LATVIJAS UNIVERSITĀTE
FIZIKAS UN MATEMĀTIKAS FAKULTĀTE
MATEMĀTIKAS NODAĻA

NEIRONU TĪKLI UN TO LIETOJUMS LAIKRINDU PROGNOZĒŠANĀ

DIPLOMDARBS

Autors: **Zane Tiltānova**

Stud. apl. zt06003

Darba vadītājs: doc. Dr.math. Jānis Valeinis

RĪGA 2011

Anotācija

Darbs ir veltīts ar vienvirziena mākslīgo neironu tīklu saistīto jautājumu apskatei, uzmanību pievēršot kļūdu atgriezeniskās izplatīšanās apmācības algoritmam un neironu tīklu pielietojumam īstermiņa prognozēšanā. Darba mērķis ir noskaidrot, vai ar apskatīto mākslīgo neironu tīklu palīdzību USD/EUR, USD/GBP un JPY/USD valūtu kursu gadījumā iespējams iegūt precīzākas prognozes vienam laika solim uz priekšu nekā ar tradicionālajiem $ARIMA$ modeliem. Darba noslēgumā ir salīdzināti ar dažādām aprakstītajām metodēm iegūtie rezultāti.

Atslēgas vārdi: mākslīgie neironu tīkli, kļūdu atgriezeniskās izplatīšanās algoritms, prognozēšana, $ARIMA$ modeļi

Abstract

The thesis is dedicated to artificial neural networks and their application to short term time series forecasting with emphasis on the backpropagation learning algorithm. The purpose of this thesis is to find out whether the herein described neural networks, in case of USD/EUR, USD/GBP and JPY/USD, can show more precise forecasts for one time step than traditional $ARIMA$ models. Finally, the thesis also contains a comparison of the described models.

Keywords: artificial neural networks, backpropagation, forecasting, $ARIMA$ models

Saturs

Ievads

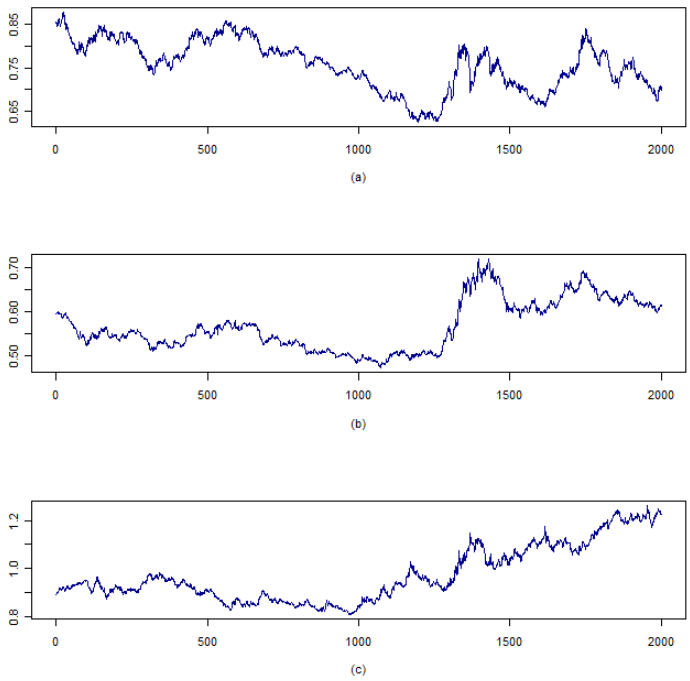
Valūtu tirgus šā brīža izpratnē ar peldošo valūtas kursu eksistē kopš 20.gadsimta septiņdesmitajiem gadiem, kad sabruka Bretonvudas sistēma, saskaņā ar kuru valstu valūtas bija piesaistītas ASV dolāram, kas savukārt bija piesaistīts zeltam. Protams, arī mūsdienās pastāv tāds jēdziens kā fiksētais valūtas kurss, piemēram, Latvijas lats ir piesaistīts eiro kursam. Tomēr, neskatoties uz to, šobrīd valūtu kursu svārstības ir lielākas nekā jebkad agrāk, un līdz ar pasaules ekonomikas globalizāciju ir pieaugusi arī valūtu tirgu nozīme, jo straujas valūtas kursa svārstības ietekmē gan starptautisku uzņēmumu peļņu, gan investīciju apjomus konkrētajā valstī un rezultātā arī pašu valsts ekonomiku. Tādēļ jau kopš pagājušā gadsmita astoņdesmitajiem gadiem pētnieki ir centušies izskaidrot valūtu kursu svārstības un prognozēt to dinamiku nākotnē.

Valūtu kursu svārstības ietekmē daudzi faktori, kas bieži vien ir cieši saistīti arī savā starpā. To vidū ietilpst ekonomiska, politiska un pat psiholoģiska rakstura faktori. Daži nozīmīgākie no tiem ir ekonomikas izaugsme, procentu likmju izmaiņas, un inflācija. Tieši daudzo faktoru un to savstarpējās saistības dēļ valūtas kursus prognozēt ir ļoti grūti. Valūtu kursu prognozēšanu sarežģī arī salīdzinoši lielās svārstības. Pie tam vairāki pētījumi rāda, ka daudzi modeļi nespēj sniegt labāku rezultātu kā gadījuma klejošanas modelis, [1, 65. lpp.].

Gadiem ilgi, runājot par valūtu tirgu, ir pastāvējuši divi pretēji viedokļi. Cilvēki, kas nodarbojas ar valūtu tirdzniecību ar mērķi nopelnīt no valūtu kursu svārstībām, uzskata, ka valūtas kurss seko tendencei, un tas savukārt ļauj veidot mehāniskas tirdzniecības sistēmas. No otras puses, pastāv uzskats, ka valūtu kursa svārstībām ir gadījuma raksturs, jeb ir spēkā tā sauktā *gadījuma klejošanas hipotēze*. Pirmo rezi ierosinājums, ka finanšu tirgi seko *gadījuma klejošanas procesam* parādījās 1900. gadā publicētajā Louis Bachelier darbā "*Theorie de la Speculation*", un vēlāk šis apgalvojums kļuva par daudzu pētījumu stūrakmeni dažādu finanšu instrumentu kontekstā. Būtībā gadījuma klejošanas hipotēze apgalvo, ka valūtu kursu svārstībām ir gadījuma raksturs un to prognozēšanā saistoša ir tikai iepriekšējā laikrindas vērtība. No šīs hipotēzes izriet, ka vērtību Y_t var izteikt kā

$$Y_t = Y_{t-1} + e_t,$$

kur e_t ir baltā trokšņa process ar $Ee_t = 0$, $De_t = \sigma^2$ un $Ee_te_{t-1} = 0$. Gadījuma klejošanas process $Y_t = Y_{t-1} + e_t$ tiek bieži pielietots nestacionāriem datiem, kuriem bieži vien ir labi



1. att. Valūtu kursu dinamika, (a) USD/EUR, (b) USD/GBP, (c) JPY/USD

saskatāmas ilgstošas tendneces, kas pēkšņi var strauji mainīt virzienu. Tā, piemēram, Bekaert un Hodrick (1992), Fong un Ouliaris (1995), LeBarton (1999), Levich un Thomas (1993), Liu un He (1991), McCurdy un Morgan (1988), Baillie un Bollerslev (1989), Sweeney (1986) un Soofi (2006) savos pētījumos, apskatot dažādu valūtu kursus, ir atraduši pierādījumus, kas ir pretrunā ar gadījuma klejošanas hipotēzi. Savukārt Diebold un Nason (1990), Fong, Koh un Oularis (1997), Hsieh (1988, 1989, 1993), McCurdy un Morgan (1987), kā arī Meese un Rigoff (1983a, 1983b) savos pētījumos neatrada pierādījumus, kas ir pretrunā ar gadījuma klejošanas hipotēzi, [1, 3. lpp.]. Tomēr, ņemot vērā jau pieminēto valūtas tirgus nozīmību, pētījumi par iespējām prognozēt valūtas kursu nezaudē savu aktualitāti par spīti neviennozīmīgajiem rezultātiem.

Bieži literatūrā saistībā ar laikrindu prognozēšanu tiek minēti Boksa un Dženkinsa uzvārdi un slavenais Boxa-Dženkinsa modelis, skatīt [2]. Taču kā viena no valūtas kursa prognozēšanas iespējām tiek piedāvāta arī neironu tīklu izmantošana, kas ir šī darba centrālā tēma. Salīdzinoši nesen veiktie pētījumi liecina, ka strap nelineārajiem modeļiem viena no alternatīvām prognožu veidošanai ir mākslīgie neironu tīkli. Iemesls, kas literatūrā tiek minēts kā arguments par labu neironu tīklu izmantošanai, ir atsevišķas neironu tīklu īpašības, kas ļauj modelēt sarežģītas sakarības. Kā vienu no šādām īpašībām var

minēt faktu, ka neironu tīkls kalpo kā universāls funkciju aproksimators, kas bez jebkādiem pieņēmumiem attiecībā uz datiem spēj attēlot jebkuru nelineāru funkciju, [1, 65. lpp.]. Pirmo rezi daudzslāņu vienvirziena neironu tīklu (*multi layer feedforward neural network*) nelineāru signālu prognozēšanai 1987.gadā izmantoja Lapedes un Farber, skatīt [3]. Kopš tā laika daudzi pētījumi ir apliecinājuši neironu tīklu lietošanas lietderību laikrindu prognozēšanā, [1, 4. lpp.]. Tiesa gan neviena no piedāvātajām prognozēšanas tehnikām, izmantojot neironu tīklus, nav spējusi 100% gadījumu pārspēt citas metodes, [1, 4. lpp.]. Kā viens no neironu tīklu mīnusiem tiek minēts fakts, ka neironu tīkls darbojas pēc "melnās kastes" principa, kas var apgrūtināt iegūto rezultātu interpretāciju.

1. tabula: *[1] autoru apkopotā informācija par 45 dažādiem pētījumiem, kas publiskoti dažādos zinātniskajos žurnālos, piemēram, Journal of Forecasting un Computers & Operations Research, laika posmā no 1995.gada līdz 2004.gadam*

Kategorija	Publikāciju skaits	% no kopējā publikāciju skaita
I	27	60.00
II	2	4.44
III	16	35.56

I Neironu tīkli uzrāda labākus rezultātus nekā citi modeļi;

II Neironu tīkli neuzrāda labākus rezultātus nekā citi modeļi;

III Neironu tīklu uzrādītais sniegums attiecībā pret citiem modeļiem atkarīgs no situācijas.

Valūtu tirgus. No pasaules finanšu tirgiem nenoliedzami lielākais un likvīdākais ir tieši valūtu tirgus. Pēc Starptautiskās norēķinu bankas (*angl. Bank for International Settlements*) aptaujas datiem, vidējais dienas apgrozījums pasaules valūtu tirgū 2010. gada aprīlī bija 4 triljoni ASV dolāru, kas ir par 20% vairāk, salīdzinot ar 2007. gada aprīli, kad tika veikta iepriekšējā aptauja. Ar aptaujas rezultātiem iespējams iepazīties [4]. Salīdzinājumam, pēc Starptautiskās fondu biržu federācijas (*angl. World Federation of Exchanges*) datiem, mēneša apgrozījums pasaules akciju tirgos 2011. gada aprīlī bija nedaudz virs 5 triljoniem ASV dolāru. Kaut arī valūtu gadījumā nevar runāt par vienu

centralizētu tirgu, jo lielākā daļa darījumu tiek noslēgti starpbanku tirgos, t.i. ārpus regulētajām biržām, turpmāk tekstā vārdu salikums "pasaules valūtu tirgus" tiks lietos, lai apzīmētu visu šo starpbanku tirgu kopumu, tātad pasaules valūtu tirgu.

Pirmsākumi aktīvai valūtu tirdzniecībai meklējami 1973. gadā, kad sabrukumu piedzīvoja jau pieminētais Bretonvudas akords, kas iepriekš paredzēja valūtu ciešu piesaisti ASV dolāram, kurš savukārt bija piesaistīts zeltam. Galvenais Bretonvudas akorda mērķis bija nodrošināt stabilu pasaules ekonomikas izaugsmi pēc Otrā pasaules kara.

Šodien valūtu tirdzniecība visaktīvāk iesaistās pasaules centrālās bankas, starptautiskie uzņēmumi, komercbankas, kā arī ieguldījumu fondi un citas organizācijas. Protams, nenoliedzami, savu lomu spēlē arī spekulanti. Katram darījumam ir divas puses un divas valūtas.

Pēc *Euromoney* datiem, pasaules valūtu tirgū desmit nozīmīgākie dalībnieki šobrīd ir Deutsche Bank, Barclays, UBS, Citi, JPMorgan, HSBC, Royal Bank of Scottland, Credit Suisse, Goldman Sachs un Morgan Stanley.

Ņemot vērā pasaules valūtu tirgus likviditāti, liela daļa darījumu, kas tiek veikti šajā tirgū, ir darījumi ar aizņemtiem līdzekļiem. Būtībā tie ir darījumi, kuros darījuma puses aizņemās naudu no citiem tirgus dalībniekiem, lai varētu veikt plānotās transakcijas. Tādā veidā pasaules valūtu tirgus, bez valūtas maiņas funkcijas, var piedāvāt tā dalībniekiem vēl vienu svarīgu funkciju - palīdzēt izvairīties no nevēlamiem valūtas riskiem.

Piemēram, ja kāds uzņēmums nākotnē gaida maksājumu ārvalstu valūtā, un ir pamats satraukumam par iespējamo valūtas kursu par kuru nāksies mainīt šo valūtu, uzņēmums var doties uz valūtu tirgu, aizņemties no kāda dalībnieka līdzīgu summu un apmainīt to pret sev nepieciešamo valūtu jau šodien, maksājot par aizdevumu tikai bāzes procentu likmju starpību, kāda pastāv starp abām valūtām konkrētajā brīdī. Vēlāk, kad ienāks gaidāmais maksājums, uzņēmums var samainīt valūtas atpakaļ, atgriežot aizdevumu un realizējot peļņu vai zaudējumus, atkarībā no tā, kur būs valūtas kurss transakcijas datumā, un paralēli konvertēt arī ienākošo maksājumu. Rezultātā tas, ko uzņēmums iegūs vai zaudēs no reālās valūtas konvertācijas tiks neitralizēts ar darījumu valūtu tirgū. Tādā veidā, ar ļoti vienkāršu darījumu, uzņēmums ir pasargājis sevi no nevēlama riska. Pieņemsim, ka EUR un USD bāzes procentu likmes ir vienādas un pašreizējais valūtas kurss EUR/USD = 1,45 (1 EUR = 1,45 USD). Uzņēmums kaut kad nākotnē plāno saņemt 1 000 000 EUR. Pie pašreizējā kursa tie būtu 1 450 000 USD. Gadījumā, ja uzņēmums

prognozē EUR vērtības kritumu, t.i. viens eiro nākotnē spēs nopirkt mazāk dolārus, tas var doties uz valūtu tirgu un no kāda tirgus dalībnieka aizņemties 1 000 000 EUR un tajā pašā momentā samainīt tos pret 1 450 000 USD. Pieņemot, ka brīdī, kad uzņēmums saņems plānoto 1 000 000 EUR maksājumu EUR/USD = 1,40, konvertējot valūtu, tas saņems par 50 000 USD mazāk. Taču tā kā uzņēmumam jau iepriekš valūtu tirgū bija nopirkti 1 450 000 USD, kas tagad ir kļuvuši vērtīgāki, lai atgrieztu 1 000 000 EUR aizdevumu, tam ir nepieciešami vairs tikai 1 400 000 USD. Rezultātā uzņēmums var paturēt starpību. Tā kā EUR un USD bāzes procentu likmes šajā gadījumā ir vienādas, to, ko uzņēmumam nāksies maksāt par 1 000 000 EUR aizdevumu, tas saņems vienkārši turot nopirktos 1 450 000 USD. Rezultātā 50 000 USD, ko uzņēmums ir zaudējis konvertējot ienākošo maksājumu nākotnē par neizdevīgāku kursu, tas ir atguvis valūtu tirgos.

Šādi darījumi ir iespējami, jo parasti ir nepieciešams nodrošināt tikai 1% no aizdevuma, kā garantijas depozītu nelabvēlīgu valūtas kursa svārstību gadījumam, jo kā galvenais nodrošinājums kalpos nopirktā valūta. Šo iespēju aizņemties krietni vairāk par paša kapitālu finanšu tirgos dēvē par kredītsviru (*angl. leverage*), un tā ir ļoti būtiska spekulantu piesaistei, kuri būs gatavi nostāties darījuma otrā pusē. Ar jēdzienu "spekulants" šī darba ietvaros tiek saprasta privātpersona, kas nodarbojas ar valūtu tirdzniecību ar mērķi nopelnīt no valūtu kursu svārstībām, un uzskata, ka valūtas kurss seko tendencei, kas savukārt ļauj veidot mehāniskas tirdzniecības sistēmas.

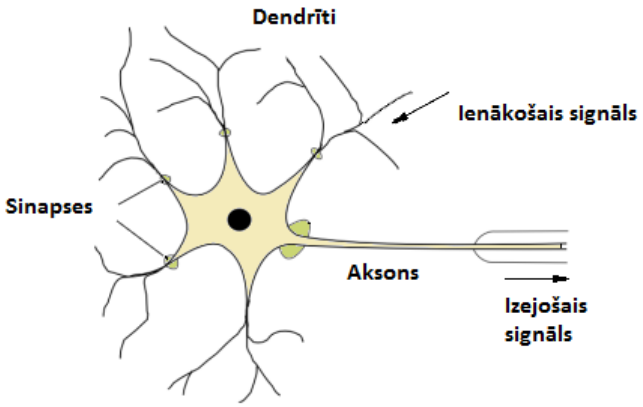
Ņemot vērā faktu, ka tā dēvētie spekulanti izmanto kredītsviru, tad jau ļoti mazas valūtu kursu svārstības var ievērojami palielināt peļņu vai zaudējumus. Tieši tādēļ īpaša uzmanība tiek pievērsta dažādām analīzes metodēm. Visbiežāk tās ir tehniskā un fundamentālā analīze, kas attiecīgi pievērš uzmanību kursu tendencei un ekonomiskājiem faktoriem. Šī darba ietvaros tiks meklēta alternatīva metode tradicionālajiem tehniskās analīzes indikatoriem, kurus iegūst izmantojot vēsturiskos kursu datus.

1. Neironu tīkli

1.1. Mākslīgo neironu tīklu attīstības vēsture

Mākslīgie neironu tīkli (*angl. artificial neural networks*) ir informācijas apstrādes rīks, kura izveidi ir iedvesmojuši pētījumi par cilvēka smadzeņu un nervu sistēmas darbību un kura mērķis ir izveidot skaitļošanas sistēmu, kas būtu līdzīga cilvēka smadzenēm. Neironu

tīklu veido liels skaits savā starpā cieši saistīti informācijas apstrādes elementi jeb neironi, kas kopā veic konkrētu uzdevumu, tendences atpazīšanu un prognozēšanu, optimizāciju vai datu klasifikāciju.



2. att. Bioloģiskā neirona uzbūve

Bioloģiskais neirons ir nervu sistēmas pamatvienība, un tas sastāv no šūnas ķermeņa un izaugumiem, kas vada nervu impulsus. Ienākošie signāli nervu šūnā nonāk caur sinapsēm, bet izejošais signāls tiek izvadīts caur aksonu, kas tālāk sazarojas un signālu novada uz citiem neironiem. Neironi savā starpā ir savienoti tīklā. Viena neirona aksons ar cita neirona dendrītu ir savienots ar sinapsi, un sinapses viena no otras atšķiras ar caurlaidību. Tādēļ vienāda stipruma impulsi, nonākot neironā caur dažādām sinapsēm, to var kairināt dažādās pakāpēs, un sinapses ietekme uz signālu var būt gan pastiprinoša, gan pavājinoša. Tādējādi katram neironā ienākošajam signālam var piekārtot skaitlisku koeficientu, kā tas tiek darīts mākslīgajos neironu modeļos, [5]. Caur sinapsēm ienākošie signāli neironu regulāri kairina, un, ja kairinājums pārsniedz noteiktu pakāpi, neirons izejā, atkarībā no tā, cik lielā mērā šis slieksnis ir pārsniegts, dod signāla impulsu. Pretēji - ja neirona kairinājuma līmenis nepārsniedz noteikto slieksni, signāla impuls netiek padots, un neirons atgriežas iepriekšējā stāvoklī.

Mākslīgie neironu tīkli (turpmāk tekstā ANN) ir mākslīgā intelekta zinātnes apakšnozare, un mākslīgā intelekta zinātnes pirmsākumi ir meklējami jau 1950.gadā, kad grāmatu "*Computing Machinery and Inteligence*" publicēja Alans Tjūrings. Savukārt jēdziens ANN parādījās jau 1943.gadā, kad Makalons un Pitss publiskoja savu darbu ar nosaukumu "*A logical calculus of the ideas immanent in nervous activity*", kurā tika aprakstīts pirmais neironu tīklu modelis. Tiesa gan pētījumi šajā nozarē tika lielā mērā pārtraukti

1969.gadā, kad Minsks un Peiperts parādīja, ka Frena Rosenblata 1958.gadā izveidotajam preceptrona modelim ir vairāki ierobežojumi un tas nespēj risināt izslēdzošā VAI problēmu *(XOR)*. Situācija mainījās aptuveni 15 gadus vēlāk, par ko jāpateicas datorzinātnes attīstībai un progresam cilvēka smadzeņu un prāta darbības izpētē. Rezultātā šo 15 gadu laikā tika pārvarēti vairāki teorētiski šķēršļi, un 1986.gadā panīkumu neironu tīklu zinātnē pārtrauca D.E. Rumelharta, G.E. Hintona un R.J. Viljamsa piedāvātais kļūdu atgriezeniskās izplatīšanās (*angl. backpropagation*) algoritms daudzslāņu preceptrona apmācībā. Šeit jāpiebilst, ka minētais algoritms paralēli tika atklāts arī citviet, taču tā kā šie trīs autori parādīja to darbībā, viņiem tiek piešķirti arī vislielākie nopelni. Arī šobrīd līdz ar datorzinātnes attīstību un plašākas sabiedrības interesi, kas savukārt nodrošina vajadzīgo finansējumu pētījumiem, šī zinātnes nozare turpina attīstīties.

2. tabula *Cilvēka smadzenes salīdzinājumā ar datoru*

Kategorija	Cilvēka smadzenes	Dators
Informācijas apstrādes ātrums	100 Hz	10^9 Hz, PC
Neironi/tranzistori	100 miljardi	100 miljoni, PC
Svars	1500 g	sākot no 1kg
Enerģijas patēriņš	20 džouli sekundē	200 džouli sekundē, PC

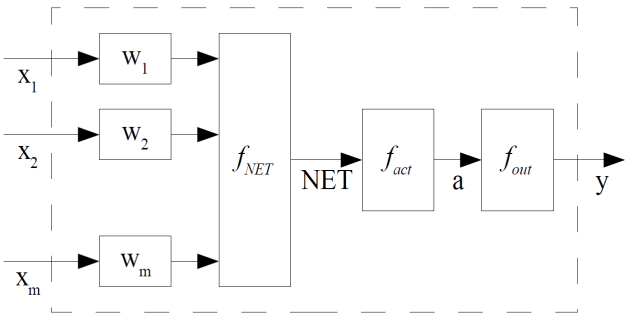
Nedaudz atgriežoties pie ANN salīdzinājuma ar cilvēka smadzeņu darbību, jāsaka, ka, salīdzinot ar cilvēka smadzenēm, ANN pagaidām ir ļoti primitīvi. Būtībā pēc [5] avota autora domām ANN ar cilvēka smadzenēm ir kopīgas tikai divas īpašības:

- Apmācība kā informācijas ieguves veids. Neironu tīkls zināšanas iegūst apmācības ceļā;
- Neironu tīkls sastāv no liela daudzuma neironu, un to savstarpējo saišu stiprums nosaka tīklā glabātās zināšanas.

1.2. Neironu tīkla uzbūve

Pats neirons ir salīdzinoši vienkāršs skaitļošanas elements ar vairākām ieejām un vienu izeju, kas atgādina daudzargumentu funkciju. Neirons sastāv no sekojošiem elementiem:

- Svariem;
- Summēšanas funkcijas (*angl. propagation function*);
- Aktivizācijas funkcijas;
- Izejas funkcijas;
- Apmācības likuma.



3. att. Neirona modelis

Šīs nodaļas ietvaros pamatā tika izmantota informācija no Jāņa Zutera, skatīt [5], sagatavotā mācību materiāla par neironu tīkliem.

Svari un summēšanas funkcija

Svari ir skaitliskas vērtības, kas tiek iegūtas apmācības procesā un nodrošina to, ka neironu tīkls apmācības procesa beigās ir ieguvis spēju risināt konkrētu problēmu. Svaru skaits parasti atbilst ieeju skaitam, jo katrai ieejai atbilst tieši viens svars. Atsevišķos gadījumos var tikt pielietots arī papildus svars. Svarus parasti apzīmē ar w_i, kur $i = 1, ..., m$ un m ir svaru skaits. Lai apzīmētu visa neironu tīkla svaru kopumu, lieto matricu pierakstu, un atbilstošo matricu apzīmē ar W.

Lai no svariem un ieejas signāliem x_i izrēķinātu vienu vērtību, kas tālāk piedalīsies neirona izejas vērtības izskaitļošanā, tiek lietota summēšanas funkcija. Summēšanas funkcijas atrasto vērtību apzīmē ar burtu virkni net, savukārt pašu funkciju ar f_{net}. Tipiskākās summēšanas funkcijas parādītas formulās (1.1- 1.5).

$$net = \sum_{i=1}^{m} w_i x_i, \tag{1.1}$$

$$net = \prod_{i=1}^{m} w_i x_i, \tag{1.2}$$

$$net = \max(w_i x_i),\ i = 1, ..., m, \tag{1.3}$$

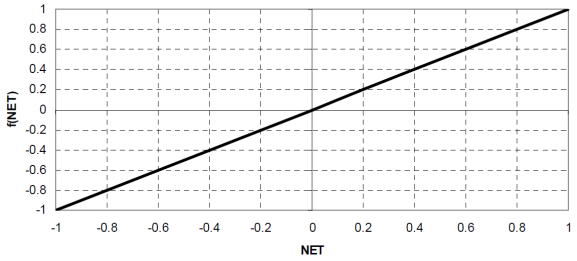
$$net = \min(w_i x_i),\ i = 1, ..., m, \tag{1.4}$$

$$net = \sqrt{\sum_{i=1}^{m} (w_i - x_i)^2}. \tag{1.5}$$

Aktivizācijas un izejas funkcijas

Ar aktivizācijas funkcijas f_{act} palīdzību, izmantojot summēšanas funkcijas vērtību net, tiek iegūts aktivizācijas stāvoklis, a. Aktivizācijas funkcija var būt gan lineāra, gan nelineāra (sliekšņveida un sigmoidāla), formulā (1.6) parādīts triviālākais aktivizācijas funkcijas veids.

$$f_{act}(net) = net. \tag{1.6}$$



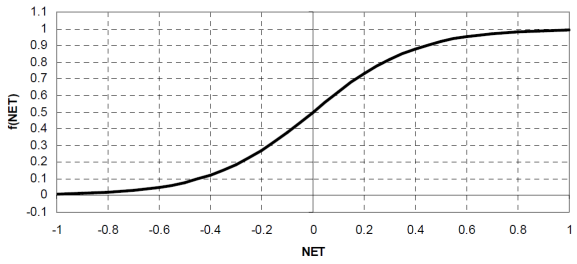
4. att. Lineāra aktivizācijas funkcija

Starp sigmoidālajām aktivizācijas funkcijām populārākās ir loģistiskā aktivizācijas funkcija (1.7) un hiperboliskais tangenss (1.8). Sigmoidālo funkciju vērtīga īpašība ir tāda, ka to atvasinājums ir izsakāms ar pašas funkcijas vērtību, kas, kā vēlāk tiks parādīts, ir nozīmīgi svaru izmaiņu noteikšanā.

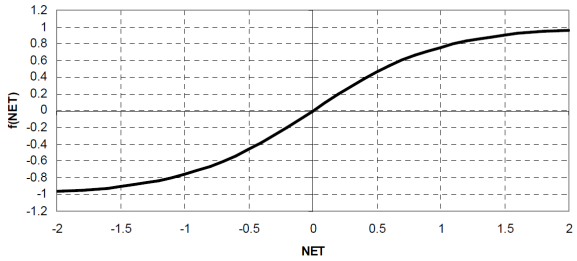
$$f_{act}(net) = \frac{1}{1 + e^{-\frac{1}{g} net}}, \tag{1.7}$$

$$f_{act}(net) = \frac{e^{net} - e^{-net}}{e^{net} + e^{-net}}. \tag{1.8}$$

Izejas funkcija f_{out} savukārt, izmantojot aktivizācijas stāvokli a, izrēķina neirona izejas vērtību y, kas reizē var būt arī cita neirona ieeja. Tiesa gan daļā neironu izejas funkcija



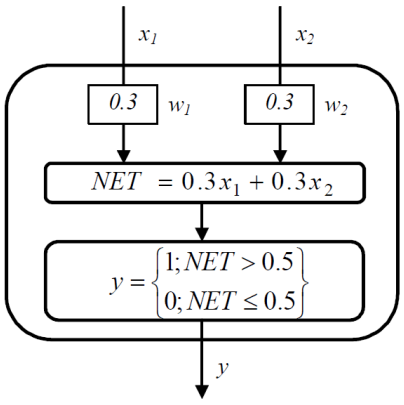
5. att. Loģistiskā aktivizācijas funkcija, g=0.2



6. att. Hiperboliskā tangensa aktivizācijas funkcija

netiek atdalīta no aktivizācijas funkcijas, un a ir ne tikai aktivizācijas stāvoklis, bet arī izejas vērtība.

Kā piemērs tiks apskatīts neirons, kas modelē divargumentu loģisko funkciju "UN". TRUE apzīmē ar 1, FALSE ar 0. Neironam ir divas ieejas un viena izeja. Par summēšanas funkciju tiek lietota (1.1), un par aktivizācijas funkciju tiek lietota sliekšņveida funkcija.



7. att. Neirons, kas modelē loģisko "UN" funkciju

1.3. Neironu tīkla arhitektūra

Definīcija 1. [5] *Neironu tīkla arhitektūra* ir neironu izvietojums tīklā un to savstarpējās

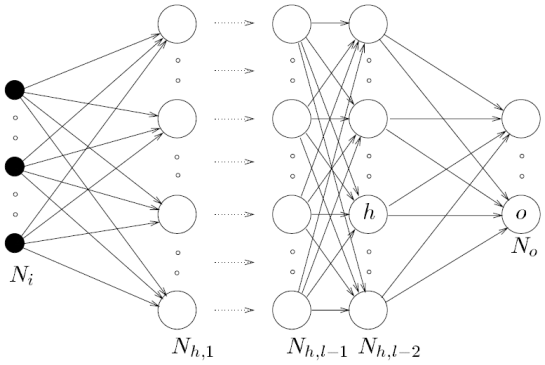
3. tabula *Konstruētā neirona pārbaude*

x_1	x_2	NET	y
0	0	0	0
0	1	0.3	0
1	0	0.3	0
1	1	0.6	1

saites.

Neironu tīkla arhitektūra ir viens no būtiskākajiem aspektiem, kas nosaka neironu tīkla darbības spēju. Neironi neironu tīklā parasti tiek grupēti slāņos.

Definīcija 2. [5] *Neironu slānis* ir neironu kopums, kas ir izvietoti kopā un kuru darbināšana un apmācība parasti notiek pēc viena algoritma.



8. att. Daudzslāņu neironu tīkla arhitektūra, [6]

Tātad neironu tīkla arhitektūru nosaka neironu slāņu savstarpējais izvietojums un to neironu sasaiste, kā arī neironu sasaiste slāņa ietvaros. Tipiska vairāku slāņu neironu tīkla arhitektūra redzama 8. attēlā. Pēc neironu slāņu izkārtojuma neironu tīkli tiek iedalīti divos galvenajos veidos:

- Vienvirziena tīkli (*angl. feedforward networks*),
- Rekursīvie tīkli (*angl. recurrent/feedback networks*).

Tā kā šī darba ietvaros tiks aprakstīti tikai vienvirziena tīkli, tad tālāk plašāk tiks aprakstīts tikai šis neironu tīklu veids.

Vienvirziena neironu tīkli

Vienvirziena tīklos neironu slāņi ir izkārtoti virknē pēc kārtas, un var izdalīt ieejas slāni, izejas slāni un slēptos slāņus. Slāņus to darbināšanas virzienā var sanumurēt, $l = 1$ piešķirot ieejas slānim, un $l = m$ piešķirot izejas slānim. Saites starp neironiem ir tikai vienā virzienā no slāņa ar mazāku numuru uz slāni ar lielāku numuru. 8. attēlā redzams l slāņu vienvirziena neironu tīkls.

Definīcija 3. [5] *Izejas slānis* ir neironu slānis, kas tieši saistīts ar apkārtējo vidi, kur tiek aizvadītas izejas vērtības. Neironu slānis, kura neironu izejas ir arī visa neironu tīkla izejas.

Definīcija 4. [5] *Ieejas slānis* ir tāds neironu slānis (iespējams mākslīgi veidots), uz kura neironu izejām tiek uzstādītas visa neironu tīkla ieejas.

Definīcija 5. [5] *Slēptais slānis* ir tāds neironu slānis, kas nav ne ieejas, ne izejas slānis.

Definīcija 6. [5] *Vairākslāņu neironu tīkls* ir tāds neironu tīkls, kuram ir vismaz viens slēptais slānis.

1.4. Neironu tīkla apmācīšanās

Literatūrā saistībā ar neironu tīklu darbināšanu eksistē divi termini - apmācība (*angl. trainig*) un apmācīšanās (*angl. learning*).

Definīcija 7. [5] *Apmācība* ir neironu tīkla svaru vērtību, dažreiz arī citu parametru, uzstādīšana, balstoties uz apmācības paraugiem, kas reprezentē noteiktu problēmu.

Definīcija 8. [5] *Apmācīšanās* ir process, kurā notiek neironu tīkla brīvo parametru pielāgošana, kas tiek veikta simulāciju vidē, kurā neironu tīkls ir ievietots.

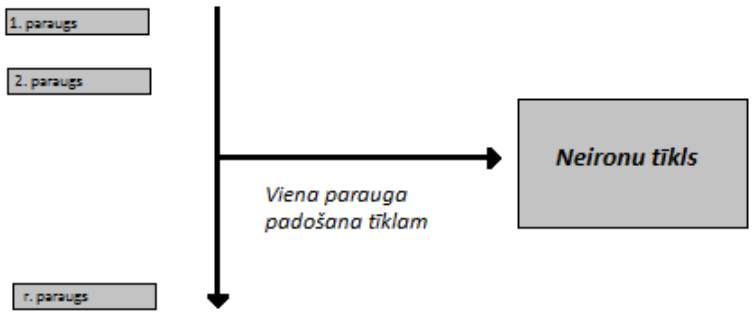
Apmācības process nosaka šādu notikumu virkni, [5]:

1. Neironu tīkls tiek darbināts noteiktā vidē;
2. Balstoties uz šo darbināšanu, neironu tīklā tiek veikas izmaiņas brīvajos parametros;
3. Ņemot vērā veiktās izmaiņas neironu tīkla iekšējā struktūrā, neironu tīkls uz vidi reaģē citā veidā.

Definīcija 9. [5] *Apmācības process* ir darbību kopums, kura laikā neironu tīklam noteiktā secībā vairākkārtīgi tiek padoti apmācības paraugi, pēc katra apmācības parauga vai paraugu kopuma padošanas katram neironam piemērojot apmācības algoritmu.

Apmācības procesa sākumā neironu tīkla svari noteiktā veidā, parasti nejauši, tiek aizpildīti ar sākuma vērtībām. Tālāk apmācības process parasti notiek pa epohām *(angl. epoch)*.

Definīcija 10. [5] *Epoha* ir apmācības procesa daļa, kuras laikā neironu tīklam tieši vienu reizi tiek padoti visi apmācības paraugi.



9. att. Epoha

Tikai neliela daļa apmācības algoritmu dod iespēju apmācīt neironu tīklu vienas epohas laikā, t.i., vienā solī. Parasti apmācības process ir garāks, un katrs paraugs tā laikā tiek vairākas reizes parādīts neironu tīklam.

Apmācības procesa organizēšanu var iedalīt divās metodēs:

- vienlaidus (*angl. continuous*) metode, kad svaru izmaiņa tiek veikta pēc katra parauga parādīšanas tīklam;
- pakešu (*angl. batch*) metode, kad svaru izmaiņa tiek veikta tikai epohas beigās, jeb pēc visu paraugu parādīšanas.

Tāpat tiek izšķirtas arī vairākas apmācīšanās kategorijas:

- apmācīšanās ar skolotāju, jeb uzraudzītā apmācīšanās (*angl. supervised learning*);
- apmācīšanās bez skolotāja;

- apmācīšanās ar pastiprinājumu;
- pašorganizēšanās, jeb neuzraudzītā apmācīšanās (*angl. unsuprevised learning*).

Šī darba ietvaros tiks izmantota vienlaidus metode un uzraudzītā apmācīšanās.

2. Datu sagatavošana

Pirms neironu tīklu pielietošanas, nozīmīgs un kritisks solis ir datu sagatavošana, un galvenais iemesls tam ir fakts, ka ieejas datu kvalitāte var spēcīgi ietekmēt neironu tīkla rezultātus, [1, 39.–40. lpp]. Turlāt pienācīgi sagatavoti dati krietni atvieglo datu analīzi. Runājot par neironu tīkliem, būtībā var izdalīt četrus galvenos procesus:

1. Problēmas identifikācija;
2. Datu sagatavošana;
3. Neironu tīkla konstruēšana;
4. Datu analīze.

Pirmajā posmā tiek definēta problēma un formulēti sagaidāmie rezultāti. Otrā procesa mērķis ir kvalitatīvu datu sagatavošana, un trešā procesa ietvaros tiek apmācīts neironu tīkls. Ja sistēma darbojas atbilstoši iepriekš noteiktajām prasībām, vispārinātie rezultāti var tikt lietoti tālākā datu analīzē, piemēram, klasifikācijas uzdevumos un lēmumu pieņemšanā.

Kā parādīts [7] avota autoru piedāvātajā datu sagatavošanas shēmā, šo procesu var iedalīt trīs posmos. Pirmais no tiem ir nepieciešamo datu un to formāta definēšana un datu iegūšana, kā arī datu integrēšana, ja dati tiek iegūti no dažādiem avotiem. Otrais posms ir datu pirmsapstrāde, kas sastāv no datu kvantitātes un kvalitātes izpētes. Šajā posmā dažas no tipiskākajām problēmām ir pārāk daudz datu, pārāk maz datu, liels tā saukto izlecēju skaits, trūkstoši dati un dati, kuri nenāk no vienas skalas, savukārt risinājumi šīm problēmām attiecīgi ir izlases veidošana, datu atkārtota vākšana, izlecēju nolīdzināšana, datu labošana un datu pielāgošana pēc vienas skalas. Trešā posma ietvaros tiek veikta datu sadale apmācīšanas un testēšanas kopās (atsevišķos gadījumos tiek veidota arī datu kopa sistēmas darbības pārbaudei). Līdz šim nav izveidots universāls likums, kā būtu jāsadala dati pirms neironu tīkla darbināšanas. Parasti apmācīšanas datu kopa ir krietni

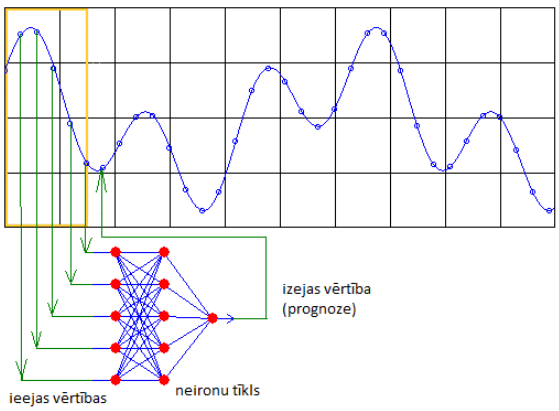
lielāka nekā testēšanas kopa. Tā kā šī darba mērķis ir aprakstīt neironu tīklu pielietojumu valūtu kursu prognozēšanā, datu sagatavošanas temats tiek atstāts lasītājam tālākai izpētei.

Darbā tiks izmantoti USD/EUR, GBP/USD un JPY/USD vēsturiskie dienas dati par iepriekšējām 2000 dienām, sākot no 20/05/2011. Ērtības labad JPY/USD kursu vēsturiskās vērtības tika pareizinātas ar 100. Protams, realitātē valūtu tirgū atšķirībā no akciju tirgus nepastāv tāds jēdziens kā dienas pēdējā cena, jo tirdzniecība norit 24 stundas dienā. Tāpēc šajā darbā izmantotajiem dienas datiem par atskaites punktu tika ņemta ASV tirdzniecības sesija.

4. tabula *Daži darbā izmantoto datu raksturlielumi*

Dati	Vidējā vērtība	Maksimālā vērtība	Minimālā vērtība	Standartnovirze
USD/EUR	0.7566	0.8766	0.3261	0.0569
USD/GBP	0.5677	0.7204	0.4725	0.0573
JPY/USD	0.9726	1.2639	0.8078	0.1176

Ja vien netiks citādi norādīts, turpmāk darbā prognozēšanā tiks izmantotas laikrindas vēsturiskās vērtības. Datu sagatavošanas shēma darbā lietoto neironu tīklu gadījumā tiek saukta par iegultu veidošanu ar laika nobīdi (*angl. time delay embedding*). Plašāku informāciju meklēt, piemēram, [8]. Darba ietvaros iegultas dimensija ir 9, jo [1] literatūras avota autoru veiktais pētījums liecina, ka valūtu kursu prognozēšanā ar neironu tīklu palīdzību labāko rezultātus sniedz tieši 9 iepriekšējās vērtības. Idejiski šī pieeja atspoguļota 10. attēlā.



10. att. Neironu tīkla apmācīšana ar vēsturiskajiem datiem

Piemēram, ja mūsu rīcībā ir x_t vērtības, kur $t = 1, ..., 9$, un prognozēšanai mēs vēlamies izmantot 3 iepriekšējās vērtības (dimensija ir 3), tad prognozēšanai izmantoti dati būtu sekojošā formā:

$$\begin{bmatrix} x_3 & x_2 & x_1 \\ x_4 & x_3 & x_2 \\ x_5 & x_4 & x_3 \\ x_6 & x_5 & x_4 \\ x_7 & x_6 & x_5 \\ x_8 & x_7 & x_6 \end{bmatrix} \longrightarrow \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix}$$

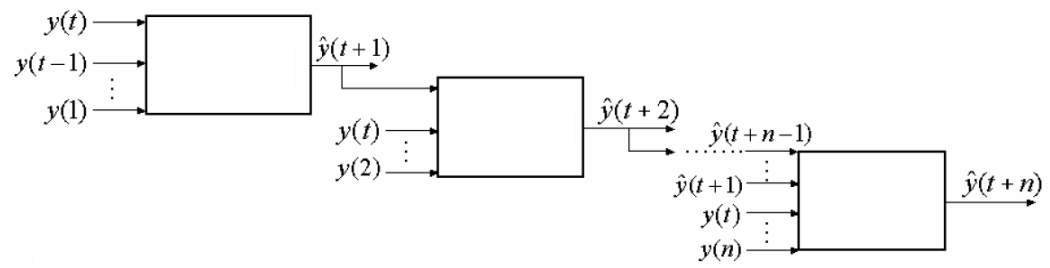
Neironu tīklu apmācīšanai darba ietvaros tiks lietoti 80% no visiem datiem, savukārt testēšanai atlikušie 20%.

3. Kļūdu atgriezeniskās izplatīšanās neironu tīkla modelis

Pirms tālāka kļūdu atgriezeniskās izplatīšanās neironu tīkla modeļa apraksta (turpmāk tekstā BPNN), atgriezīsimies pie darba mērķa, kas ir izveidot modeli, kas spētu prognozēt valūtu kursu. [1] grāmatas autoru veiktais pētījums par iespējām prognozēt valūtu kursu ar neironu tīklu palīdzību, kura ietvaros tika apskatītas 45 dažādas publikācijas (piemēram, no tādiem zinātniskajiem žurnāliem kā *Journal of Forecasting* un *Computers & Operations Research*) laika posmā no 1995.gada līdz 2004.gadam, liecina, ka vislabākos rezultātus ir sniedzis trīs slāņu vienvirziena neironu tīkls ar kļūdu atgriezeniskās izplatīšanas apmācīšanās algoritmu, sigmoidālo un hiperboliskā tangensa aktivizācijas funkciju slēptajos slāņos un lineāru izejas funkciju. Līdz ar to arī šajā darba tiks sīkāk apskatīts tikai šis viens neironu tīkla veids ar trīs dažādām modifikācijām.

Lai gan kļūdu atgriezeniskās izplatīšanās metode var tikt pielietota tīkliem ar jebkuru slāņu skaitu, ir parādīts, ka, lai aproksimētu funkciju ar ierobežotu pārrāvumu skaitu līdz iepriekš noteiktai precizitātei, pieņemot, ka slēptā slāņa neironu aktivizācijas funkcijas ir nelineāras, pietiek ar vienu slēpto slāni, [6, 33. lpp.]. Arī lielākajā daļā praktisku uzdevumu tiek lietos vienvirziena neironu tīkls ar vienu slēpto slāni un sigmoidālo neironu aktivizācijas funkciju.

Tā kā apskatītajā literatūrā ar neironu tīklu palīdzību tika veiktas prognozes vienam laika solim uz priekšu, tad arī šī darba ietvaros tiks prognozēta tikai y_{t+1} vērtība. Pēc savas būtības vairāku soļu prognozēšana ar neironu tīklu palīdzību ir sarežģīts process, ko iespējams veikt, piemēram, ar sērijveida izplatīšanās palīdzību, skatīt 11., taču šis temats tiek atstāts lasītājam tālākai izpētei.

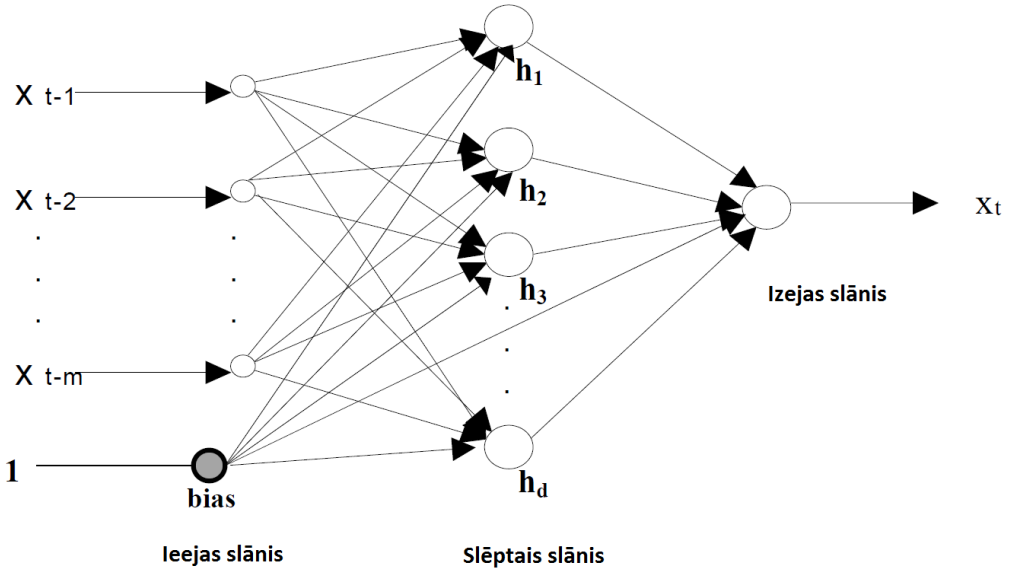


11. att. Sērijveida izplatīšanās shēma

3.1. BPNN modeļa struktūra

Jau pieminētajā Minska un Peiperta darbā, skatīt [9], kas tika publicēts 1969.gadā, tika parādīts, ka divslāņu vienvirziena neironu tīkla modelis spēj pārvarēt daudzus ierobežojumus, taču toreiz vēl nebija atrisināta problēma, kā atjaunināt svarus posmā no ieejas uz slēpto slāni. Atbildi uz šo jautājumu sniedza arī jau pieminētais Rumelharta, Hintona un Viljamsa darba rezultāts, skatīt [10]. Atbilde būtībā slēpjas vienkāršā idejā - slēptā slāņa neirona kļūdas nosaka ar nākamā slāņa kļūdu signāla apkopošanas palīdzību, jo tikai izejas slānim ir pieejama vēlamā atbilde. Kā jau redzams nosaukumā, apmācība būtībā notiek pretējā virzienā, sākot ar izejas slāni, un tad atpakaļ līdz pirmajam, tas ir, pretējā virzienā nekā notika tīkla darbināšana. Parasti neironu tīkls saņem ieejas vektora X vērtības un tīkla darbības rezultātā tiek iegūts izejas vērtību vektors Y. Savukārt saistību starp X un Y nosaka tīkla arhitektūra.

Daudzslāņu vienvirziena neironu tīkla katrs slānis sastāv no neironiem, kas saņem ieejas signālu no iepriekšējā slāņa neironiem un izdod signālu nākamā slāņa neironiem. Turklāt viena slāņa neironi savā starpā nav saistīti. Pieņemsim, ka ieejas slānī ir p neironi, slēptajā slānī ir q neironi, un izejas slānī ir k neironi. Matemātiski pamata BPNN modeļa



12. att. Daudzslāņu neironu tīkla arhitektūra ar vienu slēpto slāni, [11]

struktūru var aprakstīt ar (3.1) formulu.

$$\hat{Y}(t+1) = F_2[V^T(t)F_1(W(t)X(t))], \tag{3.1}$$

kur $X = (x_0, x_1, ..., x_p)^T \in R^{(p+1)\times 1}$ ir BPNN ieejas vērtības, $\hat{Y} = (\hat{y}_0, \hat{y}_1, ..., \hat{y}_k)^T \in R^{k\times 1}$ ir BPNN izejas vērtības, t ir laika faktors,

$$W = \begin{pmatrix} w_{10} & w_{11} & \cdots & w_{1p} \\ w_{20} & w_{21} & \cdots & w_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ w_{q0} & w_{q1} & \cdots & w_{qp} \end{pmatrix} = (W_0, W_1, ..., W_p) \in R^{q\times(p+1),}$$

$$V = \begin{pmatrix} v_{10} & v_{20} & \cdots & v_{k0} \\ v_{11} & v_{21} & \cdots & v_{k1} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1q} & v_{2q} & \cdots & v_{kq} \end{pmatrix} = (V_1, ..., V_k) \in R^{(q+1)\times k},$$

$$F_1(W(t)X(t)) = (F_1(net_0(t)), F_1(net_1(t)), ..., F_1(net_q(t)))^T \in R^{(q+1)\times 1},$$

$$net_i(t) = \sum_{j=1}^{p} w_{ij}(t)x_j(t),$$

kur $i = 0, 1, ..., q$, ir slēptā slāņa i-tā neirona izejas vērtība, $w_{i0}(t)$, $i = 1, ..., q$, ir slēptā slāņa papildus svari (*angl. bias*). $v_{ij}(t)$, $i = 1, ...q$, $j = 1, ..., k$, ir svari no slēptā slāņa

i-tā neirona uz izejas slāņa j-to neironu, un $v_{i0}(t)$, $i = 1, ..., k$ ir izejas slāņa papildus svari. Atsaucoties uz [11], papildus svaru izmantošana nodrošina efektīvāku algoritma darbību. F_1 un F_2 ir aktivizācijas funkcijas. Aktivizācijas funkcijas var būt jebkuras nelineāras, nepārtrauktas, ierobežotas un diferencējamas funkcijas. Kā jau iepriekš tika norādīts, parasti aktivizācijas funkcijas ir sigmoidālās un hiperboliskā tangensa funkcijas. Ērtības labad slēptajā slānī kā aktivizācijas funkcija tiks izmantota simetriskā hiperboliskā tangensa funkcija, [1, 29. lpp.]. Novērtējot tīkla savienojumu svarus (W, V), kas savukārt tiek darīts ar apmācības un apmācīšanās metodēm, var veikt tādus uzdevumus kā funkciju aproksimācija, tendenču atpazīšana un laikrindu prognozēšana. [1, 30. lpp.]

Vienādojuma 3.1 pierādījums

Pierādījums.

$$\begin{aligned}
\hat{Y}(t+1) &= \begin{pmatrix} \hat{y}_1(t+1) \\ \hat{y}_2(t+1) \\ \cdots \\ \hat{y}_k(t+1) \end{pmatrix} \\
&= \begin{pmatrix} f_2[\sum_{i=1}^{q} f_1(\sum_{j=1}^{p} w_{ij}(t)x_j(t) + w_{i0}(t))v_{1i}(t) + v_{10}(t)] \\ f_2[\sum_{i=1}^{q} f_1(\sum_{j=1}^{p} w_{ij}(t)x_j(t) + w_{i0}(t))v_{2i}(t) + v_{20}(t)] \\ \cdots \\ f_2[\sum_{i=1}^{q} f_1(\sum_{j=1}^{p} w_{ij}(t)x_j(t) + w_{i0}(t))v_{ki}(t) + v_{k0}(t)] \end{pmatrix} \\
&= \begin{pmatrix} f_2[\sum_{i=1}^{q} f_1(\sum_{j=1}^{p} w_{ij}(t)x_j(t)))v_{1i}(t)] \\ f_2[\sum_{i=1}^{q} f_1(\sum_{j=1}^{p} w_{ij}(t)x_j(t)))v_{2i}(t)] \\ \cdots \\ f_2[\sum_{i=1}^{q} f_1(\sum_{j=1}^{p} w_{ij}(t)x_j(t))v_{ki}(t)] \end{pmatrix} \\
&= \begin{pmatrix} f_2[\sum_{i=1}^{q} f_1(net_i)v_{1i}(t)] \\ f_2[\sum_{i=1}^{q} f_1(net_i)v_{2i}(t)] \\ \cdots \\ f_2[\sum_{i=1}^{q} f_1(net_i)v_{ki}(t)] \end{pmatrix} = \begin{pmatrix} f_2[V_1^T(t)F_1(W(t)X(t))] \\ f_2[V_2^T(t)F_1(W(t)X(t))] \\ \cdots \\ f_2[V_k^T(t)F_1(W(t)X(t))] \end{pmatrix} \\
&= F_2[V^T(t)F_1(W(t)X(t))],
\end{aligned}$$

kur f_1 ir slēptā slāņa neironu aktivizācijas funkcija, un f_2 ir izejas slāņa neironu aktivizācijas funkcija, savukārt t ir laika faktors. □

3.2. BPNN algoritma apmācīšanās process

Viena no neironu tīklu raksturīgajām īpašībām ir spēja uzlabot rezultātu, mācoties no piemēriem. BPNN ir neuzraudzītās apmācīšanās algoritms, un apmācīšanās notiek, sākot ar izejas slāni, un tad atpakaļ līdz pirmajam, t.i., pretējā virzienā nekā notiek tīkla darbināšana. Kļūdu atgriezeniskās izplatīšanās metodes ietvaros apmācīšanās izejas slānim notiek atšķirīgi, jo tam ir pieejama vēlamā atbilde. Tā kā pārējiem slāņiem tādas nav, tā vietā tiek veikta kļūdu signāla apkopošana nākamajā slānī un šīs kļūdas izmantošana vēlamās atbildes vietā. Būtībā kļūdu atgriezeniskās izplatīšanās apmācīšanās metode sastāv no divām fāzēm - izplatīšanās uz priekšu un atgriezeniskās izplatīšanās.

Pieņemsim, ka mūsu rīcībā ir n izlases (paraugi), un katru var aprakstīt ar ieejas vektoru $X_i = (x_{i1}, ..., x_{ip})$ un izejas vektoru $Y_i = (y_{i1}, ..., y_{ik})$, kur $1 \leq i \leq n$.

Pirmajā fāzē (izplatīšanās uz priekšu) tīklam tiek padotas sākotnējās vērtības X_i, un, balstoties uz esošajiem svariem W, tiek ģenerētas izejas vērtības $\hat{Y}_i = (\hat{y}_{i1}, ..., \hat{y}_{ik})$. Mērķis ir minimizēt kļūdas funkciju, kas tiek definēta kā

$$E = \sum_{i=1}^{n} \sum_{j=1}^{k} \frac{(y_{ij} - \hat{y}_{ij})^2}{2}. \tag{3.2}$$

Kļūdas, kas definēta ar (3.2), minimizācija notiek, mainot svarus W. Svaru atjaunošanas formulas izvedums tiks parādīts nākamajā nodaļā.

Otrajā fāzē (atgriezeniskā izplatīšanās) saskaņā ar formulu (3.3) tiek aprēķinātas svaru izmaiņas.

$$\triangle w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}, \tag{3.3}$$

kur $0 \leq \eta \leq 1$ ir apmācīšanās parametrs, kas kontrolē algoritma konverģences ātrumu.

Kvadrātiskā kļūda, kas tiek iegūta ar (3.2) formulas palīdzību, tiek izplatīta atpakaļ slāni pa slānim, sākot ar izejas slāni, līdz pat ieejas slānim. Savukārt svaru vērtības tiek atjauninātas atgriezeniskās izplatīšanās gaitā katrā slānī. Tā abas fāzes tiek atkārtotas tik ilgi, kamēr kļūda pārsniedz noteiktu maksimālo kļūdas līmeni ε vai kamēr nav izsmelts maksimāli pieļaujamo epohu skaits. Ja apmācības process ir beidzies, pateicoties pietiekami mazai kvadrātiskajai kļūdai, t.i. $E \leq \varepsilon$, tad tiek uzskatīts, ka neirons ir veiksmīgi apmācījies. Taču, ja apmācības process ir beidzies, jo ir pārsniegts maksimāli pieļaujamo epohu skaits, neironu tīkls nav apmācījies.

Ņemot vērā, ka katru reizi apmācīšanās process sākas no cita svaru komplekta, jo sākotnējie svari tiek piešķirti gadījumu veidā, dažādi apmācības procesi pie vienādiem

paraugiem un vienādas neironu tīklu arhitektūras var novest pie dažādiem rezultātiem.

3.3. Svaru izmaiņas BPNN algoritmā

Modeļa parametru vektors jeb neironu tīkla svarus (W, V) iespējams iegūt ar iterāciju palīdzību, minimizējot izmaksu funkciju $E(X : W, V)$. Būtībā $E(X : W, V)$ ir kļūdu kvadrātu summa modelim ar k izejas neironiem un n apmācības pāriem jeb paraugiem (*angl. patterns*), t.i.,

$$\begin{aligned} E(X : W, V) &= \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{k} e_{ij}^2 = \frac{1}{2}\sum_{j=1}^{n} e_j^T e_j \\ &= \frac{1}{2}\sum_{j=1}^{n}[y_j - \hat{y}_j(X : W, V)]^T[y_j - \hat{y}_j(X : W, V)], \end{aligned} \tag{3.4}$$

kur y_i ir patiesā vērtība un $\hat{y}_j(X : W, V)$ ir novērtētā vērtība. Ņemot vērā laika faktoru t, vienādojumu (3.4) var pārrakstīt kā

$$\begin{aligned} E(t) &= \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{k} e_{ij}^2(t) = \frac{1}{2}\sum_{j=1}^{n} e_j^T(t) e_j(t) \\ &= \frac{1}{2}\sum_{j=1}^{n}[y_j(t) - \hat{y}_j(t)]^T[y_j(t) - \hat{y}_j(t)], \end{aligned} \tag{3.5}$$

kur $e_j = (e_{1j}, e_{2j}, ..., e_{kj})^T \in R^{k\times 1}$, $= 1, 2, ..., n$, $y_j(t)$ īstā vērtība laikā t, savukārt $\hat{y}_j(t)$ ir prognozētā vērtība laikā t.

Gradienta metode

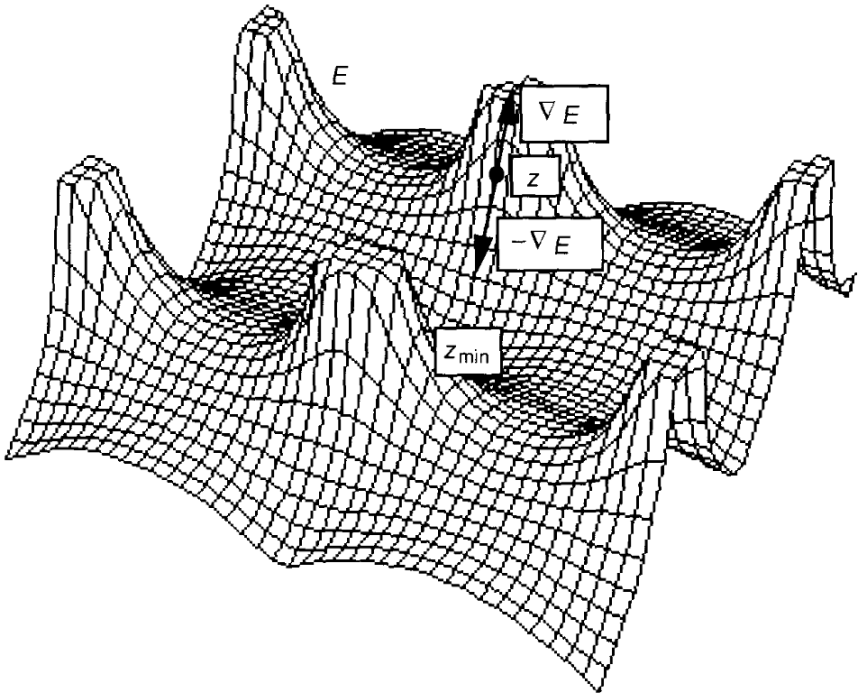
Apmācīšanās ar kļūdu atgriezenisko izplatīšanos balstās uz gradienta metodes pielietojumu kļūdas funkcijai. Par skalārā lauka u gradientu fiksētā punktā M_0 sauc vektoru, kas nosaka virzienu, kādā skalārā lieluma vērtība palielinās visātrāk, [5, 42. lpp]. Gradienta koordinātas tiek aprēķinātas, izmantojot funkcijas u parciālos atvasinājumus punktā M_0. Ja $u = u(x, y, z)$, tad u gradients ir

$$\nabla u = \left[\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial u}{\partial z}\right].$$

Pielietojot gradientu metodi neironu tīkla apmācībā, par skalārā lauka funkciju tiek ņemta kļūdas funkcija E, savukārt par argumentiem tiek ņemti neironu tīkla svari W. Tā kā šajā gadījumā ir svarīgs virziens, kurā kļūdas funkcijas vērtība samazinās, gradients jāņem ar mīnuss zīmi.

Kā jau tika minēt iepriekš, apmācības procesa mērķis ir minimizēt kvadrātisko kļūdu, kas izsakāma ar formulu (3.2). Tātad svaru izmaiņas var noformulēt sekojoši:

$$\triangle W = -\nabla E_j(W).$$



13. att. Kļūdas virsma svaru telpā

Svaru izmaiņu formulas iegūšana

Lai iegūtu svaru atjaunošanas formulas, ir nepieciešams formulēt trīs lemmas.

Lemma 1. *Slēptā slāņa aktivizācijas funkcijas* $F_1(WX) = [f_1(net_1)\, f_1(net_1)\, \cdots\, f_1(net_1)]^T$ *atvasinājums pēc NET jeb* WX*, kas izsakāms kā (3.6), ir* $F_1(WX)$ *Jakobiāna matrica,* $f_1'(net_i) = \frac{\partial f_1(net_i)}{\partial net_i}$*,* $i = 0, 1, ..., q$*,* $f_1(x) = tanh(u_0^{-1}x)$ *un tā atvasinājums ir* $f_1' = u_0^{-1}[1 - f_i^2(x)]$.

$$\begin{aligned} F_1' &= diag[f_1'(net_1)\, f_1'(net_1)\, \cdots\, f_1'(net_1)] \\ &= \frac{1}{u_0} \begin{pmatrix} 1 - f_1^2(net_1) & 0 & \cdots & 0 \\ 0 & 1 - f_1^2(net_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 - f_1^2(net_q) \end{pmatrix} \end{aligned}$$

$$= \frac{1}{u_0}\begin{pmatrix} 1-f_1^2(W_1X) & 0 & \cdots & 0 \\ 0 & 1-f_1^2(W_2X) & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1-f_1^2(W_qX) \end{pmatrix}. \tag{3.6}$$

Pierādījums. Tā kā $f_1(x) = tanh(u_0^{-1}x)$ un tā atvasinājums ir $f_1' = u_0^{-1}[1-f_1^2(x)]$, $net_i = W_iX$, $F_1(WX) = [f_1(net_0)\, f_1(net_1)\ \cdots\ f_1(net_q)]^T$, slēptā slāņa aktivizācijas funkcijas $F_1(WX)$ atvasinājums pēc NET var tikt izteikts kā

$$\begin{aligned} F_1' &= \begin{pmatrix} \frac{\partial f_1(net_1)}{\partial net_1} & \cdots & \frac{\partial f_1(net_1)}{\partial net_q} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_1(net_q)}{\partial net_1} & \cdots & \frac{\partial f_1(net_q)}{\partial net_q} \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1(net_1)}{\partial net_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\partial f_1(net_q)}{\partial net_q} \end{pmatrix} \\ &= \frac{1}{u_0}\begin{pmatrix} 1-f_1^2(W_1X) & 0 & \cdots & 0 \\ 0 & 1-f_1^2(W_2X) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1-f_1^2(W_qX) \end{pmatrix}. \end{aligned}$$

□

Lemma 2. *Slēptajā slānī iegūtā rezultāta $V^TF_1(WX)$ parciālais atvasinājums pēc W ir pierakstāms sekojoši:*

$$\frac{\partial[V^TF_1(WX)]}{\partial W} = F_1'(WX)VX^T. \tag{3.7}$$

Pierādījums.

$$\begin{aligned} \frac{\partial[V^TF_1(WX)]}{\partial W} &= \left[\frac{\partial[\sum_{i=0}^{q} v_i f_1(\sum_{j=0}^{p} w_{ij}x_j)]}{\partial w_{ij}}\right]_{q\times(p+1)} = [v_i f_1'(net_i)x_j]_{q\times(p+1)} \\ &= \begin{pmatrix} v_1f_1'(net_1)x_0 & v_1f_1'(net_1)x_1 & \cdots & v_1f_1'(net_1)x_p \\ v_2f_1'(net_2)x_0 & v_2f_1'(net_2)x_1 & \cdots & v_2f_1'(net_2)x_p \\ \vdots & \vdots & \ddots & \vdots \\ v_qf_1'(net_q)x_0 & v_qf_1'(net_q)x_1 & \cdots & v_qf_1'(net_q)x_p \end{pmatrix} \\ &= \begin{pmatrix} f_1'(net_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & f_1'(net_q) \end{pmatrix} \times \begin{pmatrix} v_1 \\ \vdots \\ v_q \end{pmatrix} \times \begin{pmatrix} x_0 & x_1 & \cdots & x_p \end{pmatrix} \\ &= F_1'(WX)VX^T. \end{aligned}$$

□

Lemma 3. *Slēptajā slānī iegūtā rezultāta $V^T F_1(WX)$ parciālais atvasinājums pēc V ir pierakstāms sekojoši:*

$$\frac{\partial[V^T F_1(WX)]}{\partial V} = F_1(WX). \tag{3.8}$$

Pierādījums.

$$\begin{aligned}\frac{\partial[V^T F_1(WX)]}{\partial V} &= \left[\frac{\partial[\sum_{i=0}^{q} v_i f_1(net_i)]}{\partial v_i}\right]_{(q+1)\times 1} = [f_1(net_i)]_{(q+1)\times 1} \\ &= F_1(WX).\end{aligned}$$

□

Tālāk, lai atrastu svaru atjaunošanas formulas, ir jāatrod $E(t)$ gradients attiecībā pret W un V. No formulas (3.5), izmantojot 1., 2. un 3. lemmu, $E(t)$ gradients attiecībā pret W ir

$$\begin{aligned}\nabla_W E(t) &= \frac{\partial E(t)}{\partial W(t)} = \sum_{j=1}^{n}\sum_{i=1}^{k} e_{ij}(t)\frac{\partial e_{ij}(t)}{\partial W(t)} \\ &= -\sum_{j=1}^{n}\sum_{i=1}^{k} e_{ij}(t)\frac{\partial \hat{y}_{ij}(t)}{\partial W(t)} \\ &= -\sum_{j=1}^{n}\sum_{i=1}^{k} e_{ij}(t)F'_{2(j)}[v_i^T F_{1(j)}(Wx_j)]\frac{\partial[v_i^T F_1(Wx_j)]}{\partial W(t)} \\ &= -\sum_{j=1}^{n}\sum_{i=1}^{k} e_{ij}F'_{2(j)}F'_{1(j)}VX^T \\ &= -\sum_{j=1}^{n} F'_{1(j)}\left(\sum_{j=1}^{k} e_{ij}F'_{2(j)}v_i\right)x_j^T \\ &= -\sum_{j=1}^{n} F'_{1(j)}\begin{pmatrix} e_{1j}f'_{2(1j)}v_{11} + e_{2j}f'_{2(2j)}v_{21} + \cdots + e_{kj}f'_{2(kj)}v_{k1} \\ e_{1j}f'_{2(1j)}v_{12} + e_{2j}f'_{2(2j)}v_{22} + \cdots + e_{kj}f'_{2(kj)}v_{k2} \\ \vdots \\ e_{1j}f'_{2(1j)}v_{1q} + e_{2j}f'_{2(2j)}v_{2q} + \cdots + e_{kj}f'_{2(kj)}v_{kq}\end{pmatrix}x_j^T \\ &= -\sum_{j=1}^{n} F'_{1(j)}\begin{pmatrix} v_{11} & v_{21} & \cdots v_{k1} \\ v_{12} & v_{22} & \cdots v_{k2} \\ \vdots & \vdots & \ddots \quad \vdots \\ v_{1q} & v_{2q} & \cdots v_{kq}\end{pmatrix} \times \begin{pmatrix} e_{1j}f'_{2(1j)} \\ e_{2j}f'_{2(2j)} \\ \vdots \\ e_{kj}f'_{2(kj)}\end{pmatrix}x_j^T\end{aligned}$$

$$= -\sum_{j=1}^{n} F'_{1(j)} \begin{pmatrix} v_{11} & v_{21} & \cdots v_{k1} \\ v_{12} & v_{22} & \cdots v_{k2} \\ \vdots & \vdots & \ddots \quad \vdots \\ v_{1q} & v_{2q} & \cdots v_{kq} \end{pmatrix} \times \begin{pmatrix} f'_{2(1j)} & 0 & \cdots & 0 \\ 0 & f'_{2(2j)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f'_{2(kj)} \end{pmatrix}$$

$$\times \begin{pmatrix} e_{1j} \\ e_{2j} \\ \vdots \\ e_{kj} \end{pmatrix} x_j^T = -\sum_{j=1}^{n} F'_{1(j)} V F'_{2(j)} e_j x_j^T.$$

Līdzīgā veidā iegūstams arī $E(t)$ gradients attiecībā pret V.

$$\begin{aligned}
\nabla_V E(t) &= \frac{\partial E(t)}{\partial V(t)} = \sum_{j=1}^{n}\sum_{i=1}^{k} e_{ij}(t)\frac{\partial e_{ij}(t)}{\partial V(t)} \\
&= -\sum_{i=1}^{k} e_{ij}(t)\frac{\partial \hat{y}_{ij}(t)}{\partial V(t)} \\
&= -\sum_{j=1}^{n}\sum_{i=1}^{k} e_{ij}(t) \begin{pmatrix} \frac{\partial \hat{y}_{ij}}{\partial v_{10}} & \frac{\partial \hat{y}_{ij}}{\partial v_{20}} & \cdots & \frac{\partial \hat{y}_{ij}}{\partial v_{i0}} \cdots \frac{\partial \hat{y}_{ij}}{\partial v_{k0}} \\ \frac{\partial \hat{y}_{ij}}{\partial v_{11}} & \frac{\partial \hat{y}_{ij}}{\partial v_{21}} & \cdots & \frac{\partial \hat{y}_{ij}}{\partial v_{i1}} \cdots \frac{\partial \hat{y}_{ij}}{\partial v_{k1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}_{ij}}{\partial v_{1q}} & \frac{\partial \hat{y}_{ij}}{\partial v_{2q}} & \cdots & \frac{\partial \hat{y}_{ij}}{\partial v_{iq}} \cdots \frac{\partial \hat{y}_{ij}}{\partial v_{kq}} \end{pmatrix} \\
&= -\sum_{j=1}^{n}\sum_{i=1}^{k} e_{ij}(t) F'_2[V^T F_1(WX)] \begin{pmatrix} 0 & 0 & \cdots & f_{1(0j)} & \cdots & 0 \\ 0 & 0 & \cdots & f_{1(1j)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f_{1(qj)} & \cdots & 0 \end{pmatrix} \\
&= -\sum_{j=1}^{n} \begin{pmatrix} e_{1j} f'_{2(1j)} f_{1(0j)} & \cdots & e_{ij} f'_{2(ij)} f_{1(0j)} & \cdots e_{kj} f'_{2(kj)} f_{1(0j)} \\ e_{1j} f'_{2(1j)} f_{1(1j)} & \cdots & e_{ij} f'_{2(ij)} f_{1(1j)} & \cdots e_{kj} f'_{2(kj)} f_{1(1j)} \\ \vdots & \ddots & \vdots & \vdots \\ e_{1j} f'_{2(1j)} f_{1(qj)} & \cdots & e_{ij} f'_{2(ij)} f_{1(qj)} & \cdots e_{kj} f'_{2(kj)} f_{1(qj)} \end{pmatrix} \\
&= -\sum_{j=1}^{n} \begin{pmatrix} f_{1(0j)} \\ f_{1(1j)} \\ \vdots \\ f_{1(qj)} \end{pmatrix} \begin{pmatrix} e_{1j} f'_{2(1j)} & e_{2j} f'_{2(2j)} & \cdots & e_{kj} f'_{2(kj)} \end{pmatrix}
\end{aligned}$$

$$= -\sum_{j=1}^{n} \begin{pmatrix} e_{1j} & e_{2j} & \cdots & e_{kj} \end{pmatrix} \begin{pmatrix} f'_{2(1j)} & 0 & \cdots & 0 \\ 0 & f'_{2(2j)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f'_{2(kj)} \end{pmatrix}$$

$$= -\sum_{j=1}^{n} F_{1(j)} e_j^T F'_{2(j)}.$$

Rezultātā, pieņemot, ka η ir apmācīšanās parametrs, un svaru W un V izmaiņas attiecīgi apzīmējot ar $\triangle W(t)$ un $\triangle V(t)$, tiek iegūtas svaru izmaiņu formulas

$$\triangle W(t) = -\eta \nabla_W E(t) = \eta(t) \sum_{j=1}^{n} F'_{1(j)} V F'_{2(j)} e_j x_j^T, \tag{3.9}$$

$$\triangle V(t) = -\eta \nabla_V E(t) = \eta(t) \sum_{j=1}^{n} F_{1(j)} e_j^T F'_{2(j)}. \tag{3.10}$$

Tā kā $\triangle W(t) = W(t) - W(t-1)$ un $\triangle V(t) = V(t) - V(t-1)$, formulas (3.9) un (3.10) var pārrakstīt formā

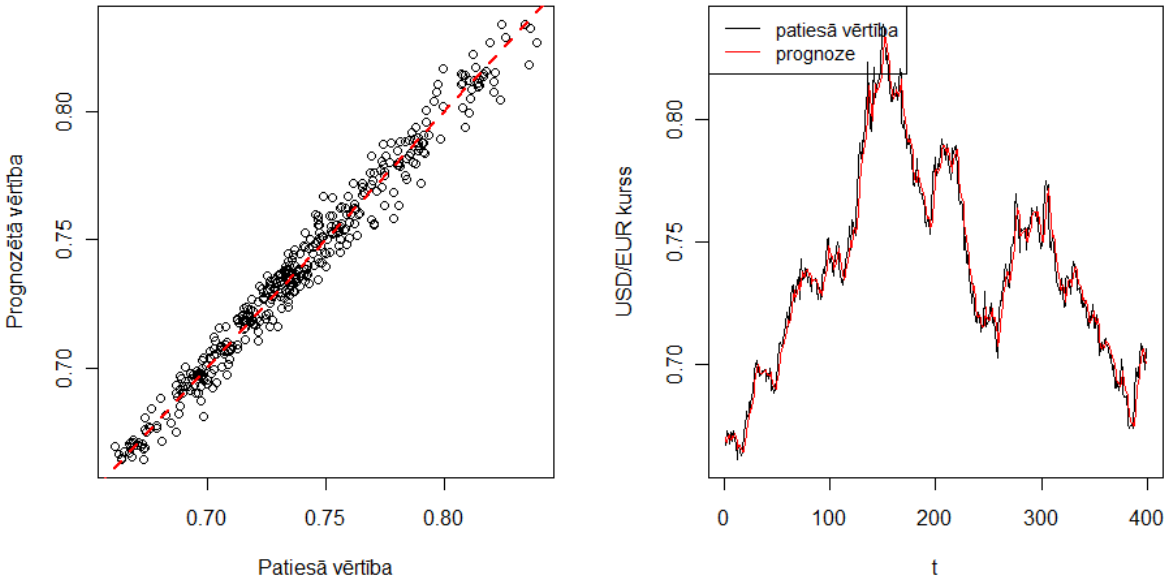
$$W(t) = W(t-1) + \eta(t) \sum_{j=1}^{n} F'_{1(j)} V F'_{2(j)} e_j x_j^T, \tag{3.11}$$

$$V(t) = V(t-1) + \eta(t) \sum_{j=1}^{n} F_{1(j)} e_j^T F'_{2(j)}. \tag{3.12}$$

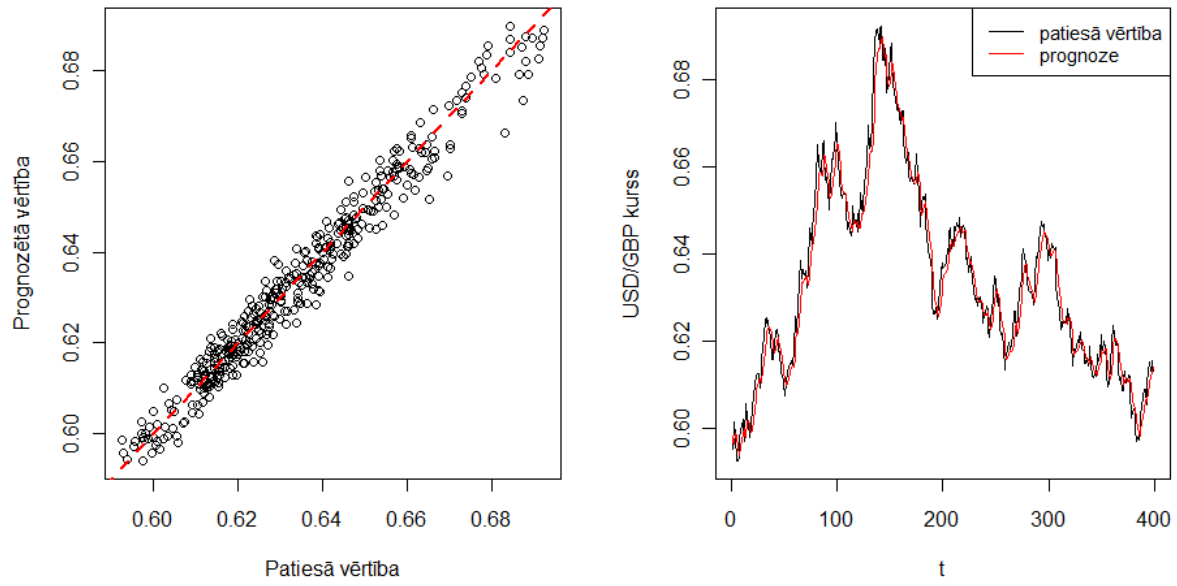
3.4. Iegūtie rezultāti

Ar R iebūvētās funkcijas *nnet* palīdzību, balstoties uz piemēriem no [12] literatūras avota, tika konstruēts neironu tīkls prognozēšanas uzdevuma veikšanai. Balstoties uz [1] autoru veiktā pētījuma, prognozes veidošanai tika izmantotas 9 iepriekšējās vērtības, slēptajā slānī tika izmantots 21 neirons, un problēmas sarežģītības dēļ epohu skaits tika noteikts 2000, savukārt $\eta = 0.3$.

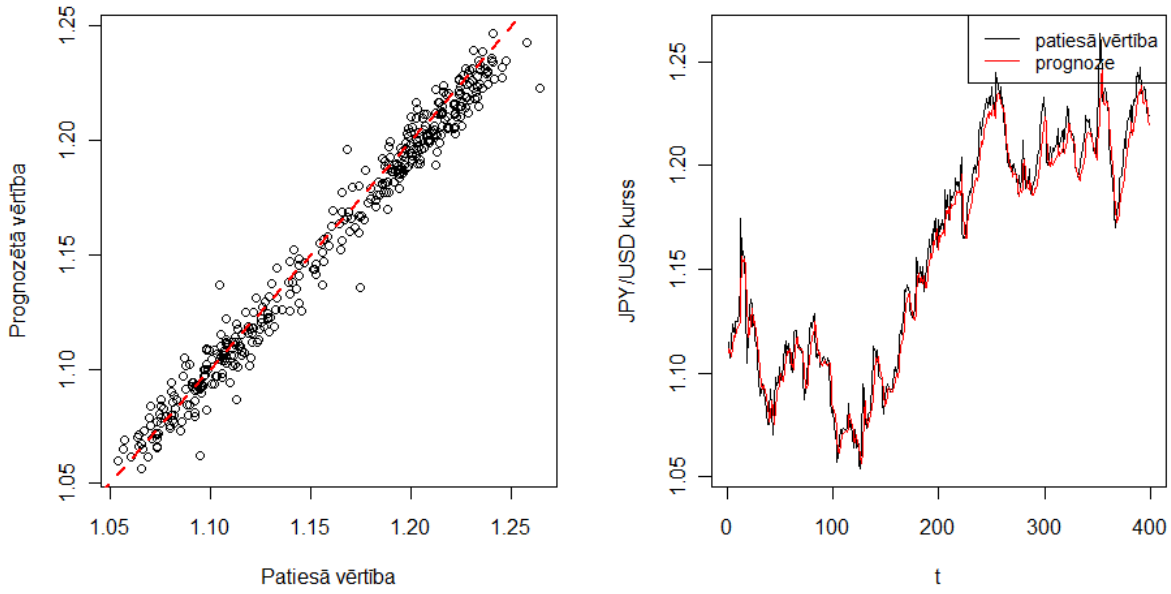
14., 15. un 16. attēli liecina, ka parastā BP modeļa prognozes ir salīdzinoši akurātas. Plašāku rezultātu analīzi un salīdzinājumu ar citiem tālāk apskatītajiem modeļiem skatīt darba pēdējā nodaļā.



14. att. USD/EUR prognoze ar BPNN



15. att. USD/GBP prognoze ar BPNN



16. att. JPY/USD prognoze ar BPNN

4. Adaptīvais BPNN ar optimālo apmācīšanās parametru un momenta faktoru

Lai gan valūtu kursu prognozēšanā tiek lietoti vairāki dažādu veidu neironu tīkli, bieži vien novārtā tiek atstāts jautājums par optimālā apmācīšanās koeficienta un momenta faktoru izvēli. [1] literatūras avota autoru veiktais pētījums liecina, ka apmācīšanās koeficients un momenta faktori vairākumā gadījumu tiek fiksēti. Tomēr šie koeficienti neironu tīkla apmācības procesā ir pietiekami nozīmīgi, [1, 67. lpp.]. Piemēram, ja apmācīšanās koeficients ir liels, apmācīšanās process var būt ātrs, taču tajā pašā laikā var kļūt nestabils un iespējama situācija, kad apmācība vispār nav notikusi. Lai nodrošinātu stabilu apmācības procesu, apmācīšanās koeficientam ir jābūt pietiekami mazam. Tomēr pārāk mazs apmācīšanās koeficients var krietni paildzināt apmācības procesu un samazināt konverģences ātrumu. Turklāt apmācīšanās koeficienta lielums ir atkarīgs arī no neironu tīkla struktūras un risināmās problēmas. Ne mazāk svarīga ir arī momenta faktora izvēle, jo momenta faktors palīdz novērst svaru izmaiņu oscilēšanu. Tomēr esošie pētījumi nesniedz precīzu algoritmu, kā momenta faktors būtu nosakāms, [1, 67. lpp.]. Apmācīšanās parametra un momenta faktoru atrašanas problēmu [1] autori piedāvā risināt ar gradienta un optimizācijas metodēm, kas arī tiks aprakstītas šīs nodaļas ietvaros.

4.1. Algoritma formulējums

Turpmāk tiks lietotas jau iepriekš formulētā BPNN modeļa reprezentācija (3.1) tāpat tiek saglabāti visi iepriekš ieviestie apzīmējumi. Tālākā aprakstā tiks izmantotas svaru atjaunināšanas formulas (3.9) un (3.10).

Optimālā apmācīšanās parametra noteikšana

Lai atrastu optimālo apmācīšanās parametra vērtību, tiks lietots kļūdu pieaugums,

$$\triangle e(t+1) = e(t+1) - e(t) = y(t+1) - \hat{y}(t+1) - y(t) + \hat{y}, \tag{4.1}$$

kur $\triangle y(t+1) = y(t+1) - y(t)$ ir patiesās vērtības izmaiņas, savukārt $\triangle \hat{y}(t+1) = \hat{y}(t+1) - \hat{y}(t)$ ir izejas vērtības izmaiņas. Pie tam tiek pieņemts, ka $|\triangle y(t+1)| \ll |\triangle \hat{y}(t+1)|$. Līdz ar to var teikt, ka patiesās vērtības izmaiņu, salīdzinājumā ar izejas vērtības izmaiņu, var neņemt vērā. Plašāku pamatojumu šim pieņēmumam iespējams atrast [1, 70. lpp.]. Rezultātā tiek iegūts, ka vienādojums (4.1) var tikt aproksimēts ar

$$\triangle e(t+1) = e(t+1) - e(t) = \triangle y(t+1) - \triangle \hat{y}(t+1) \approx - \triangle \hat{y}(t+1). \tag{4.2}$$

Lemma 4. *BPNN slēptā slāņa izejas $V^T F_1(WX)$ atvasinājums pēc laika t ir pierakstāms formā*

$$\begin{aligned} \frac{d[V^T F_1(WX)]}{dt} &= F_1(WX)\frac{dV}{dt} + V^T F_1'(WX)\frac{dW}{dt}X \\ &= \frac{dV^T}{dt}F_1(WX) + V^T F_1'(WX)\frac{dW}{dt}X, \end{aligned} \tag{4.3}$$

kur $V^T F_1(WX)$ ir BPNN slēptā slāņa izeja, $\frac{dW}{dt}$ un $\frac{dW}{dt}$ ir svaru W un V atvasinājumi pēc laika t, $X = [x_0, x_1, ..., x_p]^T$ ir ieejas vektors, $F_1 = [f_1(net_0), f_1(net_1), ..., f_1(net_q)]^T$,

$$F_1'(WX) = \begin{pmatrix} f_1'(net_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & f_1'(net_q) \end{pmatrix}, \frac{dW}{dt} = \begin{pmatrix} \frac{dw_{00}}{dt} & \frac{dw_{01}}{dt} & \cdots & \frac{dw_{0p}}{dt} \\ \frac{dw_{10}}{dt} & \frac{dw_{11}}{dt} & \cdots & \frac{dw_{1p}}{dt} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{dw_{q0}}{dt} & \frac{dw_{q1}}{dt} & \cdots & \frac{dw_{qp}}{dt} \end{pmatrix}, V = [v_1, v_2, ..., v_q]^T,$$

$\frac{dV}{dt} = \left[\frac{dv_1}{dt}, \frac{dv_2}{dt}, ..., \frac{dv_q}{dt}\right]^T$.

Pierādījums. Tā kā $f_1(net_0) \equiv 1$, $f_1'(net_0) = 0$.

$$\begin{aligned} \frac{d[V^T F_1(WX)]}{dt} &= \frac{d\left[\sum_{i=0}^{q} v_i f_1\left(\sum_{j=0}^{p} w_{ij}x_j\right)\right]}{dt} \\ &= \sum_{i=0}^{q} \frac{\partial\left[\sum_{i=0}^{q} v_i f_1\left(\sum_{j=0}^{p} w_{ij}x_j\right)\right]}{\partial v_i}\frac{dv_i}{dt} \end{aligned}$$

$$
\begin{aligned}
+ & \sum_{i=0}^{q}\sum_{j=0}^{p}\frac{\partial\left[\sum_{i=0}^{q}v_i f_1\left(\sum_{j=0}^{p}w_{ij}x_j\right)\right]}{\partial w_{ij}}\frac{dw_{ij}}{dt} \\
= & \sum_{i=0}^{q} f_1\left(\sum_{j=0}^{p}w_{ij}x_j\right)\frac{dv_i}{dt}+\sum_{i=0}^{q}\sum_{j=0}^{p}v_i f_1'\left(\sum_{j=0}^{p}w_{ij}x_j\right)x_j\frac{dw_{ij}}{dt} \\
= & \left[f_1(net_0)\frac{dv_0}{dt}+f_1(net_1)\frac{dv_1}{dt}+...+f_1(net_q)\frac{dv_q}{dt}\right] \\
+ & v_0 f_1'(net_0)\left[x_0\frac{dw_{00}}{dt}+x_1\frac{dw_{01}}{dt}+...+x_p\frac{dw_{0p}}{dt}\right] \\
+ & v_1 f_1'(net_1)\left[x_0\frac{dw_{10}}{dt}+x_1\frac{dw_{11}}{dt}+...+x_p\frac{dw_{1p}}{dt}\right] \\
+ & \cdots+v_q f_1'(net_q)\left[x_0\frac{dw_{q0}}{dt}+x_1\frac{dw_{q1}}{dt}+...+x_p\frac{dw_{qp}}{dt}\right] \\
= & [f_1(net_0),f_1(net_1),...,f_1(net_q)]\left[\frac{dv_0}{dt},\frac{dv_1}{dt},...,\frac{dv_q}{dt}\right]^T \\
+ & [v_1,v_2,...,v_q]\begin{bmatrix} f_1'(net_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & f_1'(net_q)\end{bmatrix} \\
\times & \begin{bmatrix} \frac{dw_{00}}{dt} & \frac{dw_{01}}{dt} & \cdots & \frac{dw_{0p}}{dt} \\ \frac{dw_{10}}{dt} & \frac{dw_{11}}{dt} & \cdots & \frac{dw_{1p}}{dt} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{dw_{q0}}{dt} & \frac{dw_{q1}}{dt} & \cdots & \frac{dw_{qp}}{dt}\end{bmatrix}\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_p\end{bmatrix} \\
= & F_1(WX)\frac{dV}{dt}+V^T F_1'(WX)\frac{dW}{dt}X \\
= & \frac{dV^T}{dt}F_1(WX)+V^T F_1'(WX)\frac{dW}{dt}X.
\end{aligned}
$$

□

Izmantojot 4, un svaru izmaiņu formulas (3.9), (3.10), m-tā parauga izejas vērtības izmaiņu $\triangle\hat{y}^m(t+1)$ var pierakstīt sekojošā formā (pilnu izvedumu skatīt 1. Pielikumā):

$$
\triangle\hat{y}^m(t+1) \approx \left(\frac{d\hat{y}^m(t+1)}{dt}\right)\triangle t=\begin{bmatrix}\frac{d\hat{y}_1^m(t+1)}{dt} \\ \frac{d\hat{y}_2^m(t+1)}{dt} \\ \vdots \\ \frac{d\hat{y}_k^m(t+1)}{dt}\end{bmatrix}\triangle t=\ldots=
$$

$$
\begin{aligned}
&= \eta \sum_{j=1}^{N} F'_{2,m}(F'_{2j} e_j F^T_{1,m} F_{1j} + V^T F'_{1,m} F'_{1j} V F'_{2j} e_j x_j^T x_m) \\
&= \eta \sum_{j=1}^{N} F'_{2,m}(F^T_{1,m} F_{1j} I_{k^2} + V^T F'_{1,m} F'_{1j} V x_j^T x_m) F'_{2j} e_j.
\end{aligned}
$$

Līdz ar to visu paraugu izejas vērtības izmaiņas var pierakstīt sekojošā formā (pilnu izvedumu skatīt 2. Pielikumā):

$$
\begin{aligned}
\triangle\hat{y}(t+1) &= \begin{bmatrix} \triangle\hat{y}_1(t+1) \\ \triangle\hat{y}_2(t+1) \\ \vdots \\ \triangle\hat{y}_m(t+1) \\ \vdots \\ \triangle\hat{y}_N(t+1) \end{bmatrix} \approx \begin{bmatrix} \eta \sum_{j=1}^{N} F'_{2,1}(F^T_{1,1} F_{1j} I_{k^2} + V^T F'_{1,1} F'_{1j} V x_j^T x_1) F'_{2j} e_j \\ \eta \sum_{j=1}^{N} F'_{2,2}(F^T_{1,2} F_{1j} I_{k^2} + V^T F'_{1,2} F'_{1j} V x_j^T x_2) F'_{2j} e_j \\ \vdots \\ \eta \sum_{j=1}^{N} F'_{2,m}(F^T_{1,m} F_{1j} I_{k^2} + V^T F'_{1,m} F'_{1j} V x_j^T x_m) F'_{2j} e_j \\ \vdots \\ \eta \sum_{j=1}^{N} F'_{2,N}(F^T_{1,N} F_{1j} I_{k^2} + V^T F'_{1,N} F'_{1j} V x_j^T x_N) F'_{2j} e_j \end{bmatrix} \\
&= \eta F'_2[(F_1^T F_1) I_{k^2} + (X^T X)(V F'_1 F'_1 V)] F'_2 e(t) = \eta \xi(t) e(t). \qquad (4.4)
\end{aligned}
$$

Ievietojot iegūto rezultātu formulā (4.2), iegūstam, ka

$$
e(t+1) \approx e(t) - \eta \xi(t) e(t). \qquad (4.5)
$$

Optimālo apmācīšanās parametra vērtību t-tajā iterācijā $\eta^*(t)$ iespējams atrast, minimizējot kļūdas funkciju $E(t+1) = \frac{1}{2} e^T(t+1) e(t+1)$, ko, izmantojot formulu (4.2), var pārrakstīt formā

$$
E(t+1) = \frac{1}{2} e^T(t+1) e(t+1) = \frac{1}{2} [e(t) - \eta \xi(t) e(t)]^T [e(t) - \eta \xi(t) e(t)] \qquad (4.6)
$$

Tātad

$$
\begin{aligned}
\left. \frac{dE(t+1)}{d\eta} \right|_{\eta=\eta^*(t)} &= -\frac{1}{2} [\xi(t) e(t)]^T [e(t) - \eta^*(t) \xi(t) e(t)] \\
&- \frac{1}{2} [e(t) - \eta^*(t) \xi(t) e(t)]^T \xi(t) e(t) = 0.
\end{aligned}
$$

$$
\left. \frac{d^2 E(t+1)}{d\eta^2} \right|_{\eta=\eta^*(t)} = e^T(t) \xi^T(t) \xi(t) e(t) > 0,
$$

jo $\xi(t)$ ir pozitīvi definīta. Rezultātā tiek iegūts, ka optimālā apmācīšanās parametra vērtība

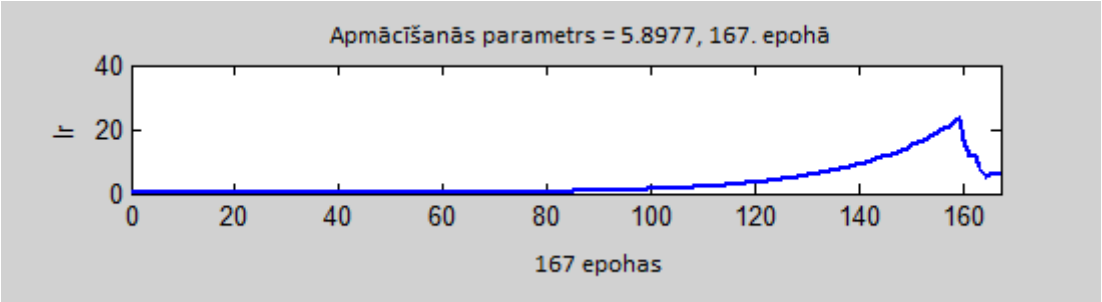
$$
\eta^*(t) = \frac{e^T(t) \xi^T(t) e(t)}{e^T(t) \xi^T(t) \xi(t) e(t)}. \qquad (4.7)
$$

Tiesa gan [1] grāmatas autori norāda, ka citos pētījumos ir iegūtas atšķirīgas optimālā apmācīšanās parametra vērtības.

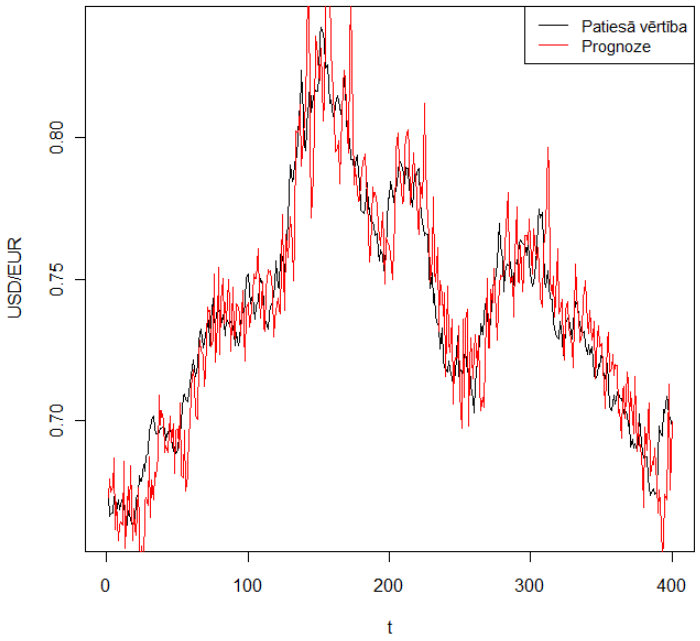
Momenta faktors. Runājot par kļūdu atgriezeniskās izplatīšanās algoritmu, ar jēdzienu momenta faktors, kas šī darba ietvaros tiks apzīmēts ar $\mu(t)$, tiek saprasta daļa no iepriekšējā svaru pieauguma $\triangle W(t-1)$ un $\triangle V(t-1)$. Galvenais momenta faktoru lietošanas mērķis ir samazināt svaru izmaiņu oscilāciju un palielināt BP konverģences ātrumu. Tomēr līdzīgi kā ar optimālo apmācīšanās parametra vērtību, nav universālas metodes, kā atrast labāko momenta faktoru, [1, 76. lpp.]. Bieži momenta faktors tiek noteikts ar izmēģinājumu un kļūdu palīdzību eksperimentālā ceļā, tomēr nepareizi izvēlēts fiksēts momenta faktors var palēināt BP algoritma konverģenci, [1, 76. lpp.].

4.2. Iegūtie rezultāti

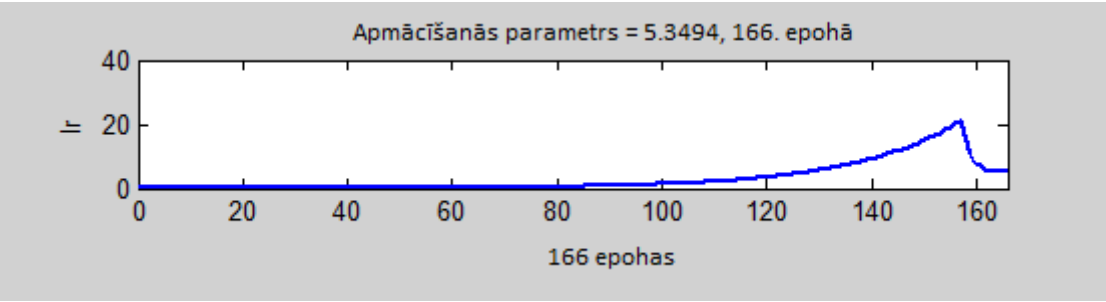
Lai iegūtu prognozes ar iepriekš aprakstīto apmācīšanās algoritmu, tika izmantotas *Matlab Neural Netwok Toolbox* iebūvētās komandas *traingda* (kļūdu atgriezeniskās izplatīšanās algoritms ar optimālo apmācīšanās parametru) un *traingdx* (kļūdu atgriezeniskās izplatīšanās algoritms ar optimālo apmācīšanās parametru un momenta faktoru). Līdzīgi kā iepriekš, balstoties uz [1] autoru veiktā pētījuma, prognozes veidošanai tika izmantotas 9 iepriekšējās vērtības, slēptajā slānī tika izmantots 21 neirons, un problēmas sarežģītības dēļ epohu skaits tika noteikts 2000. Kā momenta faktors tika izmantota konstante 0.9, kas ir *Matlab Neural Netwok Toolbox* piedāvātā noklusētā vērtība. Tā kā algoritma konverģences ātrums konkrētā uzdevuma veikšanā nebija šķērslis, tad noklusētā vērtība netika mainīta. Pateicoties *Matlab Neural Netwok Toolbox*, datu iepriekšēja manuāla apstrāde nav nepieciešama, līdz ar to tika izmantotas sākotnējās valūtu kursu laikrindas. Atsevišķi tiks apskatīti neironu tīkls ar optimālo apmācīšanās parametru un neironu tīkls ar optimālo apmācīšanās parametru un momenta faktoru.



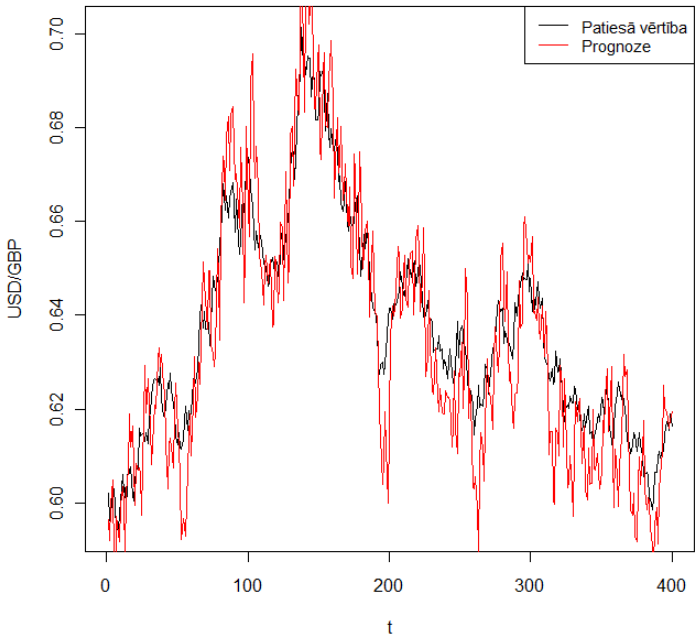
17. att. Apmācīšanās parametra izmaiņas neironu tīkla darbības laikā, USD/EUR



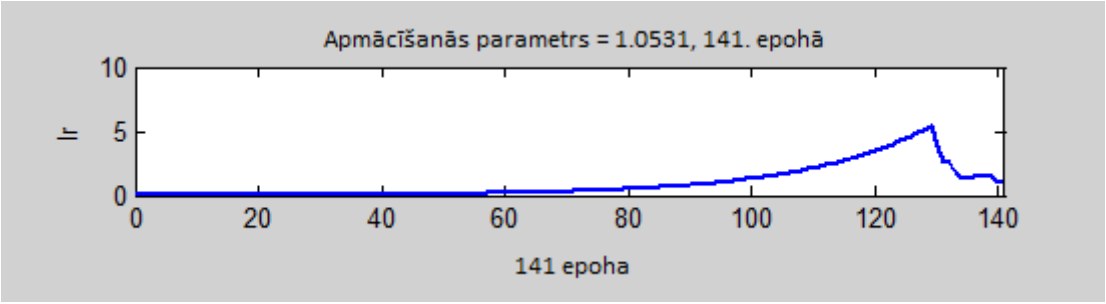
18. att.: USD/EUR prognoze un patiesā vērtība, BP ar optimālo apmācīšanās parametru



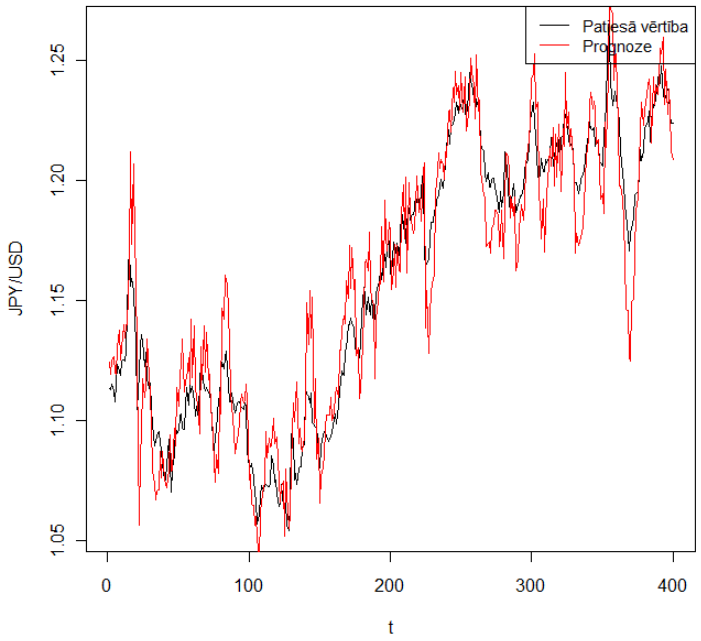
19. att. Apmācīšanās parametra izmaiņas neironu tīkla darbības laikā, USD/GBP



20. att.: USD/GBP prognoze un patiesā vērtība, BP ar optimālo apmācīšanās parametru



21. att. Apmācīšanās parametra izmaiņas neironu tīkla darbības laikā, JPY/USD

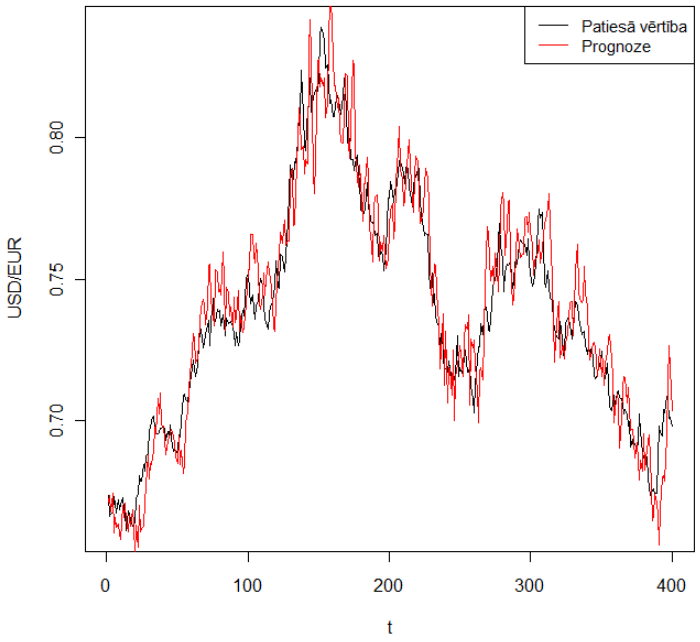


22. att. JPY/USD prognoze un patiesā vērtība, BP ar optimālo apmācīšanās parametru

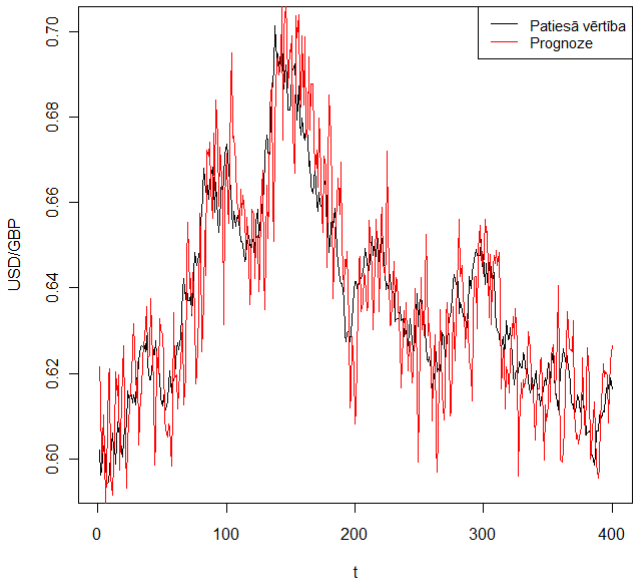
Kļūdu atgriezeniskās izplatīšanās algoritms ar optimālo apmācīšanās parametru. Kā redzams 18., 20. un 22. attēlos, BP apmācīšanās algoritms ar optimālo apmācīšanās parametru, kas pēdējā tīkla darbības epohā USD/EUR, USD/GBP un JPY/USD ir attiecīgi 5.8977, 5.3494 un 1.0531 (skatīt 17.., 19.. un 21.. attēlu), nesniedz pārāk labus rezultātus. Tomēr salīdzinoši labāku rezultātu algoritms uzrāda USD/EUR laikrindai. Tāpat redzams, ka BP algoritms ar optimālo apmācīšanās parametru pie salīdzinoši straujām laikrindas vērtību izmaiņām USD/GBP un JPY/USD gadījumā uzrāda daudz asākus kāpumus un kritumus, kas krietni pasliktina rezultātu. Un no grafikiem var izdarīt pieņēmumu, ka īpaši sliktus rezultātus šī apmācīšanās tehnika uzrāda pie ekstremālām svārstībām. Iemesli, kādēļ tas tā notiek tiek atstāti kā viena no iespējamām turpmākas izpētes tēmām.

Kļūdu atgriezeniskās izplatīšanās algoritms ar optimālo apmācīšanās parametru un momenta faktoru. Kā redzams 23., 24. un 25. attēlos, BP apmācīšanās algoritms ar optimālo apmācīšanās parametru un momenta faktoru nesniedz pārāk labus rezultātus. JPY/USD gadījumā salīdzinoši labu rezultātu algoritms uzrāda pie ilgstošas tendences klātbūtnes. Arī no USD/EUR grafika var izdarīt līdzīgu pieņēmumu, savukārt

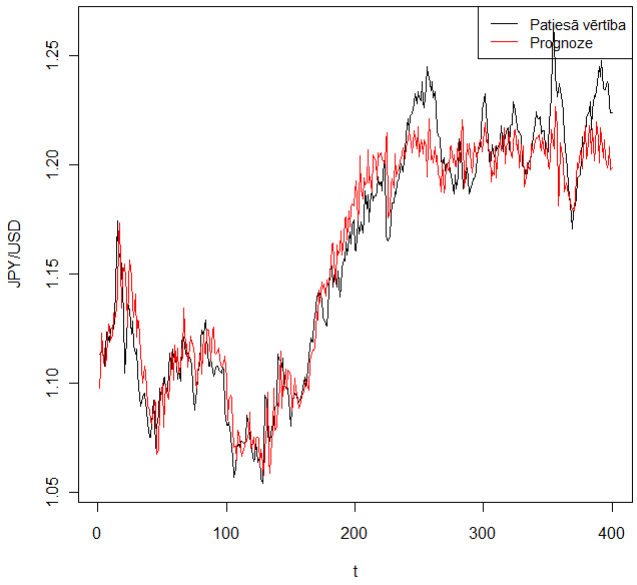
vissliktāko rezultātu algoritms uzrāda USD/GBP laikrindai.



23. att.: USD/EUR prognoze un patiesā vērtība, BP ar optimālo apmācīšanās parametru un momenta faktoru



24. att.: USD/GBP prognoze un patiesā vērtība, BP ar optimālo apmācīšanās parametru un momenta faktoru



25. att.: JPY/USD prognoze un patiesā vērtība, BP ar optimālo apmācīšanās parametru un momenta faktoru

5. Boksa Dženkinsa metodoloģija ARIMA modeļiem

Boksa-Dženkinsa pieeja ARIMA procesu modelēšanā ir aprakstīta [2], un tā pēc savas būtības ir trīs soļu iteratīvs process, kura laikā tiek identificēts piemērots ARIMA modelis, atrastā procesa pielāgošanas datiem un pielāgotā ARIMA modeļa pārbaude. Savukārt vēlāk tika pievienoti vēl divi soļi - datu pirmsapstrāde, kas ietver datu transformāciju un diferencēšanu, un prognozēšana ar atrastā modeļa palīdzību.

Šīs nodaļas ietvaros īsi tiks aprakstīta Boksa-Dženkinsa metodoloģija, un paralēli tā tiks pielietota jau iepriekš apskatītajiem datiem. Nodaļas ietvaros lasītājam tiks piedāvātas tikai svarīgākās definīcijas un rezultāti. Padziļināts izklāsts atrodams, piemēram, [2], tomēr šīs nodaļas ietvaros pamatā tika izmantota grāmata [13].

5.1. Svarīgākās definīcijas un rezultāti

Definīcija 11. Par p-tās kārtas autoregresīvo procesu $AR(p)$ sauc gadījuma procesu $\{Y_t\}_{t=1}^{\infty}$, kas apmierina vienādojumu

$$Y_t = \phi_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \varepsilon_t, \tag{5.1}$$

kur ε_t ir baltais troksnis ar $\mathbb{E}\varepsilon_t = 0$, $D\varepsilon_t = \sigma^2$ un $\phi_p \neq 0$. Izmantojot laika nobīdes operatoru B ($B^k x_t = x_{t-k}$, $k \geq 0$), vienādojumu (5.1) var pārrakstīt formā

$$(1 - \phi_1 B - \ldots - \phi_p B^p) Y_t = \phi_0 + \varepsilon_t. \tag{5.2}$$

Definīcija 12. Par q-tās kārtas slīdošā vidējā procesu $MA(q)$ sauc gadījuma procesu $\{Y_t\}_{t=1}^{\infty}$, kas apmierina vienādojumu

$$Y_t = \theta_0 + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q}, \tag{5.3}$$

kur ε_t ir baltais troksnis ar $\mathbb{E}\varepsilon_t = 0$, $D\varepsilon_t = \sigma^2$. Izmantojot laika nobīdes operatoru B, vienādojumu (5.3) var pārrakstīt formā

$$Y_t = \theta_0 + (1 + \theta_1 B + \ldots + \theta_q B^q) \varepsilon_t. \tag{5.4}$$

Definīcija 13. Par jaukto (p, q) kārtas autoregresīvo slīdošā vidējā procesu $ARMA(p, q)$ sauc gadījuma procesu $\{Y_t\}_{t=1}^{\infty}$, kas apmierina vienādojumu

$$Y_t = \phi_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q} \tag{5.5}$$

kur ε_t ir baltais troksnis ar $\mathbb{E}\varepsilon_t = 0$, $D\varepsilon_t = \sigma^2$. Izmantojot laika nobīdes operatoru B, vienādojumu (5.5) var pārrakstīt formā

$$(1 - \phi_1 B - \ldots - \phi_p B^p) Y_t = \phi_0 + (1 + \theta_1 B + \ldots + \theta_q B^q)\varepsilon_t. \tag{5.6}$$

$ARMA(p, q)$ procesu iespējams paplašināt, lai tas ietvertu arī nestacionaritāti. Tādējādi tiek iegūts autoregresīvais integrētais slīdošā vidējā process $ARIMA(p, d, q)$.

Definīcija 14. Par autoregresīvais integrētais slīdošā vidējā procesu $ARIMA(p, d, q)$ sauc gadījuma procesu $\{Y_t\}_{t=1}^{\infty}$, kas apmierina vienādojumu

$$\alpha(B)(1 - B)^d Y_t = \phi_0 + \beta(B)\varepsilon_t, \tag{5.7}$$

kur $\alpha(B) = 1 - \phi_1 B - \ldots - \phi_p B^p$ ir autoregresīvais polinoms, $\beta(B) = 1 + \theta_1 B + \ldots + \theta_q B^q$ ir slīdošā vidējā raksturīgais polinoms. Ja konstante $\phi_0 \neq 0$, tad modelī tiek pieļauts d-tās kārtas polinomiālais trends.

Autoregresīvo parametru ϕ_p un slīdošā vidējā parametru θ_q vērtībām ir noteikti ierobežojumi, kas šī darba ietvaros sīkāk netiks apskatīt.

Definīcija 15. Par autokovariāciju funkciju sauc

$$\gamma(\tau) = cov(x_t, x_{t+\tau}) = \mathbb{E}(x_t - \mu)(x_{t+\tau} - \mu), \tag{5.8}$$

kur τ ir laika atstarpe.

Definīcija 16. Par autokorelāciju funkciju sauc

$$\rho(\tau) = corr(x_t, x_{t+\tau}) = \frac{\gamma(\tau)}{\gamma(0)}, \tag{5.9}$$

kur τ ir laika atstarpe. Autokorelācija ir lineārās atkarības mērs starp procesa vērtībām ar laika atstarpi τ. Pieņemsim, ka mūsu rīcībā ir laikrinda $Y_1, Y_2, ..., Y_N$.

Definīcija 17. Par empīrisko autokovariāciju funkciju sauc

$$c(\tau) = \frac{1}{N} \sum_{t=1}^{N-\tau} (Y_t - \bar{Y})(Y_{t+\tau} - \bar{Y}), \tag{5.10}$$

kur $\bar{Y} = \frac{1}{N} \sum_{i=1}^{N} Y_i$.

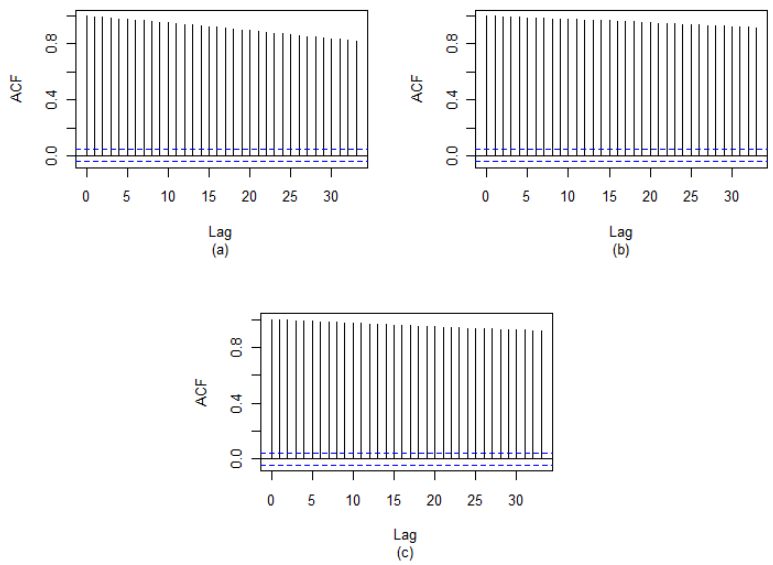
Definīcija 18. Par empīrisko autokorelāciju funkciju sauc

$$r(\tau) = \frac{c(\tau)}{c(0)} = \frac{\sum_{t=1}^{N-\tau}(Y_t - \bar{Y})(Y_{t+\tau} - \bar{Y})}{\sum_{t=1}^{N}(Y_t - \bar{Y})^2}, \tag{5.11}$$

kur $\bar{Y} = \frac{1}{N}\sum_{i=1}^{N} Y_i$. Turpmāk darbā tiks lietots arī tāds jēdziens kā parciālā autokorelāciju funkcija, ko definē kā korelāciju, kas ir attīrīta no citu mainīgo ietekmes. Piemēram, parciālā autokorelācija ar laika nobīdi τ ir autokorelācija starp Y_t un $Y_{t+\tau}$, kas ir attīrīta no lineārās mainīgo $Y_{t+1}, ..., Y_{t+\tau-1}$ ietekmes. Plašāka informācija par parciālās autokorelācijas novērtēšanu atrodama [2].

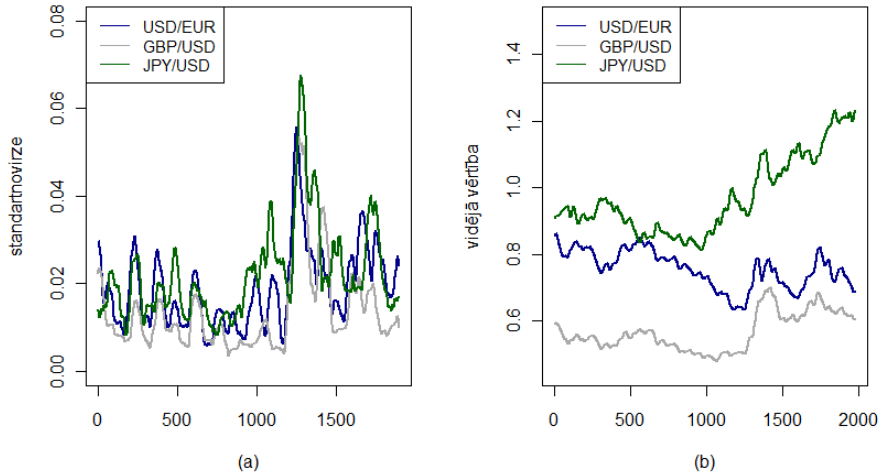
5.2. Piemērota ARIMA modeļa izvēle

Sākotnējā modeļa izvēla pamatā balstās uz laikrindas, autokorelāciju funkcijas (turpmāk tekstā ACF) un parciālās autokorelāciju funkcijas (turpmāk tekstā PACF) attēlu grafiskās analīzes. Piemēram, no 1.. attēla varam izvirzīt pieņēmumu, ka apskatītajām valūtu kursu laikrindām piemīt nestacionaritāte, ko var pamatot ar lēni dilstošo autokorelāciju funkciju (skatīt 26. attēlu), kā arī ar faktu, ka visu trīs valūtu kursu vidējās vērtības un arī dispersijas ir atkarīgas no laika, skatīt 27. attēlu.



26. att. Valūtu kursu ACF, (a) USD/EUR, (b) USD/GBP, (c) JPY/USD

Nestacionāru procesu modelēšanai ir divas populāras metodes - laikrindu modeļi ar determinētu trendu un procesi ar vienības sakni. Sākotnēji ar vienības saknes testiem,



27. att.: Valūtu kursu iepriekšējo 100 novērojumu (a) standartnovirze un (b) iepriekšejo 20 novērojumu vidējā vērtība

piemēram, ar Dikeja - Fullera tesu, tiek pārbaudīts, vai procesam ir vienības sakne. Darba ietvaros tiks lietots paplašinātais Dikeja - Fullera tests (turpmāk tekstā ADF), kura vērtības tika aprēķinātas ar R iebūvētās funkcijas *adf.test* palīdzību, un kas pārbauda nulles hipotēzi, ka laikrinda ir gadījuma klejošanas process ar sanesi tātad nestacionāra. Darbā apskatītajiem USD/EUR, GBP/USD, JPY/USD valūtu kursiem ADF testa p-vērtības ir attiecīgi 0.3489, 0.5399 un 0.6464, līdz ar to netiek noraidīta nulles hipotēze. Plašāku informāciju par ADF testu iesējams atrast, piemēram, [14].

Tā kā netika noraidīta nulles hipotēze, tad, lai varētu laikrindai piemeklēt piemērotu $ARMA$ modeli, ar diferencēšanas palīdzību ir jāpanāk laikrindas stacionaritāte.

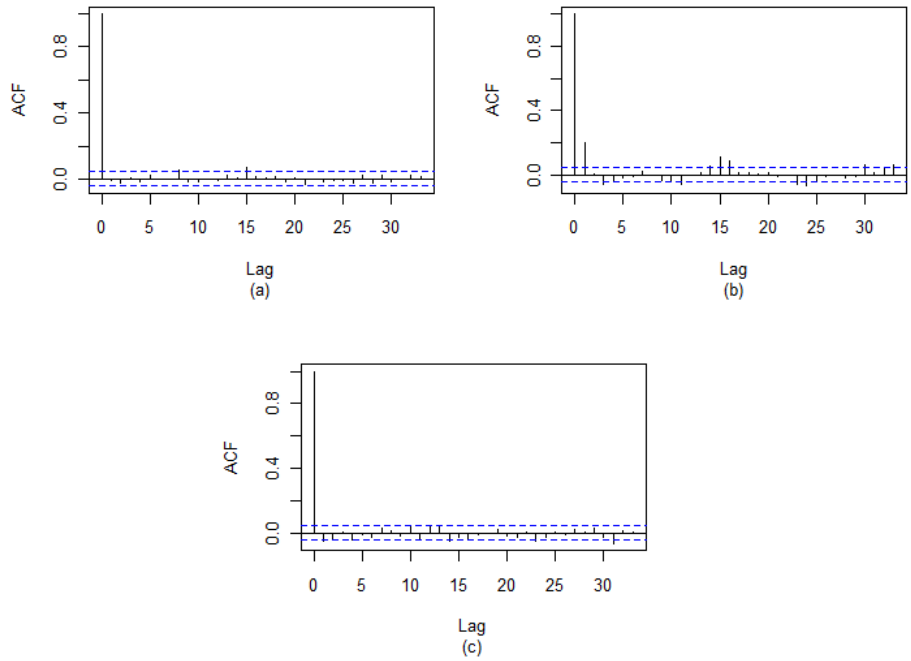
Definīcija 19. Par diferenču operatoru $\triangle$ skailiskās virknēs sauc

$$\triangle Y_t = Y_t - Y_{t-1}. \tag{5.12}$$

Otrās kātras diference $\triangle^2 Y_t = \triangle (\triangle Y_t)$.

No 28. attēla varam izdarīt pieņēmumu, ka diferencētajām laikrindām piemīt stacionaritāte. Arī ADF testa p-vērtības dirferencētajām USD/EUR, GBP/USD, JPY/USD kursu laikrindām ir zem 0.01. Līdz ar to nulles hipotēze Dikeja - Fullera testā tiek noraidīta, un nu jau stacionārajai laikrindai var piemeklēt piemērotāko $ARMA$ modeli. Arī šajā posmā ļoti noderīgi ir ACF un PACF grafiki, skatīt 5. tabulu.

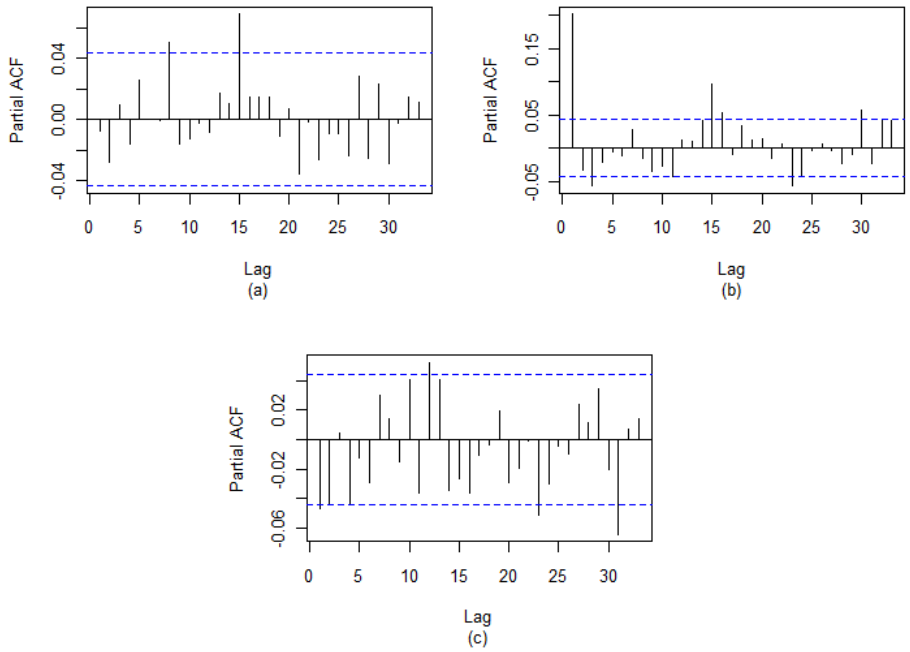
No attēliem 28. un 29., atsaucoties uz 5., varam izdarīt pieņēmumu, ka visām trīs diferencētajām valūtu kursu laikrindām ir piemērots $ARMA(p, q)$ modelis. Izmantojot



28. att.: Valūtu kursu diferencēto laikrindu ACF, (a) USD/EUR, (b) USD/GBP, (c) JPY/USD

5. tabula *ACF un PACF izskats, [15]*

Process	ACF	PACF
AR(p)	Samazinās ģeometriski	0, sākot ar laika atstarpi $p+1$
MA(q)	0, sākot ar laika atstarpi $q+1$	samazinās ģeometriski
ARMA(p,q)	samazinās ģeometriski	samazinās ģeometriski



29. att.: Valūtu kursu diferencēto laikrindu ACF, USD/EUR (a), USD/GBP (b), JPY/USD (c)

R iebūvēto funkciju *auto.arima*, kas balstās uz Akaike informācijas un Švarca-Beijesa kritērijiem, tika iegūts, ka piemērotākie $ARIMA(p,d,q)$, modeļi USD/EUR, GBP/USD un JPY/USD valūtu kursu laikrindām ir attiecīgi $ARIMA(2,1,2)$, $ARIMA(2,1,1)$ un $ARIMA(2,1,0)$. Plašāka informācija par Akaike informācijas un Švarca-Beijesa kritērijiem pieejama, piemēram, [14].

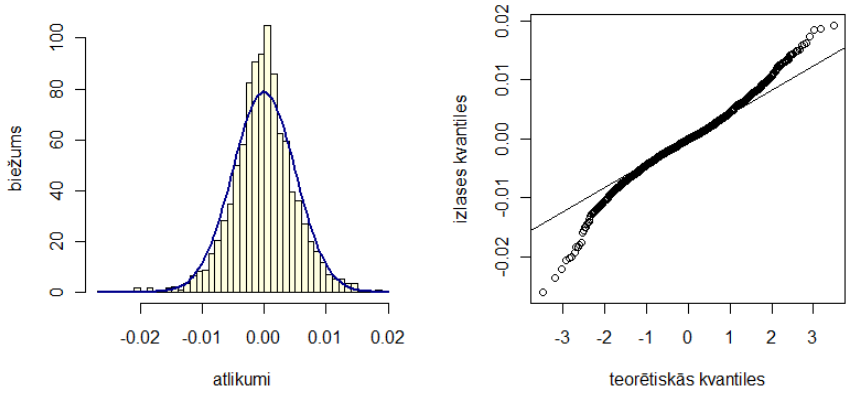
5.3. Izvēlētā ARIMA modeļa piemērotības noteikšana

Trešais solis Boksa-Dženkinsa pieejā ir izvēlētā $ARIMA$ modeļa piemērotības noteikšana, un šajā solī tiek pārbaudīti atlikumi. Pieņemot, ka ir novērtēti parametri ϕ_p un θq, atlikumus $ARIMA(p,d,q)$ modelim var definēt kā

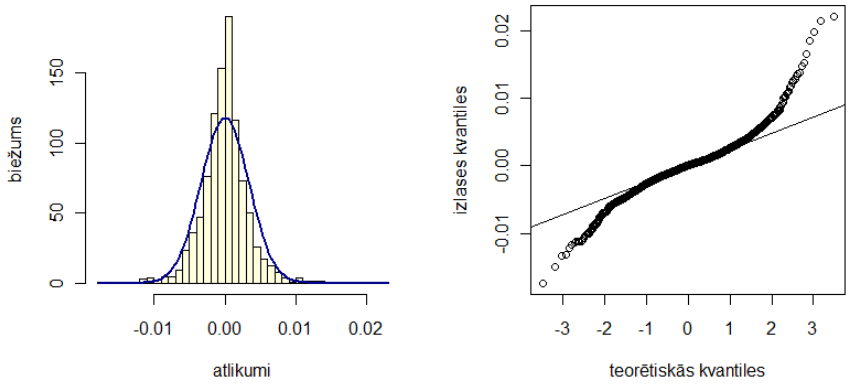
$$\hat{\varepsilon}_t = \hat{\beta(B)}^{-1}\hat{\alpha}(B)(1-B)^d Y_t. \tag{5.13}$$

Lai noteiktu modeļa piemērotību tiks pārbaudīta atlikumu nekorelētība, jo var pierādīt, ka labi piemērotam modelim $\hat{\varepsilon}_t$ ir tuvi baltajam troksnim. Baltajam troksnim empīriskās atokorelācijas var uzskatīt par neatkarīgiem vienādi, normāli sadalītiem gadījuma lielumiem ar vidējo vērtību) un dispersiju $\frac{1}{N}$, [15]. Lai pārbaudītu autokorelāciju atlikumos

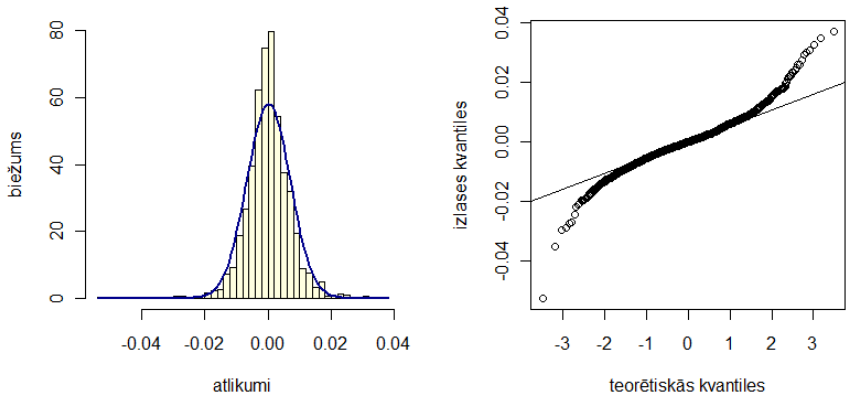
tiek lietota Boksa - Ļuinga statistika, kuras definīciju var atrast, piemēram, [15]. Boksa - Ļuinga statistika liecina, ka nulles hipotēzi, ka apskatīto pielāgoto modeļu atlikumi nav autokorelēti, noraidīt nevar. Taču tā kā pie parametru novērtēšanas un laikrindu prognozēšanas ir svarīga baltā trokšņa normalitāte, tad atlikumiem tiek veikta arī normalitātes pārbaude. Kā jau redzams 30., 31. un 32. attēlos, visticamāk normalitātes pieņēmums atbilstošajiem atlikumiem nav spēkā. To apstiprina arī Kolmogoriva - Smirnova statistikas p-vērtības.



30. att. USD/EUR laikrindai pielāgotā ARIMA(2,1,2) atlikumi



31. att. USD/GBP laikrindai pielāgotā ARIMA(2,1,1) atlikumi



32. att. JPY/USD laikrindai pielāgotā ARIMA(2,1,0) atilkumi

5.4. Izvēlētā modeļa pielietošana prognozēšanā

Lai gan pielāgoto modeļu atlikumi neizturēja normalitātes pārbaudi, šīs apakšnodaļas ietvaros tiks pārbaudīta iepriekš atrasto modeļu spēja veikt prognozes vienam solim uz priekšu. Pieņemsim, ka mūsu rīcībā ir t laikrindas vērtības, un uzdevums ir prognozēt $t+1$ laikrindas vērtību, izmantojot visas iepriekšējās laikrindas vērtības. Pielāgojot kādu no $ARIMA$ modeļiem tika atrasti parametru novērtējumi, un šīs apakšnodaļas ietvaros tiks pieņemts, ka novērojumu sērijas garums ir pietiekams un tādēļ novērtējumus var pieņemt par īstajām parametru vērtībām. Plašāku izklāstu par parametru novērtējumiem meklēt, piemēram, [14].

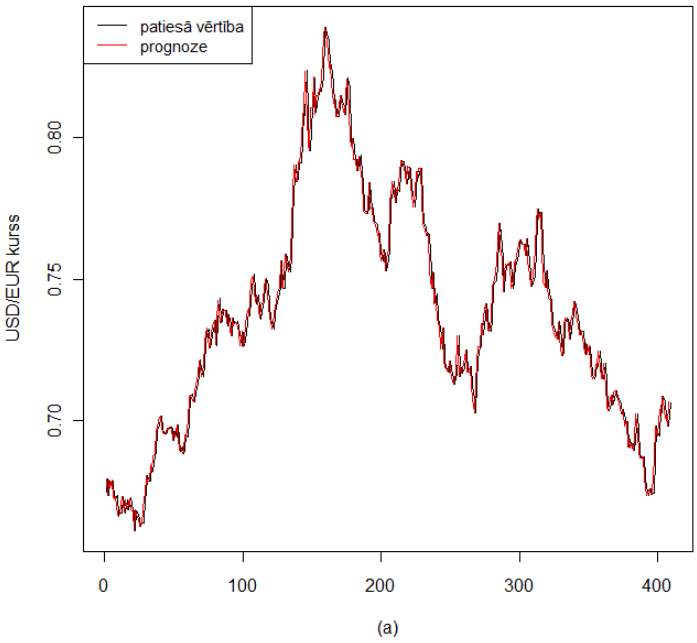
Definīcija 20. Par p*rognozi ar mazāko vidējo kvadrātisko kļūdu* $\hat{y}_t(h)$ sauc funkciju $\hat{y}$ no novērojumiem F_t, kas minimizē nosacīto vidējo vērtību $\mathbb{E}[(Y_{t+h} - \hat{y})^2/F_t]$, kur $F_t = \sigma(Y_1, ..., Y_t)$ ir gadījuma lielumu $Y_1, ..., Y_t$ ģenerētā σ-algebra.

Tātad prognoze ar mazāko vidējo kvadrātisko kļūdu ir $\hat{y}_t(h) = \mathbb{E}[Y_{t+h}|F_t]$, [15].

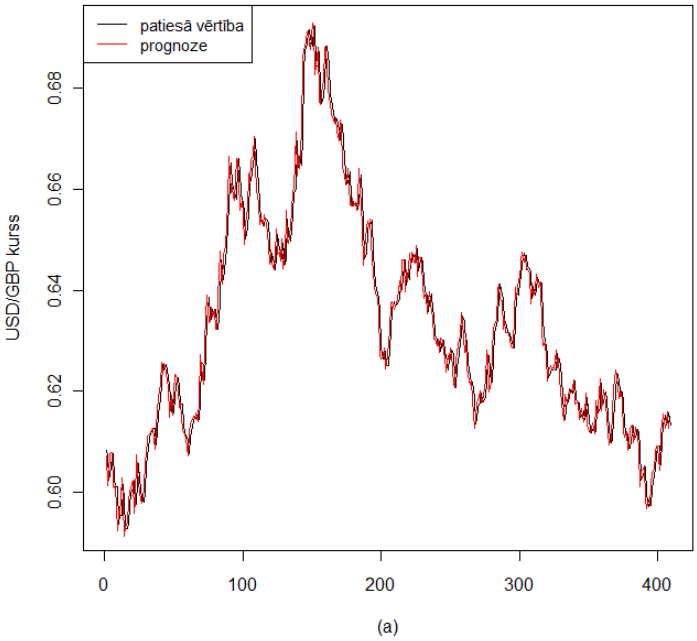
Definīcija 21. Par *prognozes kļūdu* $e_t(h)$ sauc $e_t(h) = Y_{t+h} - \hat{y}_t(h)$.

Tātad mūsu gadījumā prognoze vienam solim uz priekšu $e_t(1) = Y_{t+h} - \hat{y}_t(1) = \varepsilon_{t+1}$, [15].

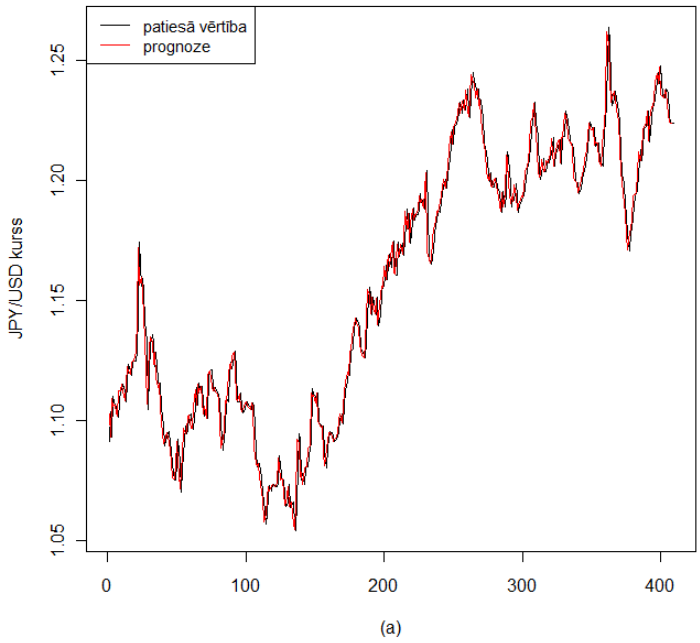
Prognožu veidošanai tika izmantotas R iebūvētā komandas *arima* un *forecast*, $\hat{y}_{t+1}$ atrašanai tika izmantotas visas iepriekšējās t vērtības un pielāgoti iepriekš atrastie $ARIMA$ modeļi. 33., 34. un 35. attēlos redzams, ka pielāgotie modeļi vienam solim uz priekšu spēj sniegt ļoti akurātu prognozi.



33. att. USD/EUR laikrindai pielāgotā $ARIMA(2,1,2)$ modeļa prognoze, $h = 1$



34. att. USD/GBP laikrindai pielāgotā $ARIMA(2,1,1)$ modeļa prognoze, $h = 1$



35. att. JPY/USD laikrindai pielāgotā $ARIMA(2,1,0)$ modeļa prognoze, $h = 1$

6. Apskatīto modeļu salīdzinājums

Lai salīdzinātu iepriekš aprakstītās prognozēšanas tehnikas, tiks izmantoti kļūdu novērtējumi - vidējā absolūtā kļūda (turpmāk tekstā MAD), vidējā kvadrātiskā kļūda (turpmāk tekstā MSE) un vidējā absolūtā procentuālā kļūda (turpmāk tekstā MAPE).

$$MAD = \frac{1}{2}\sum_{i=1}^{n} |e_i|, \tag{6.1}$$

$$MSE = \frac{1}{2}\sum_{i=1}^{n} e_i^2, \tag{6.2}$$

$$MAPE = \frac{1}{2}\left(\sum_{i=1}^{n}\left|\frac{e_i}{y_i}\right|100\right), \tag{6.3}$$

kur $e_i = y_i - \hat{y}_i$.

Apskatītajiem datiem, ja par mērauklu tiek izmantoti MAD, MSE un $MAPE$ kļūdu novērtējumi, labākus rezultātus nekā apskatītie neironu tīklu modeļi, uzrāda pielāgotie $ARIMA$ modeļi. Otro vietu ieņem neironu tīkls ar kļūdu atgriezeniskās izplatīšanās apmācīšanās algoritmu un apmācīšanās parametru $\eta = 0.3$. Savukārt USD/GBP gadījumā visvājākos rezultātus uzrāda kļūdu atgriezeniskās izplatīšanās algoritms ar optimālo ap-

6. tabula *Kļūdas neironu tīkla modelim ar BP apmācīšanās alogritmu*

Valūtu pāris	MAD	MSE	MAPE
USD/EUR	0.00488	0.0000385911	0.65497
USD/GBP	0.00487	0.0000386047	0.65509
JPY/USD	0.00692	0.0000798926	0.59511

7. tabula: *Kļūdas neironu tīkla modelim ar BP algoritmu un optimālo apmācīšanās parametru*

Valūtu pāris	MAD	MSE	MAPE
USD/EUR	0.02737	0.0002586	1.71953
USD/GBP	0.00906	0.0001276	1.42256
JPY/USD	0.01387	0.0003068	1.20241

8. tabula *Kļūdas neironu tīkla modelim ar adaptīvo BP apmācīšanās alogritmu*

Valūtu pāris	MAD	MSE	MAPE
USD/EUR	0.01037	0.0001692	1.39782
USD/GBP	0.01053	0.0001742	1.64929
JPY/USD	0.01214	0.0002441	1.03946

9. tabula *Kļūdas pielāgotajam ARIMA modelim*

Valūtu pāris	MAD	MSE	MAPE
USD/EUR	0.004126	0.00002868	0.5568
USD/GBP	0.002966	0.00001435	0.4676
JPY/USD	0.005133	0.00004829	0.4448

mācīšanās parametru un momenta faktoriem un USD/EUR, JPY/USD gadījumos - kļūdu atgriezeniskās izplatīšanās algoritms ar optimālo apmācīšanās parametru.

Viens no iespējamiem iemesliem, kādēļ, atšķirībā no [1] grāmatas autoru veiktā pētījuma, adaptīvais BP algoritms uzrāda sliktākus rezultātus nekā parastais BP algoritms, ir lielākas svārstības testa datu kopā. Tāpat redzams, ka BP algoritms ar optimālo apmācīšanās parametru pie salīdzinoši straujām laikrindas vērtību izmaiņām USD/GBP un JPY/USD gadījumā uzrāda daudz asākus kāpumus un kritumus, kas krietni pasliktina rezultātu. Un no grafikiem var izdarīt pieņēmumu, ka īpaši sliktus rezultātus šī apmācīšanās tehnika uzrāda pie ekstremālām svārstībām. Iemesli, kādēļ tas tā notiek tiek atstāti kā viena no iespējamām turpmākas izpētes tēmām.

7. Secinājumi

Darbā tika aplūkoti un teorētiski pamatoti divi dažādi neironu tīklu apmācīšanās aloritmi - kļūdu atgriezeniskās izplatīšanās algoritms ar fiksētu apmācīšanās parametru, adaptīvais kļūdu atgriezeniskās izplatīšanās algoritms ar optimālo apmācīšanās parametru, un adaptīvais kļūdu atgriezeniskās izplatīšanās algoritms ar optimālo apmācīšanās parametru un momenta faktoru. Kļūdu atgriezeniskās izplatīšanās algoritmiem ar fiksētu apmācīšanās parametru, ar optimālo apmācīšanās parametru un gan ar optimālo apmācīšanās parametru, gan momenta faktoru tika parādīts, kā notiek svaru izmaiņas. Apskatītajiem USD/EUR, USD/GBP un JPY/USD valūtu kursiem tika pielietoti minētie apmācīšanās algoritmi un iegūtas prognozes vienam solim uz priekšu. Iegūtie rezultāti tika salīdzināti ar pielāgotajiem $ARIMA$ modeļiem.

Iemesls, kas literatūrā tiek minēts kā arguments par labu neironu tīklu izmantošanai, ir atsevišķas neironu tīklu īpašības, kas ļauj modelēt sarežģītas sakarības. Kā vienu no šādām īpašībām var minēt faktu, ka neironu tīkls kalpo kā universāls funkciju aproksimators, kas bez jebkādiem pieņēmumiem attiecībā uz datiem spēj attēlot jebkuru nelineāru funkciju, [1, 65. lpp.].

Ņemot vērā faktu, ka tā dēvētie spekulanti izmanto kredītsviru, tad jau ļoti mazas valūtu kursu svārstības var ievērojami palielināt peļņu vai zaudējumus. Līdz ar to parasti spekulanti cer nopelnīt no īstermiņa svārstībām, un tādēļ īpaša uzmanība tiek pievērsta dažādām analīzes metodēm. Visbiežāk tās ir tehniskā un fundamentālā analīze, kas at-

tiecīgi pievērš uzmanību kursu tendencei un ekonomiskājiem faktoriem. Šī darba ietvaros tika meklēta alternatīva metode tradicionālajiem tehniskās analīzes indikatoriem, kurus iegūst, izmantojot vēsturiskos kursu datus. Balstoties uz šī darba ietvaros veikto pētījumu, var teikt, ka praksē no apskatītajiem neironu tīklu apmācīšanās algoritmiem saistošs varētu šķists klasiskais kļūdu atgriezeniskās izplatīšanās algoritms ar konstantu apmācīšanās parametru. Tā kā ir iespējama situācija, kurā prognozētā $\hat{y}_{t+1}$ vērtība ir tuva y_t, lai izslēgtu reakciju uz mazām izmaiņām, tiek piedāvāts lietot sekojošus valūtu tirdzniecības likumus:

- Ja $(\hat{y}_{t+1} - y_t) > c$, tad tendence ir augšupejoša un tirdzniecības stratēģija ir "pirkt";
- Ja $(\hat{y}_{t+1} - y_t) < c$, tad tendence ir lejupejoša un tirdzniecības stratēģija ir "pārdot";
- Ja $(\hat{y}_{t+1} - y_t) = c$, tad tirdzniecības stratēģija ir "turēt" vai "fiksēt peļņu/zaudējumus",

kur c ir individuāli noteikts līmenis, pie kura ir izdevīgi reaģēt uz modeļa piedāvātajiem signāliem.

No iegūtajām modeļu prognožu kļūdām un apskatītajiem grafikiem var secināt, ka vislabāko rezultātu pēc iepriekš aprakstītajiem tirdzniecības likumiem uzrādītu $ARIMA$ modeļi, kam seko BP algoritms ar konstantu apmācīšanās parametru $\eta = 0.3$. Viens no iespējamiem iemesliem, kādēļ, adaptīvais BP algoritms uzrāda vēl sliktākus rezultātus nekā parastais BP algoritms, ir salīdzinoši lielas svārstības datu kopā. Tāpat redzams, ka BP algoritms ar optimālo apmācīšanās parametru pie salīdzinoši straujām laikrindas vērtību izmaiņām USD/GBP un JPY/USD gadījumā uzrāda daudz asākus kāpumus un kritumus, kas krietni pasliktina rezultātu. Savukārt pie ilgstošākas tendences adaptīvais BP algoritms uzrāda salīdzinoši labākus rezultātus. Līdz ar to var izdarīt pieņēmumu, ka īpaši sliktus rezultātus šī apmācīšanās tehnika uzrāda pie ekstremālām svārstībām. Iemesli, kādēļ tas tā ir, tiek atstāti kā viena no iespējamām turpmākas izpētes tēmām.

Izmantotā literatūra un avoti

[1] L. Yu, S. Wang, and K.K. Lai. *Foreign exchange rate forecasting with artificial neural networks*, volume 107. Springer Verlag, 2007.

[2] G.E.P. Box, G.M. Jenkins, and G.C. Reinsel. *Time series analysis*. Holden-day San Francisco, 1970.

[3] A. Lapedes and R. Farber. Nonlinear signal processing using neural networks: Prediction and system modelling. Technical report, 1987.

[4] Bank for International Settlements. *Report on global foreign exchange market activity in 2010*. 2010.

[5] J. Zuters. *Neironu tīkli*. Rīga, 2010. Mācību materiāls.

[6] B. Krose, B. Krose, P. van der Smagt, and P. Smagt. *An introduction to neural networks*. Citeseer, 1993.

[7] Lean Yu, Shouyang Wang, and K.K. Lai. An integrated data preparation scheme for neural network data analysis. *IEEE Transactions on Knowledge and Data Engineering*, 18:217–230, 2006.

[8] M. Small. *Applied nonlinear time series analysis: applications in physics, physiology and finance*. World Scientific Pub Co Inc, 2005.

[9] M. Minsky and P. Seymour. *Perceptrons*. MIT press, 1969.

[10] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. parallel distributed processing, vol. 1. *Foundations. MIT Press, Cambridge*, 1986.

[11] A.S. Andreou and G.A. Zombanakis. Computational intelligence in exchange-rate forecasting. *Working Papers*, 2006.

[12] L. Torgo. *Data Mining with R: Learning with Case Studies*. Chapman and Hall/CRC, 2010.

[13] S. Makridakis, S.C. Wheelwright, and F. Hyndman. *Forecasting: Methods and Applications*. John Wiley and Sons, USA, 1998.

[14] J.D. Hamilton. *Time series analysis*. Princeton Univ Pr, 1994.

[15] N. Siņenko. *Laikrindu analīze*. Rīga. Lekciju konspekts.

1. Pielikums

$\triangle\hat{\mathbf{y}}^{\mathbf{m}}(\mathbf{t}+\mathbf{1})$ izvedums

$$
\begin{aligned}
\triangle\hat{y}^m(t+1) &\approx \left(\frac{d\hat{y}^m(t+1)}{dt}\right)\triangle t = \begin{bmatrix} \dfrac{d\hat{y}_1^m(t+1)}{dt} \\ \dfrac{d\hat{y}_2^m(t+1)}{dt} \\ \vdots \\ \dfrac{d\hat{y}_k^m(t+1)}{dt} \end{bmatrix}\triangle t \\
&= \begin{bmatrix} f'_{2(1),m}\left(F_{1,m}^T\dfrac{dV_1}{dt}+V_1^TF'_{1,m}\dfrac{dW}{dt}x_m\right) \\ f'_{2(2),m}\left(F_{1,m}^T\dfrac{dV_2}{dt}+V_2^TF'_{1,m}\dfrac{dW}{dt}x_m\right) \\ \vdots \\ f'_{2(k),m}\left(F_{1,m}^T\dfrac{dV_k}{dt}+V_k^TF'_{1,m}\dfrac{dW}{dt}x_m\right) \end{bmatrix}\triangle t \\
&\approx \begin{bmatrix} f'_{2(1),m}\left(F_{1,m}^T\dfrac{\triangle V_1}{\triangle t}+V_1^TF'_{1,m}\dfrac{\triangle W}{\triangle t}x_m\right) \\ f'_{2(2),m}\left(F_{1,m}^T\dfrac{\triangle V_2}{\triangle t}+V_2^TF'_{1,m}\dfrac{\triangle W}{\triangle t}x_m\right) \\ \vdots \\ f'_{2(k),m}\left(F_{1,m}^T\dfrac{\triangle V_k}{\triangle t}+V_k^TF'_{1,m}\dfrac{\triangle W}{\triangle t}x_m\right) \end{bmatrix}\triangle t \\
&= \begin{bmatrix} f'_{2(1),m}(F_{1,m}^T\triangle V_1+V_1^TF'_{1,m}\triangle Wx_m) \\ f'_{2(2),m}(F_{1,m}^T\triangle V_2+V_2^TF'_{1,m}\triangle Wx_m) \\ \vdots \\ f'_{2(k),m}(F_{1,m}^T\triangle V_k+V_k^TF'_{1,m}\triangle Wx_m) \end{bmatrix} \\
&= \begin{bmatrix} f'_{2(1),m}(F_{1,m}^T\sum_{j=1}^N\triangle V_{1j}+V_1^TF'_{1,m}\triangle Wx_m) \\ f'_{2(2),m}(F_{1,m}^T\sum_{j=1}^N\triangle V_{2j}+V_2^TF'_{1,m}\triangle Wx_m) \\ \vdots \\ f'_{2(k),m}(F_{1,m}^T\sum_{j=1}^N\triangle V_{kj}+V_k^TF'_{1,m}\triangle Wx_m) \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
&= \begin{bmatrix} f'_{2(1),m}[F_{1,m}^T \left(\eta \sum_{j=1}^N F_{1j} e_{1j} f'_{2(1),j}\right) + V_1^T F'_{1,m} \left(\eta \sum_{j=1}^N F'_{1j} V F'_{2j} e_j x_j^T\right) x_m] \\ f'_{2(2),m}[F_{1,m}^T \left(\eta \sum_{j=1}^N F_{1j} e_{2j} f'_{2(2),j}\right) + V_2^T F'_{1,m} \left(\eta \sum_{j=1}^N F'_{1j} V F'_{2j} e_j x_j^T\right) x_m] \\ \vdots \\ f'_{2(k),m}[F_{1,m}^T \left(\eta \sum_{j=1}^N F_{1j} e_{kj} f'_{2(k),j}\right) + V_k^T F'_{1,m} \left(\eta \sum_{j=1}^N F'_{1j} V F'_{2j} e_j x_j^T\right) x_m] \end{bmatrix} \\
&= \eta \sum_{j=1}^N \begin{bmatrix} f'_{2(1),m}(F_{1,m}^T F_{1j} e_{1j} f'_{2(1),j} + V_1^T F'_{1,m} F'_{1j} V F'_{2j} e_j x_j^T x_m) \\ f'_{2(2),m}(F_{1,m}^T F_{1j} e_{2j} f'_{2(2),j} + V_2^T F'_{1,m} F'_{1j} V F'_{2j} e_j x_j^T x_m) \\ \vdots \\ f'_{2(k),m}(F_{1,m}^T F_{1j} e_{kj} f'_{2(k),j} + V_k^T F'_{1,m} F'_{1j} V F'_{2j} e_j x_j^T x_m) \end{bmatrix} \\
&= \eta \sum_{j=1}^N \begin{bmatrix} f'_{2(1),m} & 0 & \cdots & 0 \\ 0 & f'_{2(2),m} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f'_{2(k),m} \end{bmatrix} \\
&\times \begin{bmatrix} F_{1,m}^T F_{1j} e_{1j} f'_{2(1),j} + V_1^T F'_{1,m} F'_{1j} V F'_{2j} e_j x_j^T x_m \\ F_{1,m}^T F_{1j} e_{2j} f'_{2(2),j} + V_2^T F'_{1,m} F'_{1j} V F'_{2j} e_j x_j^T x_m \\ \vdots \\ F_{1,m}^T F_{1j} e_{kj} f'_{2(k),j} + V_k^T F'_{1,m} F'_{1j} V F'_{2j} e_j x_j^T x_m \end{bmatrix} \\
&= \eta \sum_{j=1}^N F'_{2,m}(F'_{2j} e_j F_{1,m}^T F_{1j} + V^T F'_{1,m} F'_{1j} V F'_{2j} e_j x_j^T x_m) \\
&= \eta \sum_{j=1}^N F'_{2,m}(F_{1,m}^T F_{1j} I_{k^2} + V^T F'_{1,m} F'_{1j} V x_j^T x_m) F'_{2j} e_j.
\end{aligned}
$$

2. Pielikums

$\triangle\hat{\mathbf{y}}(\mathbf{t}+\mathbf{1})$ izvedums

$$
\begin{aligned}
\triangle\hat{y}(t+1) &= \begin{bmatrix} \triangle\hat{y}_1(t+1) \\ \triangle\hat{y}_2(t+1) \\ \vdots \\ \triangle\hat{y}_m(t+1) \\ \vdots \\ \triangle\hat{y}_N(t+1) \end{bmatrix} \approx \begin{bmatrix} \eta\sum_{j=1}^{N} F'_{2,1}(F_{1,1}^T F_{1j} I_{k^2} + V^T F'_{1,1} F'_{1j} V x_j^T x_1) F'_{2j} e_j \\ \eta\sum_{j=1}^{N} F'_{2,2}(F_{1,2}^T F_{1j} I_{k^2} + V^T F'_{1,2} F'_{1j} V x_j^T x_2) F'_{2j} e_j \\ \vdots \\ \eta\sum_{j=1}^{N} F'_{2,m}(F_{1,m}^T F_{1j} I_{k^2} + V^T F'_{1,m} F'_{1j} V x_j^T x_m) F'_{2j} e_j \\ \vdots \\ \eta\sum_{j=1}^{N} F'_{2,N}(F_{1,N}^T F_{1j} I_{k^2} + V^T F'_{1,N} F'_{1j} V x_j^T x_N) F'_{2j} e_j \end{bmatrix} \\
&= \eta \begin{bmatrix} F'_{21} & 0 & \cdots & 0 \\ 0 & F'_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & F'_{2N} \end{bmatrix} \\
&\times \begin{bmatrix} F_{11}^T F_{11} I_{k^2} & F_{11}^T F_{12} I_{k^2} & \cdots & F_{11}^T F_{1N} I_{k^2} \\ F_{12}^T F_{11} I_{k^2} & F_{12}^T F_{12} I_{k^2} & \cdots & F_{12}^T F_{1N} I_{k^2} \\ \vdots & \vdots & \ddots & \vdots \\ F_{1N}^T F_{11} I_{k^2} & F_{1N}^T F_{12} I_{k^2} & \cdots & F_{1N}^T F_{1N} I_{k^2} \end{bmatrix} \\
&+ \begin{bmatrix} x_1^T x_1 & x_1^T x_2 & \cdots & x_1^T x_N \\ x_2^T x_1 & x_2^T x_2 & \cdots & x_2^T x_N \\ \vdots & \vdots & \ddots & \vdots \\ x_N^T x_1 & x_N^T x_2 & \cdots & x_N^T x_N \end{bmatrix} \\
&\times V^T \begin{bmatrix} F_{11}^T F_{11} & F_{11}^T F_{12} & \cdots & F_{11}^T F_{1N} \\ F_{12}^T F_{11} & F_{12}^T F_{12} & \cdots & F_{12}^T F_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ F_{1N}^T F_{11} & F_{1N}^T F_{12} & \cdots & F_{1N}^T F_{1N} \end{bmatrix} \bar{V} \\
&\times \begin{bmatrix} F'_{21} & 0 & \cdots & 0 \\ 0 & F'_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & F'_{2N} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix} \\
&= \eta F'_2[(F_1^T F_1) I_{k^2} + (X^T X)(V F'_1 F'_1 V)] F'_2 e(t) = \eta\xi(t) e(t).
\end{aligned}
$$

3. Pielikums

Programmu kodi no R

```
library(xlsReadWrite)
library(fArma)
library(ts)
library(nnet)
library(neuralnet)
xls.getshlib()
eur<- read.xls("C:/Users/Maris/Kvants/Diplomdarbs/usdeur.xls",
colNames = c("DATE", "VOLUME", "OPEN", "CLOSE", "LOW", "HIGH"),
from = 1, rowNames = FALSE, type = "data.frame")
eur<-as.matrix(eur)
jpy<- read.xls("C:/Users/Maris/Kvants/Diplomdarbs/jpyusd.xls",
colNames = c("DATE", "VOLUME", "OPEN", "CLOSE", "LOW", "HIGH"),
from = 1, rowNames = FALSE, type = "data.frame")
jpy<-as.matrix(jpy)
gbp<- read.xls("C:/Users/Maris/Kvants/Diplomdarbs/usdgbp.xls",
colNames = c("DATE", "VOLUME", "OPEN", "CLOSE", "LOW", "HIGH"),
from = 1, rowNames = TRUE, type = "data.frame")
gbp<-as.matrix(gbp)

eur.c<-c()
for(i in 1:2000){
eur.c[i]<-as.numeric(eur[i+1,4])
}

gbp.c<-c()
for(i in 1:2000){
gbp.c[i]<-as.numeric(gbp[i+1,4])
}
gbp.c[1763]<-mean(gbp.c[1762],gbp.c[1764])
gbp.c[1869]<-mean(gbp.c[1868],gbp.c[1870])
```

```
jpy.c<-c()
for(i in 1:2000){
jpy.c[i]<-as.numeric(jpy[i+1,4])
}

par(mfrow=c(3,1))
plot(eur.c, type="l", col="dark blue", xlab="(a)", ylab="")
plot(gbp.c, type="l", col="dark blue", xlab="(b)", ylab="")
plot(jpy.c, type="l", col="dark blue", xlab="(c)", ylab="")

########Ienesīgumiem########

h.returns <- function(x,h=1) {
diff(x,lag=h)/x[1:(length(x)-h)]
}
eur.r<-h.returns(eur.c)
gbp.r<-h.returns(gbp.c)
jpy.r<-h.returns(jpy.c)

par(mfrow=c(3,1))
plot(eur.r, type="l", col="dark blue", xlab="(a)", ylab="")
plot(gbp.r, type="l", col="dark blue", xlab="(b)", ylab="")
plot(jpy.r, type="l", col="dark blue", xlab="(c)", ylab="")

mean(eur.r)
mean(gbp.r)
mean(jpy.r)
sd(eur.r)
sd(gbp.r)
sd(jpy.r)
```

```
par(mfrow=c(3,1))
g<-eur.r
hist(g, prob=TRUE, breaks=40, main="", ylab="Biežums",
col="light yellow")
curve(dnorm(x, mean=mean(g), sd=sd(g)), add=TRUE,
col="dark blue", lwd=2)
g<-gbp.r
hist(g, prob=TRUE, breaks=40, main="", ylab="Biežums",
col="light yellow")
curve(dnorm(x, mean=mean(g), sd=sd(g)), add=TRUE,
col="dark blue", lwd=2)
g<-jpy.r
hist(g, prob=TRUE, breaks=40, main="", ylab="Biežums",
col="light yellow")
curve(dnorm(x, mean=mean(g), sd=sd(g)), add=TRUE,
col="dark blue", lwd=2)


########Prognozesana ar BPNN########


########EUR########

embeded.dataset <- function(data,quote="Close",hday=1,emb=9) {
ds <- data.frame(embed(data,emb+hday))
ds <- ds[,c(1,(1+hday):(hday+emb))]
names(ds) <- c(paste("r",hday,".f",hday,sep=""),
paste("r",hday,".t",0:(emb-1),sep=""))
ds
}
eur<-as.matrix(eur.c)
eur.data <- embeded.dataset(eur,hday=1)
```

```
names(eur.data)

eur.train <- eur.data[1:1592,]
eur.test <- eur.data[1593:1991,]

nn <- nnet(r1.f1 ~ .,data=eur.train[,-ncol(eur.train)],
linout=T,size=21,decay=0.01,maxit=2000)
summary(nn)
nn.preds <- predict(nn,eur.test)
par(mfrow=c(1,2))
plot(eur.test[,1],nn.preds,
main="Neural Net Results",xlab="Patiesā vērtība",
ylab="Prognozētā vērtība")
abline(h=0,v=0); abline(0,1,lty=2, lwd=2,col="red")
plot(eur.test[,1], type="l",col="black", xlab="t",
ylab="USD/EUR kurss")
lines(nn.preds, col="red")
legend("topleft", cex=0.9, c("patiesā vērtība",
"prognoze"), lty=1, col=c("black", "red"))

madd<-function(c,f){
m<-c()
m[1]<-abs(f[1]-c[1])
for (i in 2:(length(c)))
{
m[i]<-m[i-1]+abs(f[i]-c[i])
}
m
}
madd(eur.test[,1],nn.preds)[399]/399
mse<-function(c,f){
m<-c()
```

```
m[1]<-(f[1]-c[1])^2
for (i in 2:(length(c)))
{
m[i]<-m[i-1]+(f[i]-c[i])^2
}
m
}
mse(eur.test[,1],nn.preds)[399]/399
mape<-function(c,f){
m<-c()
m[1]<-abs((f[1]-c[1])/c[1])*100
for (i in 2:(length(c)))
{
m[i]<-m[i-1]+abs((f[i]-c[i])/c[i])*100
}
m
}
mape(eur.test[,1],nn.preds)[399]/399


########GBP########

gbp<-as.matrix(gbp.c)
eur.data <- embeded.dataset(gbp,hday=1)
names(eur.data)
eur.data[1:2,]
length(eur.data[,3])*0.8
eur.train <- eur.data[1:1592,]
eur.test <- eur.data[1593:1991,]

nn <- nnet(r1.f1 ~ .,data=eur.train[,-ncol(eur.train)],
linout=T,size=21,decay=0.01,maxit=2000)
```

```
summary(nn)
nn.preds <- predict(nn,eur.test)
par(mfrow=c(1,2))
plot(eur.test[,1],nn.preds,
main="Neural Net Results",xlab="Patiesā vērtība",
ylab="Prognozētā vērtība")
abline(h=0,v=0); abline(0,1,lty=2, lwd=2,col="red")
plot(eur.test[,1], type="l",col="black", xlab="t",
ylab="USD/GBP kurss")
lines(nn.preds, col="red")
legend("topright", cex=0.9, c("patiesā vērtība",
"prognoze"), lty=1, col=c("black", "red"))

madd(eur.test[,1],nn.preds)[399]/399
mse(eur.test[,1],nn.preds)[399]/399
mape(eur.test[,1],nn.preds)[399]/399

########JPY########

jpy<-as.matrix(jpy.c)
eur.data <- embeded.dataset(jpy,hday=1)
names(eur.data)
eur.data[1:2,]
length(eur.data[,3])*0.8
eur.train <- eur.data[1:1592,]
eur.test <- eur.data[1593:1991,]

nn <- nnet(r1.f1 ~ .,data=eur.train[,-ncol(eur.train)],
linout=T,size=21,decay=0.01,maxit=2000)
summary(nn)
nn.preds <- predict(nn,eur.test)
par(mfrow=c(1,2))
```

```
plot(eur.test[,1],nn.preds,
main="Neural Net Results",xlab="Patiesā vērtība",
ylab="Prognozētā vērtība")
abline(h=0,v=0); abline(0,1,lty=2, lwd=2,col="red")
plot(eur.test[,1], type="l",col="black", xlab="t",
ylab="JPY/USD kurss")
lines(nn.preds, col="red")
legend("topright", cex=0.9, c("patiesā vērtība",
"prognoze"), lty=1, col=c("black", "red"))

madd(eur.test[,1],nn.preds)[399]/399
mse(eur.test[,1],nn.preds)[399]/399
mape(eur.test[,1],nn.preds)[399]/399

########ARIMA########

library(xlsReadWrite)
library(forecast)
library(tseries)
xls.getshlib()
eur<- read.xls("C:/Users/Maris/Kvants/Diplomdarbs/usdeur.xls",
colNames = c("DATE", "VOLUME", "OPEN", "CLOSE", "LOW", "HIGH"),
from = 1, rowNames = FALSE, type = "data.frame")
eur<-as.matrix(eur)
jpy<- read.xls("C:/Users/Maris/Kvants/Diplomdarbs/jpyusd.xls",
colNames = c("DATE", "VOLUME", "OPEN", "CLOSE", "LOW", "HIGH"),
from = 1, rowNames = FALSE, type = "data.frame")
jpy<-as.matrix(jpy)
gbp<- read.xls("C:/Users/Maris/Kvants/Diplomdarbs/usdgbp.xls",
colNames = c("DATE", "VOLUME", "OPEN", "CLOSE", "LOW", "HIGH"),
from = 1, rowNames = TRUE, type = "data.frame")
gbp<-as.matrix(gbp)
```

```
eur.c<-c()
for(i in 1:2000){
eur.c[i]<-as.numeric(eur[i+1,4])
}

gbp.c<-c()
for(i in 1:2000){
gbp.c[i]<-as.numeric(gbp[i+1,4])
}
gbp.c[1763]<-mean(gbp.c[1762],gbp.c[1764])
gbp.c[1869]<-mean(gbp.c[1868],gbp.c[1870])

jpy.c<-c()
for(i in 1:2000){
jpy.c[i]<-as.numeric(jpy[i+1,4])
}

par(mfrow=c(3,1))
plot(eur.c, type="l", col="dark blue", xlab="(a)", ylab="")
plot(gbp.c, type="l", col="dark blue", xlab="(b)", ylab="")
plot(jpy.c, type="l", col="dark blue", xlab="(c)", ylab="")

mean(eur.c)
mean(gbp.c)
mean(jpy.c)
sd(eur.c)
sd(gbp.c)
sd(jpy.c)
min(eur.c)
min(gbp.c)
min(jpy.c)
```

```
max(eur.c)
max(gbp.c)
max(jpy.c)

par(mfrow=c(2,2))
acf(eur.c, xlab=c("Lag", "(a)"), main="")
acf(gbp.c, xlab=c("Lag", "(b)"),main="")
acf(jpy.c, xlab=c("Lag", "(c)"),main="")

########datu disp un vid vert atkariba no laika########

n<-length(eur.c)
volatility<-function(r){
kl_20<-c()
n2<-n-1900
n3<-1900
nr1<-c()
sk<-1
for (sk in 1:1900)
{
i<-sk
j<-sk+n2-1
kl_20[sk]<-sd(r[i:j])
nr1[sk]<-sk
}
kl_20
}

eur.v<-volatility(eur.c)
gbp.v<-volatility(gbp.c)
jpy.v<-volatility(jpy.c)
```

```
par(mfrow=c(1,2))
plot(eur.v, type="l", col="dark blue", lwd=2,
xlab="(a)",ylab="standartnovirze", ylim=c(0,0.08))
lines(gbp.v, lwd=2, col="dark grey")
lines(jpy.v, lwd=2, col="dark green")
legend("topleft", cex=0.9, c("USD/EUR",
"GBP/USD", "JPY/USD"), lty=1, col=c("dark blue",
"darkgrey", "dark green"))


vid<-function(r){
kl_20<-c()
n2<-n-1980
n3<-1980
nr1<-c()
sk<-1
for (sk in 1:1980)
{
i<-sk
j<-sk+n2-1
kl_20[sk]<-mean(r[i:j])
nr1[sk]<-sk
}
kl_20
}

eur.m<-vid(eur.c)
gbp.m<-vid(gbp.c)
jpy.m<-vid(jpy.c)

plot(eur.m, type="l", col="dark blue", lwd=2,
xlab="(b)",ylab="vidējā vērtība", ylim=c(0.45,1.5))
```

```
lines(gbp.m, lwd=2, col="dark grey")
lines(jpy.m, lwd=2, col="dark green")
legend("topleft", cex=0.9, c("USD/EUR",
"GBP/USD", "JPY/USD"), lty=1, col=c("dark blue",
"darkgrey", "dark green"))

########ADF tests########

adf.test(rnorm(1000))
adf.test(eur.c)
adf.test(gbp.c)
adf.test(jpy.c)

########Diferencešana########

a<-c()
diference <- function(x) {
for (i in 2:length(x)){
a[i-1]<-x[i]-x[i-1]
}
a
}
eur.d<-diference(eur.c)
gbp.d<-diference(gbp.c)
jpy.d<-diference(jpy.c)

par(mfrow=c(2,2))
acf(eur.d, main="(a)")
acf(gbp.d,  main="(b)")
acf(jpy.d,  main="(c)")
```

```
adf.test(rnorm(1000))
adf.test(eur.d)
adf.test(gbp.d)
adf.test(jpy.d)

par(mfrow=c(2,2))
acf(eur.d, xlab=c("Lag", "(a)"), main="")
acf(gbp.d, xlab=c("Lag", "(b)"),main="")
acf(jpy.d, xlab=c("Lag", "(c)"),main="")

par(mfrow=c(2,2))
pacf(eur.d, xlab=c("Lag", "(a)"), main="")
pacf(gbp.d, xlab=c("Lag", "(b)"),main="")
pacf(jpy.d, xlab=c("Lag", "(c)"),main="")

########arima pielagosana########

library(forecast)

auto.arima(eur.d)
auto.arima(gbp.d)
auto.arima(jpy.d)

auto.arima(eur.c)
auto.arima(gbp.c)
auto.arima(jpy.c)

########arima kludu analize########

eur.arima<-arima(eur.c,order=c(2,1,2))
Box.test(eur.arima$residuals)
par(mfrow=c(1,2))
```

```
plot(eur.arima$residuals, type="l")
acf(eur.arima$residuals)
hist(eur.arima$residuals, prob=TRUE, breaks=40,
main="", ylab="biežums", xlab="atlikumi", col="light yellow")
curve(dnorm(x, mean=mean(eur.arima$residuals),
sd=sd(eur.arima$residuals)),
add=TRUE, col="dark blue", lwd=2)
qqnorm(eur.arima$residuals, main="", xlab="teorētiskās kvantiles",
 ylab="izlases kvantiles")
qqline(eur.arima$residuals)
ks.test(eur.arima$residuals, "pnorm", mean=mean(eur.arima$residuals)
, sd=sd(eur.arima$residuals))

gbp.arima<-arima(gbp.c,order=c(2,1,1))
Box.test(gbp.arima$residuals)
par(mfrow=c(1,2))
plot(gbp.arima$residuals, type="l")
acf(gbp.arima$residuals)
hist(gbp.arima$residuals, prob=TRUE, breaks=40,
main="", ylab="biežums", xlab="atlikumi", col="light yellow")
curve(dnorm(x, mean=mean(gbp.arima$residuals), s
d=sd(gbp.arima$residuals)),
add=TRUE, col="dark blue", lwd=2)
qqnorm(gbp.arima$residuals, main="", xlab="teorētiskās kvantiles",
 ylab="izlases kvantiles")
qqline(gbp.arima$residuals)
ks.test(gbp.arima$residuals, "pnorm", mean=mean(gbp.arima$residuals),
 sd=sd(gbp.arima$residuals))

jpy.arima<-arima(jpy.c,order=c(2,1,1))
Box.test(jpy.arima$residuals)
par(mfrow=c(1,2))
```

```
plot(jpy.arima$residuals, type="l")
acf(jpy.arima$residuals)
hist(jpy.arima$residuals, prob=TRUE, breaks=40, main="",
ylab="biežums", xlab="atlikumi", col="light yellow")
curve(dnorm(x, mean=mean(jpy.arima$residuals),
sd=sd(jpy.arima$residuals)),
add=TRUE, col="dark blue", lwd=2)
qqnorm(jpy.arima$residuals, main="", xlab="teorētiskās kvantiles",
ylab="izlases kvantiles")
qqline(jpy.arima$residuals)
ks.test(jpy.arima$residuals, "pnorm", mean=mean(jpy.arima$residuals),
sd=sd(jpy.arima$residuals))

plot(gbp.c-forecast(gbp.arima, h=1)$fitted, type="l")

########prognozes########

f<-c()
sk<-0
for (i in 1592:2000){
eur1<-eur.c[1+sk:i]
eur.arima<-arima(eur1,order=c(2,1,2))
f[i-1591]<-forecast(eur.arima, h=1)$mean
sk<-sk+1
}
f



f1<-c()
sk<-0
for (i in 1592:2000){
gbp1<-gbp.c[1+sk:i]
```

```
gbp.arima<-arima(gbp1,order=c(2,1,1))
f1[i-1591]<-forecast(gbp.arima, h=1)$mean
sk<-sk+1
}
f1


f2<-c()
sk<-0
for (i in 1592:2000){
jpy1<-jpy.c[1+sk:i]
jpy.arima<-arima(jpy1,order=c(2,1,0))
f2[i-1591]<-forecast(jpy.arima, h=1)$mean
sk<-sk+1
}
f2

par(mfrow=c(1,1))
plot(eur.c[1592:2000], col="black", lwd=1, xlab="(a)",
type="l", ylab="USD/EUR kurss")
lines(f, type="l", lwd=1, col="red")
legend("topleft", cex=0.9, c("patiesā vērtība",
"prognoze"), lty=1, col=c("black", "red"))
plot(gbp.c[1592:2000], col="black", lwd=1, xlab="(a)",
type="l", ylab="USD/GBP kurss")
lines(f1, type="l", lwd=1, col="red")
legend("topleft", cex=0.9, c("patiesā vērtība",
"prognoze"), lty=1, col=c("black", "red"))
plot(jpy.c[1592:2000], col="black", lwd=1, xlab="(a)",
type="l", ylab="JPY/USD kurss")
lines(f2, type="l", lwd=1, col="red")
legend("topleft", cex=0.9, c("patiesā vērtība",
```

```
"prognoze"), lty=1, col=c("black", "red"))

########Kludas########

madd<-function(c,f){
m<-c()
m[1]<-abs(f[1]-c[1])
for (i in 2:(length(c)))
{
m[i]<-m[i-1]+abs(f[i]-c[i])
}
m
}

madd(eur.c[1592:2000],f)[408]/408
madd(gbp.c[1592:2000],f1)[408]/408
madd(jpy.c[1592:2000],f2)[408]/408

mse<-function(c,f){
m<-c()
m[1]<-(f[1]-c[1])^2
for (i in 2:(length(c)))
{
m[i]<-m[i-1]+(f[i]-c[i])^2
}
m
}


mse(eur.c[1592:2000],f)[408]/408
mse(gbp.c[1592:2000],f1)[408]/408
mse(jpy.c[1592:2000],f2)[408]/408
```

```
mape<-function(c,f){
m<-c()
m[1]<-abs((f[1]-c[1])/c[1])*100
for (i in 2:(length(c)))
{
m[i]<-m[i-1]+abs((f[i]-c[i])/c[i])*100
}
m
}


mape(eur.c[1592:2000],f)[408]/408
mape(gbp.c[1592:2000],f1)[408]/408
mape(jpy.c[1592:2000],f2)[408]/408
```

Programmas kods no Matlab

```
targetSeries = eur;

feedbackDelays = 1:9;
hiddenLayerSize = 21;
net = narnet(feedbackDelays,hiddenLayerSize);

net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};

[inputs,inputStates,layerStates,targets] = preparets(net,{},{},
targetSeries);

net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'time';  % Divide up every value
net.divideParam.trainRatio = 70/100;
```

```
net.divideParam.valRatio = 10/100;
net.divideParam.testRatio = 20/100;

net.trainParam.epochs = 2000;
net.trainFcn = 'traingda';

net.performFcn = 'mse';  % Mean squared error

net.plotFcns = {'plotperform','plottrainstate','plotresponse', ...
  'ploterrcorr', 'plotinerrcorr'};

net.trainParam.lr = 0.3;
net.trainParam.epochs = 2000;
[net,tr] = train(net,inputs,targets,inputStates,layerStates);

outputs = net(inputs,inputStates,layerStates);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

trainTargets = gmultiply(targets,tr.trainMask);
valTargets = gmultiply(targets,tr.valMask);
testTargets = gmultiply(targets,tr.testMask);
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)

view(net)

netc = closeloop(net);
[xc,xic,aic,tc] = preparets(netc,{},{},targetSeries);
yc = netc(xc,xic,aic);
perfc = perform(net,tc,yc)
```

```
nets = removedelay(net);
[xs,xis,ais,ts] = preparets(nets,{},{},targetSeries);
ys = nets(xs,xis,ais);
closedLoopPerformance = perform(net,tc,yc)
```

DIPLOMDARBS darbs “Neironu tīkli un to lietojums laikrindu prognozēšanā” 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 Tiltānova

_____ _____

(paraksts) (datums)

Rekomendēju darbu aizstāvēšanai.
Vadītājs: doc. Dr.math. Jānis Valeinis

_____ _____

(paraksts) (datums)

Recenzents:

Darbs iesniegts Matemātikas nodaļā _____

(datums)

_____

(darbu pieņēma)

DIPLOMDARBS aizstāvēts valsts pārbaudījuma komisijas sēdē

_____ prot. Nr. _____, vērtējums_____

(datums)

Komisijas sekretārs/-e: asoc. prof. Dr.math. Inese Bula