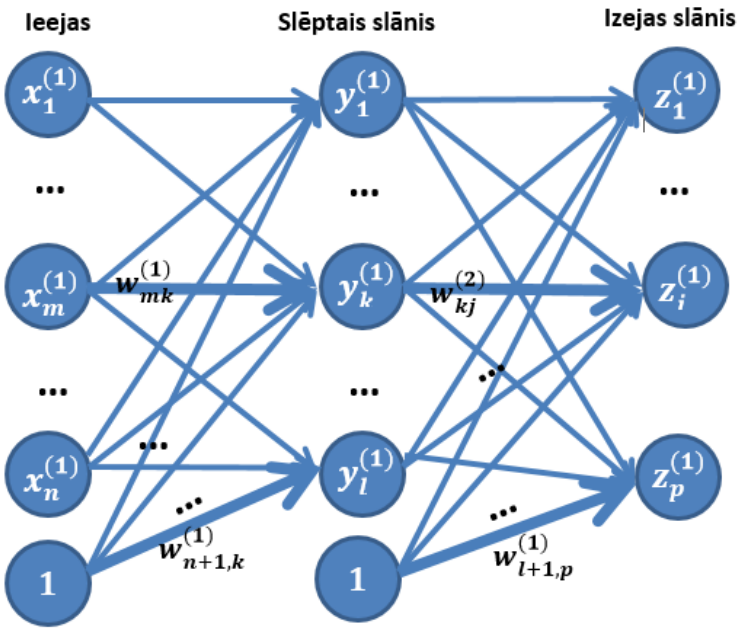
Piemērs

Attēlā 7.1 dots divslāņu neironu tīkls. Tīklam ir n ieejas vērtības $x^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)})$, viens slēptais slānis, kas sastāv no l neironiem $y^{(i)} = (y_1^{(i)}, \ldots, y_l^{(i)})$. Ieejas ar slēpto slāni savieno svari $w_{ij}^{(1)}$, $i = 1, \ldots, n$, $j = 1, \ldots, l$. Slēpto slāni ar izeju savieno svari $w_{ij}^{(2)}$, $i = 1, \ldots, l$, $j = 1, \ldots, p$. Tīklam ir p izejas $z^{(i)} = (z_1^{(i)}, \ldots, z_p^{(i)})$. Jau iepriekš ir dots vektors ar vēlamajām izejām $d^{(1)} = (d_1^{(1)}, .., d_p^{(1)})$.



Att. 7.1: **Mākslīgā divslāņu neironu tīkla uzbūve**

Tālāk ar $w_{mk}^{(1)}$ apzīmē svaru starp ieejas virsotni m un slēpto neironu k, un $m = 1, ..., n$, $k = 1, ..., l$. Ar $w_{kj}^{(2)}$ apzīmē svaru starp slēpto neironu k un izejas neironu j, un $k = 1, ..., l$, $j = 1, ..., p$ (attēls 7.2). Līdz ar to izejas vērtība neironam j izejas slānī ir:



Att. 7.2: **Atgriezeniskā izplatīšanās algoritma tīkla struktūra**

$$z_j^{(i)} = \sigma(\sum_{k=1}^{l} w_{kj}^{(2)} \sigma(\sum_{m=1}^{n} w_{mk}^{(1)} x_m^{(i)} - b_k^{(1)}) - b_j^{(2)}), j = 1, .., p. \tag{7.01}$$

Formulā 7.01 var redzēt, ka otrajās iekavās ir aprēķināta izejas vērtība, kas tiek iegūta no slēptā slāņa un attiecīgi šī vērtība tiek sareizināta ar svariem, kas savieno slēpto slāni un izejas slāni, tādā veidā iegūstot to, ka $z_j^{(i)}$ ir izejas vērtība visam neironu tīklam.

Tālāk pievieno papildu neironu gan ieejas slānim, gan slēptajam slānim, kura vērtība vienmēr būs 1 (attēls 7.2). Tā drīkst darīt, jo svars starp slieksni un neironu slēptajā slānī ir 1, bet tagad slieksnis būs attiecīgais svars starp pievienoto virsotni, kuras vērtība ir 1, līdz ar to rezultāts nemainās. Formulu 7.01 ir ērti sadalīt divās daļās tā, lai tiktu uzrādītas formulas, kā tiek aprēķinātas izejas vērtības no slēptā slāņa un izejas vērtības no izejas slāņa attiecīgi:

$$y_j^{(i)} = \sigma(\sum_{m=1}^{n+1} w_{mj}^{(1)} x_m^{(i)}), j = 1, \ldots, l, \tag{7.02}$$

$$z_j^{(i)} = \sigma(\sum_{k=1}^{l+1} w_{kj}^{(2)} y_k^{(i)}), j = 1, \ldots p. \tag{7.03}$$

Kā var redzēt, formulās 7.02 un 7.03 ir pievienota viena ieeja un viena virsotne slēptajam slānim. Kad tiek aprēķināts $y_j^{(i)}$, sākumā algoritms aprēķina summu no ieejas, lietojot formulu:

$$s_j^{(i)} = \sum_{m=1}^{n+1} w_{mj}^{(1)} x_m^{(i)}, j = 1, \ldots, l.$$

Tālāk tiek aprēķināta izeja ar loģistisko sigmoīda funkciju:

$$\sigma = \frac{1}{1 + e^{-s_j^{(i)}}}.$$

Tagad ir aprēķināta izejas vērtība no slēptā slāņa (formula 7.02). Tālāk var aprēķināt izejas vērtību $z_j^{(i)}$ no izejas slāņa, ko dara tieši tāpat, tikai šoreiz ieejas vērtības ir $y_j^{(i)}$, līdz ar to summācijas funkcija izskatās šādi:

$$s_j^{(i)} = \sum_{k=1}^{l+1} w_{kj}^{(2)} y_k^{(i)}, j = 1, \ldots, p$$

un aktivizācijas funkcija šādi:

$$\sigma = \frac{1}{1 + e^{-s_j^{(i)}}}.$$

Līdz ar to tiek iegūta formula 7.03.

Kā jau iepriekš tika teikts, lai apmācītu neironu tīklu ar atgriezenisko izplatīšanās algoritmu, ir jāminimizē kļūdas funkcija 5.21. Kļūdu $E : \mathbb{R}^{(n+1)l+p(l+1)} \to \mathbb{R}$ minimizē, mainot svaru vērtības. [3] [13] Kļūda ir atkarīga no neironu tīkla katra slāņa svaru vērtībām. Apmācības procesa ātrums un precizitāte ir atkarīga no mācīšanās koeficienta φ. Tālāk visos modeļos kļūda tiks atvasināta pēc $w_{kj}^{(1)}$, $w_{kj}^{(2)}$, lai iegūtu nepieciešamos rezultātus.

1. pielikumā tiek parādīts vispārīgā neironu tīkla formulu iegūšana, kur nav konkrēts skaits slēpto slāņu, konkrētu aktivizācijas funkciju, līdz ar to ir iespējams lasītājam izveidot jebkuru modeli.

7.1 Pirmais modelis

Šajā modelī slēptajiem neironiem aktivizācijas funkcija formulā 7.02 ir loģistiskā sigmoīda funkcija un izejā aktivizācijas funkcija formulā 7.03 arī ir loģiskā sigmoīda funkcija. Tiks izmantoti visi tie paši apzīmējumi, kas 7. nodaļā, ja netiks īpaši norādīts, ka būs citādi.

Tagad ir jāaprēķina parciālais atvasinājums 7.11 jeb kļūdas 5.21 visi parciālie atvasinājumi pēc visiem svariem, kas savieno slēpto slāni ar izejas neironu.

$$\frac{\partial E}{\partial w_{kj}^{(2)}} \tag{7.11}$$

7.11 izmanto diferenciāļa formas invarianci.

$$\frac{\partial E}{\partial w_{kj}^{(2)}} = \sum_{i=1}^{N} \frac{\partial E}{\partial z_j^{(i)}} \frac{\partial z_j^{(i)}}{\partial w_{kj}^{(2)}} \tag{7.12}$$

Tālāk aprēķinus ir ērtāk veikt, ja 7.12 sadala attiecīgi 7.13 un 7.14.

$$\sum_{i=1}^{N} \frac{\partial E}{\partial z_j^{(i)}} = \sum_{i=1}^{N} \frac{1}{2} 2(z_j^{(i)} - d_j^{(i)}) \frac{\partial z_j^{(i)}}{\partial z_j^{(i)}} = \sum_{i=1}^{N} (z_j^{(i)} - d_j^{(i)}) \tag{7.13}$$

$$\begin{aligned} \sum_{i=1}^{N} \frac{\partial z_j^{(i)}}{\partial w_{kj}^{(2)}} &= \sum_{i=1}^{N} \frac{\partial (\frac{1}{1+e^{-w_{kj}^{(2)} y_k^{(i)}}})}{\partial w_{kj}^{(2)}} = \sum_{i=1}^{N} y_k^{(i)} \frac{e^{-w_{kj}^{(2)} y_k^{(i)}}}{(e^{-w_{kj}^{(2)} y_k^{(i)}} + 1)^2} = \\ &= \sum_{i=1}^{N} z_j^{(i)} (1 - z_j^{(i)}) y_k^{(i)}. \end{aligned} \tag{7.14}$$

Šeit 7.14 tika izmantots tas, ka $z_j^{(i)}$ var pārrakstīt kā $\sigma(s_j^{(i)})$. Līdz ar to ir iegūts rezultāts 7.15.

$$\frac{\partial E}{\partial w_{kj}^{(2)}} = \sum_{i=1}^{N} \frac{\partial E}{\partial z_j^{(i)}} \frac{\partial z_j^{(i)}}{\partial w_{kj}^{(2)}} = \sum_{i=1}^{N} (z_j^{(i)} - d_j^{(i)}) z_j^{(i)} (1 - z_j^{(i)}) y_k^{(i)}. \tag{7.15}$$

Tālāk jāaprēķina parciālo atvasinājumu (7.16) jeb kļūdas (5.21) visi parciālie atvasinājumi pēc svariem, kas savieno ieejas virsotnes ar slēpto slāni.

$$\frac{\partial E}{\partial w_{kj}^{(1)}} \tag{7.16}$$

7.16 tiek izmantota diferenciāļa formas invariance.

$$\frac{\partial E}{\partial w_{kj}^{(1)}} = \sum_{i=1}^{N} (\sum_{q=1}^{p} \frac{\partial E}{\partial z_q^{(i)}} \frac{\partial z_q^{(i)}}{\partial y_j^{(i)}}) \frac{\partial y_j^{(i)}}{\partial w_{kj}^{(1)}}, \tag{7.17}$$

7.17 diferenciāļa formas invariance tiek izmantota divas reizes. Tālāk tiks aprēķināts katrs parciālais atvasinājums atsevišķi attiecīgi 7.18, 7.19 un 7.110.

$$\sum_{q=1}^{p} \frac{\partial E}{\partial z_q^{(i)}} = \sum_{q=1}^{p} \frac{1}{2} 2(z_q^{(i)} - d_q^{(i)}) \frac{\partial z_q^{(i)}}{\partial z_q^{(i)}} = \sum_{q=1}^{p} (z_q^{(i)} - d_q^{(i)}), \tag{7.18}$$

$$\sum_{q=1}^{p} \frac{\partial z_q^{(i)}}{\partial y_j^{(i)}} = \sum_{q=1}^{p} \frac{\partial(\frac{1}{1+e^{-w_{jq}^{(2)} y_j^{(i)}}})}{\partial y_j^{(i)}} = \sum_{q=1}^{p} w_{jq}^{(2)} \frac{e^{-w_{jq}^{(2)} y_j^{(i)}}}{(e^{-w_{jq}^{(2)} y_j^{(i)}} + 1)^2} =$$
$$= \sum_{q=1}^{p} z_q^{(i)}(1 - z_q^{(i)}) w_{jq}^{(2)}, \tag{7.19}$$

$$\sum_{i=1}^{N} \frac{\partial y_j^{(i)}}{\partial w_{kj}^{(1)}} = \sum_{i=1}^{N} \frac{\partial(\frac{1}{1+e^{-w_{kj}^{(1)} x_k^{(i)}}})}{\partial w_{kj}^{(1)}} = \sum_{i=1}^{N} x_k^{(i)} \frac{e^{-w_{kj}^{(1)} x_k^{(i)}}}{(e^{-w_{kj}^{(k)} x_k^{(i)}} + 1)^2} =$$
$$= \sum_{i=1}^{N} y_j^{(i)}(1 - y_j^{(i)}) x_k^{(i)}. \tag{7.110}$$

Līdz ar to ir iegūts rezultāts 7.111.

$$\frac{\partial E}{\partial w_{kj}^{(1)}} = \sum_{i=1}^{N} (\sum_{q=1}^{p} \frac{\partial E}{\partial z_q^{(i)}} \frac{\partial z_q^{(i)}}{\partial y_j^{(i)}}) \frac{\partial y_j^{(i)}}{\partial w_{kj}^{(i)}} =$$
$$= \sum_{i=1}^{N} (\sum_{q=1}^{p} (z_q^{(i)} - d_q^{(i)}) z_q^{(i)}(1 - z_q^{(i)}) w_{jq}^{(2)}) y_j^{(i)}(1 - y_j^{(i)}) x_k^{(i)}. \tag{7.111}$$

Tālāk apzīmē:

$$\delta_j^{(i)} = (z_j^{(i)} - d_j^{(i)}) z_j^{(i)}(1 - z_j^{(i)}), \tag{7.112}$$
$$\Delta_j^{(i)} = y_j^{(i)}(1 - y_j^{(i)}) \sum_q \delta_q^{(i)} w_{jq}^{(2)}. \tag{7.113}$$

$\delta_j^{(i)}$ un $\Delta_j^{(i)}$ sauc par kļūdām atgriezeniskajā algoritma daļā. Var pārrakstīt iegūtos rezultātus:

$$\frac{\partial E}{\partial w_{kj}^{(2)}} = \sum_{i=1}^{N} \delta_j^{(i)} y_k^{(i)},$$
$$\frac{\partial E}{\partial w_{kj}^{(1)}} = \sum_{i=1}^{N} \Delta_j^{(i)} x_k^{(i)}.$$

Tagad var pārrakstīt formulas, kā tiks uzlaboti svari, jo ir iegūti nepieciešamie parciālie atvasinājumi. Formula 7.114 tiek izmantota, lai uzlabotu svarus no slēptā slāņa uz izejas neironu, un formula 7.115 tiek izmantota, lai uzlabotu svarus no ieejas virsotnēm uz slēpto slāni.

$$w_{kj}^{(2)}(t+1) = w_{kj}^{(2)}(t) - \varphi \sum_{i=1}^{N} \delta_j^{(i)} y_k^{(i)}, \tag{7.114}$$

kur $k = 1, \ldots, l+1$ un $j = 1, \ldots, p$.

$$w_{kj}^{(1)}(t+1) = w_{kj}^{(1)}(t) - \varphi \sum_{i=1}^{N} \Delta_j^{(i)} x_k^{(i)}, \tag{7.115}$$

kur $k = 1, \ldots, n+1$ un $j = 1, \ldots, l$.

7.2 Otrais modelis

Šajā modelī slēptajiem neironiem aktivizācijas funkcija formulā 7.02 ir loģistiskā sigmoīda funkcija un izejā aktivizācijas funkcija formulā 7.03 ir identitātes funkcija. Tiks izmantoti visi tie paši apzīmējumi, kas 7. nodaļā, ja netiks īpaši norādīts, ka būs citādi.

Tālāk kļūda tiks atvasināta pēc $w_{kj}^{(1)}$, $w_{kj}^{(2)}$. Arī šeit ir jāaprēķina parciālais atvasinājums 7.11, kur izmanto diferenciāļa formas invarianci 7.12. Tiks izmantoti iepriekš iegūtie rezultāti - 7.13. Atliek iegūt parciālos atvasinājumus:

$$\sum_{i=1}^{N} \frac{\partial z_j^{(i)}}{\partial w_{kj}^{(2)}} = \sum_{i=1}^{N} \frac{\partial w_{kj}^{(2)} y_k^{(i)}}{\partial w_{kj}^{(1)}} = \sum_{i=1}^{N} y_k^{(i)}. \tag{7.21}$$

Līdz ar to, vienādojumā 7.12 ievietojot 7.13 un 7.21, ir iegūts rezultāts:

$$\frac{\partial E}{\partial w_{kj}^{(2)}} = \sum_{i=1}^{N} \frac{\partial E}{\partial z_j^{(i)}} \frac{\partial z_j^{(i)}}{\partial w_{kj}^{(2)}} = \sum_{i=1}^{N} (z_j^{(i)} - d_j^{(i)}) y_k^{(i)}. \tag{7.22}$$

Tālāk atkal kļūda tiek atvasināta pēc svariem, kas savieno ieejas ar slēpto slāni 7.16, kur atkal izmanto diferenciāļa formas invarianci un jāaprēķina ir 7.17. Tiks izmantoti iepriekš iegūtie rezultāti - 7.18 un 7.110. Atliek iegūt:

$$\sum_{q=1}^{p} \frac{\partial z_q^{(i)}}{\partial y_j^{(i)}} = \sum_{i=1}^{N} \frac{\partial w_{kj}^{(2)} y_k^{(i)}}{\partial w_{kj}^{(1)}} = w_{kj}^{(2)}. \tag{7.23}$$

Līdz ar to,vienādojumā 7.17 ievietojot 7.18, 7.110 un 7.23 , ir iegūts rezultāts:

$$\frac{\partial E}{\partial w_{kj}^{(1)}} = \sum_{i=1}^{N}(\sum_{q=1}^{p} \frac{\partial E}{\partial z_q^{(i)}} \frac{\partial z_q^{(i)}}{\partial y_j^{(i)}}) \frac{\partial y_j^{(i)}}{\partial w_{kj}^{(i)}} =$$
$$= \sum_{i=1}^{N}(\sum_{q=1}^{p} (z_q^{(i)} - d_q^{(i)}) w_{jq}^{(2)}) y_k^{(i)} (1 - y_j^{(i)}) x_k^{(i)} \tag{7.24}$$

Apzīmē:

$$\delta_j^{(i)} = (z_q^{(i)} - d_q^{(i)}), \tag{7.25}$$
$$\Delta_j^{(i)} = y_k^{(i)} (1 - y_j^{(i)}) \sum_q \delta_j^{(i)} w_{jq}^{(2)}. \tag{7.26}$$

Šos rezultātus var tālāk ievietot 7.114 un 7.115, lai iegūtu formulas, kā uzlabot svarus.

7.3 Trešais modelis

Šajā modelī slēptajiem neironiem aktivizācijas funkcija formulā 7.02 ir loģistiskā sigmoīda funkcija un izejā aktivizācijas funkcija formulā 7.03 ir hiperboliskā tangensa funkcija. Tiks izmantoti visi tie paši apzīmējumi, kas 7. nodaļā, ja netiks īpaši noādīts, ka būs citādi.

Tālāk kļūda tiks atvasināta pēc $w_{kj}^{(1)}$, $w_{kj}^{(2)}$. Arī šeit ir jāaprēķina parciālais atvasinājums 7.11, kur izmanto diferenciāļa formas invarianci 7.12. Tiks izmantoti iepriekš iegūtie rezultāti - 7.13. Atliek aprēķināt:

$$\sum_{i=1}^{N} \frac{\partial z_j^{(i)}}{\partial w_{kj}^{(2)}} = \frac{\partial(\frac{e^{w_{kj}^{(1)} x_k^{(i)}} - e^{-w_{kj}^{(1)} x_k^{(i)}}}{e^{w_{kj}^{(1)} x_k^{(i)}} + e^{-w_{kj}^{(1)} x_k^{(i)}}})}{\partial w_{kj}^{(2)}} = \sum_{i=1}^{N} y_k^{(i)} (1 - (z_j^{(i)})^2). \tag{7.31}$$

Līdz ar to, vienādojumā 7.12 ievietojot 7.13 un 7.31, ir iegūts rezultāts:

$$\frac{\partial E}{\partial w_{kj}^{(2)}} = \sum_{i=1}^{N} \frac{\partial E}{\partial z_j^{(i)}} \frac{\partial z_j^{(i)}}{\partial w_{kj}^{(2)}} = \sum_{i=1}^{N} (z_j^{(i)} - d_j^{(i)}) y_k^{(i)} (1 - (z_j^{(i)})^2). \tag{7.32}$$

Tālāk atkal kļūda tiek atvasināta pēc svariem, kas savieno ieejas ar slēpto slāni 7.16, kur atkal izmanto diferenciāļa formas invarianci un jāaprēķina ir 7.17. Tiks izmantoti iepriekš iegūtie rezultāti - 7.18 un 7.110. Atliek iegūt:

$$\sum_{q=1}^{p} \frac{\partial z_q^{(i)}}{\partial y_j^{(i)}} = \frac{\partial(\frac{e^{w_{kj}^{(1)} x_k^{(i)}} - e^{-w_{kj}^{(1)} x_k^{(i)}}}{e^{w_{kj}^{(1)} x_k^{(i)}} + e^{-w_{kj}^{(1)} x_k^{(i)}}})}{\partial y_j^{(i)}} = w_{kj}^{(2)}(1 - (z_q^{(i)})^2). \tag{7.33}$$

Līdz ar to,vienādojumā 7.17 ievietojot 7.18, 7.110 un 7.33, ir iegūts rezultāts:

$$\begin{aligned} \frac{\partial E}{\partial w_{kj}^{(1)}} &= \sum_{i=1}^{N}(\sum_{q=1}^{p} \frac{\partial E}{\partial z_q^{(i)}} \frac{\partial z_q^{(i)}}{\partial y_j^{(i)}}) \frac{\partial y_j^{(i)}}{\partial w_{kj}^{(i)}} = \\ &= \sum_{i=1}^{N}(\sum_{q=1}^{p}(z_q^{(i)} - d_q^{(i)}) w_{jq}^{(2)}(1 - (z_q^{(i)})^2)) y_k^{(i)}(1 - y_j^{(i)}) x_k^{(i)} \end{aligned} \tag{7.34}$$

Apzīmē:

$$\delta_j^{(i)} = (z_j^{(i)} - d_j^{(i)})(1 - (z_k^{(i)})^2), \tag{7.35}$$

$$\Delta_j^{(i)} = y_k^{(i)}(1 - y_j^{(i)}) \sum_q \delta_j^{(i)} w_{jq}^{(2)}. \tag{7.36}$$

Šos rezultātus var tālāk ievietot 7.114 un 7.115, lai iegūtu formulas, kā uzlabot svarus.

7.4 Ceturtais modelis

Šajā modelī slēptajiem neironiem aktivizācijas funkcija formulā 7.02 ir hiperboliskā tangensa funkcija un izejā aktivizācijas funkcija formulā 7.03 ir identitātes funkcija. Tiks izmantoti visi tie paši apzīmējumi, kas 7. nodaļā, ja netiks īpaši norādīts, ka būs citādi.

Tālāk kļūda tiks atvasināta pēc $w_{kj}^{(1)}$, $w_{kj}^{(2)}$. Arī šeit ir jāaprēķina parciālais atvasinājums 7.11, kur izmanto diferenciāļa formas invarianci 7.12. Tiks izmantoti iepriekš iegūtie rezultāti - 7.13 un 7.21. Līdz ar to,ievietojot 7.13 un 7.21 vienādojumā 7.12, ir iegūts rezultāts:

$$\frac{\partial E}{\partial w_{kj}^{(2)}} = \sum_{i=1}^{N} \frac{\partial E}{\partial z_j^{(i)}} \frac{\partial z_j^{(i)}}{\partial w_{kj}^{(2)}} = \sum_{i=1}^{N}(z_j^{(i)} - d_j^{(i)}) y_k^{(i)}. \tag{7.41}$$

Tālāk atkal kļūda tiek atvasināta pēc svariem, kas savieno ieejas ar slēpto slāni 7.16, kur atkal izmanto diferenciāļa formas invarianci un jāaprēķina ir 7.17. Tiks izmantoti iepriekš iegūtie rezultāti - 7.18 un 7.23. Atliek aprēķināt:

$$\sum_{i=1}^{N}\frac{\partial y_j^{(i)}}{\partial w_{kj}^{(1)}}=\sum_{i=1}^{N}\frac{\partial(\frac{e^{w_{kj}^{(1)}x_k^{(i)}}-e^{-w_{kj}^{(1)}x_k^{(i)}}}{e^{w_{kj}^{(1)}x_k^{(i)}}+e^{-w_{kj}^{(1)}x_k^{(i)}}})}{\partial w_{kj}^{(1)}}=\sum_{i=1}^{N}x_k^{(i)}(1-(y_j^{(i)})^2). \tag{7.42}$$

Līdz ar to,ievietojot 7.18, 7.23 un 7.42 vienādojumā 7.17, ir iegūts rezultāts:

$$\begin{aligned}\frac{\partial E}{\partial w_{kj}^{(1)}}&=\sum_{i=1}^{N}(\sum_{q=1}^{p}\frac{\partial E}{\partial z_q^{(i)}}\frac{\partial z_q^{(i)}}{\partial y_j^{(i)}})\frac{\partial y_j^{(i)}}{\partial w_{kj}^{(i)}}=\\&=\sum_{i=1}^{N}(\sum_{q=1}^{p}(z_q^{(i)}-d_q^{(i)})w_{jq}^{(2)})x_k^{(i)}(1-(y_j^{(i)})^2)\end{aligned} \tag{7.43}$$

Apzīmē:

$$\delta_j^{(i)}=z_j^{(i)}–d_j^{(i)}, \tag{7.44}$$

$$\Delta_j^{(i)}=(1-(y_j^{(i)})^2)\sum_q\delta_q^{(i)}w_{jq}^{(2)}. \tag{7.45}$$

Šos rezultātus var tālāk ievietot 7.114 un 7.115, lai iegūtu formulas, kā tiek uzlaboti svari.

7.5 Piektais modelis

Šajā modelī slēptajiem neironiem aktivizācijas funkcija formulā 7.02 ir hiperboliskā tangensa funkcija un izejā aktivizācijas funkcija formulā 7.03 arī ir hiperboliskā tangensa funkcija. Tiks izmantoti visi tie paši apzīmējumi, kas 7. nodaļā, ja netiks īpaši norādīts, ka būs citādi.

Tālāk kļūda tiks atvasināta pēc $w_{kj}^{(1)}$, $w_{kj}^{(2)}$. Arī šeit ir jāaprēķina parciālais atvasinājums 7.11, kur izmanto diferenciāļa formas invarianci 7.12. Tiks izmantoti iepriekš iegūtie rezultāti - 7.13 un 7.31.

Līdz ar to,ievietojot 7.13 un 7.31 vienādojumā 7.12, ir iegūts rezultāts:

$$\frac{\partial E}{\partial w_{kj}^{(2)}}=\sum_{i=1}^{N}\frac{\partial E}{\partial z_j^{(i)}}\frac{\partial z_j^{(i)}}{\partial w_{kj}^{(2)}}=\sum_{i=1}^{N}(z_j^{(i)}-d_j^{(i)})y_k^{(i)}(1-(z_j^{(i)})^2). \tag{7.51}$$

Tālāk atkal kļūda tiek atvasināta pēc svariem, kas savieno ieejas ar slēpto slāni 7.16, kur atkal izmanto diferenciāļa formas invarianci, un ir tālāk jāaprēķina 7.17. Tiks izmantoti iepriekš iegūtie rezultāti - 7.18, 7.33 un 7.42.

Līdz ar to,ievietojot 7.18, 7.33 un 7.42 vienādojumā 7.17, ir iegūts rezultāts:

$$\frac{\partial E}{\partial w_{kj}^{(1)}} = \sum_{i=1}^{N}(\sum_{q=1}^{p}\frac{\partial E}{\partial z_q^{(i)}}\frac{\partial z_q^{(i)}}{\partial y_j^{(i)}})\frac{\partial y_j^{(i)}}{\partial w_{kj}^{(i)}} = \\ = \sum_{i=1}^{N}(\sum_{q=1}^{p}(z_q^{(i)} - d_q^{(i)})w_{jq}^{(2)}(1-(z_q^{(i)})^2))x_k^{(i)}(1-(y_j^{(i)})^2) \tag{7.52}$$

Apzīmē:

$$\delta_j^{(i)} = (z_j^{(i)} - d_j^{(i)})(1-(z_j^{(i)})^2), \tag{7.53}$$

$$\Delta_j^{(i)} = (1-(y_j^{(i)})^2)\sum_q \delta_j^{(i)} w_{jk}^{(2)}. \tag{7.54}$$

Šos rezultātus var tālāk ievietot 7.114 un 7.115, lai iegūtu formulas, kā tiek uzlaboti svari.

7.6 Sestais modelis

Šajā modelī slēptajiem neironiem aktivizācijas funkcija formulā 7.02 ir hiperboliskā tangensa funkcija un izejā aktivizācijas funkcija formulā 7.03arī ir loģistiskā sigmoīda funkcija. Tiks izmantoti visi tie paši apzīmējumi, kas 7. nodaļā, ja netiks īpaši norādīts, ka būs citādi.

Tālāk kļūda tiks atvasināta pēc $w_{kj}^{(1)}$, $w_{kj}^{(2)}$. Arī šeit ir jāaprēķina parciālais atvasinājums 7.11, kur izmanto diferenciāļa formas invarianci 7.12. Tiks izmantoti iepriekš iegūtie rezultāti - 7.13 un 7.14. Līdz ar to,ievietojot 7.13 un 7.14 vienādojumā 7.12, ir iegūts rezultāts:

$$\frac{\partial E}{\partial w_{kj}^{(2)}} = \sum_{i=1}^{N}\frac{\partial E}{\partial z_j^{(i)}}\frac{\partial z_j^{(i)}}{\partial w_{kj}^{(2)}} = \sum_{i=1}^{N}(z_j^{(i)} - d_j^{(i)})z_j^{(i)}(1-z_j^{(i)})y_k^{(i)}. \tag{7.61}$$

Tālāk atkal kļūda tiek atvasināta pēc svariem, kas savieno ieejas ar slēpto slāni 7.16, kur atkal izmanto diferenciāļa formas invarianci un jāaprēķina ir 7.17. Tiks izmantoti iepriekš iegūtie rezultāti - 7.18, 7.19 un 7.42. Līdz ar to,ievietojot 7.18, 7.19 un 7.42 vienādojumā 7.17, ir iegūts rezultāts:

$$\frac{\partial E}{\partial w_{kj}^{(1)}} = \sum_{i=1}^{N}(\sum_{q=1}^{p}\frac{\partial E}{\partial z_q^{(i)}}\frac{\partial z_q^{(i)}}{\partial y_j^{(i)}})\frac{\partial y_j^{(i)}}{\partial w_{kj}^{(i)}} = \\ = \sum_{i=1}^{N}(\sum_{q=1}^{p}(z_q^{(i)} - d_q^{(i)})z_q^{(i)}(1-z_q^{(i)})w_{kj}^{(2)})x_k^{(i)}(1-(y_j^{(i)})^2) \tag{7.62}$$

Apzīmē:

$$\delta_j^{(i)} = (z_j^{(i)} - d_j^{(i)}) z_j^{(i)} (1 - z_j^{(i)}), \tag{7.63}$$

$$\Delta_j^{(i)} = (1 - (y_j^{(i)})^2) \sum_q \delta_j^{(i)} w_{jq}^{(2)}. \tag{7.64}$$

Šos rezultātus var tālāk ievietot 7.114 un 7.115, lai iegūtu formulas, kā tiek uzlaboti svari.

7.7 Algoritma kopsavilkums

0. Patvaļīgi izvēlas svarus starp ieejām un slēpto slāni $w_{kj}^{(1)}$ un svarus starp slēpto slāni un izejām $w_{kj}^{(2)}$, jāatzīmē, ka tos nedrīkst izvēlēties visus vienādus, jo citādi visas izejas iegūs vienādas kļūdas, plašāku informāciju var iegūt [22]. Fiksē $i = 1$ un $t = 0$. Izvēlas mācīšanās ātrumu $\varphi > 0$, apstāšanās parametru, piemēram, $\varepsilon > 0$ vai iterāciju skaitu, kā arī aktivizācijas funkciju slēptajiem neironiem un izejas neironiem.

1. Aprēķina vērtības vienvirziena tīklā uz priekšu. Ieejas vektors $x^{(i)}$ tiek ievadīts tīklā, aprēķinātas vērtības $y_k^{(i)}$ un $z_j^{(i)}$ attiecīgi pēc formulām 7.02 un 7.03. Saglabā aktivizācijas funkcijas atvasinājums katrā neironā.

2. Sākas atgriezeniskā izplatīšanās uz izejas slāni. Meklē parciālos atvasinājumus $\frac{\partial E}{\partial w_{ij}^{(2)}}$. Aprēķina $\delta_j^{(i)}$ attiecīgajam modelim.

3. Atgriezeniskā izplatīšanās no izejas uz slēpto slāni. Meklē parciālos atvasinājumus $\frac{\partial E}{\partial w_{ij}^{(1)}}$. Aprēķina $\Delta_j^{(i)}$ attiecīgajam modelim. Palielina i par viens. Ja $i > N$, dodas uz 4. soli, citādi dodas uz 1. soli.

4. Izeja no neironu tīkla tiek salīdzināta ar vēlamo atbildi $d^{(i)}$, un kļūdas kopsumma E tiek aprēķināta pēc formulas 5.21. Ja $E < \varepsilon$ vai iterāciju skaits ir izpildīts, tad apstājas. Citādi uzlabo svarus no izejas slāņa uz slēpto slāni pēc formulas 7.114 un no slēptā slāņa uz ieejām pēc formulas 7.115.

   Fiksē $i = 1$ un t palielina par viens. Dodas uz 1. soli.

Piemērs

Par ieejas datiem tiks ņemts EUR/AUD valūtu kurss. Visi dati tiek normalizēti pēc formulas 6.01. Ērtības labad ieejas 7.71 tiek noapaļotas līdz diviem skaitļiem aiz komata, bet

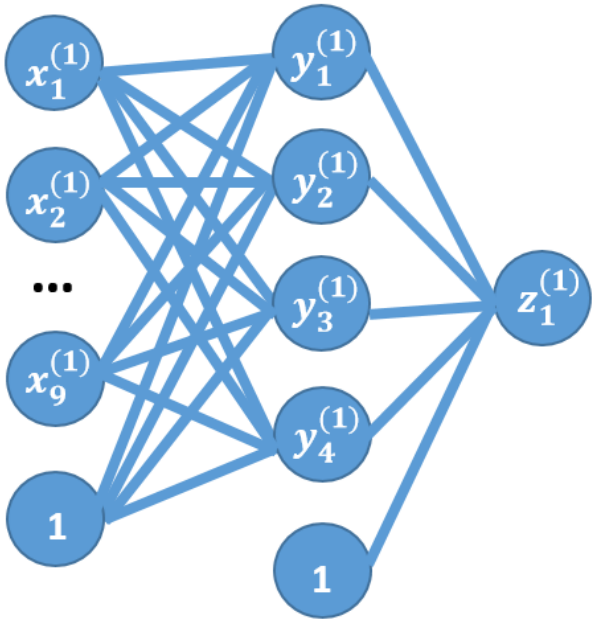
aprēķiniem tiks izmantoti četri cipari aiz komata. Kopā ir 2295 ieejas vektori. Kā jau iepriekš minēts, tad 80% datu tiek izmantoti tīkla apmācībai un 20% datu tiek izmantoti testēšanai.

$$x^{(1)} = \begin{pmatrix} 0.14 & 0.15 & 0.17 & 0.17 & 0.18 & 0.19 & 0.20 & 0.20 & 0.17 & 1 \end{pmatrix} \tag{7.71}$$

Attiecīgā vēlamā atbilde ieejas vektoram $x^{(1)}$ ir $d^{(1)} = 0.1497$.

0. solis

Tiek izvēlēts 4. modelis ar hiperboliskā tangensa aktivizācijas funkciju slēptajos neironos un identitātes aktivizācijas funkciju izejas neironos. Ieejas virsotnes ir 10, slēptie neironi ir 4 un ir viens izejas neirons, skatīt attēlu 7.3. Apmācīšanās ātrums ir $\varphi = 0.0003$. Kā apstāšanās parametrs tiek izvēlētas 20000 iterācijas. Fiksē $i = 1$ un $t = 0$. Tiek patvaļīgi izvēlēti sākuma svari, kas savieno ieejas ar slēptajiem neironiem 7.72.



Att. 7.3: **Piemēra neironu tīkla uzbūve**

$$\begin{pmatrix} -0.1501 & -0.1309 & -0.2577 & -0.1316 \\ 0.0650 & 0.0184 & -0.2514 & -0.0675 \\ -0.0843 & 0.0840 & -0.1306 & 0.0490 \\ -0.0367 & 0.1312 & -0.1230 & 0.0327 \\ -0.1743 & -0.1474 & 0.0026 & 0.0351 \\ -0.0073 & -0.0235 & -0.0558 & -0.0059 \\ 0.0448 & -0.0555 & 0.0623 & 0.0552 \\ -0.1166 & 0.1052 & -0.0867 & -0.0762 \\ 0.1302 & -0.1492 & -0.0220 & -0.0669 \\ -0.1458 & 0.0374 & -0.0124 & -0.0924 \end{pmatrix} \tag{7.72}$$

Tiek patvaļīgi izvēlēti sākuma svari, kas savieno slēptos neironus ar izejas neironiem 7.73.

$$\begin{pmatrix} -0.1149 \\ 0.0681 \\ -0.1258 \\ -0.1610 \\ -0.1346 \end{pmatrix} \tag{7.73}$$

1. solis

Tagad ir vienvirziena tīkls, tātad kustība notiek no ieejas virsotnēm uz izejas neironu. Izmanto svarus 7.72 un formulu 7.02. Tiek aprēķinātas izejas no slēptajiem neironiem $y_1^{(1)}$, $y_2^{(1)}$, $y_3^{(1)}$ un $y_4^{(1)}$.

$$\begin{aligned} y_1^{(1)} &= \sigma(w_{11}^{(1)}x_1^{(1)} + w_{21}^{(1)}x_2^{(1)} + w_{31}^{(1)}x_3^{(1)} + w_{41}^{(1)}x_4^{(1)} + w_{51}^{(1)}x_5^{(1)} + \\ &+ w_{61}^{(1)}x_6^{(1)} + w_{71}^{(1)}x_7^{(1)} + w_{81}^{(1)}x_8^{(1)} + w_{91}^{(1)}x_5^{(1)}9 + w_{101}^{(1)}x_{10}^{(1)}) = \\ &= \sigma((-0.1501) \times 0.1409 + 0.065 \times 0.1494 + (-0.0843) \times 0.173 + \\ &+ (-0.0367) \times 0.1737 + (-0.1743) \times 0.1814 + (-0.0073) \times 0.1888 + \\ &+ 0.0448 \times 0.2015 + (-0.1166) \times 0.2022 + 0.1302 \times 0.1668 + \\ &+ (-0.1458) \times 1) = \sigma(-0.204) = -0.2013 \end{aligned}$$

$$
\begin{aligned}
y_2^{(1)} &= \sigma(w_{12}^{(1)}x_1^{(1)} + w_{22}^{(1)}x_2^{(1)} + w_{32}^{(1)}x_3^{(1)} + w_{42}^{(1)}x_4^{(1)} + w_{52}^{(1)}x_5^{(1)} + \\
&+ w_{62}^{(1)}x_6^{(1)} + w_{72}^{(1)}x_7^{(1)} + w_{82}^{(1)}x_8^{(1)} + w_{92}^{(1)}x_5^{(1)}9 + w_{102}^{(1)}x_{10}^{(1)}) = \\
&= \sigma((-0.1309) \times 0.1409 + 0.0184 \times 0.1494 + 0.0840 \times 0.173 + \\
&+ 0.1312 \times 0.1737 + (-0.1474) \times 0.1814 + (-0.0235) \times 0.1888 + \\
&+ (-0.0555) \times 0.2015 + 0.1052 \times 0.2022 + (-0.1492) \times 0.1668 + \\
&+ (0.0374) \times 1) = 0.0131
\end{aligned}
$$

$$
\begin{aligned}
y_3^{(1)} &= \sigma(w_{13}^{(1)}x_1^{(1)} + w_{23}^{(1)}x_2^{(1)} + w_{33}^{(1)}x_3^{(1)} + w_{43}^{(1)}x_4^{(1)} + w_{53}^{(1)}x_5^{(1)} + \\
&+ w_{63}^{(1)}x_6^{(1)} + w_{73}^{(1)}x_7^{(1)} + w_{83}^{(1)}x_8^{(1)} + w_{93}^{(1)}x_5^{(1)}9 + w_{103}^{(1)}x_{10}^{(1)}) = \\
&= \sigma((-0.2577) \times 0.1409 + (-0.2514) \times 0.1494 + (-0.1306) \times 0.173 + \\
&+ (-0.123) \times 0.1737 + 0.0026 \times 0.1814 + (-0.0558) \times 0.1888 + \\
&+ 0.0623 \times 0.2015 + (-0.0867) \times 0.2022 + (-0.022) \times 0.1668 + \\
&+ (-0.0124) \times 1) = -0.1478
\end{aligned}
$$

$$
\begin{aligned}
y_4^{(1)} &= \sigma(w_{14}^{(1)}x_1^{(1)} + w_{24}^{(1)}x_2^{(1)} + w_{34}^{(1)}x_3^{(1)} + w_{44}^{(1)}x_4^{(1)} + w_{54}^{(1)}x_5^{(1)} + \\
&+ w_{64}^{(1)}x_6^{(1)} + w_{74}^{(1)}x_7^{(1)} + w_{84}^{(1)}x_8^{(1)} + w_{94}^{(1)}x_5^{(1)}9 + w_{104}^{(1)}x_{10}^{(1)}) = \\
&= \sigma((-0.1316) \times 0.1409 + (-0.0675) \times 0.1494 + 0.049 \times 0.173 + \\
&+ 0.0327 \times 0.1737 + 0.0351 \times 0.1814 + (-0.0059) \times 0.1888 + \\
&+ 0.0552 \times 0.2015 + (-0.0762) \times 0.2022 + (-0.0669) \times 0.1668 + \\
&+ (-0.0924) \times 1) = -0.1165
\end{aligned}
$$

Izmantojot aprēķinātās izejas $y_1^{(1)}$, $y_2^{(1)}$, $y_3^{(1)}$ un $y_4^{(1)}$ no slēptajiem neironiem, svarus 7.73 un formulu 7.03, tiek aprēķinātas izejas no visa tīkla jeb no izejas neirona $z_1^{(1)}$.

$$
\begin{aligned}
z_1^{(1)} &= \sigma(w_{11}^{(2)}y_1^{(1)} + w_{21}^{(2)}y_2^{(1)} + w_{31}^{(2)}y_3^{(1)} + w_{41}^{(2)}y_4^{(1)} + w_{51}^{(2)}y_5^{(1)} = \\
&= \sigma((-0.1149) \times (-0.2013) + 0.0881 \times 0.0131 + \\
&+ (-0.1258) \times (-0.1478) + (-0.161) \times (-0.1165)) + \\
&+ (-0.1346) \times 1) = -0.0732
\end{aligned}
$$

2. solis

Tagad sākas atgriezeniskā izplatīšanās, kad tīklā pārvietojas virzienā no izejas neirona uz ieejas virsotnēm, tiek meklēti visi parciālie atvasinājumi 7.11. Tā kā šie parciālie atvasinājumi jau ir atrasti 4. modeļa aprakstā, tad atliek tikai aprēķināt $\delta_1^{(1)}$ pēc formulas 7.44.

$$\delta_1^{(1)} = z_1^{(1)} - d_1^{(1)} = -0.0732 - 0.1497 = -0.2228$$

3. solis

Tiek meklēti visi parciālie atvasinājumi 7.12. Tā kā šie parciālie atvasinājumi jau ir atrasti 4. modeļa aprakstā, tad atliek tikai aprēķināt $\Delta_1^{(1)}$, $\Delta_2^{(1)}$, $\Delta_3^{(1)}$ un $\Delta_4^{(1)}$ pēc formulas 7.45.

$$\Delta_1^{(1)} = (1 - (y_1^{(1)})^2)\delta_1^{(1)} w_{11}^{(2)} = (1 - (-02013)^2)(-0.2228) \times (-0.1149) = 0.0246$$

$$\Delta_2^{(1)} = (1 - (y_2^{(1)})^2)\delta_1^{(1)} w_{21}^{(2)} = (1 - 0.0131^2)(-0.2228) \times 0.0681 = -0.0152$$

$$\Delta_3^{(1)} = (1 - (y_3^{(1)})^2)\delta_1^{(1)} w_{31}^{(2)} = (1 - (-0.1478)^2)(-0.2228) \times (-0.1258) = 0.0274$$

$$\Delta_4^{(1)} = (1 - (y_4^{(1)})^2)\delta_1^{(1)} w_{41}^{(2)} = (1 - (-0.1165)^2)(-0.2228) \times (-0.161) = 0.0354$$

Tagad i tiek palielināts par viens, tātad $i = 2$, kas nozīmē, ka $i < N$. Tātad jāatgriežas uz 1. soli, un šiem pirmajiem trīs soļiem vēl ir jāiziet cauri 1842 reizes, jo $N = 1843$.

Kad $i = 1844$, tad $i > N$, kas nozīmē, ka jādodas uz 4. soli.

4. solis

Aprēķina kļūdu (5.21). $E = 11.6571$, bet, tā kā apstāšanās ir uzstādīta pēc 20000 iterācijām, tad kļūdu pēc katras iterācijas ir vērts rēķināt tikai tādēļ, lai noskaidrotu, vai tīkls vispār apmācās un kļūda samazinās, jo nav uzstādīts parametrs, kad tīkls apstājas pie kāda parametra ε, ($\varepsilon < E$).

Uzlabo svarus no izejas neirona uz slēptajiem neironiem pēc formulas 7.114.

$$w_{11}^{(2)}(2) = w_{11}^{(2)}(1) - \varphi \sum_{i=1}^{1843} \delta_1^{(i)} y_1^{(i)}$$

Kad visi svari ir uzlaboti no izejas neirona uz slēptajiem neironiem, iegūst 7.74:

$$\begin{pmatrix} -0.1921 \\ 0.0558 \\ -0.2312 \\ -0.2069 \\ -0.1346 \end{pmatrix} \tag{7.74}$$

Uzlabo svarus no slēptajiem neironiem uz ieejas virsotnēm pēc formulas 7.115.

$$w_{11}^{(1)}(2) = w_{11}^{(1)}(1) - \varphi \sum_{i=1}^{1843} \Delta_1^{(i)} x_1^{(i)}$$

$$w_{12}^{(1)}(2) = w_{12}^{(1)}(1) - \varphi \sum_{i=1}^{1843} \Delta_2^{(i)} x_1^{(i)}$$

$$w_{13}^{(1)}(2) = w_{13}^{(1)}(1) - \varphi \sum_{i=1}^{1843} \Delta_3^{(i)} x_1^{(i)}$$

...

$$w_{91}^{(1)}(2) = w_{91}^{(1)}(1) - \varphi \sum_{i=1}^{1843} \Delta_1^{(i)} x_9^{(i)}$$

...

$$w_{102}^{(1)}(2) = w_{102}^{(1)}(1) - \varphi \sum_{i=1}^{1843} \Delta_2^{(i)} x_{10}^{(i)}$$

$$w_{103}^{(1)}(2) = w_{103}^{(1)}(1) - \varphi \sum_{i=1}^{1843} \Delta_3^{(i)} x_{10}^{(i)}$$

Kad visi svari ir uzlaboti no slēptajiem neironiem uz ieejas virsotnēm, iegūst 7.75:

$$\begin{pmatrix} -0.1635 & -0.1220 & -0.2700 & -0.1519 \\ 0.0517 & 0.0273 & -0.2638 & -0.0878 \\ -0.0977 & 0.0929 & -0.1430 & 0.0287 \\ -0.0500 & 0.1401 & -0.1353 & 0.0125 \\ -0.1876 & -0.1386 & -0.0097 & 0.0149 \\ -0.0206 & -0.0146 & -0.0681 & -0.0261 \\ 0.0315 & -0.0467 & 0.0500 & 0.0350 \\ -0.1298 & 0.1140 & -0.0990 & -0.0963 \\ 0.1169 & -0.1403 & -0.0343 & -0.0870 \\ -0.1458 & 0.0374 & -0.0124 & -0.0924 \end{pmatrix} \tag{7.75}$$

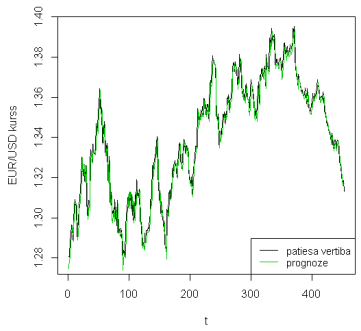
Kad šādi ir iziets cauri 20000 iterāciju, tīkls ir apmācīts, un kļūda ir $E = 0.5037$. Pēdējie uzlabotie svari no izejas neirona uz slēptajiem neironiem ir 7.76.

$$\begin{pmatrix} -0.2745 \\ 0.3875 \\ -1.1933 \\ -0.5875 \\ -0.1346 \end{pmatrix} \tag{7.76}$$

Pēdējie uzlabotie svari no slēptajiem neironiem uz ieejas virsotnēm ir 7.77.

$$\begin{pmatrix} -0.3348 & 0.0015 & -0.6103 & -0.3933 \\ 0.0913 & 0.0382 & -0.0241 & -0.0094 \\ -0.0497 & 0.1312 & 0.0254 & 0.0843 \\ 0.0055 & 0.1780 & 0.0337 & 0.0723 \\ -0.1792 & -0.0631 & -0.0098 & -0.0126 \\ 0.0135 & 0.0338 & 0.0325 & -0.0030 \\ 0.0285 & 0.0334 & 0.0192 & -0.0103 \\ -0.0755 & 0.1539 & 0.0523 & -0.0460 \\ 0.1480 & -0.0784 & 0.0233 & -0.0828 \\ -0.1458 & 0.0374 & -0.0124 & -0.0924 \end{pmatrix} \tag{7.77}$$

Tagad var sākt testēt atlikušos 452 ieejas vektorus. Tagad tiks izmantoti svari 7.76 no izejas neirona uz slēptajiem neironiem un svari 7.77 no slēptajiem neironiem uz ieejas virsotnēm. Tagad vairs nav vēlamās izejas, bet izeja, ko dos tīkls attiecīgajai ieejai, ir prognozētais datu punkts. Attēlā 7.4 ir EUR/USD prognozētā vērtība ar zaļo krāsu un ar melno krāsu ir patiesā vērtība.



Att. 7.4: **EUR/USD prognoze**

Izrēķinot kļūdas prognozei pēc formulām 6.06, 6.07 un 6.08, attiecīgi, iegūst $MAD =$

0.01, $MSE = 0.0002$ un $MAPE = 0.026$. Kā var redzēt gan attēlā 7.4, gan spriežot pēc kļūdām, iegūtais rezultāts ir samērā precīzs, bet ne pietiekami labs.

8 Rezultāti ar mākslīgo neironu tīklu katram modelim

Lai iegūtu rezultātus, aprēķiniem tika izmantotas datorpaketes *Matlab* un *R*. Mākslīgais neironu tīkls tika uzprogrammēts gan *Matlab*, gan *R*. Eksperimentu rezultātā noskaidro, ka *R* uzprogrammētā tīkla apmācība strādā daudz lēnāk, tāpēc neironu tīkla apmācībai un valūtu kursu prognozēšanai izmanto *Matlab*. Sākumā netika izmantotas iebūvētās funkcijas programmā *R*.

Parametru izvēle katram modelim

Jau iepriekš veiktajos pētījumos Zinātniskās prakses ietvaros, kur tika apskatītas loģiskās operācijas - UN, VAI un XOR, un divi modeļi - 1. modelis un 4. modelis, tika noskaidrots, ka katram modelim ir jāizvēlas cits mācīšanās ātrums un citi sākuma svari, jo katra aktivizācijas funkcija tīkla apmācīšanos ietekmē citādi.

Tas nozīmē, ka arī šeit katram modelim ir jāmeklē sākuma svari un mācīšanās ātrums. Kā sākuma parametri tiek izvēlēti autora [8] ieteiktie - divslāņu tīkls ar 21 slēpto neironu. Autors [8] neiesaka mācīšanās ātrumu, tādēļ tiek izmēģināti dažādi mācīšanās ātrumi φ.

1. modelis

Sākumā pirmajam modelim visiem eksperimentiem tika izvēlēts 21 slēptais neirons, kā arī tika izmēģināts ņemt mazāk un vairāk slēpto neironu. Ja slēpto neironu skaitu slēptajā slānī palielina līdz 25 vai samazina līdz 18, tad rezultāts neuzlabojas. Ja slēpto neironu skaits tiek izvēlēts zem 18 vai virs 25, tad neironu tīkls neapmācās un nav iespējams prognozēt valūtu kursus. Svari tiek patvaļīgi izvēlēti, bet, lai eksperimentus varētu salīdzināt, kā arī vajadzības gadījumā atkārtot vienu un to pašu gadījumu, izmanto *Matlab* iebūvēto funkciju *rng(seed)*, lai patvaļīgi izvēlētie svari vienmēr būtu vienādi. Mācīšanās ātrums arī tika izvēlēts dažāds EUR/AUD un EUR/CAD valūtu kursiem $\varphi = 0.002$, $\varphi = 0.003$ un $\varphi = 0.001$. Pārējiem valūtu kursiem tika izvēlēts $\varphi = 0.001$. Tīkls netika apmācīts līdz kādai konkrētai kļūdai 5.21, bet tika uzstādīts, ka apmācība apstājas pēc 20000 iterācijām, līdz ar to katram valūtu kursam tīkls apmācījās līdz citai kļūdai E.

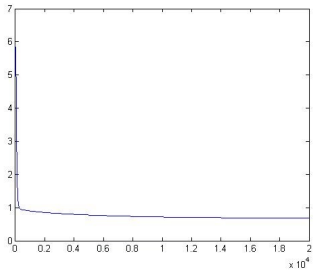
Tabulā 8.1 ir apkopoti visi 1. modeļa rezultāti. Kolonna *"E"* norāda, līdz kādai kļūdai tika apmācīts tīkls. Pēc rezultātiem var spriest, ka, piemēram, valūtu kursam EUR/AUD labākais

Tabula 8.1: **1. modeļa prognozēšanas precizitātes rezultāti valūtu kursiem, kas noteikti ar MAD, MSE un $MAPE$ pie attiecīgā mācīšanās ātruma un neironu tīkla apmācības kļūdas E**

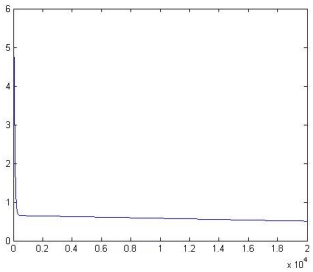
	Iterācijas	φ	MAD	MSE	$MAPE$	E
EUR/AUD	20000	0.001	0.0092	0.0002	0.0470	0.5569
kurss	20000	0.002	0.0216	0.0006	0.1054	0.5334
	20000	0.003	0.0218	0.0006	0.1067	0.5248
	5000	0.001	0.0216	0.0007	0.0986	0.6444
EUR/CAD	20000	0.001	0.0137	0.0003	0.0428	0.5466
kurss	20000	0.002	0.0133	0.0003	0.0407	0.5103
	5000	0.002	0.0149	0.0003	0.0468	0.5970
EUR/CZK	20000	0.001	0.0137	0.0003	0.0428	0.5440
kurss	5000	0.001	0.0159	0.0004	0.0302	0.6697
EUR/GBP	20000	0.001	0.0129	0.0003	0.0232	0.5074
kurss	5000	0.001	0.0158	0.0004	0.0286	0.6182
EUR/JPY	20000	0.001	0.0205	0.0006	0.0438	0.6799
kurss	5000	0.001	0.2373	0.0008	0.0524	0.7780
EUR/USD	20000	0.001	0.0133	0.0003	0.0355	0.6854
kurss	5000	0.001	0.0149	0.0004	0.0394	0.7867

mācīšanās ātrums ir $\varphi = 0.001$, bet tīkls tika apmācīts līdz kļūdai $E = 0.5569$, bet pie mācīšanās ātruma $\varphi = 0.003$ tīkls tika apmācīts līdz kļūdai $E = 0.5248$. Tomēr labāku prognozi dod tīkls, kas tika apmācīts ar ātrumu $\varphi = 0.003$, tas nozīmē, ka ir arī robeža, kad tīkls ir pārmācīts un prognozes būs sliktākas. Tādēļ tika nolemts, ka jāpārbauda, vai neironu tīkls, kas apmācīts līdz lielākai kļūdai E, vai šāds tīkls neprognozē precīzāk. Tādēļ neironu tīklam tika uzstādīts, ka apmācīšanās apstājas, kad ir iziets cauri 5000 iterācijām, līdz ar to iegūst, ka kļūda ir lielāka par iepriekš iegūtajām. Šie rezultāti ir apkopoti tabulā 8.1. No iegūtajiem rezultātiem var spriest, ka mazāk apmācīts tīkls šajā gadījumā nedod labākus prognozēšanas rezultātus.

Attēlos 8.1 ir parādīts, kā mainās kļūda pēc katras iterācijas valūtu kursiem EUR/JPY un EUR/GBP. Var redzēt, ka sākumā kļūda E samazinās strauji, bet pēc tam ļoti lēni, kas ir skaidrojams ar mazo mācīšanās ātrumu φ.



(a) **EUR/JPY,** $\varphi = 0.001$



(b) **EUR/GBP,** $\varphi = 0.001$

Att. 8.1: **Attēlos (a) un (b) uzzīmēti 1. modeļa kļūdu grafiki**

2. modelis

Arī otrajam modelim visiem eksperimentiem tika izvēlēts 21 slēptais neirons, kā arī tika izmēģināts ņemt mazāk un vairāk slēpto neironu. Iegūtie rezultāti ir tādi paši kā 1. modelim, mainot slēpto neironu skaitu slēptajā slānī, tādēļ arī tiek atstāts 21 slēptais neirons, kā to iesaka [8]. Arī šeit sākuma svari tiek izvēlēti patvaļīgi, bet tāpat kā pirmajam modelim izmanto *Matlab* iebūvēto funkciju *rng(seed)*, lai patvaļīgi izvēlētie svari vienmēr būtu vienādi. Mācīšanās ātrums tika izvēlēts $\varphi = 0.0004$ un $\varphi = 0.0006$. Tīkls netika apmācīts līdz kādai konkrētai kļūdai 5.21, bet tika uzstādīts, ka apmācība apstājas pēc 20000 iterācijām, līdz ar to katram valūtu kursam tīkls apmācījās līdz citai kļūdai 5.21.

Tabulā 8.2 ir apkopoti visi 2. modeļa rezultāti. Kolonna *"E"* norāda, līdz kādai kļūdai tika apmācīts tīkls. Pēc rezultātiem var spriest, ka EUR/AUD un EUR/JPY valūtu kursiem

Tabula 8.2: **2. modeļa prognozēšanas precizitātes rezultāti valūtu kursiem, kas noteikti ar MAD, MSE un $MAPE$ pie attiecīgā mācīšanās ātruma un neironu tīkla apmācības kļūdas E**

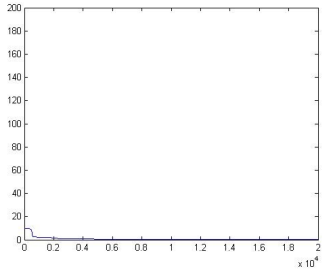
	Iterācijas	φ	MAD	MSE	$MAPE$	E
EUR/AUD	20000	0.0004	0.0081	0.0001	0.0367	0.3223
kurss	20000	0.0006	0.0075	0.00009	0.0345	0.3069
	5000	0.0006	0.0126	0.0003	0.0579	0.5428
EUR/CAD	20000	0.0004	0.0098	0.0002	0.0317	0.4180
kurss	20000	0.0006	0.0099	0.0002	0.0320	0.4105
	5000	0.0004	0.0133	0.0003	0.0433	0.5427
EUR/CZK	20000	0.0004	0.0065	0.0001	0.0140	0.3768
kurss	20000	0.0006	0.0067	0.0001	0.0143	0.3700
	5000	0.0004	0.0116	0.0003	0.0243	0.5246
EUR/GBP	20000	0.0004	0.0083	0.0001	0.0151	0.3349
kurss	20000	0.0006	0.0085	0.0001	0.0153	0.3302
	5000	0.0004	0.0166	0.0004	0.0315	0.5326
EUR/JPY	20000	0.0004	0.0097	0.0002	0.0255	0.4662
kurss	20000	0.0006	0.0097	0.0002	0.0256	0.4586
	5000	0.0004	0.0116	0.0002	0.0272	0.4127
EUR/USD	20000	0.0004	0.0091	0.0002	0.0213	0.3327
kurss	20000	0.0006	0.0090	0.0002	0.0210	0.3237
	5000	0.0006	0.0132	0.0003	0.0343	0.5810

labāko prognozēšanas rezultātu dod pie mācīšanās ātruma, $\varphi = 0.0006$, bet pārējiem valūtu kursiem labāku prognozēšanas rezultātu dod mācīšanās ātrums $\varphi = 0.0004$. Šeit ļoti izteikti var redzēt, ka tīklu nevajadzētu apmācīt līdz pārāk mazai kļūdai, jo visi tīkli vismazāko kļūdu E sasniedza pie mācīšanās ātruma $\varphi = 0.0006$, bet valūtu kursiem EUR/CAD, EUR/CZK, EUR/GBP un EUR/USD labāks prognozēšanas rezultāts ir pie lielākas kļūdas E.

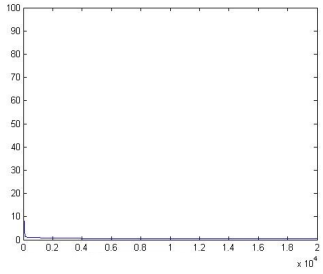
Arī šim modelim tika nolemts, ka tīkls ir jāapmāca līdz lielākai kļūdai E, kā apstāšanās parametrs tika izvēlēts 5000 iterācijas. Kā var spriest no tabulas 8.2, tad iegūtie prognozēšanas

rezultāti nav precīzāki par iepriekš iegūtajiem.

Attēlos 8.2 ir parādīts, kā mainās kļūda pēc katras iterācijas valūtu kursiem EUR/CAD un EUR/AUD. Var redzēt, ka sākumā kļūda E samazinās strauji, bet pēc tam ļoti lēni, kas ir skaidrojams ar mazo mācīšanās ātrumu φ.



(a) **EUR/CAD,** $\varphi = 0.0006$



(b) **EUR/AUD,** $\varphi = 0.004$

Att. 8.2: **Attēlos (a) un (b) uzzīmēti 2. modeļa kļūdu grafiki**

Šis ir vienīgais modelis, kuram rezultātus ir iespējams iegūt arī ar programmā *R* iebūvēto funkciju *nnet*. Tabulā 8.3 ir apkopoti rezultāti, kas tika iegūti ar iebūvēto funkciju. Iegūtie

Tabula 8.3: **2. modeļa prognozēšanas precizitātes rezultāti valūtu kursiem ar iebūvēto funkciju *nnet*, kas noteikti ar MAD, MSE un $MAPE$ pie attiecīgā mācīšanās ātruma un neironu tīkla apmācības kļūdas E**

	Iterācijas	φ	MAD	MSE	$MAPE$
EUR/AUD kurss	20000	0.0001	0.0065	0.00007	0.0046
EUR/CAD kurss	20000	0.0001	0.0048	0.000004	0.0034
EUR/CZK kurss	20000	0.0001	0.0127	0.0004	0.0270
EUR/GBP kurss	20000	0.0001	0.0027	0.00001	0.0032
EUR/JPY kurss	20000	0.0001	0.0089	0.0001	0.0207
EUR/USD kurss	20000	0.0001	0.0004	0.00003	0.003

rezultāti ir nedaudz labāki par programmētajiem rezultātiem, izņemot valūtu kursam EUR/CZK. Tas varētu būt skaidrojams ar to, ka sākuma svari tika izvēlēti labāki un tīkls tika apmācīts labāk, bet diemžēl šī iebūvētā funkcija nedod iespēju redzēt, līdz kādai kļūdai E tīkls apmācās.

3. modelis

Arī trešajam modelim visiem eksperimentiem tika izvēlēts 21 slēptais neirons, kā arī tika izmēģināts ņemt mazāk un vairāk slēpto neironu. Iegūtie rezultāti ir tādi paši kā 1. modelim, mainot slēpto neironu skaitu slēptajā slānī, tādēļ arī tiek atstāts 21 slēptais neirons, kā to iesaka [8]. Arī šeit sākuma svari tiek izvēlēti patvaļīgi, bet tāpat kā pirmajam modelim izmanto *Matlab* iebūvēto funkciju *rng(seed)*, lai patvaļīgi izvēlētie svari vienmēr būtu vienādi. Mācīšanās ātrums arī tika izvēlēts $\varphi = 0.0004$ un $\varphi = 0.0006$. Tīkls netika apmācīts līdz kādai konkrētai kļūdai 5.21, bet tika uzstādīts, ka apmācība apstājas pēc 20000 iterācijām, līdz ar to katram valūtu kursam tīkls apmācījās līdz citai kļūdai 5.21.

Tabulā 8.4 ir apkopoti visi 3. modeļa rezultāti. Kolonna *"E"* norāda, līdz kādai kļūdai tika apmācīts tīkls. No rezultātiem var redzēt, ka vislabāk tīkls apmācījās pie mācīšanās ātruma

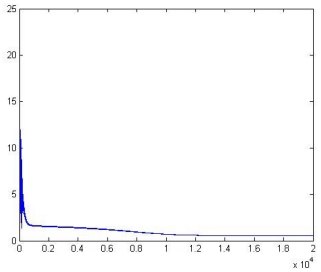
Tabula 8.4: **3. modeļa prognozēšanas precizitātes rezultāti valūtu kursiem, kas noteikti ar MAD, MSE un $MAPE$ pie attiecīgā mācīšanās ātruma un neironu tīkla apmācības kļūdas E**

	Iterācijas	φ	MAD	MSE	$MAPE$	E
EUR/AUD	20000	0.0004	0.0176	0.0004	0.0754	0.6538
kurss	20000	0.0006	0.0103	0.0002	0.0560	0.4834
	5000	0.0006	0.0414	0.0025	0.2654	1.2481
EUR/CAD	20000	0.0004	0.0205	0.0006	0.0608	0.6084
kurss	20000	0.0006	0.0145	0.0003	0.0433	0.5298
	5000	0.0006	0.0403	0.0023	0.1569	1.3445
EUR/CZK	20000	0.0004	0.0089	0.0002	0.0186	0.7273
kurss	20000	0.0006	0.0139	0.0003	0.0258	0.5171
	5000	0.0004	0.0126	0.0003	0.0269	1.2281
EUR/GBP	20000	0.0004	0.0299	0.0011	0.0571	0.6687
kurss	20000	0.0006	0.0091	0.0001	0.0165	0.4224
	5000	0.0006	0.0147	0.0004	0.0257	1.0978
EUR/JPY	20000	0.0004	0.0168	0.0004	0.0004	0.5797
kurss	20000	0.0006	0.0165	0.0004	0.0345	0.5522
	5000	0.0004	0.0739	0.0059	0.1551	1.4091
EUR/USD	20000	0.0004	0.0141	0.0003	0.0356	0.6577
kurss	20000	0.0006	0.0138	0.0003	0.0374	0.6352
	5000	0.0006	0.0266	0.0010	0.0713	1.2397

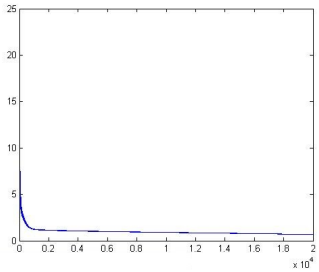
$\varphi = 0.006$, bet valūtu kursi EUR/CZK un EUR/USD labākus prognozēšanas rezultātus uzrāda pie mācīšanās ātruma $\varphi = 0.0004$. Pārējiem valūtu kursiem labāku prognozēšanas precizitāte ir pie mācīšanās ātruma $\varphi = 0.0006$.

Arī šim modelim tika nolemts, ka tīkls ir jāapmāca līdz lielākai kļūdai E, kā apstāšanās parametrs tika izvēlēts 5000 iterācijas. Kā var spriest no tabulas 8.4, tad iegūtie prognozēšanas rezultāti nav precīzāki par iepriekš iegūtajiem.

Attēlos 8.3 ir parādīts, kā mainās kļūda pēc katras iterācijas valūtu kursiem EUR/CAD un EUR/GBP. Arī šiem grafikiem var redzēt, ka sākumā kļūda E samazinās strauji, bet pēc tam ļoti lēni, kas ir skaidrojams ar mazo mācīšanās ātrumu φ.



(a) **EUR/CAD,** $\varphi = 0.0006$



(b) **EUR/GBP,** $\varphi = 0.004$

Att. 8.3: **Attēlos (a) un (b) uzzīmēti 3. modeļa kļūdu grafiki**

4. modelis

Ceturtajam modelim visiem eksperimentiem tika izvēlēts 21 slēptais neirons. Slēpto neironu skaits slēptajā slānī arī tiek mainīts, kā jau var redzēt piemērā, kas aplūkots 7.7 apakšnodaļā, tīkls apmācās arī pie četriem slēptajiem neironiem, bet tas nedod labākus prognozēšanas rezultātus, tādēļ arī šeit paliek pie 21 slēptā neirona slēptajā slānī. Šis modelis apskatītajā literatūrā ir atzīts par labāko, tādēļ tas arī tiek apskatīts plašāk. Arī šeit sākuma svari tiek izvēlēti patvaļīgi, bet tāpat kā pirmajam modelim izmanto *Matlab* iebūvēto funkciju *rng(seed)*, lai patvaļīgi izvēlētie svari vienmēr būtu vienādi. Mācīšanās ātrums tika izvēlēts $\varphi = 0.0004$ un $\varphi = 0.0003$ visiem valūtu kursiem, bet valūtu kursiem EUR/CZK, EUR/JPY un EUR/USD tika izvēlēts arī mācīšanās ātrums $\varphi = 0.0002$ un $\varphi = 0.0001$. Tīkls netika apmācīts līdz kādai konkrētai kļūdai 5.21, bet tika uzstādīts, ka apmācība apstājas pēc 20000 iterācijām, līdz ar to katram valūtu kursam tīkls apmācījās līdz citai kļūdai 5.21.

Tabulā 8.5 ir apkopoti visi 4. modeļa rezultāti. Kolonna *"E"* norāda, līdz kādai kļūdai tika apmācīts tīkls. Pēc rezultātiem var spriest, ka visi valūtu kursi izņemot EUR/JPY vislabāk

Tabula 8.5: **4. modeļa prognozēšanas precizitātes rezultāti valūtu kursiem, kas noteikti ar MAD, MSE un $MAPE$ pie attiecīgā mācīšanās ātruma un neironu tīkla apmācības kļūdas E**

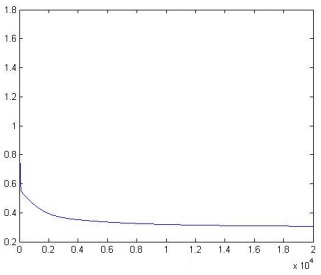
	Iterācijas	φ	MAD	MSE	$MAPE$	E
EUR/AUD	20000	0.0004	0.0072	0.0001	0.0337	0.2982
kurss	20000	0.0003	0.0072	0.0001	0.0335	0.2961
	5000	0.0003	0.0082	0.0001	0.0378	0.3329
EUR/CAD	20000	0.0004	0.0093	0.0016	0.0301	0.4049
kurss	20000	0.0003	0.0093	0.0002	0.0299	0.3982
	5000	0.0003	0.0100	0.0002	0.0325	0.4251
EUR/CZK	20000	0.0004	0.0074	0.0001	0.0153	0.3795
kurss	20000	0.0003	0.0062	0.0001	0.0135	0.3600
	20000	0.0002	0.0063	0.0001	0.0136	0.3768
	5000	0.0003	0.0066	0.0001	0.0143	0.3876
EUR/GBP	20000	0.0004	0.0075	0.0001	0.0135	0.3070
kurss	20000	0.0003	0.0077	0.0001	0.0139	0.3066
	5000	0.0004	0.0084	0.0001	0.0153	0.3322
EUR/JPY	20000	0.0004	0.0921	0.0002	0.0216	0.3406
kurss	20000	0.0003	0.0090	0.0002	0.0209	0.3239
	20000	0.0002	0.0087	0.0002	0.0204	0.3186
	5000	0.0002	0.0101	0.0002	0.0236	0.3644
EUR/USD	20000	0.0004	0.0096	0.0002	0.0253	0.4516
kurss	20000	0.0003	0.0096	0.0002	0.0255	0.4483
	20000	0.0001	0.0095	0.0002	0.0252	0.4580
	5000	0.0001	0.0109	0.0002	0.0288	0.5210

tika apmācīti pie mācīšanās ātruma $\varphi = 0.0003$, kamēr valūtu kursa EUR/JPY tīkls tika apmācīts vislabāk pie mācīšanās ātruma $\varphi = 0.0002$. Valūtu kursi EUR/AUD, EUR/CAD un EUR/CZK

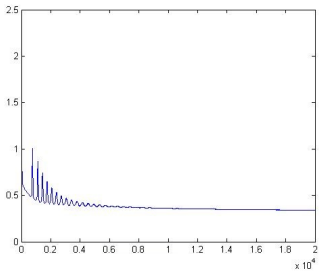
vislabākās prognozes uzrāda pie mācīšanās ātruma $\varphi = 0.0003$, EUR/GBP pie $\varphi = 0.0004$, EUR/JPY pie $\varphi = 0.0002$ un EUR/USD pie $\varphi = 0.0001$. Atkal tiek novērots, ja tīkls tiek apmācīts līdz pārāk mazai kļūdai, tad tas vairs neprognozē precīzi.

Arī šim modelim tika nolemts, ka tīkls ir jāapmāca līdz lielākai kļūdai E, kā apstāšanās parametrs tika izvēlēts 5000 iterācijas. Kā var spriest no tabulas 8.5, tad iegūtie prognozēšanas rezultāti nav precīzāki par iepriekš iegūtajiem.

Attēlos 8.4 ir parādīts, kā mainās kļūda pēc katras iterācijas valūtu kursiem EUR/GBP un EUR/JPY. Attēlā (a) kļūdu grafikam var redzēt, ka sākumā kļūda E samazinās strauji, bet pēc tam ļoti lēni, kas ir skaidrojams ar mazo mācīšanās ātrumu φ. Attēlā (b) sākumā kļūda E lēkā no lielākas uz mazāku, kas ir skaidrojams ar to, ka mācīšanās ātrums φ ir izvelēts liels, tomēr pēc aptuveni 800 iterācijām kļūda beidz lēkāt, bet lēni samazinās.



(a) **EUR/GBP,** $\varphi = 0.0003$



(b) **EUR/JPY,** $\varphi = 0.004$

Att. 8.4: **Attēlos (a) un (b) uzzīmēti 4. modeļa kļūdu grafiki**

5. modelis

Arī piektajam modelim visiem eksperimentiem tika izvēlēts 21 slēptais neirons, kā arī tika izmēģināts ņemt mazāk un vairāk slēpto neironu, bet tas nedeva labāku rezultātu. Šis ir vienīgais modelis, kuram netika atrasti piemēroti parametri, lai prognozes būtu precīzas. Arī šeit sākuma svari tiek izvēlēti patvaļīgi, bet tāpat kā pirmajam modelim izmanto *Matlab* iebūvēto funkciju *rng(seed)*, lai patvaļīgi izvēlētie svari vienmēr būtu vienādi. Tīkls netika apmācīts līdz kādai konkrētai kļūdai 5.21, bet tika uzstādīts, ka apmācība apstājas pēc 20000 iterācijām, līdz ar to katram valūtu kursam tīkls apmācījās līdz citai kļūdai 5.21.

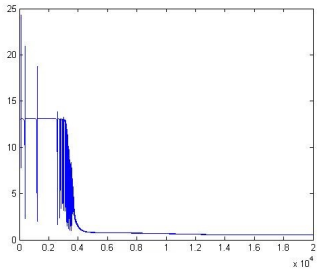
Tabulā 8.6 ir apkopoti visi 5. modeļa rezultāti, kur mācīšanās ātrums ir $\varphi = 0.006$. Kolonna *"E"* norāda, līdz kādai kļūdai tika apmācīts tīkls. Pēc rezultātiem var spriest, kā jau tas tika minēts, ka nav atrasti piemēroti parametri modelim, lai tas varētu prognozēt.

Arī šim modelim tika nolemts, ka ir tīkls jāapmāca līdz lielākai kļūdai E, kā apstāšanās parametrs tika izvēlēts 5000 iterācijas. Kā var spriest no tabulas 8.5, tad iegūtie prognozēšanas rezultāti vēl joprojām nav precīzi, lai arī valūtu kursiem EUR/CZK, EUR/GBP un EUR/JPY tagad ir nedaudz precīzākas prognozes.

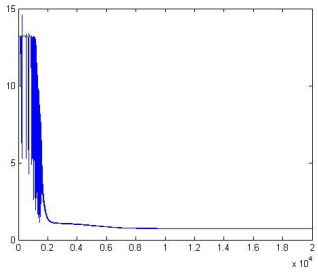
Tabula 8.6: **5. modeļa prognozēšanas precizitātes rezultāti valūtu kursiem, kas noteikti ar MAD, MSE un $MAPE$ pie attiecīgā mācīšanās ātruma un neironu tīkla apmācības kļūdas E**

	Iterācijas	φ	MAD	MSE	$MAPE$	E
EUR/AUD	20000	0.0006	0.6956	0.4946	3.6623	0.5194
kurss	5000	0.0006	0.6703	0.4580	3.5146	0.7933
EUR/CAD	20000	0.0006	0.6699	0.4675	2.3430	0.4971
kurss	5000	0.0006	0.6886	0.4950	2.4129	0.6912
EUR/CZK	20000	0.0006	0.4771	0.2416	1.0449	0.5974
kurss	5000	0.0006	0.4655	0.2319	1.0229	0.7515
EUR/GBP	20000	0.0006	0.5238	0.2805	0.9799	0.8867
kurss	5000	0.0006	0.2313	0.0594	0.4422	3.3906
EUR/JPY	20000	0.0006	0.4891	0.2494	1.1667	0.6640
kurss	5000	0.0006	0.4739	0.2355	1.1287	0.8842
EUR/USD	20000	0.0006	0.6661	0.4477	1.7426	0.7196
kurss	5000	0.0006	0.6857	0.4745	1.7940	0.9474

Attēlos 8.5 ir parādīts, kā mainās kļūda pēc katras iterācijas valūtu kursiem EUR/AUD un EUR/USD. Attēlā (a) un (b) sākumā kļūda E lēkā no lielākas uz mazāku, kas ir skaidrojams ar to, ka mācīšanās ātrums φ ir izvelēts liels, tomēr pēc aptuveni 400 iterācijām kļūda beidz lēkāt, bet lēni samazinās. Tomēr, kā jau tika teikts, tīkls netiek apmācīts pietiekami, lai tas varētu prognozēt.



(a) **EUR/AUD,** $\varphi = 0.0006$



(b) **EUR/USD,** $\varphi = 0.0006$

Att. 8.5: **Attēlos (a) un (b) uzzīmēti 5. modeļa kļūdu grafiki**

6. modelis

Arī sestajam modelim visiem eksperimentiem tika izvēlēts 21 slēptais neirons, kā arī tika izmēģināts ņemt mazāk un vairāk slēpto neironu. Iegūtie rezultāti ir tādi paši kā 1. modelim, mainot slēpto neironu skaitu slēptajā slānī, tādēļ arī tiek atstāts 21 slēptais neirons, kā to iesaka [8]. Arī šeit sākuma svari tiek izvēlēti patvaļīgi, bet tāpat kā pirmajam modelim izmanto *Matlab* iebūvēto funkciju *rng(seed)*, lai patvaļīgi izvēlētie svari vienmēr būtu vienādi. Mācīšanās ātrums tika izvēlēts $\varphi = 0.001$ un $\varphi = 0.0001$. Tīkls netika apmācīts līdz kādai konkrētai kļūdai 5.21, bet tika uzstādīts, ka apmācība apstājas pēc 20000 iterācijām, līdz ar to katram valūtu kursam tīkls apmācījās līdz citai kļūdai 5.21.

Tabulā 8.7 ir apkopoti visi 6. modeļa rezultāti. Kolonna *"E"* norāda, līdz kādai kļūdai tika apmācīts tīkls. Pēc rezultātiem var spriest, ka tīkls vislabāk apmācās pie mācīšanās ātruma

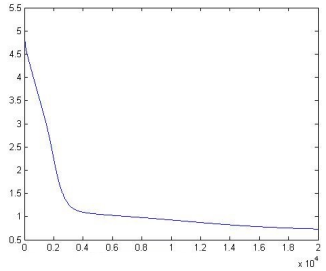
Tabula 8.7: **6. modeļa prognozēšanas precizitātes rezultāti valūtu kursiem, kas noteikti ar MAD, MSE un $MAPE$ pie attiecīgā mācīšanās ātruma un neironu tīkla apmācības kļūdas E**

	Iterācijas	φ	MAD	MSE	$MAPE$	E
EUR/AUD	20000	0.001	0.0117	0.0002	0.0530	0.5235
kurss	20000	0.0001	0.0376	0.0017	0.2106	0.7537
	5000	0.001	0.0237	0.0009	0.1593	0.9172
EUR/CAD	20000	0.001	0.0138	0.0003	0.0426	0.5116
kurss	20000	0.0001	0.0166	0.0004	0.0569	0.7312
	5000	0.001	0.0195	0.0005	0.0612	0.7467
EUR/CZK	20000	0.001	0.0136	0.0003	0.0315	0.5836
kurss	20000	0.0001	0.0184	0.0005	0.0369	0.6262
	5000	0.001	0.0165	0.0004	0.0322	0.5731
EUR/GBP	20000	0.001	0.0133	0.0002	0.0243	0.4194
kurss	20000	0.0001	0.0123	0.0002	0.2234	0.6100
	5000	0.0001	0.0156	0.0004	0.0293	1.1810
EUR/JPY	20000	0.001	0.0268	0.0009	0.0526	0.6514
kurss	20000	0.0001	0.0217	0.0007	0.0505	0.8795
	5000	0.0001	0.0277	0.0012	0.0655	1.0938
EUR/USD	20000	0.001	0.0107	0.0002	0.0278	0.6301
kurss	20000	0.0001	0.0169	0.0004	0.0442	0.8668
	5000	0.001	0.0221	0.0007	0.0534	0.9065

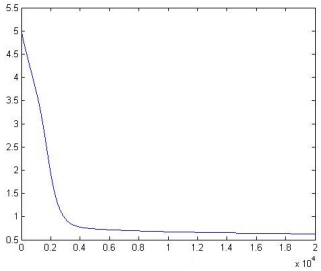
$\varphi = 0.001$. Tomēr valūtu kursi EUR/GBP un EUR/JPY labākus prognozes rezultātus uzrāda pie mācīšanās ātruma $\varphi = 0.0001$.

Arī šim modelim tika nolemts, ka ir tīkls jāapmāca līdz lielākai kļūdai E, kā apstāšanās parametrs tika izvēlēts 5000 iterācijas. Kā var spriest no tabulas 8.4, tad iegūtie prognozēšanas rezultāti nav precīzāki par iepriekš iegūtajiem.

Attēlos 8.6 ir parādīts, kā mainās kļūda pēc katras iterācijas valūtu kursiem EUR/CAD un EUR/CZK. Arī šiem grafikiem var redzēt, ka sākumā kļūda E samazinās strauji, bet pēc tam ļoti lēni, kas ir skaidrojams ar mazo mācīšanās ātrumu φ.



(a) **EUR/CAD,** $\varphi = 0.0001$



(b) **EUR/CZK,** $\varphi = 0.0001$

Att. 8.6: **Attēlos (a) un (b) uzzīmēti 6. modeļa kļūdu grafiki**

8.1 Mākslīgo neironu tīklu rezultātu apkopojums

Šajā apakšnodaļā tiek apkopoti visi vislabākie rezultāti, balstoties uz kļūdu novērtējumiem MAD, MSE un $MAPE$, kas tika iegūti katrai laikrindai ar katru modeli. Tabulā 8.8 ir apkopoti katra modeļa vsmazakās kļūdas MAD, MSE un $MAPE$. No tabulas 8.8 var secināt, ka

Tabula 8.8: **Mākslīgo neironu tīklu prognozēšanas precizitātes novērtēšana, vismazāko kļūdu apkopojums katram modelim**

Modelis	MAD	MSE	$MAPE$	Modelis	MAD	MSE	$MAPE$
	EUR/AUD kurss				EUR/CAD kurss		
1.	0.0092	0.0002	0.0470	1.	0.0133	0.0003	0.0407
2.	0.0065	0.00007	0.0046	2.	0.0048	0.00004	0.0034
3.	0.0103	0.0002	0.0560	3.	0.0145	0.0003	0.0433
4.	0.0072	0.0001	0.033	4.	0.0093	0.0002	0.0299
5.	0.6956	0.4946	3.6624	5.	0.6699	0.4675	2.3430
6.	0.0117	0.0002	0.0530	6.	0.0138	0.0003	0.0426
	EUR/CZK kurss				EUR/GBP kurss		
1.	0.0137	0.0003	0.0428	1.	0.0129	0.0003	0.0232
2.	0.0065	0.0001	0.0140	2.	0.0027	0.00001	0.0032
3.	0.0089	0.0002	0.0186	3.	0.0091	0.0001	0.0165
4.	0.0062	0.0001	0.0135	4.	0.0075	0.0001	0.0135
5.	0.4655	0.2319	1.0229	5.	0.2313	0.0594	0.4422
6.	0.0136	0.0003	0.0315	6.	0.0123	0.0002	0.2234
	EUR/JPY kurss				EUR/USD kurss		
1.	0.0205	0.0006	0.0438	1.	0.0133	0.0003	0.0355
2.	0.0089	0.000	0.0207	2.	0.0004	0.00003	0.003
3.	0.0165	0.0004	0.0345	3.	0.0141	0.0003	0.0356
4.	0.0087	0.0002	0.0204	4.	0.0095	0.0002	0.0252
5.	0.4739	0.2355	1.1287	5.	0.6661	0.4477	1.7426
6.	0.0217	0.0007	0.0505	6.	0.0107	0.0002	0.0278

vislabāk izdevās prognozēt valūtu kursu EUR/USD un vissliktāk tika prognozēts valūtu kurss EUR/JPY, balstoties uz kļūdu novērtejumiem MAD, MSE un $MAPE$. Vispiemērotākais modelis katram valūtu kursam ir:

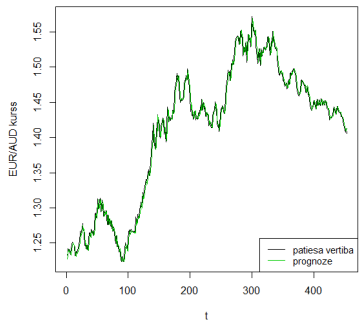
1. EUR/AUD - 2. modelis, kur $\varphi = 0.0006$ un $E = 0.3069$,
2. EUR/CAD - 2. modelis, kur $\varphi = 0.0004$ un $E = 0.418$,
3. EUR/CZK - 4. modelis, kur $\varphi = 0.0003$ un $E = 0.36$,
4. EUR/GBP - 2. modelis, kur $\varphi = 0.0004$ un $E = 0.3349$,
5. EUR/JPY - 4. modelis, kur $\varphi = 0.0002$ un $E = 0.3186$,
6. EUR/USD - 2. modelis, kur $\varphi = 0.0006$ un $E = 0.3237$.

Tabulā 8.9 tiek apkopoti iegūtie rezultāti. Var redzēt, ka labākais modelis ir 2., kur slēptajos neironos aktivizācijas funkcija ir loģistiskais sigmoīds un izejas neironā - identitātes funkcija. Otrais labākais modelis ir 4., kur aktivizācijas funkcija slēptajos neironos ir hiperboliskais tangens, bet izejas neironam - identitātes funkcija. 3. modelis un 1. modelis arī prognozē samērā precīzi.

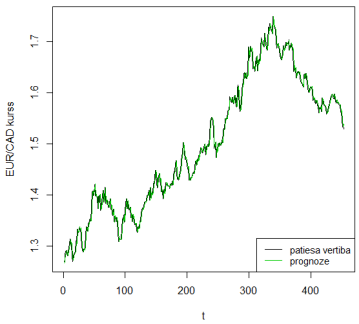
Tabula 8.9: **Modeļu sarindoti pa vietām no 1. vietas līdz 6.vietai pēc to prognozēšanas precizitātes, balstoties uz kļūdu novērtējumiem** MAD**,** MSE **un** $MAPE$

Valūtu kurss	1.	2.	3.	4.	5.	6
EUR/AUD	2.	4.	1.	3.	6.	5.
EUR/CAD	2.	4.	1.	6.	3.	5.
EUR/CZK	4.	2.	3.	6.	1.	5.
EUR/GBP	2.	4.	3.	6.	1.	5.
EUR/JPY	4.	2.	3.	1.	6.	5.
EUR/USD	2.	4.	6.	1.	3.	5.

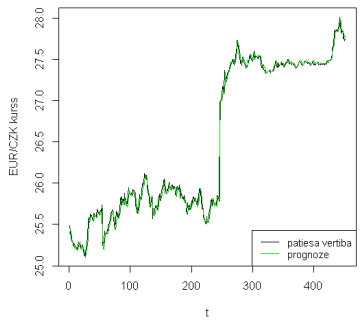
Attēlos 8.7 ar zaļo līniju ir parādītas prognozes ar 2. modeli, bet ar melno līniju ir patiesās vērtības. Attēlos 8.8 ar zaļo līniju ir parādītas prognozes ar 4. modeli, bet ar melno līniju ir patiesās vērtības.



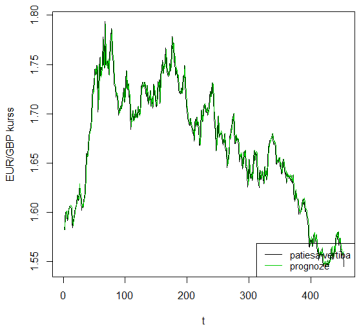
(a) **EUR/AUD prognoze**



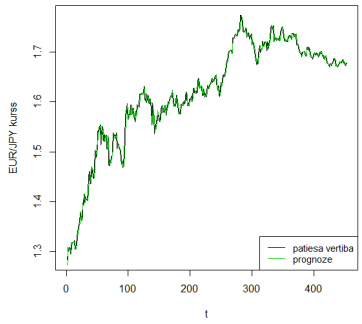
(b) **EUR/CAD prognoze**



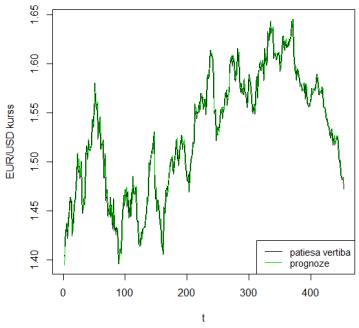
(c) **EUR/CZK prognoze**



(d) **EUR/GBP prognoze**

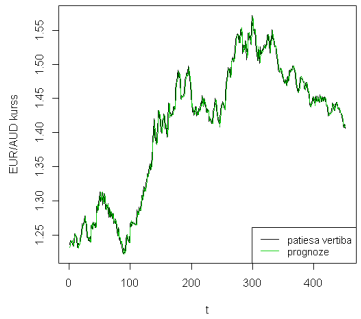


(e) **EUR/JPY prognoze**

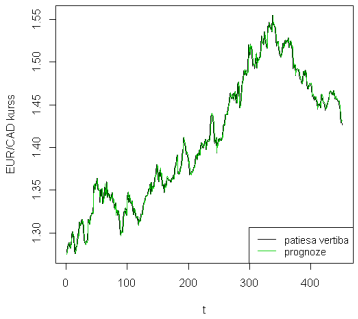


(f) **EUR/USD prognoze**

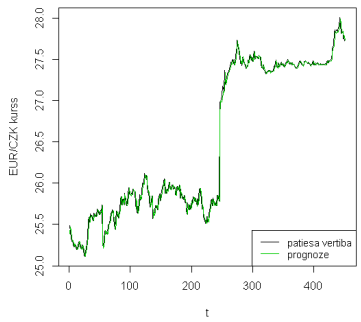
Att. 8.7: **Attēlos (a) - (f) uzzīmēti 2. modeļa vislabāk, spriežot pēc** MAD, MSE, $MAPE$, **prognozēto valūtu kursu prognožu un patieso vērtību grafiki**



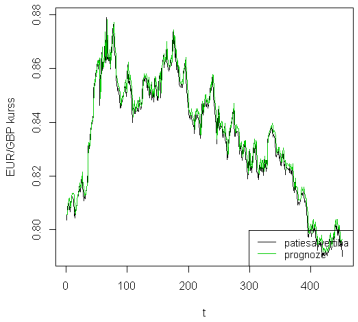
(a) **EUR/AUD prognoze**



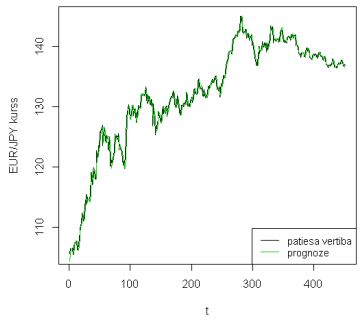
(b) **EUR/CAD prognoze**



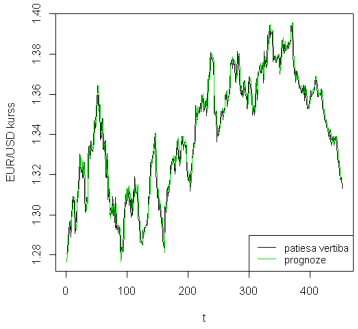
(c) **EUR/CZK prognoze**



(d) **EUR/GBP prognoze**



(e) **EUR/JPY prognoze**



(f) **EUR/USD prognoze**

Att. 8.8: **Attēlos (a) - (f) uzzīmēti 4. modeļa vislabāk, spriežot pēc** MAD**,** MSE**,** $MAPE$**, prognozēto valūtu kursu prognožu un patieso vērtību grafiki**

9 $ARIMA$ modeļi

Plašāk par prognozēšanu ar $ARIMA$ modeļiem var iepazīties [9] un [4]. Šajā darbā tiks iekļautas tikai svarīgākās definīcijas un tiks konspektīvi aprakstīta prognozēšana ar $ARIMA$ modeļiem. Boksa Dženkinsa metodoloģija prognozēšanai ietver – datu pirmapstrādi (datu transformāciju un diferencēšanu), piemērota $ARIMA$ modeļa identificēšana, $ARIMA$ pielāgošana datiem, pielāgotā $ARIMA$ modeļa pārbaude un prognozēšana ar modeli.

Svarīgākās definīcijas

Definīcija 11. [9] Par gadījuma procesu $\{x_t(\omega), t \in T\}$ sauc gadījuma lielumu saimi $x_t = x(t, \omega)$, kuri ir uzdoti varbūtības telpā $\langle \Omega, F, P \rangle$ un ir atkarīgi no parametra $t \in T$.

Definīcija 12. [9] Gadījuma procesu $\{x_t\}$ sauc par (p, d, q) kārtas autoregresīvo integrēto slīdošo vidējo procesu $ARIMA(p.d.q)$, ja $w_t = \Delta^d x_t$ ir $ARIMA(p, d, q)$ process.

Definīcija 13. [9] Gadījuma procesu $\{x_t\}, t = 0, 1, ...$ sauc par jaukto (p, q) kārtas autoregresīvo slīdošā vidējā $(ARMA(p, q))$ procesu, ja tas apmierina vienādojumu: $x_t = a_0 + a_1 x_{t-1} + ... + a_p x_{t-p} + \mu_0 + b_1 \varepsilon_{t-1} + ... + b_q \varepsilon_{t-q} + \varepsilon_t$, kur ε_t baltais troksnis ar $E\varepsilon_t = 0$ un $D\varepsilon_t = \sigma^2$.

Definīcija 14. [9] Gadījuma procesu $\{x_t\}_{t=-\infty}^{\infty}$ sauc par q-tās kārtas slīdošā vidējā procesu $MA(q)$, ja tas apmierina vienādojumu: $x_t = \mu + b_1 \varepsilon_{t-1} + b_2 \varepsilon_{t-2} + ... + b_q \varepsilon_{t-q} + \varepsilon_t$, kur ε_t baltais troksnis ar $E\varepsilon_t = 0$ un $D\varepsilon_t = \sigma^2$.

Definīcija 15. [9] Gadījuma procesu $\{x_t\}_{t=1}^{\infty}$ sauc par p-tās kārtas autoregresīvo procesu $AR(p)$, ja tas apmierina vienādojumu: $x_t = a_0 + a_1 x_{t-1} + a_2 x_{t-2} + ... + a_p x_{t-q} + \varepsilon_t$, kur ε_t baltais troksnis ar $E\varepsilon_t = 0$, $D\varepsilon_t = \sigma^2$ un $a_p \neq 0$.

Definīcija 16. [9] Par augstākās kārtas diferenci sauc $\Delta^d = (1 - L)^d$, kur $d \geq 1$.

Definīcija 17. [9] Gadijumu procesu $\{X_t\}_{t=-\infty}^{\infty}$ sauc par stacionāru stingrā nozimē, ja tā galīgdimensionālie sadalījumi paliek nemainīgi ie patvaļīgas laika nobīdes, t.i., ja $\forall n \leq 1, \forall t_1, t_2, ..., t_n$ un $\forall h$ n-dimensionālu gadijumu vektoru $(x_{t_1},)x_{t_2}, ..., x_{t_n})$ un $(x_{t_1+h},)x_{t_2+h}, ..., x_{t_n+h})$ sadalījuma funkcija sakrīt $F_{t_1+h, t_2+h, ..., t_n+h}(x_z, x_2, ..., x_n) = F_{t_1, t_2, ..., t_n}(x_1, x_2, ..., x_n)$.

$ARIMA$ modeļa izvēle

Lai izvēlētos atbilstošu $ARIMA$ modeli laikrindai, ļoti noderīgi ir attiecīgās laikrindas grafiks, autokorelāciju funkcijas (ACF) un parciālās autokorelāciju funkcijas ($PACF$) grafiku izveidošana, kā arī ir svarīga informācija par laikrindas datu vākšanas procedūru. Vairāk par ACF un $PACF$ skatīt [9], [4]. Pēc šiem grafikiem var noteikt, vai datiem ir nepieciešama transformācija, lai izvairītos no trenda. Tomēr transformācijas pielietošana ir jāizvērtē, jo ir iespēja arī laikrindu sabojāt. Pēc datu transformēšanas, stacionārajai laikrindai ir jāpiemeklē piemērotu stacionāro $ARIMA$ modeli. [9] Tabulā 9.1 ir parādīts, kā izskatās attiecīgā modeļa ACF un $PACF$ funkciju grafiskais attēlojums.

Tabula 9.1: ACF **un** $PACF$ **funkciju grafiskā izskata apraksts, [9]**

	ACF	$PACF$
$AR(p)$	samazinās ģeometriski	0, sākot ar laikrindas atstarpi $p+1$
$MA(q)$	0, sākot ar laikrindas atstarpi $q+1$	samazinās ģeometriski
$ARMA(p,q)$	samazinās ģeometriski	samazinās ģeometriski
$ARIMA(p,d,q)$	procesam ar $d \neq 1$ ACF samazinās lēni, aptuveni lineāri	

Modeļus var izvēlēties arī, balstoties uz atlikumiem. Divi visbiežak izmantotie kritēriji ir Akaike informācijas kritērijs (AIC) un Švarca-Beijesa kritērijs (BIC). $AIC = -2L + 2k$ un $BIC = -2L + klnT$, kur k ir novērtējamo parametru skaits, T ir parametru novērtēšanai lietojamo novērojumu skaits un L ir logaritmiskā ticamības funkcija. Jo mazāka AIC un BIC vērtība, jo izvēlētais modelis ir labāks. [9]

Izvēlētā modeļa pārbaude

Kad ir izvēlēts $ARIMA$ modelis, ir jāpārbauda, vai vispārīgāks modelis nedod labākus rezultātus nekā izvēlētais $ARIMA$ modelis. Tas nozīmē, ka, piemēram, ja izvēlētais modelis ir $AR(1)$, tad jāapskata arī modeļi $AR(2)$ un $ARMA(1,1)$. Tālāk ir jāizveido atlikumu grafiks, lai varētu redzēt izlēcējus un laika periodus, kad modelis nav labi piemērots datiem. Ja izvēlētais modelis ir piemērots, tad atlikumi ir tuvi baltā trokšņa novērtējumiem. Tā kā visas testa statistikas, kas tiek izmantotas, ir konstruētas pie pieņēmuma, ka teorētiskie ε_t ir Gausa baltā trokšņa process un, ņemot vērā, ka baltā trokšņa normalitāte ir svarīga pie parametru novērtēšanas un laikrindas prognozēšanas, ir jāveic atlikumu normalitātes pārbaude grafiski un ar Kolmogorova kritēriju palīdzību. [9]

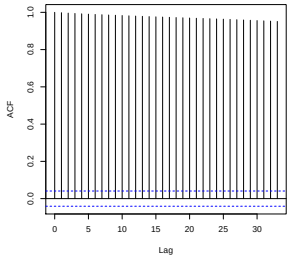
$ARIMA$ procesu prognozēšana

Vērtības $x_1, x_2, ..., x_N$ ir iegūtas N secīgos laika momentos, novērojot $ARIMA$ procesu $\{x_n\}$ un, balstoties uz to, var prognozēt procesa vērtību x_{N+h} laika momentā $N+h$, kur $h \geq 1$. Laika momentu N sauc par prognozes sākumpunktu un laika starpību h sauc par prognozes attālumu. [9]

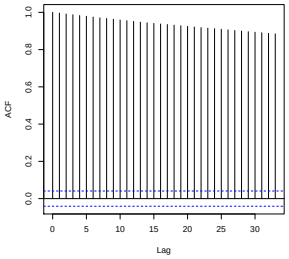
Definīcija 18. [9] Par prognozi ar mazāko vidējo kvadrātisko kļūdu $\hat{x}_N(h)$ sauc par funkciju $\hat{x}$ no novērojumiem F_N, kas minimizē nosacīto vidējo vērtību $E[X_{N+h} - \hat{x})^2 / F_N]$.

9.1 $ARIMA$ procesu prognozēšanas rezultāti

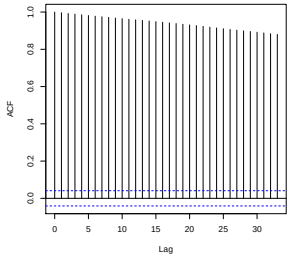
Lai prognozētu, ir svarīgi, lai laikrinda būtu stacionāra, tādēļ tiek uzzīmēts ACF. No attēliem 9.1 var redzēt, ka ACF ir lēni dilstoša, kas varētu nozīmēt, ka dati ir nestacionāri. Tādēļ ar Dikeja-Fullera vienības saknes testu tiek pārbaudīts, vai laikrindā ir vienības sakne.



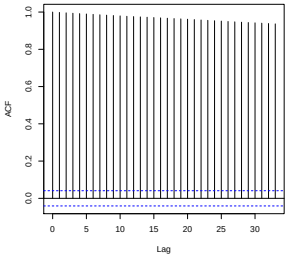
(a) **EUR/AUD** ACF **grafiks**



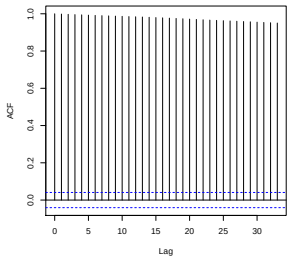
(b) **EUR/CAD** ACF **grafiks**



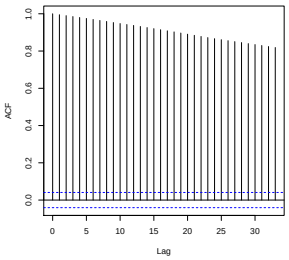
(c) **EUR/CZK** ACF **grafiks**



(d) **EUR/GBP** ACF **grafiks**



(e) **EUR/JPY** ACF **grafiks**



(f) **EUR/USD** ACF **grafiks**

Att. 9.1: **Attēlos (a) - (f) uzzīmēti valūtu kursu ACF grafiks**

Tas tiek darīts ar *EViews* un *R* iebūvēto funkciju *adf.test*.

Tabula 9.2: **Dikeja-Fullera testa p-vērtiba valūtu kursu laikrindām un diferencētajām laikrindām**

Valūtu kurss	p vērtība	p-vērtība pēc diferencēšanas
EUR/AUD	0.6678	<0.01
EUR/CAD	0.4826	<0.01
EUR/CZK	0.8028	<0.01
EUR/GBP	0.9132	<0.01
EUR/JPY	0.8654	<0.01
EUR/USD	0.2854	<0.01

Tabulā 9.2 var redzēt, ka $p > 0.05$, nulles hipotēzi nevar noraidīt, tādēļ visas laikrindas tiek diferencētas ar *R* iebūvēto funkciju *diff*, pēc tam atkal tiek pārbaudīta stacionaritāte. Tagad visi dati ir stacionāri, ko var redzēt tabulā 9.2 ($p < 0.05$), tādēļ var sākt meklēt piemērotāko modeli datiem. Visiem valūtu kursiem tiek uzzīmēti ACF un $PACF$ grafiki, skatīt attēlā 9.2. Tiek izmantota programmā *R* iebūvētā funkcija *auto.arima*, kura nosaka, kāds ir piemērotākais modelis datiem, izmantojot AIC un BIC novērtējumus. Tabulā 9.3 ir apkopoti rezultāti, kādi modeļi ir jāapskata katram valūtu kursam.

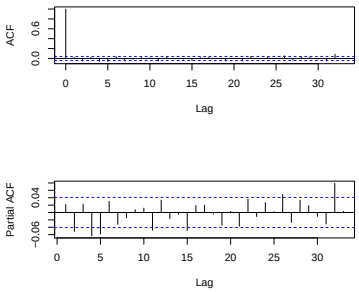
Tabula 9.3: **Izvēlētie $ARIMA$ modeļi valūtu kursiem**

Valūtu kurss	auto.arima modelis
EUR/AUD	$ARIMA(4,1,3)$
EUR/CAD	$ARIMA(1,1,2)$
EUR/CZK	$ARIMA(0,1,1)$
EUR/GBP	$ARIMA(2,1,0)$
EUR/JPY	$ARIMA(0,1,0)$
EUR/USD	$ARIMA(0,1,0)$

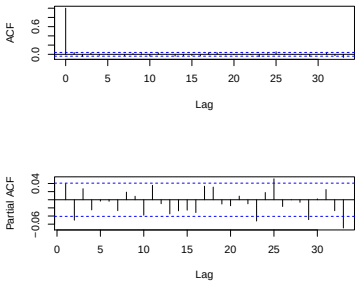
Tālāk ir jāpārbauda atlikumu nekorelētība un normalitāte. [9]. Tas tiek darīts ar programmā *R* iebūvētajām funkcijām *Box.test* un *shapiro.test* attiecīgi. No tabulas 9.4 var redzēt, ka atlikumi ir nekorelēti, bet tie nav normāli sadalīti. Bet, ņemot vērā, ka atlikumi ir nekorelēt, tad prognozēšana tiks veikta.

Tabula 9.4: **Boksa-Ļuinga korelētības testa p-vērtība, un Šapiro normalitātes testa p-vērtība**

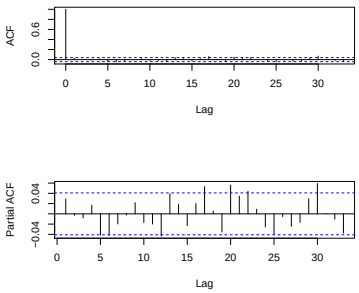
Valūtu kurss	p vērtība (Boksa-Ļuinga)	p-vērtība (Šapiro)
EUR/AUD	0.9963	<0.01
EUR/CAD	0.9947	<0.01
EUR/CZK	0.9858	<0.01
EUR/GBP	0.9927	<0.01
EUR/JPY	0.9668	<0.01
EUR/USD	0.7934	<0.01



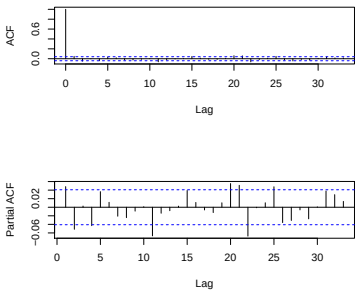
(a) **EUR/AUD** ACF **un** $PACF$ **grafiki**



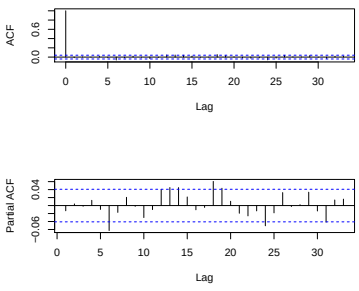
(b) **EUR/CAD** ACF **un** $PACF$ **grafiki**



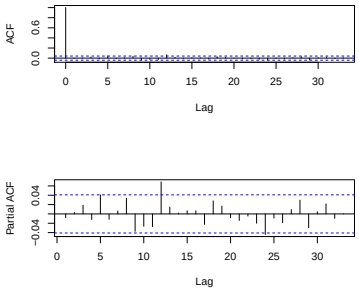
(c) **EUR/CZK** ACF **un** $PACF$ **grafiki**



(d) **EUR/GBP** ACF **un** $PACF$ **grafiki**



(e) **EUR/JPY** ACF **un** $PACF$ **grafiki**

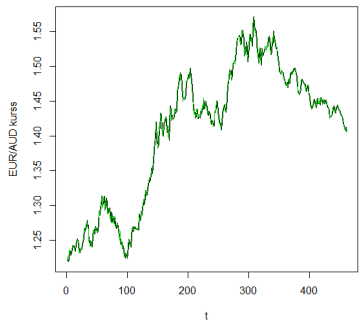


(f) **EUR/USD** ACF **un** $PACF$ **grafiki**

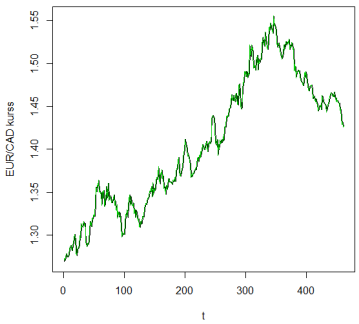
Att. 9.2: **Attēlos (a) - (f) ir uzzīmēti pirmās diferences valūtu kursu** ACF **un** $PACF$ **grafiki**

$ARIMA$ prognozēšanas rezultāti

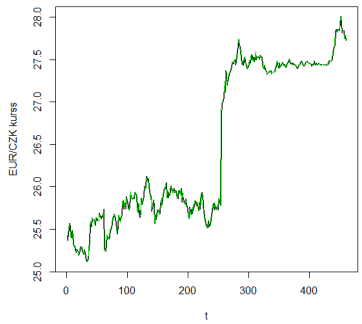
Prognozēšana tika veikta ar programmā *R* iebūvēto funkciju *forecast*. Attēlos 9.3 var redzēt prognozēto laikrindu (zaļā krāsā) un patieso laikrindu (melnā krāsā). Jau pēc visiem attēliem var spriest, ka prognozēšana ir bijusi samērā precīza. Bet, lai to noskaidrotu precīzi, ir jāaprēķina kļūdu novērtējumi 6.06, 6.08 un 6.07. Visi rezultāti ir apkopoti tabulā 9.5. Visprecīzāk, balstotis uz kļudu novērtējumiem MAD, MSE un $MAPE$, tika prognozēts valūtu kurss EUR/GBP, bet vissliktāk EUR/JPY un EUR/CZK.



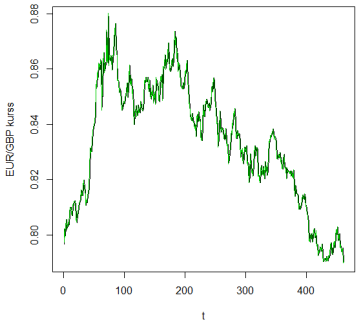
(a) **EUR/AUD prognoze**



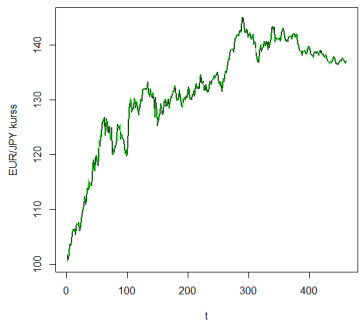
(b) **EUR/CAD prognoze**



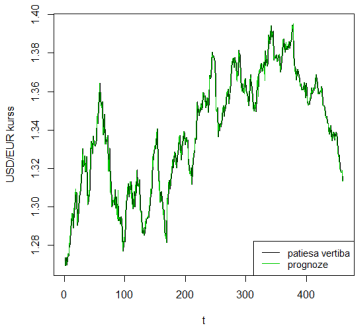
(c) **EUR/CZK prognoze**



(d) **EUR/GBP prognoze**



(e) **EUR/JPY prognoze**



(f) **EUR/USD prognoze**

Att. 9.3: **Attēlos (a) - (f) ir uzzīmēti ARIMA modeļu prognožu un patieso vērtību grafiki**

Tabula 9.5: ARIMA modeļu valūtu kursu prognozēšanas precizitātes rezultāti

Valūtu kurss	MAD	MSE	$MAPE$
EUR/AUD	0.00660	0.00007	0.00471
EUR/CAD	0.00514	0.00005	0.00366
EUR/CZK	0.04188	0.00573	0.00160
EUR/GBP	0.00259	0.00001	0.00309
EUR/JPY	0.65130	0.86075	0.0051
EUR/USD	0.00405	0.00003	0.00303

Rezultāti

Jau ar iepriekš aprakstītajiem kļūdu novērtējumiem $MAPE$ (6.08), MSE (6.07) un MAD (6.06) tiks salīdzināta prognozēšanas precizitāte. No mākslīgo neironu tīkla modeļiem tiek izvēlēts 2. un 4. modelis, jo šie modeļi uzrādīja labākos prognozēšanas precizitātes rezultātus, balstoties uz MAD, MSE un $MAPE$, ko var redzēt 8.1 apakšnodaļā. Tabulā 9.6 ir apkopoti labāko modeļu kļūdu novērtejumi MAD, MSE un $MAPE$.

Tabula 9.6: **Vislabāko modeļu, spriežot pēc MAD, MSE un $MAPE$, valūtu kursu prognozēšanas precizitātes rezultāti**

Valūtu kurss	Modelis	MAD	MSE	$MAPE$
EUR/AUD	2.	0.0065	0.00007	0.0046
	4.	0.0072	0.0001	0.0335
	ARIMA	0.0066	0.00007	0.0047
EUR/CAD	2.	0.0048	0.00004	0.0034
	4.	0.0093	0.0002	0.0299
	ARIMA	0.0051	0.00005	0.0037
EUR/CZK	2.	0.0065	0.0001	0.0140
	4.	0.0062	0.0001	0.0135
	ARIMA	0.0434	0.0062	0.0017
EUR/GBP	2.	0.0027	0.00001	0.0032
	4.	0.0075	0.0001	0.0135
	ARIMA	0.0026	0.00001	0.0031
EUR/JPY	2.	0.0089	0.0002	0.0207
	4.	0.0087	0.00002	0.0204
	ARIMA	0.6513	0.8607	0.0051
EUR/USD	2.	0.0004	0.00003	0.0030
	4.	0.0095	0.0002	0.0252
	ARIMA	0.0041	0.00003	0.0030

Tabulā 9.7 ir šie trīs modeļi sarindoti pēc vietām sākot ar 1. vietu līdz 3. vietai. Spriežot

Tabula 9.7: **Vislabāko modeļu sadalīšana no 1. vietas līdz 3. vietai, balstoties uz MAD, MSE un $MAPE$**

Valūtu kurss	1.	2.	3.
EUR/AUD	2.	$ARIMA$	4.
EUR/CAD	2.	$ARIMA$	4.
EUR/CZK	4.	2.	$ARIMA$
EUR/GBP	$ARIMA$	2.	4.
EUR/JPY	4.	2.	$ARIMA$
EUR/USD	2.	$ARIMA$	4.

pēc rezultātiem, visprecīzāk prognozē 2. modelis, tomēr var redzēt, ka pirmajā vietā ierindojas katrs no modeļiem, kas norāda uz to, ka katram valūtu kursam ir piemērots cits modelis, ar kuru tiks visprecīzāk prognozēts. Arī autora [6] publikācijā ir secināts, ka ar mākslīgo neironu tīklu prognozē precīzāk nekā ar $ARIMU$, tomēr prognozēšanas precizitāte ir ļoti tuva.

Darba ietvaros katram modelim tika meklēti parametri. Tika noskaidrots, ka vislabākos rezultātus dod neironu tīkls ar 21 slēpto neironu, kā tas arī bija aprakstīts avotā [8]. Tika arī izmēģināts autora [1] izvēlēties 40 neironi slēptajā slānī, bet šādā gadījumā neironu tīkls nekonverģēja. Mācīšanās ātrums tika izvēlēts salīdzinoši mazs visiem modeļiem $\varphi \leq 0.001$. Tika apskatīti arī lielāki φ, bet tad apmācība kļūva nestabila un neironu tīkls neapmācijās. Tika noskaidrots, ka katram modelim ir jāizvēlas atbilstošs mācišanās ātrums φ.

Eksperimentu rezultātā tika noskaidrots arī, ka tīklu nedrīkst pārmācīt, kas nozīmē, ka kļūda E nedrīkst būt pārāk maza, jo tad neironu tīkls neprognozēs pietiekami labi, bet arī nevar izvēlēties kļūdu E pārlieku lielu, jo arī tad prognozes nebūs tik precīzas. Ir jameklē atbilstoša kļūda E, līdz kurai neironu tīklu apmācīt.

Pirmā izvirzītā hipotēze - ar mākslīgo neironu tīklu valūtu kursu prognozes ir precīzākas nekā ar $ARIMA$ modeli - izpildās daļēji, jo valūtu kursam EUR/CZK $ARIMA$ modelis prognozēja precīzāk, balstoties uz kļūdu novērtējumiem MAD, MSE un $MAPE$. Protams, iespējams arī šim valūtu kursam var piemeklēt gan modeli, gan attiecīgos parametrus, lai mākslīgo neironu tīkls prognozētu precīzāk, bet šī darba ietvaros tāds netika atrasts.

Otrā izvirzītā hipotēze - ar 1., 2. un 4. modeli var prognozēt precīzāk nekā ar 3., 5. un 6. modeli - arī izpildās daļēji, jo vislabāk prognozēja ar modeļiem ir 2. un 4., bet trešais labākais modelis, ja neskaita $ARIMA$ modeli, ir dalīts - 3. un 1. Apskatītajā literatūrā 3. modelis nav aprakstīts, bet prognozēšanas rezultāti ir pietiekami precīzi, lai modeli arī apsvērtu prognozēšanai, kā arī tas prognozē valūtu kursus EUR/CZK un EUR/JPY precīzāk par ARIMA modeli, balsoties uz kļūdu novērtējumiem MAD, MSE un $MAPE$.

Secinājumi

Darbā tika salīdzināta prognozēšanas precizitāte ar sešiem mākslīgo neironu tīklu modeļiem un $ARIMA$ modelis valūtu kursiem - EUR/AUD, EUR/CAD, EUR/CZK, EUR/GBP, EUR/JPY un EUR/USD. Tika noskaidrots, ka katram valūtu kursam ir jāpiemeklē savs modelis, kā arī jāpiemeklē attiecīgie parametri, lai neironu tīkls veiksmīgi apmācītos. Pēc izpētes veikšanas, var secināt, ka mākslīgie neironu tīkli prognozē precīzāk nekā $ARIMA$ modeļi, skatoties kļūdu novērtējumus MAD, MSE un $MAPE$, kas arī sakrīt ar [1] veikto pētījumu. Šobrīd iegūtie rezultāti ir ļoti tuvi, bet prognozēšanas precizitāti varētu uzlabot, ja, kā ieejas vērtības, izvēlētos ne tikai deviņus iepriekšējos datu punktus, bet arī citu valūtu kursu datu punktus, ar kuru attiecīgais valūtu kurss korelē, kā arī var pievienot citus datus kā ieejas.

Eksperimentu rezultātā tika noskaidrots arī, ka tīklu nedrīkst pārmācīt, kas nozīmē, ka kļūda E nedrīkst būt pārāk maza, jo tad neironu tīkls var neprognozēt pietiekami labi, bet arī nevar izvēlēties kļūdu E pārlieku lielu, jo arī tad prognozes nebūs tik precīzas. Ir jameklē atbilstoša kļūda E, līdz kurai neironu tīklu apmācīt.

Izveidotais kods programmās gan *Matlab*, gan *R* ir lēns salīdzinoši ar iebūvēto funkciju programmā *R*. Tīkla apmācība *Matlab* pie 20000 iterācijām norisinās aptuveni 20 minūtes, kamēr ar iebūvēto funkciju tas pats rezultāts tiek iegūts 3 minūšu laikā. Tas ir skaidrojams ar to, ka funkcijas *nnet* pirmkods nav programmēts *R*, kas nozīmē, ka attiecīgā programmēšanas valoda to pašu veic daudz ātrāk. Tātad ieteicams nākotnē ir uzlabot programmēto kodu. Iemesls, kādēļ netiek ieteikts izmantot jau esošo iebūvēto funkciju *nnet*, ir tāds, ka tā ir ļoti ierobežota, ar to ir iespējams iegūt tikai 2. modeli, kā arī nav iespējams zīmēt kļūdu grafiku.

Darba laikā tika secināts, ka apskatītos mākslīgo neironu tīkla modeļus var izmantot turpmākai pētniecībai, bet tos nav ieteicams izmantot praksē, jo parametru atrašana prasa laiku, prognozēšana notiek tikai vienu soli uz priekšu, kā arī mākslīgo neironu tīkla apmācība ir ļoti lēna un tā prasa daudz iterāciju. Šāda prognozēšana ir neefektīva. Lai tīkla apmācība notiktu ātrāk, to var apmācīt līdz kādai konkrētai kļūdai E un turpmāk izmantot iegūtos svarus kā sākuma svarus.

Literatūra

[1] B. Oancea, S. Ciucu *Time series forecasting using neural networks*. Research paper, 2013

[2] E. Simon *Forecasting Foreign Exchange Rates with Neural Networks*. Lausanne:Project Report, 2002

[3] E.Zwer, U.Helmke, D. Pratzel-Wolters *Mathematical Theory of Neural Networks*. Lecture script, 2011

[4] G. Box, G. Jenkins, G. Reinsel*Time Series Analysis. Forecasting and Control. Fourth edition*. New Jersy:Wiley, 2008

[5] J. Buls *Matemātiskās loģikas un kopu teorijas elementi*. Lekciju konspekts, 2008

[6] J. Yao, H. Poh, T. Jasic *Foreign Exchange rates forecasting with neural networks*. Singapore, 1996

[7] J.Zuters *Neironu tīkli*. Lekciju konspekts, 2013

[8] L. Yu, S. Wang, K. Lain *Foreign-exchange-rate forecasting with artificial neural networks*. U.S.A:Springer,2007

[9] N.Siņenko *Laikrindu analīze*. Lekciju konspekts, 2013

[10] R. Rojas *Neural Networks: A Systematic Introduction*. Berlin:Springer-Verlag, 1996

[11] S. Asmuss *Operāciju pētīšana*. Lekciju konspekts, 2013

[12] S. Haykin *Neural Networks*. India:Indian Branch, 2001

[13] S. Trenn *Introduction to Neural Networks*. Lecture script, 2013

[14] V. Pacelli, V. Bevilacqua, M. Azzollini *An artificial neural network model to forecas exchange rates*. Research paper, 2011

[15] Z. Tiltānova *Neironu tīkli un to lietojums laikrindu prognozēšanā*. Diplomdarbs, 2011

[16] *A Comparision of Traditional Forecasting Techniques and Neural Networks*. Pieejmas: http://www.eng.auburn.edu/wilambm/pap/1995/ANNIE'95_A_comparison_of_traditional_forecasting.pdf (Skat. 14.12.2014.)

[17] *Biological Neurons and Neural Networks, Artificial Neurons*. Pieejmas: http://www.cs.bham.ac.uk/jxb/INC/l2.pdf (Skat. 25.06.2014.)

[18] *Four ways to forecast currency changes*. Pieejams: http://www.investopedia.com/articles/forex/11/4-ways-to-forecast-exchange-rates.asp (Skat. 20.12.2014.)

[19] *Neuron*. Pieejams: http://en.wikipedia.org/wiki/Neuron (Skat. 22.11.2014.)

[20] *The Biological Neuron*. Pieejams: http://vv.carleton.ca/neil/neural/neuron-a.html (Skat. 25.06.2014.)

[21] *Valūtu kursi*. Pieejams: www.bank.lv (Skat. 15.09.2014.)

[22] *Weights of backpropagation algorithm*. Pieejams: http://stats.stackexchange.com/questions/45087/why doesnt-backpropagation-work-when- you-initialize-the-weights-the-same-value (Skat. 20.12.2014.)

1. pielikums. Aprēķini vispārīgajam daudzslāņu tīklam

Dots neironu tīkls, kuram ir n ieejas un p izejas. Lai apmācītu šo tīklu, ir nepieciešamas vēlamās atbildes, kuras tiks apzīmētas ar d. Apmācības laikā būtu jāminimizē atšķirība starp izeju un vēlamo atbildi. Tiks izmantota vidējā kvadrātiskā kļūda 5.21.

Neironu tīklā ir M slēptie slāņi. $M+1$ apzīmē slāņu skaitu neironu tīklā. Ar i apzīmēsim i-to slēpto slāni, $i = 1, \ldots, M$. $S^i \in \mathbb{R}^{n_i}$ ir vektors, kas sastāv no summācijas funkcijas rezultātiem i-tajā slānī, $Y^i \in \mathbb{R}^{n_i}$ ir vektors, kas sastāv no visām izejām i-tajā slānī, $W^i \in \mathbb{R}^{n_i \times n_{i-1}}$ ir visu svaru matrica, tie ir svari, kas savieno $(i-1)$-to un i-to slāni, $B^i \in \mathbb{R}^{n_i}$ ir vektors, kas sastāv no i-tā slāņa sliekšņiem. Y^{i-1} ir izejas vektors, ko dod $(i-1)$-tā slāņa neironi un Y^i ir i-tā slāņa izejas vektors.

$Y^1 = \sigma_1(W^1 X + B^1)$ ir izeja no pirmā slāņa, kur X ir ieejas vektors.
$Y^i = \sigma_i(W^i Y^{i-1} + B^i)$ ir izeja no i-tā slāņa.
$Y = Y^{M+1}$ ir izeja visam neironu tīklam.

Tālāk aprakstītais ir balstīts uz [3] [13].
Jaunie svari un slieksnis tiek atjaunoti pēc formulām attiecīgi:

$$w_{kl}^{i,new} = w_{kl}^{i,old} - \varphi \frac{\partial E}{\partial w_{kl}^i}$$

$$b_k^{i,new} = b_k^{i,old} - \varphi \frac{\partial E}{\partial b_k^i}$$

Lai aprēķinātu jaunos svarus un jauno slieksni, ir jāaprēķina parciālie atvasinājumi:

$$\frac{\partial E}{\partial w_{kl}^i}, \frac{\partial E}{\partial b_k^i}.$$

Ar $W = (W^1, B^1, W^2, B^2, ..., W^{n+1}, B^{n+1})$ tiek apzīmēts visu svaru un sliekšņu matrica tīklam. Šeit n ir skaitlis, cik ir slēpto slāņu. Sāksim ar $\frac{\partial E}{\partial w_{kl}^i}$ atvasinājumu, un tiks izmantota diferenciāļa formas invariance.

$$\frac{\partial E}{\partial w_{kl}^i} = \frac{\partial E}{\partial Y} \frac{\partial Y}{\partial w_{kl}^i}.$$

Pierakstīsim, kas ir nepieciešams, lai iegūtu izeju Y^i. Kombinē W^i, B^i un Y^{i-1}, izmantojot summācijas funkciju 2.01, iegūst rezultātu S^i. Tad šo rezultātu ievieto aktivizācijas

funkcijā, tiek iegūta i-tā slāņa izeja Y^i. Ņemot vērā, ka šeit formulas tiks aprakstītas vispārīgi, aktivizācijas funkcija netiek izvēlēta.

Sākumā tiks meklēti parciālie atvasinājumi i-tajam slānim. Ir jāaprēķina, kādu rezultātu dos summācijas funkcija i-tajā slānī. Rezultātus ir ērti sakārtot vektorā.

$$S^i(W^i, B^i, Y^{i-1}) = W^i Y^{i-1} - B^i = \begin{pmatrix} \sum_{\lambda=1}^{n_{i-1}} w_{1\lambda}^i y_\lambda^{i-1} - b_1^i \\ \sum_{\lambda=1}^{n_{i-1}} w_{2\lambda}^i y_\lambda^{i-1} - b_2^i \\ \dots \\ \sum_{\lambda=1}^{n_{i-1}} w_{n_i\lambda}^i y_\lambda^{i-1} - b_{n_i}^i \end{pmatrix}$$

Tālāk S^i ievieto aktivizācijas funkcijā, līdz ar to tiek aprēķināts rezultāts, kādu dos aktivizācijas funkcija i-tajā slānī.

$$Y^i(S^i) = \begin{pmatrix} \sigma^i(s_{n_1}^i) \\ \sigma^i(s_{n_2}^i) \\ \dots \\ \sigma^i(s_{n_i}^i) \end{pmatrix}$$

Tagad, kad šie rezultāti ir uzrakstīti, var sākt meklēt parciālos atvasinājumus $\frac{\partial Y^i}{\partial w_{kl}^i}$. Šeit atkal pielieto diferenciāļa formas invariance.

$$\frac{\partial Y^i}{\partial w_{kl}^i} = \frac{\partial Y^i}{\partial S^i} \frac{\partial S^i}{\partial w_{kl}^i}$$

Var pārrakstīt $\frac{\partial Y^i}{\partial S^i}$ kā matricu, lai vieglāk saprast, kā notiek atvasināšana.

$$\frac{\partial Y^i}{\partial S^i} \frac{\partial S^i}{\partial w_{kl}^i} = \begin{bmatrix} \frac{\partial y_1^i}{\partial s_1^i} & \frac{\partial y_1^i}{\partial s_2^i} & \dots & \frac{\partial y_1^i}{\partial s_{n_i}^i} \\ \frac{\partial y_2^i}{\partial s_1^i} & \frac{\partial y_2^i}{\partial s_2^i} & \dots & \frac{\partial y_2^i}{\partial s_{n_i}^i} \\ \dots & & & \\ \frac{\partial y_{n_i}^i}{\partial s_1^i} & \frac{\partial y_{n_i}^i}{\partial s_2^i} & \dots & \frac{\partial y_{n_i}^i}{\partial s_{n_i}^i} \end{bmatrix} \frac{\partial S^i}{\partial w_{kl}^i}$$

y_1^i, y_2^i un $y_{n_i}^i$ var pārrakstīt attiecīgi kā $y_1^i = \sigma(s_1^i)$, $y_2^i = \sigma(s_2^i)$ un $y_n^i = \sigma(s_n^i)$, līdz ar to parciālos atvasinājumus attiecīgi pa diagonāli var pārrakstīt $\sigma^{i'}(s_1^i)$, $\sigma^{i'}(s_2^i)$ un $\sigma^{i'}(s_n^i)$. Ņemot vērā, ka aktivizācijas funkcija šeit vēl nav izvēlēta, $\sigma(s_1^i)$ ir vēl jāatvasina pēc $i-t$ slāņa attiecīgās summācijas s_1^i, līdz ar to pierakstām šo atvasinājumu kā $\sigma^{i'}(s_1^i)$.

Virs diagonāles atvasinājumi ir nulles, jo y_1^i, y_2^i un $y_{n_i}^i$ ir atkarīgi tikai no s_1^i, s_2^i un s_n^i, bet zem diagonāles atvasinājumi nav nepieciešami, jo tos tīklā neapskata, tādēļ tur rakstām 0. Parciālajā

atvasinājumā $\frac{\partial S^i}{\partial w^i_{kl}}$ vienīgi k-tajā rindā elements $w^i_{kl}y^{i-1}_l - b^i_2$ ir atkarīgs no w^i_{kl}, līdz ar to tas arī ir vienīgais atvasinājums, kas nebūs nulle. Līdz ar to tiek iegūts:

$$\frac{\partial Y^i}{\partial S^i}\frac{\partial Y^i}{\partial S^i} = \begin{bmatrix} \sigma^{i'}(s^i_1) & 0 & ... & 0 \\ 0 & \sigma^{i'}(s^i_2) & ... & 0 \\ ... & & & \\ 0 & 0 & ... & \sigma^{i'}(s^i_{n_i}) \end{bmatrix} \begin{pmatrix} 0 \\ ... \\ 0 \\ y^{i-1}_l \\ 0 \\ ... \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ ... \\ 0 \\ \sigma^{i'}(s^i_k)y^{i-1}_l \\ 0 \\ ... \\ 0 \end{pmatrix}.$$

Tagad ir jāaprēķina $\frac{\partial E}{\partial b^i_k}$, kas tiek iegūts analoģiski.

$$\frac{\partial Y^i}{\partial b^i_k} = \frac{\partial Y^i}{\partial S^i}\frac{\partial S^i}{\partial b^i_k} = \begin{pmatrix} 0 \\ ... \\ 0 \\ \sigma^{i'}(s^i_k) - 1 \\ 0 \\ ... \\ 0 \end{pmatrix}$$

Jau tika iegūts rezultāts $\frac{\partial Y^i}{\partial w^i_{kl}}$. Tagad tiek pieņemts, ka $i < j$. Un ir jāiegūst parciālais atvasinājums un tiek atkal pielietota diferenciāļa formas invariance:

$$\frac{\partial Y^j}{\partial w^i_{kl}} = \frac{\partial Y^j}{\partial S^j}\frac{\partial S^j}{\partial w^i_{kl}}.$$

Tāpat kā iepriekš tiek iegūts $\frac{\partial Y^j}{\partial S^j}$, un apzīmēsim šo matricu ar D_{σ_j}.

$$\frac{\partial Y^j}{\partial S^j} = \begin{bmatrix} \sigma^{j'}(s^j_1) & 0 & ... & 0 \\ 0 & \sigma^{j'}(s^j_2) & ... & 0 \\ ... & & & \\ 0 & 0 & ... & \sigma^{j'}(s^j_{n_i}) \end{bmatrix} = D_{\sigma_j}$$

Tagad ir jāaprēķina $\frac{\partial S^j}{\partial w^i_{kl}}$. Šeit jāatceras, ka $j > i$. Tādēļ var pārrakstīt S^j kā $S^j = W^j Y^{j-1} - B^j$. Līdz ar to tiek iegūts:

$$\frac{\partial S^j}{\partial w_{kl}^i} = \begin{pmatrix} \sum_{\lambda=1}^{k_{j-1}} w_{1,\lambda}^j \frac{\partial y\lambda^{j-1}}{\partial w_{kl}^i} \\ \sum_{\lambda=1}^{k_{j-1}} w_{2,\lambda}^j \frac{\partial y\lambda^{j-1}}{\partial w_{kl}^i} \\ \dots \\ \sum_{\lambda=1}^{k_{j-1}} w_{n_j,\lambda}^j \frac{\partial y\lambda^{j-1}}{\partial w_{kl}^i} \end{pmatrix} = W^j \frac{\partial Y^{j-1}}{\partial w_{kl}^i}$$

$$\frac{\partial Y^j}{\partial w_{kl}^i} = \frac{\partial Y^j}{\partial S^j}\frac{\partial S^j}{\partial w_{kl}^i} = \begin{bmatrix} \sigma^{j'}(S_1^j) & 0 & \dots & 0 \\ 0 & \sigma^{j'}(s_2^j) & \dots & 0 \\ \dots & & & \\ 0 & 0 & \dots & \sigma^{j'}(s_{n_i}^j) \end{bmatrix} \begin{pmatrix} \sum_{\lambda=1}^{k_{j-1}} w_{1,\lambda}^j \frac{\partial y\lambda^{j-1}}{\partial w_{kl}^i} \\ \sum_{\lambda=1}^{k_{j-1}} w_{2,\lambda}^j \frac{\partial y\lambda^{j-1}}{\partial w_{kl}^i} \\ \dots \\ \sum_{\lambda=1}^{k_{j-1}} w_{n_j,\lambda}^j \frac{\partial y\lambda^{j-1}}{\partial w_{kl}^i} \end{pmatrix} =$$

$$= D_{\sigma_j} W^j \frac{\partial Y^{j-1}}{\partial w_{kl}^i}.$$

Analoģiski tiek iegūts:

$$\frac{\partial Y^j}{\partial b_k^i} = D_{\sigma_j} W^j \frac{\partial Y^{j-1}}{\partial b_k^i}$$

Tagad var aprēķināt:

$$\frac{\partial E}{\partial w_{kl}^i}, \frac{\partial E}{\partial b_k^i}$$

Sākumā tiks aprēķināts $\frac{\partial E}{\partial w_{kl}^i}$. Kā jau iepriekš teikts, šeit var pielietot diferenciāļa formas invarianci.

$$\frac{\partial E}{\partial w_{kl}^i} = \frac{\partial E}{\partial Y}\frac{\partial Y}{\partial w_{kl}^i}$$

Abi šie rezultāti ir jau iepriekš iegūti tikai i-tajam slānim. Jāatceras, ka neironu tīklam ir $M+1$ slāņi un izeja ir $Y = Y^{M+1}$. Līdz ar to tiek iegūts:

$$\frac{\partial E}{\partial w_{kl}^i} = \frac{\partial E}{\partial Y}\frac{\partial Y}{\partial w_{kl}^i} = \frac{\partial E}{\partial Y} D_{\sigma^{M+1}} W^{M+1} \frac{\partial Y^M}{\partial w_{kl}^i} =$$

$$= \frac{\partial E}{\partial Y} D_{\sigma^{M+1}} W^{M+1} D_{\sigma^M} W^M \cdot ... \cdot D_{\sigma^{i+1}} W^{i+1} \frac{\partial Y^i}{\partial w_{kl}^i} =$$

$$= \frac{\partial E}{\partial Y} D_{\sigma^{M+1}} W^{M+1} \cdot ... \cdot D_{\sigma^{i+1}} W^{i+1} D_{\sigma^i} \begin{pmatrix} 0 \\ ... \\ 0 \\ y_l^{i-1} \\ 0 \\ ... \\ 0 \end{pmatrix}.$$

Analoģiski tiek iegūts:

$$\frac{\partial E}{\partial b_k^i} = \frac{\partial E}{\partial Y} D_{\sigma^{M+1}} W^{M+1} \cdot ... \cdot D_{\sigma^{i+1}} W^{i+1} D_{\sigma^i} \begin{pmatrix} 0 \\ ... \\ 0 \\ 1 \\ 0 \\ ... \\ 0 \end{pmatrix}$$

Vēl atliek aprēķināt $\frac{\partial E}{\partial Y}$:

$$\frac{\partial E_{(x,d)}}{\partial Y} = \frac{\partial \frac{1}{2}(Y-d)^\top (Y-d)}{\partial Y} = (Y-d)^\top = e^\top$$

Apzīmēsim atgriezeniskās izplatīšanās kļūdu i-tajam slānim ar ε^i:

$$\varepsilon^{M+1} := D_{\sigma^{M+1}} e$$

$$\varepsilon^i := D_{\sigma^i} (W^{i+1})^\top \varepsilon^{i+1}, i = M, M-1, ..., 1$$

Svari un slieksnis attiecīgi tiks atjaunoti pēc formulām:

$$w_{kl}^{i,new} = w_{kl}^{i,old} - \varphi \varepsilon_k^i y_l^{i-1}$$

$$b_{kl}^{i,new} = b_{kl}^{i,old} - \varphi \varepsilon_k^i$$

2. pielikums. *Matlab* programmu kodi

```
%%%%%%%%%%%%1.modelis%%%%%%%%
%izvelas neironu sleptaja slani, epohas un macisanas atrumu
sleptie_neironi = 21;
epohas = 20000;
macisanas_atrums = 0.001;
%ievada datus
paraugi = jpyieejas;
velama_atbilde = jpyizejas;
if size(paraugi,1) = size(velama_atbilde,1)
disp('ERROR: nesakrit paraugi ar velamajam atbildem')
return
end
patterns = size(paraugi,1);
bias = ones(patterns,1);
paraugi = [paraugi bias];
ieejas = size(paraugi,2);
%pievieno pogas, lai varetu apstaties pirms laika u.c
hstop = uicontrol('Style','PushButton','String','Stop','Position',
[5 5 70 20],'callback','earlystop = 1;');
earlystop = 0;
hreset = uicontrol('Style','PushButton','String',
'Reset Wts', 'Position',
get(hstop,'position')+[75 0 0 0],'callback','reset = 1;');
reset = 0;
hlr = uicontrol('Style','slider','value',.1,
'Min',.01,'Max',1,'SliderStep',[0.01 0.1]
,'Position', get(hreset,'position')+[75 0 100 0]);
%patvaligi izveleti svari
rng(12)
svari_ieeja_sleptais = (randn(ieejas,sleptie_neironi) - 0.5)/10;
rng(1)
svari_sleptais_izeja = ((randn(1,sleptie_neironi+1) - 0.5)/10)';
%sakas apmaciba
for iter = 1:epohas
for j = 1:patterns
patnum=j;
tag_ieeja = paraugi(patnum,:);
act = velama_atbilde(patnum,1);
o=ones(sleptie_neironi,1);
Yizeja = (o'./(o'+(exp(-tag_ieeja*svari_ieeja_sleptais))));
%y izrekina
Yizeja1=[Yizeja 1]; %pievienots 1
pred = (1./(1+exp(-Yizeja1*svari_sleptais_izeja))) ;
% izeja no neironu tikla
%error = (norm(pred - act))*0.5; %error
end
ol = ones(size(paraugi*svari_ieeja_sleptais));
eXW = exp(-paraugi*svari_ieeja_sleptais); %exp(XW)
```

```
pred1=ol./(ol + eXW); %Y izeja no hidden
oo = (ones(1,patterns))';
pred11=[pred1 oo]; %pievieno vieninieku kolonnu
eYV=exp(-pred11*svari_sleptais_izeja); %exp(YV)
pred=(oo./(oo+eYV)); %Z izeja no izejas
error = (norm(pred- velama_atbilde));
err(iter) = 0.5*(sum(error));
figure(1);
plot(err);
%delta
de = (pred - velama_atbilde).*pred.*(ones(size(velama_atbilde))-pred);
%Delta
la=ones(patterns,sleptie_neironi);
pred1;
la-pred1;
a = pred1.*(la-pred1);
c=repmat(de,1,sleptie_neironi)';
d=repmat(svari_sleptais_izeja(1:sleptie_neironi,1),1,patterns);
k=(c.*d)';
De=a.*k;
la=ones(patterns,sleptie_neironi);
svari_sleptais_izeja1=(repmat(svari_sleptais_izeja
(1:sleptie_neironi,1),1,patterns))';
de1=repmat(de,1,sleptie_neironi);
ris=de1.*svari_sleptais_izeja1;
De = pred1.*(la-pred1).*ris;
% svaru uzlabosana sleptais-izeja
p=pred1;
ab=[macisanas_atrums.*(de'*p) 0];
svari_sleptais_izeja=svari_sleptais_izeja-ab';
%svari_sleptais_izeja = (svari_sleptais_izeja)-
%(repmat(macisanas_atrums,1,sleptie_neironi+1))'.*(de'*p)'
% svaru uzlabosana ieeja-sleptais
ga=ones(sleptie_neironi,1) -1;
ba=paraugi(:,1:9)'*De;
va=[ba;ga'];
svari_ieeja_sleptais=svari_ieeja_sleptais-macisanas_atrums.*va;
%stop
if earlystop
fprintf('stopped at epoch: %d\n',iter);
break
end
%stop
if err(iter) < 0.001
fprintf('converged at epoch: %d\n',iter);
break
end
end
fprintf('kopa %d epohas\n',iter);
er=(norm(pred-velama_atbilde))*0.5;
```

```
[velama_atbilde pred];
%ievada datus
paraugi_te = jpytestieejas;
velama_atbilde_te = jpytestizejas;
if size(paraugi_te,1) = size(velama_atbilde_te,1)
disp('ERROR: nesakrit paraugi ar velamajam atbildem')
return
end
patterns_te = size(paraugi_te,1);
bias_te = ones(patterns_te,1);
paraugi_te = [paraugi_te bias_te];
ieejas_te = size(paraugi_te,2);
for j = 1:patterns_te
patnum=j;
tag_ieeja_te = paraugi_te(patnum,:);
act_te = velama_atbilde_te(patnum,1);
o_te=ones(sleptie_neironi,1);
Yizeja_te = (o_te'./(o_te'+(exp(-tag_ieeja_te*svari_ieeja_sleptais))));
%y izrekina
Yizeja1_te=[Yizeja_te 1]; %pievienots 1
pred_te = (1./(1+exp(-Yizeja1_te*svari_sleptais_izeja))) ;
% izeja no neironu tikla
%error = (norm(pred - act))*0.5; %error
end
ol_te = ones(size(paraugi_te*svari_ieeja_sleptais));
eXW_te = exp(-paraugi_te*svari_ieeja_sleptais); %exp(XW)
pred1_te=ol_te./(ol_te + eXW_te); %Y izeja no hidden
oo_te = (ones(1,patterns_te))';
pred11_te=[pred1_te oo_te]; %pievieno vieninieku kolonnu
eYV_te=exp(-pred11_te*svari_sleptais_izeja); %exp(YV)
pred_te=(oo_te./(oo_te+eYV_te)); %Z izeja no izejas
[velama_atbilde_te pred_te]
%%%%%%%%%%%%2.modelis atskirigais%%%%%%%%
o=ones(sleptie_neironi,1);
Yizeja = (o'./(o'+(exp(-tag_ieeja*svari_ieeja_sleptais)))); %izeja no slepta
%y izrekina
Yizeja1=[Yizeja 1]; %pievienots 1
pred = Yizeja1*svari_sleptais_izeja ; %izeja no visa
% izeja no neironu tikla
ol = ones(size(paraugi*svari_ieeja_sleptais));
eXW = exp(-paraugi*svari_ieeja_sleptais); %exp(XW)
pred1=ol./(ol + eXW); %Y izeja no hidden
oo = (ones(1,patterns))';
pred11=[pred1 oo]; %pievieno vieninieku kolonnu
eYV=pred11*svari_sleptais_izeja; %exp(YV)
pred=eYV; %Z izeja no izejas
%delta
de = (pred - velama_atbilde);
%Delta
la=ones(patterns,sleptie_neironi);
```

```
svari_sleptais_izeja1=(repmat(svari_sleptais_izeja
(1:sleptie_neironi,1),1,patterns))';
de1=repmat(de,1,sleptie_neironi);
ris=de1.*svari_sleptais_izeja1;
De = pred1.*(la-pred1).*ris;
%%%%%%%%%3.modelis atskirigais%%%%%%%%
o=ones(sleptie_neironi,1);
Yizeja = (o'./(o'+(exp(-tag_ieeja*svari_ieeja_sleptais))));
%izeja no slepta
Yizeja1=[Yizeja 1]; %pievienots 1
na=exp(-Yizeja1*svari_sleptais_izeja);
ba=exp(Yizeja1*svari_sleptais_izeja);
pred = (ba-na)./(ba+na); %izeja no visa
% izeja no neironu tikla
ol = ones(size(paraugi*svari_ieeja_sleptais));
eXW = exp(-paraugi*svari_ieeja_sleptais); %exp(XW)
eXW2 = exp(paraugi*svari_ieeja_sleptais);
pred1=ol./(ol + eXW); %Y izeja no hidden
oo = (ones(1,patterns))';
pred11=[pred1 oo]; %pievieno vieninieku kolonnu
eYV=exp(-pred11*svari_sleptais_izeja); %exp(YV)
eYV2=exp(pred11*svari_sleptais_izeja);
pred=(eYV2-eYV)./(eYV2+eYV); %Z izeja no visa
%delta
de = (pred - velama_atbilde).*(ones(size(velama_atbilde))-pred.^2);
%Delta
la=ones(patterns,sleptie_neironi);
svari_sleptais_izeja1=(repmat(svari_sleptais_izeja
(1:sleptie_neironi,1),1,patterns))';
de1=repmat(de,1,sleptie_neironi);
ris=de1.*svari_sleptais_izeja1;
De = pred1.*(la-pred1).*ris;
%%%%%%4.modelis atskirigais%%%%%%%%%%
o=ones(sleptie_neironi,1);
Yizeja = (((exp(tag_ieeja*svari_ieeja_sleptais))
-(exp(-tag_ieeja*svari_ieeja_sleptais)))
./((exp(tag_ieeja*svari_ieeja_sleptais))
+(exp(-tag_ieeja*svari_ieeja_sleptais))));
%izeja no slepta
Yizeja1=[Yizeja 1]; %pievienots 1
pred = Yizeja1*svari_sleptais_izeja ; %izeja no visa
% izeja no neironu tikla
ol = ones(size(paraugi*svari_ieeja_sleptais));
eXW = exp(-paraugi*svari_ieeja_sleptais); %exp(XW)
eXW2 = exp(paraugi*svari_ieeja_sleptais);
pred1=(eXW2-eXW)./(eXW2 + eXW); %Y izeja no hidden
oo = (ones(1,patterns))';
pred11=[pred1 oo]; %pievieno vieninieku kolonnu
eYV=pred11*svari_sleptais_izeja; %exp(YV)
pred=(eYV); %Z izeja no izejas
```

```
%delta
de = (pred - velama_atbilde);
%Delta
la=ones(patterns,sleptie_neironi);
svari_sleptais_izeja1=(repmat(svari_sleptais_izeja
(1:sleptie_neironi,1),1,patterns))';
de1=repmat(de,1,sleptie_neironi);
ris=de1.*svari_sleptais_izeja1;
De = (la-pred1.^2).*ris;
%%%%%%%%5. Modelis atskirigais %%%%%%%%%%%
o=ones(sleptie_neironi,1);
Yizeja = (((exp(tag_ieeja*svari_ieeja_sleptais))-
(exp(-tag_ieeja*svari_ieeja_sleptais)))
./((exp(tag_ieeja*svari_ieeja_sleptais))
+(exp(-tag_ieeja*svari_ieeja_sleptais)))); %izeja no slepta
Yizeja1=[Yizeja 1]; %pievienots 1
na=exp(-Yizeja1*svari_sleptais_izeja);
ba=exp(Yizeja1*svari_sleptais_izeja);
pred = (ba-na)./(ba+na) ; %izeja no visa
% izeja no neironu tikla
ol = ones(size(paraugi*svari_ieeja_sleptais));
eXW = exp(-paraugi*svari_ieeja_sleptais); %exp(XW)
eXW2 = exp(paraugi*svari_ieeja_sleptais);
pred1=(eXW2-eXW)./(eXW2 + eXW); %Y izeja no hidden
oo = (ones(1,patterns))';
pred11=[pred1 oo]; %pievieno vieninieku kolonnu
eYV=exp(-pred11*svari_sleptais_izeja); %exp(YV)
eYV2=exp(pred11*svari_sleptais_izeja);
pred=(eYV2-eYV)./(eYV2+eYV); %Z izeja no visa
%delta
de = (pred - velama_atbilde).*
(ones(size(velama_atbilde)) - pred.^2);
%Delta
la=ones(patterns,sleptie_neironi);
svari_sleptais_izeja1=(repmat(svari_sleptais_izeja
(1:sleptie_neironi,1),1,patterns))';
de1=repmat(de,1,sleptie_neironi);
ris=de1.*svari_sleptais_izeja1;
De = (la-pred1.^2).*ris;

%%%%6. Modelis atskrigais%%%%%%%
o=ones(sleptie_neironi,1);
Yizeja = (((exp(tag_ieeja*svari_ieeja_sleptais))
-(exp(-tag_ieeja*svari_ieeja_sleptais)))
./((exp(tag_ieeja*svari_ieeja_sleptais))
+(exp(-tag_ieeja*svari_ieeja_sleptais)))); %izeja no slepta
%y izrekina
Yizeja1=[Yizeja 1]; %pievienots 1
pred = (1./(1+exp(-Yizeja1*svari_sleptais_izeja))) ;
% izeja no neironu tikla
```

```
ol = ones(size(parusdgi*svari_ieeja_sleptais));
eXW = exp(-parusdgi*svari_ieeja_sleptais); %exp(XW)
eXW2 = exp(parusdgi*svari_ieeja_sleptais);
pred1=(eXW2-eXW)./(eXW2 + eXW); %Y izeja no hidden
oo = (ones(1,patterns))';
pred11=[pred1 oo]; %pievieno vieninieku kolonnu
eYV=exp(-pred11*svari_sleptais_izeja); %exp(YV)
pred=(oo./(oo+eYV)); %Z izeja no izejas
%delta
de = (pred - velama_atbilde).*pred
.*(ones(size(velama_atbilde))-pred);
%Delta
la=ones(patterns,sleptie_neironi);
svari_sleptais_izeja1=(repmat(svari_sleptais_izeja
(1:sleptie_neironi,1),1,patterns))';
de1=repmat(de,1,sleptie_neironi);
ris=de1.*svari_sleptais_izeja1;
De = (la-pred1.^2).*ris;
```

3. pielikums. *R* programmu kodi

```
data<-scan("usd.txt")
min.data<-min(data,na.rm=TRUE)
max.data<-max(data,na.rm=TRUE)
invnorm<-function(data){
data*(max.data-min.data)+min.data}
test<-read.table("izmusd.txt")
test.i.in<-invnorm(test[,1])
test.f.in<-invnorm(test[,2])
plot(test.i.in,type="l",ylab="EUR/USD kurss",xlab="t")
lines(test.f.in,col=3)
legend("bottomright",cex=0.9,c
("patiesa vertiba","prognoze"),lty=1,col=c(1,3))
mad<-function(istie,forecasts){
(1/length(forecasts))*(sum(abs(istie-forecasts)))
}
mad(test[,1],test[,2])
mse<-function(istie,forecasts){
(1/length(forecasts))*(sum((istie-forecasts)^2))
}
mse(test[,1],test[,2])
mape<-function(istie,forecasts){
(1/length(forecasts))*(sum(abs((istie-forecasts)/istie)))
}
mape(test[,1],test[,2])
##############1.modelis####################
data<-scan('gbp.txt')
norm <- function(x){
(x-min(x,na.rm=TRUE))/(max(x,na.rm=TRUE)-min(x,na.rm=TRUE))
}
data.n<-(norm(data))
#sagataves funkcijas izveidei palidziba tika
#mekleta interneta
sagatave <- function(data,hday=1,emb=9) {
ds <- data.frame(embed(data,emb+hday))
ds <- ds[,c(1,(1+hday):(hday+emb))]
}
data.visi<-sagatave(data.n)
data.visi<-data.matrix(data.visi)
data.train<-data.visi[1:1843,]
data.test<-data.visi[1843:2295,]
ieejas.sk<-10
sleptie.sk<-10
izeja.sk<-1
macisanas.atrums<-0.001
set.seed(1)
svari.ieeja.sleptais.g<-runif(ieejas.sk*sleptie.sk,-1,1)
svari.ieeja.sleptais<-matrix
(svari.ieeja.sleptais.g,ieejas.sk,sleptie.sk)
```

```
set.seed(21)
svari.sleptais.izeja<-runif(sleptie.sk+1,-1,1)
sigma<-function(x,a=1){
rep(1,a)/(rep(1,a)+exp(-x))
}
tanh<-function(x,a=1){
(exp(x)-exp(-x))/(exp(x)+exp(-x))
}
for (l in 1:2000){
izejas.vektors<-c()
kluda<-0
Y<-matrix()
for (i in 1:1843){
ieeja<-c(data.train[i,2:10],c(1))
velama.atbilde<-data.train[i,1]
sl.izeja<-sigma(ieeja%*%svari.ieeja.sleptais,sleptie.sk)
#izeja no slepta slana
Y<-c(matrix(Y),matrix(sl.izeja))
izeja<-sigma(c(sl.izeja[1,],c(1))%*%svari.sleptais.izeja)
#izeja no visa tikla
izejas.vektors[i]<-izeja
kluda<-kluda+abs(izeja-velama.atbilde)^2
}
kluda<-0.5*(kluda)
Y.g<-matrix(Y[2:length(Y)],sleptie.sk,1843) #visas izejas
delta<-((izejas.vektors-data.train[1:1843,1]))*izejas.vektors*
(rep(1,1843)-izejas.vektors)
Delta<-Y.g*(matrix(rep(1,sleptie.sk*1843),sleptie.sk,1843)-Y.g)*
t(delta*t(replicate(1843,svari.sleptais.izeja[1:sleptie.sk])))
svari.sleptais.izeja<-svari.sleptais.izeja-c(macisanas.atrums*
(delta%*%t(Y.g)),c(macisanas.atrums))
svari.ieeja.sleptais<-svari.ieeja.sleptais-macisanas.atrums*
rbind(t((Delta%*%data.train[1:1843,2:10])),c(rep(1,sleptie.sk)))
}
plot(data.train[,1],type="l")
lines(izejas.vektors,col=2)
plot(izejas.vektors,type="l",col=2)
#####prognozesana
data.test<-data.visi[1843:2295,]
izejas.vektors.te<-c()
Y.te<-matrix()
for (k in 1:453){
ieeja.te<-c(data.test[k,2:10],c(1))
sl.izeja.te<-sigma(ieeja.te%*%svari.ieeja.sleptais,sleptie.sk)
#izeja no slepta slana
Y.te<-c(matrix(Y.te),matrix(sl.izeja.te))
izeja.te<-sigma(c(sl.izeja.te[1,],c(1))%*%svari.sleptais.izeja)
#izeja no visa tikla
izejas.vektors.te[k]<-izeja.te
}
```

```
plot(data.test[,1],type="l",main="Prognoze",xlab="t",
ylab="EUR/GBP kurss")
lines(izejas.vektors.te,col=3)
legend("bottomright",cex=0.9,c("patiesa vertiba", "prognoze"),
lty=1, col=c(1,3))
###prognozesanas kludu aprekins
mad<-function(istie,forecasts){
(1/length(forecasts))*(sum(abs(forecasts-istie)))
}
mad(data.test[,1],izejas.vektors.te)
mse<-function(istie,forecasts){
(1/length(forecasts))*(sum((forecasts-istie)^2))
}
mse(data.test[,1],izejas.vektors.te)
mape<-function(istie,forecasts){
(1/length(forecasts))*(sum(abs((forecasts-istie)/istie)))
}
mape(data.test[,1],izejas.vektors.te)
########prognozesana nnet
neironi<-21
set.seed(12)
norm <- rnorm(10*neironi+neironi+1)
nn<-nnet(X1 X2+X3+X4+X5+X6+X7+X7+X8+X9+X10,
data=data.train,
linout=T,size=neironi,decay=0.01,maxit=3000)
pred <- predict(nn,data.test)
plot(data.test[,1],type="l",main="",xlab="t",
ylab="EUR/JPY kurss")
lines(pred,col=3)
legend("bottomright",cex=0.9,c("patiesa vertiba", "prognoze"),
lty=1, col=c(1,3))
############ARIMA###########
data1<-scan('jpy.txt')
data<-data1[1:1843]
library(tseries)
adf.test(data1)
acf(data, xlab=c("Lag"),main="")
data.d<-diff(data1)
adf.test(data.d)
library(forecast)
auto.arima(data)
acf(data.d, xlab=c("Lag"),main="")
pacf(data.d, xlab=c("Lag"),main="")
data.arima<-arima(data,order=c(0,1,0))
Box.test(data.arima$residuals)
shapiro.test(data.arima$residuals)
plot(data.arima$residuals, type="l")
acf(data.arima$residuals)
qqnorm(data.arima$residuals)
qqline(data.arima$residuals)
```

```
hist(data.arima$residuals, prob=TRUE, breaks=40)
curve(dnorm(x, mean=mean(data.arima$residuals),
sd=sd(data.arima$residuals)),
add=TRUE, col=3)
#lai veiktu prognozes arimai
#palidziba tika mekleta interneta
forecasts<-c()
a<-0
for (i in 1844:2304){
data.izm<-data1[1+a:i]
data.arima<-arima(data.izm,order=c(0,1,0))
forecasts[i-1843]<-forecast(data.arima, h=1)$mean
a<-a+1
}
data.test<-data1[1844:2304]
plot(data1[1844:2304], col="black",
type="l", ylab="EUR/JPY kurss",xlab="t")
lines(forecasts,type="l",col=3)
```

Bakalaura darbs “Valūtas kursu prognozēšana ar atgriezenisko izplatīšanās algoritmu” izstrādāts LU Fizikas un Matemātikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: Zane Balode

_____ _____

(paraksts) (datums)

Rekomendēju darbu aizstāvēšanai.

Vadītājs: prof. Dr.mat. Jānis Buls

_____ _____

(paraksts) (datums)

Recenzents: Mat.mag. Māra Vēliņa

_____ _____

(paraksts) (datums)

Darbs iesniegts Matemātikas nodaļā _____

(datums)

_____

(darbu pieņēma)

Darbs aizstāvēts Valsts pārbaudījuma komisijas sēdē

_____ prot. Nr. _____, vērtējums_____

(datums)

Komisijas sekretārs/-e: _____ _____

(Vārds, Uzvārds) (paraksts)