VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

MATEMATINĖS STATISTIKOS KATEDRA

Vidas Paulauskas

FINANSINIŲ RODIKLIŲ ANALIZĖ, NAUDOJANT NEURONINIUS TINKLUS

Magistro baigiamasis darbas

Vilnius, 2012

Darbo vadovas:

Doc. Dr. Pranas Vaitkus (parašas)

Recenzentas:

Doc. Dr. Rūta Levulienė (parašas)

Registracijos Nr.: .....

Darbo gynimo data: 2012-06-04

T U R I N Y S

ĮVADAS

Darbo aktualumas. Šiuo metu tiek Europai, tiek visam pasauliui brendant iš ekonominės krizės, rinkos yra vis dar labai jautrius tiek įvairioms kalboms, tiek įvairioms naujienoms. Tai sąlygoja didelį rinkų nepastovumą ir greitą kintamumą. Prekiaujant valiutų rinkoje šios visos problemos itin jaučiamos, bet kokia netikėta naujiena gali valiutos kainą santykinai labai stipriai pakelti arba sumažinti. Todėl yra kuriami įvairūs nauji metodai, kurie pakankamai greitai sureaguotų į rinkos pokyčius.

Problema. Yra sukurtą daug įvairių prognozavimo, identifikavimo modelių, bet nėra aišku, kurie metodai yra geresni tam tikromis sąlygomis – šiuo atveju valiutų rinkoje. Taip pat nežinia, kaip galėtų veikti įvairūs jungtiniai metodai. Todėl šiame darbe nagrinėjama neuroniniu tinklu paremta strategija, Markovo strategija, branduolių metodas ir šių strategijų bei metodų junginiai.

Darbo tikslas. Išnagrinėti minėtus metodus, sukurti sudėtingesnį neuroninį tinklą ir šiuos metodus pritaikyti valiutų rinkos duomenims.

Darbo uždaviniai.

1. Išnagrinėti neuroniniu tinklu paremtą strategiją, Markovo strategiją ir branduolių metodą.
2. Sukurti sudėtingesnį neuroninio tinklo atvejį.
3. Palyginti minėtų metodų tinkamumą prekiaujant valiutų rinkoje ir išbandyti jų junginius.
4. Realizuoti minėtus metodus ir uždavinius SAS programoje.
5. Sukurti programą MQL4 programavimo kalba, kurios pagalba iš elektroninės prekybos platformos „MetaTrader 4“ galima eksportuoti duomenis.

Objektas. Nagrinėjama euro ir JAV dolerio valiutų poros kaina 2011 lapkričio 1 – 2012 gegužės 15 laikotarpiu.

Darbo metodai. Valiutų kainų analizėje taikomi minėti metodai. Šiam darbui pasitelkta „MS Office Excel“ programa, „MetaTrader 4“ elektroninės prekybos platforma ir SAS statistinis paketas.

Darbo rezultatai. Teorinėje dalyje aprašytos neuroniniu tinklu paremta strategija, sudėtingesnis neuroninio tinklo atvejis, taip pat Markovo strategija ir branduolių metodas, bei kelių hipotezių tikrinimas. Praktinėje dalyje aprašoma valiutų rinka, valiutų poros, prekybos platforma, su ja susijusi programavimo kalba, taip pat aprašomas SAS programos naudojimas. Taip pat pateikta teorinėje dalyje nagrinėjamų metodų panaudojimo valiutų rinkoje analizė. Darbo pabaigoje pateikiamos išvados, literatūros sąrašas, santrauka lietuvių ir anglų kalbomis bei prieduose pateikiami naudotų programų kodai.

1. TEORINĖ DALIS

Teorinėje dalyje aprašomi metodai, kurie taikomi praktinėje dalyje. Jie yra tokie: nuoseklaus optimizavimo strategija naudojant neuroninį tinklą, Markovo proporcingų lažybų strategija bei branduolių rekursinis mažiausių kvadratų metodas.

1.1. Nuoseklaus optimizavimo strategija naudojant neuroninį tinklą

Šiame poskyryje aprašoma nuoseklaus optimizavimo strategija naudojant neuroninį tinklą (dėl trumpumo ir patogumo toliau vadinama tiesiog neuroninio tinklo strategija). Šio poskyrio teorija daugiausiai remiasi autorių R. Arachi ir A. Takemura straipsniu [2] bei G. Shafer it V. Vovk idėjomis [13].

1.1.1. Riboto prognozavimo žaidimas

Tarkime, kad pradinis investuotojo kapitalas K_0 lygus 1 ir turimas kapitalas po n raundų yra K_n. Kiekvieną kartą investuotojas turi nuspręsti, kokią sumą M_n jis pasiryžęs investuoti. Žinoma, ši suma negali būti didesnė už turimą kapitalą $\left(|M_n| < K_{n-1}\right)$. Apsisprendus dėl investuojamos sumos, belieka sulaukti rinkos pokyčio $x_n \in [-1,1]$. Čia x_n atitinka vienetinio finansinio instrumento (akcijos, valiutos kurso) kainos pokytį n-jame raunde. Taigi žinant K_n, M_n ir x_n, randamas investuotojo kapitalas po n-jame raunde:

$$K_n = K_{n-1} + M_n x_n . \tag{1}$$

Visa tai galima pateikti tokiu ciklu:

CIKLAS:
$K_0 = 1$.
Kiekviename raunde (n = 1, 2, ...) turima :
Investuotojo sprendimas $M_n \in R$.
Sulaukiama rinkos pokyčio $x_n \in [-1,1]$.
$K_n = K_{n-1} + M_n x_n$.
Perėjus per visus n, baigiamas ciklas.

Pasak Shafer ir Vovk [13], investuotojas laimi prieš rinką, jei, pirma, jo kapitalas K_n niekada nebūna neigiamas ir, antra, viena iš šių išraiškų yra teisinga:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} x_i = 0 \tag{2}$$

arba

$$\lim_{n\to\infty} K_n = \infty . \tag{3}$$

Įvedamas naujas kintamasis α_n, kuris vadinamas investicijų santykiu raunde *n*. Jis gaunamas kaip santykis investuojamos sumos M_n raunde *n* ir turimo kapitalo po $n-1$ raundų K_{n-1}:

$$\alpha_n = M_n / K_{n-1}. \tag{4}$$

Kad būtų išvengta investuotojo bankroto, padaromi apribojimai investicijų santykiui: $-1 < \alpha_n < 1$. Šaltinyje [13] teigiama, jog prireikus, investuotojas gali pasiskolinti pinigų. Šis faktas nekeičia modelio esmės, nes tariant, kad ilguoju laikotarpiu galima pasiskolinti daugiausiai $\widetilde{K}$ pinigų, yra tas pats, kas laikyti, jog pradinis kapitalas yra lygus $1+\widetilde{K}$.

Apjungus (1) ir (4) formules, gaunama:

$$K_n = K_{n-1} + M_n x_n = K_{n-1} + \alpha_n K_{n-1} x_n = K_{n-1}(1+\alpha_n x_n) = \prod_{k=1}^{n}(1+\alpha_k x_k). \tag{5}$$

Logaritmuota K_n išraiška:

$$\log K_n = \sum_{k=1}^{n} \log(1+\alpha_k x_k). \tag{6}$$

Laikui bėgant investuotojo kapitalas (5) kinta priklauso nuo α_k. Ir apibrėžus jo funkcinę formą, bus gaunama investavimo strategija. Logišką, kad dabartinis apsisprendimas priklauso nuo praeities įvykių (rinkos pokyčių), todėl apibrėžti α_k naudojami praėję kainų skirtumai $x_{k-1}, x_{k-2}, \ldots$. Ir bandoma rasti tokius α_k, kad ateities kapitalas K_n būtų maksimalus (čia $n > k$). Reikia atkreipti dėmesį, kad α_k yra ne tik investicijų santykis, bet ir rinkos pokyčio (krypties) prognozė. Nes jei gaunama, kad $\alpha_k > 0$, reiškia manoma, kad ir $x_k > 0$, ir atvirkščiai. Taigi šiuo atveju α_k atlieka dvi funkcijas.

Apibrėžiamas $\mathbf{u_{n-1}} = (x_{n-1}, \ldots, x_{n-L})$ kaip vektorius, sudarytas iš *L* praeitų rinkos pokyčių, ir tariama, kad α_n priklauso nuo $\mathbf{u_{n-1}}$ ir svorių matricos **w** (ji bus apibrėžta kitame skyrelyje) šitaip: $\alpha_n = f(\mathbf{u}_{n-1}, \mathbf{w})$. Tada

$$\mathbf{w}^*_{\mathbf{n-1}} = \arg\max \sum_{t=1}^{n-1} \log(1 + f(\mathbf{u}_{n-1}, \mathbf{w}) x_t) \tag{7}$$

yra geriausia parametrų matrica iki praėjusio raundo $n-1$. Taigi nuoseklaus optimizavimo strategijoje yra naudojami $\mathbf{u_{n-1}}$ ir $\mathbf{w}^*_{\mathbf{n-1}}$, kad nustatyti investavimo sumą M_n raunde *n*:

$$M_n = K_{n-1} \times f(\mathbf{u_{n-1}}, \mathbf{w}^*_{\mathbf{n-1}}). \tag{8}$$

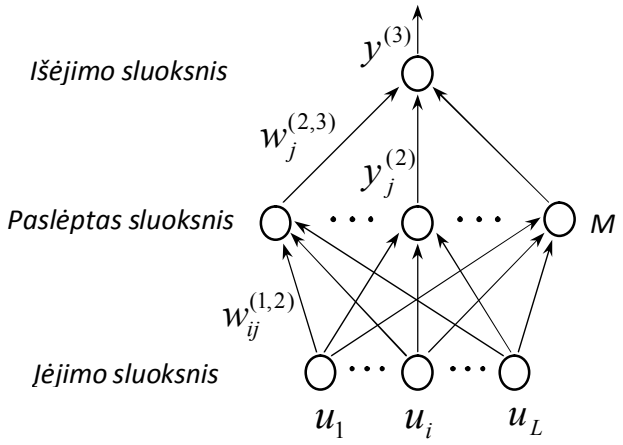
Funkcijai *f* apibrėžti naudojamas neuroninis tinklas, apie kurį rašoma kitame skyrelyje.

1.1.2. Neuroninio tinklo kūrimas

Dirbtiniai neuroniniai tinklai yra informacijos apdorojimo struktūros, kurios netiksliai imituoja kai kuriuos smegenyse vykstančius informacijos apdorojimo procesus. Šie tinklai sudaromi iš daugelio tarpusavyje sujungtų itin paprastų skaičiavimo elementų. Jungiant šiuos elementus vienus su kitais, yra gaunamas apytikris biologinių neuronų modelis [16].

Neuroniniai tinklai gali būti apmokomi be jokių apribojimų, taip gaunant modelio parametrus ir atrandant įvairiausius sąryšius, nulemtus vien tik duomenų struktūros. Šios savybės turi didelę praktinę reikšmę ir pritaikymo galimybes finansų srityje [17]. O Daugiasluoksnis perceptronas (neuroninis tinklas su šuolinėmis perdavimo funkcijomis [16]) gali būti efektyviai naudojamas prognozuojant valiutų kursus. Jis prognozavimo kokybe nenusileidžia ARCH modelių šeimai (GARCH, IGARCH, GARCH_M ir pan) [1].

Kaip minėta ankščiau, neuroninius tinklus gerai galima pritaikyti finansiniuose skaičiavimuose. Ne išimtis ir šiame darbe. Yra sukonstruojamas trijų sluoksnių tiesioginio sklidimo neuroninis tinklas, kuriame signalai sklinda iš įėjimų per visus paslėptus elementus ir pasiekiamas išėjimo neuronas (1 pav).



1 pav. Neuroninis tinklas.

Įėjimo sluoksnis turi L neuronų, kurie įveda kintamųjų u_i $(i = 1,...,L)$ vertes į tinklą. O paslėptas sluoksnis turi M neuronų, ir kiekvieno šio sluoksnio įėjimo elementai yra svertinė u_i elementų suma:

$$I_j^{(2)} = \sum_{i=1}^{L} w_{ij}^{(1,2)} u_i \,. \tag{9}$$

Pažymėjimas $w_{ij}^{(1,2)}$ vadinamas svoriu, kuris atspindį sinapsinį ryšį tarp *i*-tojo neurono įėjimo sluoksnyje ir *j*-tojo neurono paslėptame sluoksnyje. *J*-tojo neurono išėjimas aprašomas taip:

$$y_j^{(2)} = \tanh(I_j^{(2)}) \,. \tag{10}$$

Čia perdavimo funkcijai naudojamas hiperbolinis tangentas:

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}. \tag{11}$$

Akivaizdu, kad perdavimo funkcijos reikšmė gali kisti tarp -1 ir 1. Reikia paminėti, kad visi vidinio sluoksnio neuronai turi tą pačią perdavimo funkciją.

Tuo pačiu principu aprašoma išėjimo sluoksnio neurono įėjimo vertė $I^{(3)}$:

$$I^{(3)} = \sum_{j=1}^{M} w_j^{(2,3)} y_j^{(2)}, \tag{12}$$

čia $w_j^{(2,3)}$ yra svoris tarp *j*-tojo neurono paslėptame sluoksnyje ir neurono, esančio išėjimo sluoksnyje. Analogiškai (10) formulei, gaunama išėjimo iš neuroninio tinklo vertė:

$$y^{(3)} = \tanh(I^{(3)}). \tag{13}$$

Ši gauta išėjimo vertė $y^{(3)}$ ir naudojama neuroninio tinklo investavimo strategijoje:

$$\alpha_k = y^3 = f(\mathbf{u_{n-1}}, \mathbf{w}). \tag{14}$$

Dabar galima pateikti svorių matricų pavidalus:

$$\mathbf{w} = (\mathbf{w^{(1,2)}}, \mathbf{w^{(2,3)}}) = ((w_{ij}^{(1,2)})_{i=1,\ldots,L,\, j=1,\ldots,M}, (w_j^{(2,3)})_{j=1,\ldots,M}). \tag{15}$$

$$\mathbf{w^{(1,2)}} = \begin{pmatrix} w_{1,1}^{(1,2)} & \cdots & w_{1,M}^{(1,2)} \\ \vdots & \ddots & \vdots \\ w_{L,1}^{(1,2)} & \cdots & w_{L,M}^{(1,2)} \end{pmatrix}, \quad \mathbf{w^{(2,3)}} = \begin{pmatrix} w_1^{(2,3)} & \cdots & w_M^{(2,3)} \end{pmatrix}$$

Galiausiai galima užrašyti investuotojo kapitalo pavidalą:

$$K_n = K_{n-1}(1 + f(\mathbf{u_{n-1}}, \mathbf{w}) x_n). \tag{16}$$

Kyla dar viena užduotis: kokį imti įėjimo sluoksnio neuronų skaičių L ir kokį imti paslėpto sluoksnio neuronų skaičių M. Tai yra pakankamai sudėtinga užduotis. Praktinės dalies 2.6. poskyryje plačiau nagrinėjami šių pasirinkimų variantai. Verta paminėti, kad ne vien tik rinkos pokyčiai u_i gali dalyvauti verčių įvedime į neuroninį tinklą, taip pat galima naudoti bet kokią prieinamą informaciją iki raundo n: slenkantį vidurkį, kainų santykinės galios indeksą (angl. RSI), įvairius sezoninius indikatorius ar kitokius ekonominius duomenis. Apie konkretesnius papildomus rodiklius plačiau rašoma 2.4. poskyryje.

1.1.3. Nuoseklaus optimizavimo strategija naudojant neuroninį tinklą

Šiame skyrelyje aprašoma nuoseklaus optimizavimo strategija naudojant neuroninį tinklą.

Pirmiausia yra randama tokia svorių matrica $\mathbf{w}^* = \mathbf{w}^*_{\mathbf{n-1}}$, kuri maksimizuoja ϕ funkciją:

$$\phi = \sum_{k=1}^{n-1} \log(1 + f(\mathbf{u_{k-1}}, \mathbf{w}^*) x_k). \tag{17}$$

Tai yra geriausios svorių matricos reikšmės iki praėjusio raundo imtinai. Tada investicijų santykį *n*-jame raunde galima užrašyti tokiu pavidalu:

$$\alpha_n = f(\mathbf{u_{n-1}}, \mathbf{w^*_{n-1}}), \tag{18}$$

o investuotojo kapitalas aprašomas analogiškai formulei (16):

$$K_n = K_{n-1}(1 + f(\mathbf{u_{n-1}}, \mathbf{w^*_{n-1}})x_n). \tag{19}$$

Pastarosios funkcijos (19) maksimizavimui panaudojamas gradientinis nusileidimo metodas. Šis metodas yra klasikinis iteracinis neuroninių tinklų apmokymo būdas [18]. Pasitelkiant šį metodą bei apmokomąją konstantą β, svorių $w_j^{(2,3)}$ atnaujinimo algoritmas atrodo taip:

$$w_j^{(2,3)} = w_j^{(2,3)} + \Delta w_j^{(2,3)} = w_j^{(2,3)} + \beta \frac{\partial \phi}{\partial w_j^{(2,3)}}, \tag{20}$$

čia

$$\frac{\partial \phi}{\partial \omega_j^{(2,3)}} = \frac{\partial \phi}{\partial f}\frac{\partial f}{\partial \omega_j^{(2,3)}} = \sum_{k=1}^{n-1}\frac{\partial \phi}{\partial f}\frac{\partial f}{\partial {}^k I^{(3)}}\frac{\partial {}^k I^{(3)}}{\partial \omega_j^{(2,3)}} = \sum_{k=1}^{n-1}\frac{x_k}{1 + f(\mathbf{u_{k-1}}, \mathbf{w})x_k}(1 - \tanh^2({}^k I^{(3)}))^k y_j^{(2)} =$$

$$= \sum_{k=1}^{n-1} {}^k\delta_1 \, {}^k y_j^{(2)}, \tag{21}$$

o kairieji viršutiniai indeksai *k* prie $I^{(3)}$ ir $y_j^{(2)}$ nurodo raundo numerį. Taigi iš (20) ir (21) gaunama, kad

$$\Delta w_j^{(2,3)} = \beta \frac{\partial \phi}{\partial w_j^{(2,3)}} = \beta \sum_{k=1}^{n-1} {}^k\delta_1 \, {}^k y_j^{(2)}. \tag{22}$$

Pasinaudojant (13) ir (14) išraiškomis, gali paprasčiau užrašyti ${}^k\delta_1$:

$${}^k\delta_1 = \frac{x_k}{1 + f(\mathbf{u_{k-1}}, \mathbf{w})x_k}(1 - \tanh^2({}^k I^{(3)})) = \frac{x_k}{1 + {}^k y^{(3)} x_k}(1 - ({}^k y^{(3)})^2). \tag{23}$$

Analogiškai galima užrašyti ir svorių $w_{ij}^{(1,2)}$ atnaujinimo algoritmo išraišką:

$$w_{ij}^{(1,2)} = w_{ij}^{(1,2)} + \Delta w_{ij}^{(1,2)} = w_{ij}^{(1,2)} + \beta \frac{\partial \phi}{\partial w_{ij}^{(1,2)}}, \tag{24}$$

čia

$$\frac{\partial \phi}{\partial w_{ij}^{(1,2)}} = \sum_{k=1}^{n-1}\frac{\partial \phi}{\partial {}^k I_j^{(2)}}\frac{\partial {}^k I_j^{(2)}}{\partial w_{ij}^{(1,2)}} = \sum_{k=1}^{n-1}\frac{\partial \phi}{\partial f}\frac{\partial f}{\partial {}^k I^{(3)}}\frac{\partial {}^k I^{(3)}}{\partial {}^k y_j^{(2)}}\frac{\partial {}^k y_j^{(2)}}{\partial {}^k I_j^{(2)}}\frac{\partial {}^k I_j^{(2)}}{\partial w_{ij}^{(1,2)}} =$$

$$= \sum_{k=1}^{n-1}\frac{x_k}{1 + f(\mathbf{u_{k-1}}, \mathbf{w})x_k}(1 - \tanh^2({}^k I^{(3)}))w_j^{(2,3)}(1 - \tanh^2({}^k I_j^{(2)}))(u_{k-1})_i =$$

$$= \sum_{k=1}^{n-1} {}^k\delta_1 w_j^{(2,3)}(1 - \tanh^2({}^k I_j^{(2)}))(u_{k-1})_i = \sum_{k=1}^{n-1} {}^k\delta_2 (u_{k-1})_i \tag{25}$$

Iš (24) ir (25) gaunama, kad

$$\Delta w_{ij}^{(1,2)} = \beta \frac{\partial \phi}{\partial w_{ij}^{(1,2)}} = \beta \sum_{k=1}^{n-1} {}^{k}\delta_2 (u_{k-1})_i \,. \tag{26}$$

Pasinaudojant (10) išraiška, galima truputį kitaip užrašyti ${}^{k}\delta_2$:

$${}^{k}\delta_2 = {}^{k}\delta_1 w_j^{(2,3)} (1 - \tanh^2({}^{k}I_j^{(2)})) = {}^{k}\delta_1 w_j^{(2,3)} (1 - ({}^{k}y_j^{(2)})^2)\,, \tag{27}$$

o vietoj ${}^{k}\delta_1$ galima naudoti (23) formulę.

Sutrumpintas nagrinėjamas algoritmas *n*-tajame raunde aprašomas taip:

1. Duotam įėjimo vektoriui $\mathbf{u_{k-1}} = (x_{k-1},....,x_{k-L}), (k = 1,....,n-1)$ ir svorių matricai $\mathbf{w_{n-1}}$ iš pradžių suskaičiuojama ${}^{k}I_j^{(2)} = \sum_{i=1}^{L} w_{ij}^{(1,2)} (\mathbf{u_{k-1}})_i$ ir ${}^{k}y_j^{(2)} = \tanh({}^{k}I_j^{(2)})$. Taip pat apsisprendžiama dėl apmokomosios konstantos β reikšmės.

2. Suskaičiuojama ${}^{k}I^{(3)} = \sum_{j=1}^{M} w_j^{(2,3)} {}^{k}y_j^{(2)}$ ir ${}^{k}y^{(3)} = \tanh({}^{k}I^{(3)})$, o ${}^{k}I_j^{(2)}$ ir ${}^{k}y_j^{(2)}$ imami iš 1-ojo žingsnio. Tada yra atnaujinama svorių matricą $\mathbf{w}$ su svorių atnaujinimo funkcijomis $w_j^{(2,3)} = w_j^{(2,3)} + \beta \sum_{k=1}^{n-1} {}^{k}\delta_1 {}^{k}y_j^{(2)}$ ir $w_{ij}^{(1,2)} = w_{ij}^{(1,2)} + \beta \sum_{k=1}^{n-1} {}^{k}\delta_2 (u_{k-1})_i$.

3. Grįžtama į 1-ąjį žingsnį, kuriame svorių matricos reikšmės $\mathbf{w_{n-1}}$ pakeičiamos atnaujintomis reikšmėmis iš 2-ojo žingsnio.

Po pakankamo iteracijų skaičiaus, funkcija ϕ (17) konverguoja į lokalųjį maksimumo tašką atsižvelgiant į svorių matricas $\mathbf{w^{(1,2)}}$ ir $\mathbf{w^{(2,3)}}$. Tada priskiriamos svorių matricos $\mathbf{w^{(1,2)}} = \mathbf{w^{(1,2)*}}$, $\mathbf{w^{(2,3)}} = \mathbf{w^{(2,3)*}}$, kurios sudaro svorių matricą $\mathbf{w^{*}_{n-1}}$. Galiausiai pagal (19) formulę suskaičiuojamas investuotojo kapitalas *n*-jame raunde, $K_n = K_{n-1}(1 + f(\mathbf{u_{n-1}}, \mathbf{w^{*}_{n-1}}) x_n)$.

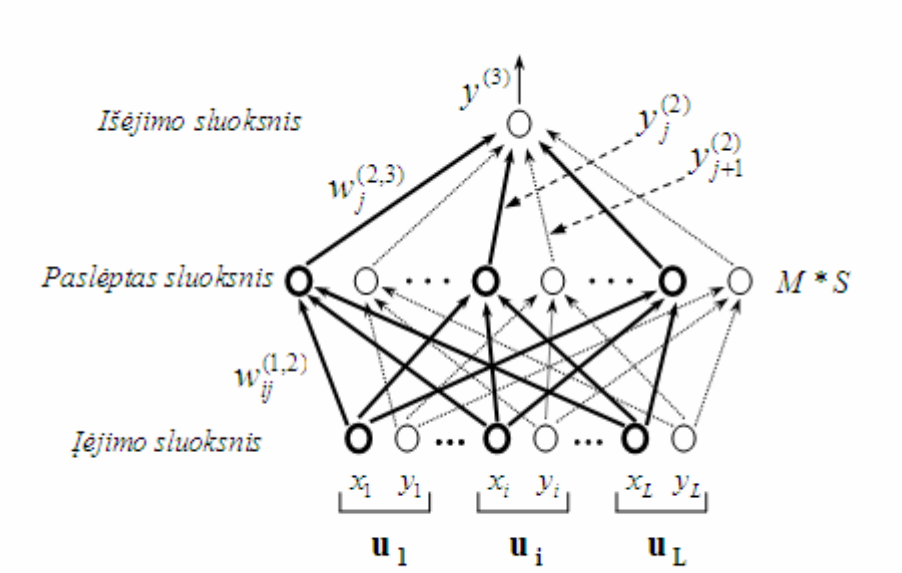
1.1.4. Nuoseklaus optimizavimo strategija naudojant neuroninį tinklą daugiamačiu atveju

Šiame skyrelyje aprašomas nuoseklaus optimizavimo strategija naudojant neuroninį tinklą daugiamačiu atveju, t.y. kai įvestyje į neuroninį tinklą dalyvauja ne vektorius, o matrica. Apie tokią galimybę užsimename nagrinėjamame straipsnyje [2], bet plačiau ji nėra aprašoma ir nagrinėjama, todėl tas padaroma šiame skyrelyje.

Tarkime turima tokią duomenų matrica:

$$\mathbf{u_{n-1}} = \begin{pmatrix} x_{n-1} & \dots & x_{n-L} \\ y_{n-1} & \dots & y_{n-L} \\ z_{n-1} & \dots & z_{n-L} \end{pmatrix},$$

čia pirmoji eilutė atitinka rinkos pokyčius, o likusiose eilutėse yra kitokia su rinka susijusi informacija atitinkamuose raunduose. Žinoma, tų eilučių gali būti ir daugiau, ir mažiau, priklausomai nuo turimos informacijos kiekio. Kadangi pasikeitė duomenų kiekis, todėl ir pats neuroninis tinklas atrodys šiek tiek kitaip (2 pav.). Čia pateiktas neuroninis tinklas, kai įvedime dalyvauja rinkos pokyčiai ir vienas kintamasis su papildoma informacija. Pačių neuronų skaičius paslėptame sluoksnyje šiuo atveju padvigubėja, jų bus atitinkamai $M * S$, $S = 2$. Atrodytų, kad ir įėjimo sluoksnyje turėtų neuronų padvigubėti (kaip ir parodyta (2 pav.)), bet laikome, kad ten yra vienas neuronas $\mathbf{u_i}$, kuris turi savyje informacijos apie (x_i, y_i). Naujasis kintamasis S nurodo visos informacijos kiekį (modelyje, kur įėjime dalyvauja tik rinkos pokyčiai, $S = 1$).



2 pav. Neuroninis tinklas su vektoriniais įėjimais.

Kad svoriai nesigautų labai dideli, padaroma jų reguliarizaciją. Tada tikslo funkcija (17) atrodo taip:

$$\phi = \sum_{k=1}^{n-1} \log(1 + f(\mathbf{u_{k-1}}, \mathbf{w}^*) x_k) + \frac{1}{2}\lambda \left(\sum_{\substack{i=1,\dots,L \\ s=1,\dots,S}} \left(w_{ij,s}^{(1,2)}\right)^2 + \sum_{j=1}^{M*S} \left(w_j^{(2,3)}\right)^2 \right) \tag{28}$$

Dabar (9) ir (12) formulės pasikeis taip:

$$I_j^{(2)} = \sum_{\substack{i=1,\dots,L \\ s=1,\dots,S}} w_{ij,s}^{(1,2)} u_{i,s}\,, \tag{29}$$

$$I^{(3)} = \sum_{j=1}^{M*S} w_j^{(2,3)} y_j^{(2)}\,. \tag{30}$$

čia $i = 1,\dots,L$, $j = 1,\dots,M * S$, $s = 1,\dots,S$.

Tokiu daugiamačių atveju svorių matricos bus didesnės negu kad aprašytos 1.1.2. skyrelyje:

$$\mathbf{w} = (\mathbf{w}^{(1,2)}, \mathbf{w}^{(2,3)}) = ((w_{i,j,s}^{(1,2)})_{i=1,...,L,\, j=1,...,M*S,\, s=1,...,S}, (w_j^{(2,3)})_{j=1,...,M*S}) \tag{31}$$

$$\mathbf{w}^{(1,2)} = \begin{pmatrix} w_{1,1,1}^{(1,2)} & \cdots & w_{1,S,S}^{(1,2)} & \cdots & w_{1,(M-1)*S+1,1}^{(1,2)} & \cdots & w_{1,M*S,S}^{(1,2)} \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ w_{L,1,1}^{(1,2)} & \cdots & w_{L,S,S}^{(1,2)} & \cdots & w_{L,(M-1)*S+1,1}^{(1,2)} & \cdots & w_{L,M*S,S}^{(1,2)} \end{pmatrix}, \mathbf{w}^{(2,3)} = \begin{pmatrix} w_1^{(2,3)} & \cdots & w_{M*S}^{(2,3)} \end{pmatrix}$$

Svorių atnaujinimo formulės ne daug skiriasi nuo 1.1.3. skyrelyje aprašytų, tik kai kur prisideda apatinis indeksas *s* bei reguliarizacijos konstanta su pačiu svoriu:

$$w_j^{(2,3)} = w_j^{(2,3)} + \Delta w_j^{(2,3)} = w_j^{(2,3)} + \beta\left(\sum_{k=1}^{n-1} {}^k\delta_1 \, {}^k y_j^{(2)} + \lambda w_j^{(2,3)}\right), \tag{32}$$

$$w_{ij,s}^{(1,2)} = w_{ij,s}^{(1,2)} + \Delta w_{ij,s}^{(1,2)} = w_{ij,s}^{(1,2)} + \beta\left(\sum_{k=1}^{n-1} {}^k\delta_2 (u_{k-1})_{i,s} + \lambda w_{ij,s}^{(1,2)}\right). \tag{33}$$

Svarbu atkreipti dėmesį, kad šiuose formulėse indeksai kinta taip: $i = 1,...,L$, $j = 1,...,M*S$, $s = 1,...,S$.

Galiausiai investuotojo kapitalas raundo *n* pabaigoje užrašomas taip pat kaip ir (19) formulė.

1.2. Markovo proporcingų lošimų strategija

Naudojantis šaltiniu [2], šiame poskyryje aprašoma paprastesnė lošimų strategija negu aprašyta 1.1 poskyryje. Ši strategija yra apibendrintas Markovo strategijos monetos mėtymo riboto prognozavimo žaidime atvejis [14].

Pradedama nuo to, kad yra maksimizuojamas logaritmuotas investuotojo kapitalas kaip (6) formulėje atsižvelgiant į α_k:

$$\log K_n = \sum_{k=1}^{n} \log(1 + \alpha_k x_k) \longrightarrow \max_{\alpha} \tag{34}$$

Laikomasi paprastos strategijos, kurioje $\alpha_n = \alpha_{n-1}^*$, o α_{n-1}^* gaunama iš tokios išraiškos:

$$\alpha_{n-1}^* = \arg\max \prod_{k=1}^{n-1} (1 + \alpha x_k). \tag{35}$$

Ši strategija pažymima kaip MKV0.

Toliau nagrinėjamas apibendrintas MKV0 strategijos atvejis. Čia naudojami skirtingi investavimo santykiai, kurie priklauso nuo rinkos pokyčio praeitame raunde. Pažymima $\alpha_k = \alpha_k^+$, jei x_{k-1} buvo teigiamas (rinka kilo) ir $\alpha_k = \alpha_k^-$, jei x_{k-1} buvo neigiamas (rinka krito). Tokia strategija vadinama MKV1. Tada *n*-jame raunde naudojami $\alpha_k = \alpha_k^{+*}$ ir $\alpha_k = \alpha_k^{-*}$ šitaip:

$$(\alpha_{n-1}^{+*}, \alpha_{n-1}^{-*}) = \arg\max \prod_{k=1}^{n-1} (1 + f(\mathbf{u_{k-1}}, \alpha^+, \alpha^-) x_k) \tag{36}$$

$$f(\mathbf{u_{k-1}}, \alpha^+, \alpha^-) = \alpha^+ I\{x_{k-1} > 0\} + \alpha^- I\{x_{k-1} < 0\}, \tag{37}$$

čia $\mathbf{u_{k-1}} = (x_{k-1})$, o $I\{\bullet\}$ yra indikatoriaus funkcija tam tikro įvykio $\{\bullet\}$. Tada investuotojo kapitalas aprašomas tokia forma:

$$\log K_n = \sum_{k=1}^{n} \log(1 + f(\mathbf{u_{k-1}}, \alpha^+, \alpha^-) x_k) = \sum_{k=1}^{n} I\{x_{k-1} > 0\} \log(1 + \alpha^+ x_k) +$$

$$+ \sum_{k=1}^{n} I\{x_{k-1} < 0\} \log(1 + \alpha^- x_k) \longrightarrow \max_{\alpha^+, \alpha^-}. \qquad (38)$$

Po to kiekvienas dėmuo iš (38) formulės maksimizuojamas atskirai, atitinkamai pagal α^+ ir α^-.

Galima naudoti ir kitokį apibendrinimą, atsižvelgiant į rinkos pokyčius per paskutinius du raundus. Tada $\mathbf{u_{k-1}} = (x_{k-1}, x_{k-1})$ ir

$$f(u_{k-1}, \alpha^{++}, \alpha^{+-}, \alpha^{-+}, \alpha^{--}) = \alpha^{++} I\{x_{k-1} > 0, x_{k-2} > 0\} + \alpha^{+-} I\{x_{k-1} > 0, x_{k-2} < 0\} +$$

$$+ \alpha^{-+} I\{x_{k-1} < 0, x_{k-2} > 0\} + \alpha^{--} I\{x_{k-1} < 0, x_{k-2} < 0\} \qquad (39)$$

O investuotojo kapitalas atrodys taip:

$$\log K_n = \sum_{k=1}^{n} \log(1 + f(u_{k-1}, \alpha^{++}, \alpha^{+-}, \alpha^{-+}, \alpha^{--}) x_k) = \sum_{k=1}^{n} I\{x_{k-1} > 0, x_{k-2} > 0\} \log(1 + \alpha^{++} x_k) +$$

$$+ \sum_{k=1}^{n} I\{x_{k-1} > 0, x_{k-2} < 0\} \log(1 + \alpha^{+-} x_k) + \sum_{k=1}^{n} I\{x_{k-1} < 0, x_{k-2} > 0\} \log(1 + \alpha^{-+} x_k) +$$

$$+ \sum_{k=1}^{n} I\{x_{k-1} < 0, x_{k-2} < 0\} \log(1 + \alpha^{--} x_k). \qquad (40)$$

Šiuo atveju taip pat paskutiniosios formulės (40) dėmenys maksimizuojami atskirai, atitinkamai pagal α^{++}, α^{+-}, α^{-+} ir α^{--}. Tokia strategija vadinama MKV2.

Markovo strategijų MKV0, MKV1 ir MKV2 atitinkamiems parametrams skaičiuoti (tuo pačiu maksimizuoti logaritmuoto kapitalo funkciją) naudojama netiesinio programavimo optimizavimo pasitikėjimo regione paprogramė, kitaip NLPTR (angl. Nonlinear Optimization by Trust Region). NLPTR metodas naudoja gradientą $g^{(k)} = \nabla f(x^{(k)})$ ir Hessian matricą $G^{(k)} = \nabla^2 f(x^{(k)})$, bei yra reikalaujama, kad tikslo funkcija $f = f(x)$ turėtų tolydžias pirmos ir antros eilės išvestines [12]. Šiuo atveju tikslo funkcija yra $\log K_n$.

Kaip ir 1.1. poskyryje, taip ir čia investicijų santykiai $\alpha, \alpha^{\bullet}, \alpha^{\bullet\bullet}$ naudojami ir kaip rinkos pokyčio x prognozės.

1.3. Slenkančio lango branduolių rekursinis mažiausių kvadratų metodas

Šiame skyrelyje aprašomas branduolių rekursinis mažiausių kvadratų (RMK) metodas, kuris taikomas netiesinės sistemos identifikavimui bei prognozavimui, ir tai daroma „online“ metodu, pasitelkiant slenkantį langą. Poskyrio medžiaga daugiausiai remiasi [15] šaltiniu.

1.3.1. Mažiausių kvadratų ir branduolių metodas

Klasikinis mažiausių kvadratų metodas duotam vektoriui $\mathbf{y} \in R^{N\times 1}$ ir duomenų matricai $\mathbf{X} \in R^{N\times M}$ ieško geriausio sprendimų vektoriaus $\mathbf{h} \in R^{M\times 1}$, kuris minimizuoją tokią funkciją:

$$J = \min_{\mathbf{h}} \left\| \mathbf{y} - \mathbf{X}\mathbf{h} \right\|^2 . \tag{41}$$

Tokio uždavinio sprendinys užrašomas taip:

$$\hat{\mathbf{h}} = [\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{y} . \tag{42}$$

Sprendžiant tokius uždavinius kartais iškyla problemų dėl to, kad negalima rasti $\mathbf{X}^T\mathbf{X}$ atvirkštinės. Šiam tikslui yra panaudojama reguliarizacija ir (41) formulę užrašome taip:

$$J = \min_{\mathbf{h}} [\left\| \mathbf{y} - \mathbf{X}\mathbf{h} \right\|^2 + c\mathbf{h}^T\mathbf{h}] , \tag{43}$$

čia c yra reguliarizacijos konstanta. Tada sprendinys gaunamas toks:

$$\hat{\mathbf{h}} = [\mathbf{X}^T\mathbf{X} + c\mathbf{I}]^{-1}\mathbf{X}^T\mathbf{y} , \tag{44}$$

kur $\mathbf{I}$ yra vienetinė matrica.

Sprendžiant kai kuriuos uždavinius, iš pradžių ne visi duomenys yra žinomi. Todėl sprendiniai turi būti vis perskaičiuojami su lyg kiekviena nauja informacija. Tokiu atveju yra naudojamas rekursinis mažiausių kvadratų metodas. Dažniausiai yra sutinkamas rekursinis mažiausių kvadratų metodas su eksponentiniais svoriais, kur didesnius svorius gauna naujesni duomenis, o mažesni svoriai lieka senesniems duomenims. Tada duotam skaliarui $\lambda < 1$ (užmiršimo faktorius), algoritmas rekursiškai suskaičiuoja sprendinį šiai formulei:

$$J = \min_{\mathbf{h}} \left[\sum_{j=0}^{N} \lambda^{N-j} \left\| y(j) - \mathbf{x}_j^T\mathbf{h} \right\|^2 + c\mathbf{h}^T\mathbf{h} \right] , \tag{45}$$

čia $\mathbf{x}_j^T$- matricos $\mathbf{X}$ j-toji eilutė. Šį algoritmą galima taikyti vis augančiam stebėjimų skaičiui N.

Branduolių metodas yra paremtas duomenų $\mathbf{x}_i$ iš įvesties erdvės (angl. input space) netiesine transformacija į taip vadinamą aukštos dimensijos požymių erdvę (angl. feature space) $\tilde{\mathbf{x}}_i = \Phi(\mathbf{x}_i)$. Pagrindinė branduolių metodo savybė yra ta, kad skaliaras požymių erdvėje gaunamas kaip netiesinė (branduolio) funkcija iš duomenų, esančių įvesties erdvėje. Šiuo atveju netiesinei branduolio funkcijai yra naudojamas taip vadinamas Gauso branduolys:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\left\| \mathbf{x}_i - \mathbf{x}_j \right\|^2}{2\delta^2} \right) = \exp\left(-\frac{(x_{i,1} - x_{j,1})^2 + \ldots + (x_{i,N} - x_{j,N})^2}{2\delta^2} \right). \tag{46}$$

Tada mažiausių kvadratų netiesinis branduolių algoritmas yra gaunamas transformuojant duomenis į požymių erdvę. Transformuotam sprendimo vektoriui $\widetilde{\mathbf{h}} \in R^{M' \times 1}$ ir transformuotai duomenų matricai $\widetilde{\mathbf{X}} \in R^{N \times M'}$, mažiausių kvadratų lygtį (41) galima užrašoma taip:

$$J' = \min_{\widetilde{\mathbf{h}}} \left\| y - \widetilde{\mathbf{X}}\widetilde{\mathbf{h}} \right\|^2 . \tag{47}$$

Minėtas transformuotas sprendinio vektorius gali būti išreikštas kaip tam tikra transformuotos duomenų matricos $\widetilde{\mathbf{X}}$ eilučių išraiška:

$$\widetilde{\mathbf{h}} = \widetilde{\mathbf{X}}^T \boldsymbol{\alpha} \tag{48}$$

Dar daugiau, formulėje (41) galima naudoti branduolių matricą $\mathbf{K} = \widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T$. Tada

$$J' = \min_{\widetilde{\mathbf{h}}} \left\| y - \mathbf{K}\boldsymbol{\alpha} \right\|^2 , \tag{49}$$

čia $\alpha \in R^{N \times 1}$ yra naujasis sprendinio vektorius. Taigi norint naudoti mažiausių kvadratų branduolio metodą, tereikia apskaičiuoti branduolių matricą, pasinaudojant (46) formule:

$$\mathbf{K}(i, j) = k(\mathbf{x}_i, \mathbf{x}_j) , \tag{50}$$

o čia $\mathbf{x}_i^T$ ir $\mathbf{x}_j^T$ yra *i*-toji ir *j*-toji duomenų matricos $\mathbf{X}$ eilutės.

Daugeliui naudingų branduolių funkcijų požymių erdvės dimensija M' yra gaunama daug didesnė nei turimų stebėjimų skaičius N. Tam, kad išvengtume taip vadinamo persimokymo, yra naudojama branduolių keteros regresija (angl. ridge regression). Tokio tipo regresijoje, transformuotas sprendinys $\widetilde{\mathbf{h}}$ yra gaunamas išsprendus šią problemą:

$$J'' = \min_{\widetilde{\mathbf{h}}} [\left\| \mathbf{y} - \widetilde{\mathbf{X}}\widetilde{\mathbf{h}} \right\|^2 + c\widetilde{\mathbf{h}}^T\widetilde{\mathbf{h}}] . \tag{51}$$

Šia lygtį, pasinaudojant (48) formule ir branduolių matricos apibrėžimu, galima perrašyti taip:

$$J'' = \min_{\alpha} [\left\| \mathbf{y} - \mathbf{K}\alpha \right\|^2 + c\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}] , \tag{52}$$

kurios sprendinys gaunamas šitaip:

$$\boldsymbol{\alpha} = \mathbf{K}_{reg}^{-1}\mathbf{y} , \tag{53}$$

$$\mathbf{K}_{reg} = (\mathbf{K} + c\mathbf{I}) , \tag{54}$$

čia c yra jau anksčiau minėta reguliarizacijos konstanta. Reikia pastebėti, kad branduolių matrica $\mathbf{K}$ yra teigiamai apibrėžta (pagal branduolio apibrėžimą, formulė (46)), o branduolių matrica $\mathbf{K}_{reg}$ yra ne tik teigiamai apibrėžta, bet ir ne vienetinė [19].

1.3.2. Slenkančio lango atvejis

Daugelyje situacijų yra pravartu naudoti „online“ metodu veikiantį algoritmą. Pavyzdžiui, jei $\mathbf{y}$ yra laike kintantis procesas, tada galima sukonstruoti „online“ metodu veikiantį algoritmą, norint tinkamai sekti vis atsirandančius pokyčius.

Tariama, kad turimi tokie duomenys $\{(y_1, \mathbf{x_1}), (y_2, \mathbf{x_2}), ...\}$, tada slenkantis langas apima tik N paskutinių duomenų. Taigi *n*-tasis langas susideda iš tokio duomenų vektoriaus $\mathbf{y}_n = [y_n, y_{n-1}, ..., y_{n-N+1}]^T$ ir duomenų matricos $\mathbf{X}_n = [\mathbf{x}_n, \mathbf{x}_{n-1}, ..., \mathbf{x}_{n-N+1}]^T$. O reguliazituota branduolių matrica apskaičiuojama taip:

$$\mathbf{K}_n = \widetilde{\mathbf{X}}_n \widetilde{\mathbf{X}}_n^T + c\mathbf{I}. \tag{55}$$

Būtina atkreipti dėmesį į tai, kad reikia riboti duomenų eilučių skaičių N (lango dydį), kuriam ir yra skaičiuojama branduolių matrica. Kitaip nei standartiniam rekursiniam mažiausių kvadratų metodui, kurio koreliacijos matrica yra fiksuoto dydžio $M \times M$, čia branduolių matricos dydis priklauso nuo stebėjimo skaičiaus ir yra $N \times N$ dydžio.

1.3.3. Atvirkštinės branduolių matricos atnaujinimas

Atnaujinant sprendinį α, reikia suskaičiuoti $N \times N$ dydžio atvirkštinę matricą $\mathbf{K}_n^{-1}$ kiekvienam slenkančiam langui. Todėl, kad būtų mažiau skaičiavimų, galimą minėtą atvirkštinę matricą suskaičiuoti naudojantis vien tik dabartinio lango duomenimis $\{\mathbf{x}_n, y_n\}$ ir žinant ankstesnio lango atvirkštinę matricą $\mathbf{K}_{n-1}^{-1}$. O atnaujintas sprendinys gali būti randamas pasinaudojant (53) formule.

Iš duotos reguliarizuotos branduolio matricos $\mathbf{K}_{n-1}$ sekančio lango branduolio matricą $\mathbf{K}_n$ galima rasti dviem žingsniais. Pirma, pašalinami matricos $\mathbf{K}_{n-1}$ pirmoji eilutė ir pirmasis stulpelis. Šis žingsnis vadinamas „sumažinimu“ ir tokia matrica žymima $\hat{\mathbf{K}}_{n-1}$. Antra, branduoliai, gauti iš naujų duomenų, pridedami prie šios $\hat{\mathbf{K}}_{n-1}$ matricos kaip paskutinė eilutė ir paskutinis stulpelis:

$$\mathbf{K}_n = \begin{bmatrix} \hat{\mathbf{K}}_{n-1} & \mathbf{k}_{n-1}(\mathbf{x}_n) \\ \mathbf{k}_{n-1}(\mathbf{x}_n)^T & k_{nn} + c \end{bmatrix}. \tag{56}$$

čia $\mathbf{k}_{n-1}(\mathbf{x}_n) = [k(\mathbf{x}_{n-N+1}, \mathbf{x}_n), \ldots, k(\mathbf{x}_{n-1}, \mathbf{x}_n)]^T$ ir $k_{nn} = k(\mathbf{x}_n, \mathbf{x}_n)$. Šis antrasis žingsnis vadinamas „padidinimu“.

Atvirkštinės matricos $\mathbf{K}_n^{-1}$ skaičiavimas taip pat susideda iš dviejų žingsnių. Pirma, duotai praėjusio lango matricai $\mathbf{K}_{n-1}$ ir jos atvirkštinei $\mathbf{K}_{n-1}^{-1}$, suskaičiuojama sumažintos matricos $\hat{\mathbf{K}}_{n-1}$ atvirkštinė matrica $\hat{\mathbf{K}}_{n-1}^{-1}$ pagal tokių formulių grupę:

$$\left.\begin{array}{l} \mathbf{K} = \begin{bmatrix} a & \mathbf{b}^T \\ \mathbf{b} & \mathbf{D} \end{bmatrix} \qquad \mathbf{K}^{-1} = \begin{bmatrix} e & \mathbf{f}^T \\ \mathbf{f} & \mathbf{G} \end{bmatrix} \\ \Rightarrow \begin{cases} \mathbf{b}e + \mathbf{D}\mathbf{f} = 0 \\ \mathbf{b}\mathbf{f}^T + \mathbf{D}\mathbf{G} = \mathbf{I} \end{cases} \\ \Rightarrow \mathbf{D}^{-1} = \mathbf{G} - \mathbf{f}\mathbf{f}^T / e \end{array}\right\} \tag{57}$$

Gauta atvirkštinė matrica $\mathbf{D}^{-1}$ ir yra ieškoma atvirkštinė matrica $\hat{\mathbf{K}}_{n-1}^{-1}$. Antrajame žingsnyje padidinama matrica $\hat{\mathbf{K}}_{n-1}$ ir gaunama dabartinio lango branduolio matrica $\mathbf{K}_n$. Tada žinant $\hat{\mathbf{K}}_{n-1}^{-1}$ ir $\mathbf{K}_n$, atvirkštinė matrica $\mathbf{K}_n^{-1}$ gaunama pagal tokių formulių grupę.

$$\left.\begin{array}{l} \mathbf{K} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{b}^T & d \end{bmatrix} \qquad \mathbf{K}^{-1} = \begin{bmatrix} \mathbf{E} & \mathbf{f} \\ \mathbf{f}^T & g \end{bmatrix} \\ \Rightarrow \begin{cases} \mathbf{A}\mathbf{E} + \mathbf{b}\mathbf{f}^T = \mathbf{I} \\ \mathbf{A}\mathbf{f} + \mathbf{b}g = \mathbf{0} \\ \mathbf{b}^T\mathbf{f} + dg = 1 \end{cases} \\ \Rightarrow \mathbf{K}^{-1} = \begin{bmatrix} \mathbf{A}^{-1}(\mathbf{I} + \mathbf{b}\mathbf{b}^T\mathbf{A}^{-1H}g) & -\mathbf{A}^{-1}\mathbf{b}g \\ -(\mathbf{A}^{-1}\mathbf{b})^T g & g \end{bmatrix} \\ g = (d - \mathbf{b}^T\mathbf{A}^{-1}\mathbf{b})^{-1} \end{array}\right\} \tag{58}$$

Pažymėtina, kad (58) formulėse naudojama $\mathbf{A}^{-1}$ yra ankstesniame žingsnyje gauta $\hat{\mathbf{K}}_{n-1}^{-1}$.

Algoritmas pradedamas nuo to, kad pradinis langas yra užpildytas duomenis, kurie visi yra lygus nuliui. Taigi ir pradiniai duomenys $\mathbf{x}_0$ taip pat sudaryti vien tik iš nulių. Tada yra suskaičiuojama reguliarizuotą branduolių matrica $\mathbf{K}_0$ ir jos išvestinė $\mathbf{K}_0^{-1}$ (skaičiuojama paprastai). Šiuo atveju $\mathbf{K}_0 = \mathbf{1} + c\mathbf{I}$, čia $\mathbf{1}$ yra $N \times N$ dydžio matrica, sudaryta vien iš 1.

Visą aprašytą sprendinio atnaujinimo algoritmą galima užrašyti tokiu ciklu:

CIKLAS:

Turimos pradinės matricos $\mathbf{K}_0$ ir $\mathbf{K}_0^{-1}$.

Kiekvienam lange (n = 1, 2, ...) skaičiuojama:

Sumažinama matrica $\mathbf{K}_{n-1}$ ir gaunama $\hat{\mathbf{K}}_{n-1}$.

Pagal (57) formules suskaičiuojama $\hat{\mathbf{K}}_{n-1}^{-1}$.

Padidinama matrica $\hat{\mathbf{K}}_{n-1}$ ir gaunama $\mathbf{K}_n$ pagal (56) formules.

Pagal (57) formules suskaičiuojama $\mathbf{K}_n^{-1}$.

Randamas atnaujintas sprendinys $\boldsymbol{\alpha} = \mathbf{K}_n^{-1}\mathbf{y}_n$.

Perėjus per visus *n*, baigiamas ciklas.

Aprašytas modelis gali būti naudojamas tiek identifikavimui, tiek prognozavimui. Šiame darbe apsiribojama vien tik prognozavimu. Tarkime *n*-tajame lange yra suskaičiuojamas sprendinys α. Tada gaunami nauji duomenys $\mathbf{x}_{n+1}$ ($\mathbf{y}_{n-1}$ kol kas nežinomas) ir suskaičiuojama $\mathbf{k}_n(\mathbf{x}_{n+1}) = [k(\mathbf{x}_{n-N+2}, \mathbf{x}_{n+1}), \ldots, k(\mathbf{x}_n, \mathbf{x}_{n+1})]^T$. Prognozė tada yra tokia:

$$\hat{y}_{n+1} = \boldsymbol{\alpha}^T \mathbf{k}_n(\mathbf{x}_{n+1}) = \alpha_{n,1} \times k(\mathbf{x}_{n-N+1}, \mathbf{x}_{n+1}) + \ldots + \alpha_{n,N} \times k(\mathbf{x}_n, \mathbf{x}_{n+1}) = \sum_{i=1}^{N} \alpha_{n,i} \times k(\mathbf{x}_{n-N+i}, \mathbf{x}_{n+1}) . \quad (59)$$

Šio metodo prognozavimo niuansai ir tikslumas aprašomas praktinėje dalyje 2.8. poskyryje.

1.4 Įvairios statistikos

Šiame poskyryje pateikiami serijų testo ir hipotezės apie proporciją aprašymai.

1.4.1 Serijų testas

Tariama, kad turima ilgio $N = n_0 + n_1$ seka, sudarytą iš dviejų rūšių simbolių A ir $\overline{A}$. Tada tokią seką galima užrašyti $C_N^{n_1}$ skirtingais būdais. Jeigu eksperimentas toks, kad visų $C_N^{n_1}$ variantų pasirodymas yra vienodai galimas, tada galima sakyti, kad simboliai A ir $\overline{A}$ yra išsidėstę atsitiktinai. Taigi hipotezę, apie šių simbolių atsitiktinį išsidėstymą galima tikrinti remiantis serijų kriterijumi [8].

Yra skiriami du atvejai: mažos imties atvejis, kai $n_0 < 20$ ir $n_1 < 20$, ir didelės imties atvejis, kai $n_0 \geq 20$ ir $n_1 \geq 20$ [4]. Šiame darbe apsiribojama didelės imties atveju.

Nagrinėjamos tokios hipotezės:

H_0 : visi simbolių A ir $\overline{A}$ išsidėstymai yra vienodai galimi;

H_1 : simbolių A ir $\overline{A}$ išsidėstymai nėra vienodai galimi;

Apibrėžiami tokie kintamieji: T_0 - skaičius serijų, sudarytų iš simbolio $\overline{A}$ ir T_1 - skaičius serijų, sudarytų iš simbolio A. Tada sekoje iš viso yra $T = T_0 + T_1$ serijų. Didelėms n_0 ir n_1 reikšmėms statistiką T galima aproksimuoti normaliuoju skirstiniu [11] su tokiais parametrais:

$$E(T) = \mu = \frac{2n_0 n_1}{N} + 1, \quad (60)$$

$$D(T) = \sigma^2 = \frac{2n_0 n_1 (2n_0 n_1 - N)}{N^2 (N-1)} = \left(\frac{2n_0 n_1}{N} + 1 - 1 \right) \left(\frac{2n_0 n_1}{N} + 1 - 2 \right) \frac{1}{N-1} = \frac{(\mu - 1)(\mu - 2)}{N-1}. \quad (61)$$

O statistika Z aprašome taip:

$$Z = \frac{T - \mu}{\sigma}. \quad (62)$$

Tada, jei

$$P(Z > Z_\alpha) < \alpha, \quad (63)$$

yra atmetama nulinė hipotezė, kad visi simbolių A ir $\overline{A}$ išsidėstymai yra vienodai galimi. Čia Z_α yra standartinio normaliojo skirstinio α -toji kritinė reikšmė.

Taip pat verta paminėti, kad jei Z reikšmė yra labai mažas skaičius, tada sekoje stebima klasterizaciją, o jei Z reikšmė yra labai didelis skaičius – sekoje stebimas stiprus susimaišymas.

1.4.2. Hipotezė apie proporciją. Normalioji aproksimacija

Šiame skyrelyje aprašoma statistinės hipotezės apie proporciją tikrinimas (praktinėje dalyje vadinama proporcijos hipoteze). Skyrelis remiasi šaltiniu [3].

Tariama, kad atsitiktinis dydis X gali įgyti dvi reikšmes: 0 arba 1 su tokiomis tikimybėmis: $P(X = 1) = p$ ir $P(X = 0) = 1 - p$. Reiškia, yra stebimas binominis atsitiktinis dydis $X \sim B(1, p)$ su nežinomu parametru p. Tada tokių nepriklausomų atsitiktinių dydžiu suma taip pat turi binominį skirstinį, kurio parametrai yra n ir p:

$$S_n = X_1 + \ldots + X_n \sim B(n, p). \quad (64)$$

Statistiką S_n galima taikyti hipotezėms tikrinti, bet esant didelei imčiai yra sunku apskaičiuoti binominio atsitiktinio dydžio reikšmių tikimybes, todėl naudojama statistikos S_n aproksimacija. Jei p reikšmė nėra mažas skaičius, tada taikoma normalioji aproksimacija:

$$Z = \frac{S_n - \mathbf{E}S_n}{\sqrt{\mathbf{D}S_n}} = \frac{S_n - np}{\sqrt{np(1-p)}} = \frac{\overline{X} - p}{\sqrt{p(1-p)/n}} = \frac{\hat{p} - p}{\sqrt{p(1-p)/n}} \approx N(0,1). \quad (65)$$

Kadangi atsitiktinis dydis X įgyja tik dvi reikšmes (0 arba 1), todėl jo vidurkis $\overline{X}$ bus tarp skaičių 0 ir 1, o tai ir bus ieškomas $\hat{p}$ įvertinys.

Suformuluojama hipotezė ir vienpusė alternatyva:

$H_0 : p = a$;

$H_1 : p > a$;

Tada apskaičiuojama statistika Z:

$$Z = \frac{m - na}{\sqrt{na(1-a)}} = \frac{\hat{p} - a}{\sqrt{a(1-a)/n}}, \quad (66)$$

čia m – imties vienetų skaičius, o $\hat{p} = m/n$.

Tegu reikšmingumo lygmuo yra lygus α. Hipotezė H_0 yra atmetama, kai $Z > z_\alpha$, o neatmetama, kai $Z \le z_\alpha$. Čia z_α yra standartinio normaliojo skirstinio α -oji kritinė reikšmė.

2. PRAKTINĖ DALIS

Praktinėje dalyje yra aprašoma valiutų rinka, prekybos joje niuansai, elektroninės prekybos platforma, keletas techninių indikatorių. Taip pat pateikiami parašytos SAS programos naudojimas, duomenų ir naudojamų statistikų aprašymai. Galiausiai pateikiama teorinėje dalyje minėtų strategijų ir metodų pritaikymo valiutų rinkos duomenims analizė.

2.1. Valiutų rinka

Šiame poskyryje aprašoma, kas yra valiutų rinka, kam ji reikalinga ir kaip veikia.

Tarpbankinė valiutų rinka (angl. foreign exchange markets, FX markets arba FOREX) yra sistema, padedanti susitikti valiutos pirkėjams ir pardavėjams. Seniau šio tipo prekyba buvo daugiausiai susijusi su eksporto ir importo apmokėjimu, o dabar didžioji dalis šios prekybos tenka tarptautiniam investavimui, arbitražui ir spekuliacijai. Ši rinka turi didžiausią dienos apyvartą iš visų rinkų (akcijų, žaliavų ir pan.), kuri siekia net 4 trilijonus JAV dolerių. Pagrindiniai prekybos valiuta centrai yra Londonas, Niujorkas ir Tokijas. Pagrindiniai valiutų rinkos dalyviai yra šie: centriniai bankai, komerciniai bankai ir kitos finansinės institucijos, tarpbankiniai brokeriai, didelės tarptautinės korporacijos ir eksporto-importo firmos [6]. Žinoma, nereikėtų pamiršti ir begalės smulkiųjų spekuliantų.

Valiutos rinkos rinka pasižymi tokiomis savybėmis [10]:

- pasaulinė rinka;
- 24 val. per parą, 5 dienas per savaitę vykstanti prekyba;
- elektroninė prekyba;
- likvidi rinka;
- svertinė rinka;
- informatyvi rinka;

Kaip žinoma, valiutos kursas yra santykinis dydis, jis tik transformuoja prekių, paslaugų ir turto kainas iš vienos valiutos į kitą. Jeigu valiutos kursas kyla kitų valiutų atžvilgiu, tai šalies ekonomika, eksportas tampa mažiau konkurencingas užsienyje, tačiau ima mažėti infliacija, kainos lieka stabilesnės, ir atvirkščiai. Tiek per stipri, tiek per silpna valiuta gali turėti neigiamos įtakos šalies ekonomikai [6].

Valiutų kursai gali būti įvairūs, bet šiame darbe yra naudojamas rinkos kursas – tai neatidėliotino sandorio kursas rinkoje realiu laiku [6].

Valiutų rinką galima analizuoti dviem metodais. Pirmasis – techninė analizė. Ji analizuoja valiutos kainos judėjimo praeitį, o po to daromos prognozė apie ateities kainos judėjimą. Ši analizė pasižymi tokiomis savybėmis [7]:

- valiutos kainų judėjimo grafikas puikiai atspindi pirkėjų ir pardavėjų santykį, bei jame jau yra visa viešai žinoma informacija;
- kainos linkusios judėti trendais (kryptingai) ir kitokiomis struktūromis, ir tada, pasiremiant praeitimi, galima gauti statistiškai reikšmingas tikimybes apie būsimus kainų judėjimus;

Kita analizės rūšis – fundamentalioji analizė. Ji bando nuspėti ateities valiutų kainos kilimus ir kritimus atsižvelgiant į būsimas naujienas ar kitus ekonominius rodiklius. Pavyzdžiui, svarbūs ekonomikos rodikliai laikomi tokie: palūkanų norma, BVP, nedarbo lygis ir pan. Kai tik svarbi informacija yra paskelbiama, rinka reaguoja akimirksniu. Jei rodiklis skiriasi nuo prognozės, valiutos kainos šuolis būna dar didesnis. Tai pat svarbūs yra įvairių centrinių bankų prezidentų, galingų šalių lyderių pasisakymai. Todėl daugumai naujienų yra sudaromi kalendoriai, ir joms galima pasiruošti iš anksti [5].

2.2. Valiutų poros

Pasaulyje šiuo metu yra daugybė skirtingų valiutų, o pasitelkiant šiuolaikines technologijas, nesunku sužinoti vienos valiutos santykį su kita. Vienų valiutų santykiai visą laiką kinta, pavyzdžiui euro ir JAV dolerio, kitos yra susietos, tokios kaip euras ir litas. Dėl patogumo valiutos santykis toliau vadinamas tiesiog kaina arba kursu.

Tiriama euro – JAV dolerio valiutos pora, kurios sutrumpinimas yra EURUSD. Būtent toks užrašymas (ir ne atvirkščias) naudojamas ir suprantamas visame pasaulyje. Šis užrašymas reiškia, kad euras yra bazinė valiuta, nes jo pozicija yra priekyje dolerio. Taigi šių valiutų santykis parodo, kiek dolerių kainuoja vienas euras. Ir jei tas santykis kyla, reiškia euras brangsta, stiprėja dolerio atžvilgiu ir atvirkščiai. Jei manoma, kad kaina kils, tada doleriai parduodami ir perkami eurai, o kainai pakilus, parduodami eurai ir vėl perkami doleriai. Žinoma, po vieną ar po du eurus niekas neperka. Yra sutartinis valiutos kiekio vienetas – lotas. Jis atitinka 100000 pagrindinės valiutos vienetų. Taip pat, valiutos kitimas dažniausiai vyksta ketvirto skaičiaus po kablelio lygmenyje, ir tas skaičius vadinamas punktu. Pavyzdžiui, 2012 m. gegužės 7 dieną 16:00 EURUSD kaina buvo 1.3029, o už pusvalandžio pakilo iki 1.3055. Reiškia, kaina iš viso pakilo 26 punktais. Taigi, jei tada 16:00 būtume pirkę vieną lotą EURUSD, o 16:30 jį pardavę, būtume uždirbę 260 dolerių. Žinoma, ne kiekvienas turi tiek pinigų, kad iš karto galėtų pirkti vieną lotą ar netgi daugiau valiutos, todėl galima pirkti jų mažiau arba pasinaudoti svertu.

Šis įrankis padidina turimus pinigus tam tikrą kartų kiekį (būna ir 50, ir 100, ir netgi 500 dydžio svertas). Bet tiek pat kartų padidėja rizika šiuos pinigus prarasti esant neteisingam spėjimui.

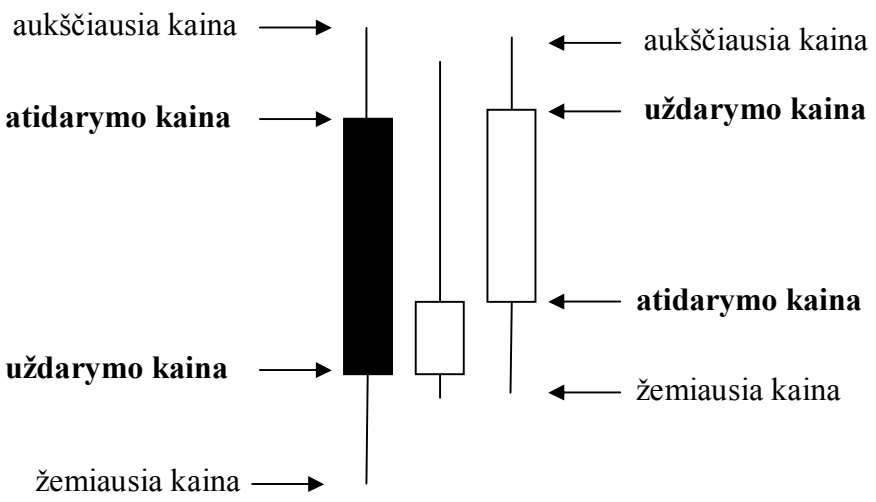
2.3. Prekybos platforma ir programavimo kalba

Norint prekiauti valiutomis realiu laiku, reikia turėti prieiga prie informacijos, duomenų. Tam yra sukurta daugybė elektroninės prekybos platformų, kurių pagalba galima prekiauti ne tik valiutomis, bet įvairiomis akcijomis, žaliavomis, išvestinėmis finansinėmis priemonėmis ir pan.

Buvo pasirinkta gana populiari „MetaTrader 4“ (MT4) prekybos platforma, joje galima prekiauti vien tik valiutomis. Bet ji turi daug techninės analizės įrankių, kurie padeda prekiaujant. Taip pat yra galimybė pasirinkti bandomąją versiją, kurioje galima išbandyti jėgas prekiaujant valiutomis, išbandyti įvairias strategijas nerizikuojant savomis lėšomis. Kaip atrodo ši platforma, parodyta (3 pav.).

3 pav. Prekybos platforma „MetaTrader 4“.

Kaip matyti iš paveikslėlio, valiutos kurso kitimo grafikas yra pats didžiausias, ir jis pavaizduotas taip vadinamomis „japoniškomis žvakėmis“. Šios žvakės atspindi tam tikro periodo kainos kitimą. Periodus galima pasirinkti tokius: 1 min., 5 min., 15 min., 30 min., 1 val., 4 val., 1 diena ir 1 savaitė. Jeigu žvakė pilnavidurė (šiuo atveju balta), tada reiškia kad per pasirinkta periodą valiutos kursas nukrito, o jei tuščiavidurė, tada valiutos kursas pakilo. Aiškesnė japoniškų žvakių struktūra pateikta paveikslėlyje (4 pav.).



4 pav. Japoniškų žvakių struktūra.

Specialiai šiai platformai yra sukurta ir programavimo kalba „MetaQuotes Language 4“ (MQL4). Tiek platforma, tiek pati programavimo kalba sukurta kompanijoje „MetaQuotes Software Corp.“. Pasinaudojant šia programavimo kalba, galima suprogramuoti savo pasirinktą prekybos strategiją, ją paleisti platformoje ir programa galėtų prekiauti be jokio įsikišimo. Taip pat šioje platformoje yra strategijos testuotojas, kuris parašytą programą išbando su istoriniais duomenimis ir parodo, kiek būtų buvę galima uždirbti arba prarasti, taip pat pateikia ir kitokią statistiką, susijusią su prekyba valiutomis.

Darbo autorius pačios prekybos strategijos su šia programa nekūrė, bet parašė programą, kuri iš šios platformos eksportuoja duomenis pasirinktu .xls formatu. Buvo pasirinkta 60 min. periodo duomenys (žymima M60). Eksportuota tokia informacija: data, laikas (žvakės suformavimo pradžia), savaitės diena, atidarymo kaina, aukščiausia kaina, žemiausia kaina, uždarymo kaina bei tokie techninės analizės rodikliai: paprastas slenkantis vidurkis, eksponentinis slenkantis vidurkis, santykinis stiprumo indeksas, slankiojo vidurkio konvergencijos ir divergencijos indikatorius. Apie šiuos techninius indikatorius daugiau rašoma 2.4 poskyryje. O pati programa pateikta 1 priede.

2.4. Techninės analizės indikatoriai

Šiame poskyryje, pasinaudojant šaltiniu [6], pateikiami slenkančio vidurkio, santykinio stiprumo indekso bei slenkančio vidurkio konvergencijos ir divergencijos indikatoriaus aprašymai.

2.4.1 Slenkantis vidurkis (SMA, EMA)

Slenkantis vidurkis (angl. moving average, MA) yra plačiausiai naudojamas techninės analizės indikatorius. Jo pagrindinė funkcija yra nustatyti rinkos kryptį, atmetant trumpalaikius kainos svyravimus, bei rodant tik esmę. Nors slenkantis vidurkis nurodo rinkos krypties tendenciją, jis nieko negali pasakyti apie krypties stiprumą ir rinkos nuotaikas. Dar vienas slenkančio vidurkio bruožas yra tas, kad jis yra atsiliekantis indikatorius, nerodantis rinkos pokyčių, kol jie neįvyksta.

Paprastas slenkantis vidurkis (angl. simple moving average, SMA) yra paprasčiausias iš slenkančiu vidurkių. Yra susumuojama keleto periodų kaina ir padalinama iš periodų (raundų) skaičiaus. Kuo daugiau periodų, tuo lygesnė ir mažiau reaguojanti gaunama indekso linija. Šis paprastas slenkantis vidurkis turi trūkumą, kad visų periodų duomenų įtaka indeksui yra vienoda.

Eksponentinis slenkantis vidurkis (angl. exponential moving average, EMA) didesnę reikšmę teikia naujesniems duomenims, tačiau nepamiršta ir senesnių laikotarpių. Eksponentinio vidurkio vertė apskaičiuojama pagal geometrinės progresijos principą

2.4.2. Santykinis stiprumo indeksas (RSI)

Santykinio stiprumo indeksas (angl. relative strenght index, RSI) yra bene labiausiai naudojamas ir žinomas impulso indikatorius. Jo tikslas yra parodyti perpirktas ir perparduotas rinkas. RSI svyruoja tarp 0 ir 100. Rinka laikoma perparduota, kai indikatoriaus reikšmė yra žemiau 30, o perpirkta - virš 70. Šie lygiai geriausiai tinka 14 periodų RSI. Žinoma, gali būti naudojami ir kitokių periodų laikotarpis. Tačiau kuo mažesnis laikotarpis, tuo kraštutinesnės gaunamos RSI reikšmės.

2.4.3. Slenkančio vidurkio konvergencijos ir divergencijos indikatorius (MACD)

Slenkančio vidurkio konvergencijos ir divergencijos (angl. moving average convergence and divergence, MACD) indikatorius yra trijų, eksponentiškai išlygintų slenkančių vidurkių derinys. Dažnai naudojami 9, 12 ir 26 laikotarpių eksponentiniai slankantys vidurkiai (EMA 9, EMA 12 ir EMA 26). Šis indikatorius sudarytas iš dviejų linijų. Pirmoji pagrindinė linija rodo skirtumą tarp EMA 12 ir EMA 26. Antroji signalizuojanti linija yra 9 periodų pirmosios pagrindinės linijos eksponentinis slenkantis vidurkis. Pirkimo/pardavimo signalus rodo linijų susikirtimas. Verta naudoti MACD histogramą, kuri yra viena iš geriausių techninės analizės priemonių. Ji ne tik rodo, kokia yra rinka – kylanti ar smunkanti – bet ir parodo, ar tas kilimas (smukimas) stiprėja ar silpnėja. MACD histograma gaunama iš pagrindinės linijos atimant signalizuojančią liniją.

2.5 SAS programos naudojimas, duomenys bei statistikos

Parašyta SAS programa apima teorinėje dalyje minimas strategijas, taip pat yra tikrinamos serijų ir proporcijų hipotezės, skaičiuojamos modelių statistikos, braižomi grafikai. Reikia pabrėžti, kad naudojami ne gryni duomenys, o jų skirtumai, iš *n*-tojo raundo duomenų reikšmės atimant *n-1* raundo duomenų reikšmes. Be to, yra naudojama tam tikra skirtumo transformaciją, kuri gautą skirtumą x transformuoja į tokį skaičių x' iš intervalo $[-1,1]$:

$$x' = \left(\frac{1}{1+e^{-\beta x}} \right) * 2 - 1 \tag{67}$$

Dydis β priklauso nuo pačių duomenų x didumo.

Makroprograma susideda iš tokių parametrų:

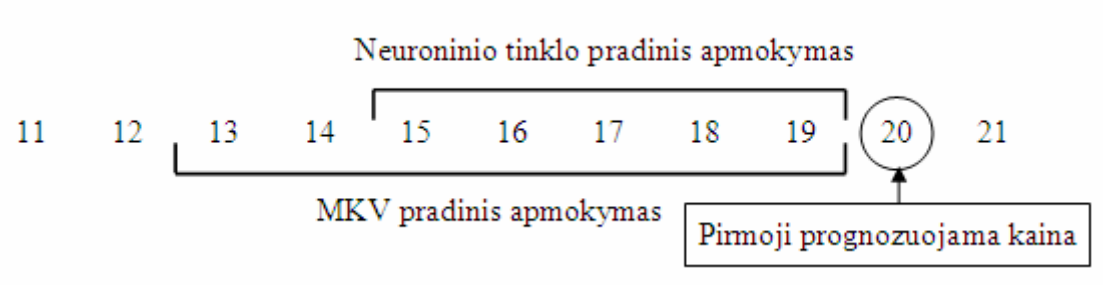
- PATH – kelias iki duomenų failo;
- Viso – kiek kainų bandoma atspėti;
- Initial – nu kurio kainų skirtumo taško prasidės prognozavimas;
- Beta – apmokomoji konstanta neuroninio tinklo svorių atnaujinimo algoritme;
- Apm – neuroninio tinklo apmokomojo lango dydis;
- Input – neuronų skaičius įėjimo sluoksnyje;
- Hidden – neuronų skaičius paslėptame sluoksnyje;
- Lambda – neuroninio tinklo svorių reguliarizacijos parametras;
- Kint – kintamojo pavadinimas, su kuriuo dirbama ir kurį mėginama prognozuoti;
- Pap_kint – papildomų kintamųjų sąrašas, naudojamas neuroniniame tinkle;
- Apm_MKV – Markovo strategijos apmokomojo lango dydis;
- MKV – Markovo strategija;
- SLW_K – branduolių metodo apmokomojo lango dydis;
- Pap_kint_kernel – papildomi kintamieji, naudojami branduolio metode kaip kovariantės;

Reikia atkreipti dėmesį, kad visiems parametras reikia nurodyti skaičius, išskyrus „PATH“, kur nurodome kelią iki duomenų (rašyti be kabučių) bei „MKV“, kur galima nurodyti tris variantus: MKV0, MKV1 ir MKV2 (taip pat rašyti be kabučių). Prie parametro „Kint“ galima rašyti tik vieną kintamąjį, o tiek prie „Pap_kint“, tiek prie „Pap_kint_kernel“ galima rašyti kiek tik norima papildomų kintamųjų, juos atskiriant tarpais.

Parašyta programa yra pritaikyta neuroniniam tinklui ir su vienmačiais, ir su daugiamačiais įėjimais. Jei prie „Pap_kint“ nėra jokių kintamųjų, tada turimas tinklas su vienmačiais įėjimais, o jei parašyta – automatiškai su daugiamačiais įėjimais. Parametro „Lambda“ reikšmė nurodo, ar neuroniniams tinklui naudojame reguliarizaciją ar ne. Jei įrašoma nulis, tada reguliarizacija nenaudojama, o jei įrašomas nelygus nuliui skaičius, tada reguliarizacija naudojama. Pagal tikslo

funkcijos apibrėžimą (28), parametras Lambda turi būti neigiamas ir gana nedidelis, kad svoriai netaptų lygiais nuliui.

Tarkime, kad Initial=20, Apm=5, Apm_MKV=7, tada vizualiai taip atrodys apmokymų ir spėjimų pradžia (5 pav.):



5 pav. Programos pradžia.

Taigi neuroninis tinklas iš pradžių apsimoko su 15-19 iteracijų duomenimis, prognozuoja 20 kainos pokytį, po to langas pasislenka per vieną iteraciją, apsimokoma su 16-20 kainų skirtumų duomenimis ir bandoma prognozuoti 21 kainos pokytį. Analogiškai veikia ir MKV bei branduolio metodai (paveikslėlyje nepavaizduotas). Suprantama, pradinio apmokymo lango dydis negali būti didesnis už „Initial“ reikšmę. Svarbu atkreipti dėmesį, kad visos trys strategijos ir metodai veikia būtent slenkančio lango principu, todėl pačioje programoje reikia nurodyti šių langų dydžius („Apm“, „Apm_MKV“, „SLW_K“).

Kaip minėta darbo įvade, yra naudojami šių strategijų ir metodų junginiai. Pagrindinei strategijai naudojamas neuroninis tinklas, o prie jo jungiami Markovo strategija ir branduolių metodas. Tarkime, kad apjungiami neuroninis tinklas ir MKV strategija. Tada, jei jų prognozės apie kainos kryptį sutampa, yra statomas neuroninio tinklo suprognozuotas kapitalas. O jei prognozės nesutampa, tada atsisakoma nuo sprendimo ir tą raundą prekyba nevyksta.

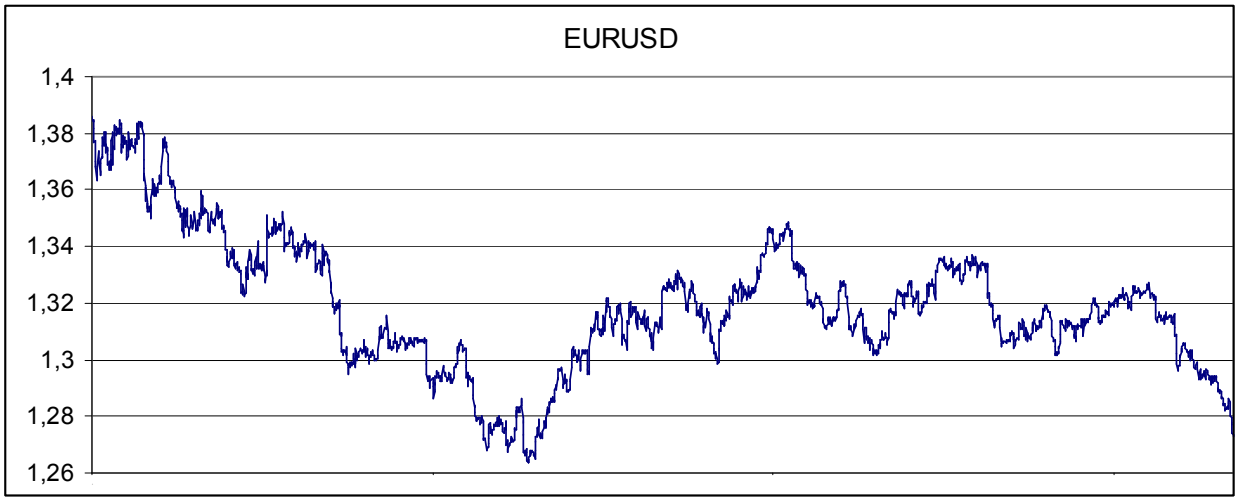
Teorinėje dalyje rašoma, kad norint maksimizuoti tikslo funkciją ϕ formulėje (17), reikia rasti tokius neuroninio tinklo svorius $\mathbf{w}^{(1,2)}$ ir $\mathbf{w}^{(2,3)}$, kurie maksimizuotų šią funkciją. Šių svorių radimo algoritmas aprašytas 1.1.3. skyrelyje. Literatūroje [2] rekomenduojama svorių atnaujinimą vykdyti tol, kol $\left|\Delta w_{ij}^{(1,2)}\right| < 0.0001$ ir $\left|\Delta w_{j}^{(2,3)}\right| < 0.0001$, bet ne daugiau nei 10000 iteracijų. Toks svorių ieškojimas ir atnaujinimas vyksta kiekviename raunde. Pačius pradinius svorius rekomenduojama imti atsitiktinius dydžius iš tolygaus intervalo $[-0.1;0.1]$. Taip pat verta paminėti, kad neuroninis tinklas yra adaptyvusis, jis einamojo lango pradinius svorius imi iš prieš tai buvusio lango, todėl jei rinka labai stipriai nesikeičia, užtenka kelių iteracijų, kad būtų išpildytos svorių atnaujinimo sąlygos dabartiniame lange.

Markovo strategijoje investicijų santykis α randamas pasinaudojant SAS programoje PROC IML procedūroje esančia paprograme NLPTR.

Aptarta SAS programa parašyta remiantis knyga [9] ir SAS Online Help [12]. O pati programa pateikta 2 priede.

Kaip minima 2.3. poskyryje, yra naudojami EURUSD valiutos valandinio periodo duomenys (M60). Periodo apibrėžimas yra tapatus raundo apibrėžimui teorinėje dalyje. 2.4. poskyryje aprašyti techninės analizės įrankiai yra sukurti atidarymo kainos pagrindu. Tai reiškia, kad visi šie rodikliai apimi vien tik atidarymo kainos duomenis. Naudojame techninius rodiklius su tokiais parametrais: turime 10, 34, 55 periodų SMA duomenis, 10, 34 periodų EMA duomenis, 14 periodų RSI bei 9, 12, 26 parametrų MACD.

Patys duomenis apima 2011-11-01 – 2012-05-15 laikotarpį. Jų viso turima 3351, bet kadangi yra skaičiuojami jų skirtumai, todėl duomenų sumažėja vienetu iki 3350. Pridedamas šio periodo kainų kitimų grafikas (6 pav.).



6 pav. EURUSD kainos kitimo grafikas per 2011-11-01 – 2012-05-15 periodą.

Naudojami modeliai bus vertinami tokiomis statistikomis:

- Tikslumas. Jis nurodo kokį procentą prognozuotų kainos pokyčių krypčių sutapo su realiomis kainų pokyčių kryptimis.
- Kapitalas. Kokio dydžio kapitalas turimas pabaigoje.
- Vidutinis kapitalas. Koks buvo vidutinis kapitalas per visą prognozavimo laikotarpi.

Šios trys statistikos skaičiuojamos neuroniniam tinklui ir Markovo strategijai, o branduolių metodui skaičiuojamas tik tikslumas.

Lyginant prognozę apie kainos pokyčio kryptį su realia kainos pokyčio kryptimi, yra naudojamas indikatorius, kuris įgyja reikšmę 1, jei prognozės ir kainos ženklai sutampa, kitu atveju reikšmė lygi 0. Tai reiškia, kad jei prognozuojamas kainos mažėjimas, ir kaina iš tikrųjų mažėja, tada indikatorius įgyją reikšmę 1, ir atvirkščiai. Taigi tada taip pat turime laiko eilutę, sudaryta vien tik iš 0 ir 1. Įdomu patikrinti, ar šioje eilutėje 0 ir 1 išsidėstę atsitiktinai, ar stebima klasterizacija, ar gaunamas stiprus susimaišymas. Tam tikslui ir yra naudojamas serijų testas, aprašytas 1.4.1. skyrelyje. Proporcijų testas naudojamas įsitikinti tam, ar gautas tikslumas

statistiškai skiriasi nuo 0.5, t.y. ar prognozuojama geriau nei 50%, ar rezultatai nesiskirtu nuo prognozių, gautų metant simetrinę monetą.

Galiausiai reikia paminėti, kad yra bandoma prekiauti 3000 raundų, o pradinis duomenų taškas („Initial“), nuo kurio pradedama yra 201.

2.6. Neuroninio tinklo analizė

Šiame poskyryje aprašoma neuroninio tinklo analizė. Ji skirstoma į du skyrelius, kur pirmajame aprašomas „vienmatis“ neuroninis tinklas, o antrajame „daugiamatis“ neuroninis tinklas. Neuroninio tinklo struktūra ieškoma tokiose rėžiuose: apmokymas (Apm) nuo 5 iki 10, įėjimo sluoksnio dydis (Input) nuo 1 iki 3, paslėptų neuronų dydis (Hidden) nuo 1 iki 10. Rasta geriausia „vienmačio“ neurono struktūra naudojama ir kitų tipų neuroniniams tikslams, nes ieškojimo algoritmas užtrunka apie 6-8 val. Ten, kur naudojama svorių reguliarizacija, jos parametras yra lygus -0.01.

2.6.1. Neuroninis tinklas su vienmačiais įėjimais

Kaip minėta poskyrio įvade, buvo ieškota geriausia neuroninio tinklo su vienmačiais įėjimais struktūra, geriausia laikoma ta, kurios tikslumas didžiausias. Rasta, kad tokia struktūra yra 7:2:8 (Apm=7, Input=2, Hidden=8). Šio neuroninio tinklo statistikos pateikiamos 1 lentelėje:

1 lentelė. Neuroninio tinklo strategijos rezultatai

	NT 7:2:8
Tikslumas	53,73 %
Kapitalas	34,11
Vidutinis kapitalas	18,19
Serijų testas	Z=-1.42, p=0.16
Proporcijos hipotezė	Z=4.09, p<0.001

Matyti, kad tikslumas nėra didelis, bet yra didesnis už 50 % (p<0.001), todėl to užtenka, kad būtų generuojamas teigiamas kapitalas. Serijų testo hipotezė neatmetama, bet p-reikšmė nėra itin didelė, todėl sprendžiant iš Z reikšmės, numanoma, kad teisingos ir neteisingos prognozės laiko eilutė yra linkusios šiek tiek klasterizuotis, bet ši nėra tokia stipri, kad sugebėtų atmesti nulinę hipotezę.

Toliau nagrinėjamas tokios pat struktūros neuroninis tinklas, bet šį kartą taikoma reguliarizacija neuroninio tinklo svoriams. Tokio tinklo rezultatai pateikiami 2 lentelėje:

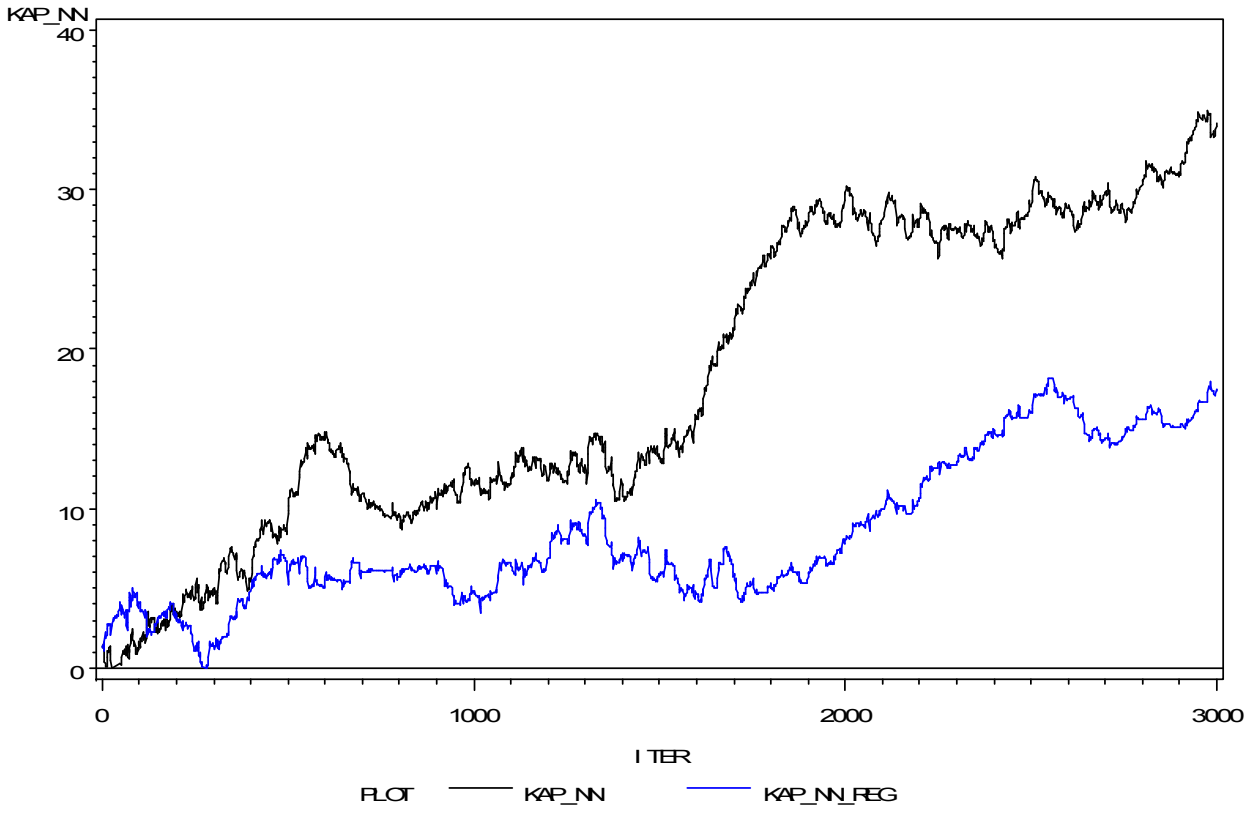
2 lentelė. Reguliarizuoto neuroninio tinklo strategijos rezultatai.

	NT 7:2:8 (reg.)
Tikslumas	50,2 %
Kapitalas	17,42
Vidutinis kapitalas	8,35
Serijų testas	Z=0.84, p=0.40
Proporcijos hipotezė	Z=0.22, p<0.413

Reguliarizuoto neuroninio tinklo rezultatai žymiai prastesni, nors to ir užtenka, kad būtų gaunamas teigiamas kapitalas. Serijų testas nieko įtartino nerodo, o proporcijos hipotezė patvirtina, kad gautas tikslumas statistiškai nesiskiria nuo 50%.

Verta atkreipti dėmesį ir į tai, kad naudojant nereguliarizuotą neuroninį tinklą, jo prognozės dažniausiai būna arba $+1$ arba -1. O reguliarizuoto tinklo prognozės kinta įvairiau. Todėl jei abiejų tinklų prognozių tikslumas būtų panašus, pirmojo tinklo kapitalas būtų vis tiek didesnis, nes jis beveik visada rizikuoja visu kapitalu (santykinai).

Iliustruota aptartų neuroninių tinklų kapitalo kitimo dinamika yra pateikiama paveikslėlyje (7 pav.)



7 pav. Neuroninio tinklo ir reguliazituoto neuroninio tinklo kapitalo dinamika.

Matyti, kad kapitalo kitimo tendencijos šiek tiek skiriasi (neatsižvelgiant į kapitalo dydį). Kada vieno linija svyruoja apie kažkokią reikšmę, kita kyla aukštyn arba leidžiasi, ir atvirkščiai. Bendra tendencija, kad abi kapitalo linijos linkusios kilti aukštyn.

2.6.2. Neuroninis tinklas su daugiamačiais įėjimais

Šiame poskyryje aprašoma neuroninio tinklo su daugiamačiais įėjimais analizė. Neuroninio tinklo struktūra naudojama tokia pat, kaip ir 2.6.1. skyrelyje, tik bus daugiau varijuojama papildomais kintamaisiais. Taip pat pridedami tokio pat tinklo reguliarizuotos versijos rezultatai.

Pradedama tik nuo vieno papildomo kintamojo RSI. Tokio tinklo rezultatai matyti 3 lentelėje.

3 lentelė. Neuroninio tinklo su daugiamačiai įėjimais rezultatai.

	NT 7:2:8	**NT 7:2:8 (reg.)**	**Papild. kintamieji**
Tikslumas	46,83 %	51,07 %	RSI
Kapitalas	-20,29	6,61	
Vidutinis kapitalas	-14,04	5,57	
Serijų testas	Z=0.33, p=0.74	Z=-0.27, p=0.79	
Proporcijos hipotezė	Z=-3.47, p=1	Z=1.16, p=0.12	

Gaunama, kad panaudojus reguliarizaciją, neuroninio tinklo tikslumas padidėja, taip pat padidėja kapitalas ir vidutinis kapitalas. Vis dėlto neuroninio tinklo su daugiamačiais įėjimais tikslumas mažesnis nei tokios pat struktūros neuroninis tinklas su vienmačiais įėjimais.

Dabar panaudojami du papildomi kintamieji: RSI ir MACD_H (slenkančio vidurkio konvergencijos ir divergencijos histograma). Rezultatai pateikiami 4 lentelėje.

4 lentelė. Neuroninio tinklo su daugiamačiai įėjimais rezultatai.

	NT 7:2:8	**NT 7:2:8 (reg.)**	**Papild. kintamieji**
Tikslumas	50,33 %	49,67 %	RSI MACD_H
Kapitalas	2,87	1,98	
Vidutinis kapitalas	4,71	-0,14	
Serijų testas	Z=3.11, p=0.002	Z=0.95, p=0.34	
Proporcijos hipotezė	Z=0.37, p=0.36	Z=-0.37, p=0.64	

Šiuo atveju reguliarizacija nepadidina tikslumo ir pats kapitalas yra mažesnis. Reikia pastebėti, kad be reguliarizacijos neuroninio tinklo kapitalo dinamika visą periodą buvo geresne nei pačioje pabaigoje, nes tada kapitalas nukrenta žemiau vidutinio kapitalo. O reguliarizuoto neuroninio tinklo atveju, kapitalas svyruoja apie nulį, nors pačioje pabaigoje pakyla beveik iki 2.

Taip pat reikėtų panagrinėti neuroninį tinklą iš karto su 5 papildomais kintamaisiais: SMA_10, SMA_34, SMA_55, EMA_10 ir EMA_34. Tokio neuroninio tinklo rezultatai matyti 5 lentelėje.

5 lentelė. EURUSD neuroninio tinklo su daugiamačiai įėjimais rezultatai.

	NT 7:2:8	**NT 7:2:8 (reg.)**	**Papild. kintamieji**
Tikslumas	47,27 %	46,83 %	SMA_10
Kapitalas	-23,25	-1,52	SMA_34
Vidutinis kapitalas	-10,25	-1,64	SMA_55
Serijų testas	Z=4.60, p<0.001	Z=1.76, p=0.09	EMA_10
Proporcijos hipotezė	Z=-2.99, p=1	Z=-3.47, p=1	EMA_34

Šiuo atveju tikslumo rezultatai vėl gana prasti, o kapitalas abiem atvejais yra neigiamas. Be to, nereguliarizuoto tinklo teisingų ir neteisingų atsakymų sekos linkusios šiek tiek klasterizuotis.

Galima padaryti išvadą, kad neuroniniam tinklui papildomi kintamieji teigiamas įtakos nepadaro, nes suprastėja tiek tikslumas, tiek kapitalas. Todėl šiuo atveju geriau naudotis vienmačiu neuroniniu tinklu. O jei vis dėlto norima neuroniniame tinkle naudoti papildomą informaciją, reikėtų keisti patį neuroninį tinklą, kuris galbūt geriau susidorotu su dideliu informacijos kiekiu.

Kadangi rezultatai nedžiuginantys, todėl nėra pateikiamas joks grafikas, nes dažniausiai kapitalas svyruoja apie nulį, arba yra netgi neigiamas.

2.7. Markovo strategijų analizė

Šiame poskyryje pateikiama Markovo strategijų MKV0, MKV1 ir MKV2 analizė.

Kaip ir neuroninio tinklo atveju, čia taip pat reikėjo rasti geriausių šių strategijų parametrus, kuris šiuo atveju tėra vienas – slenkančio lango dydis. Buvo ieškoma tokiose rėžiuose:

- MKV0: [2 : 200];
- MKV1: [10 : 200];
- MKV2: [50 : 200];

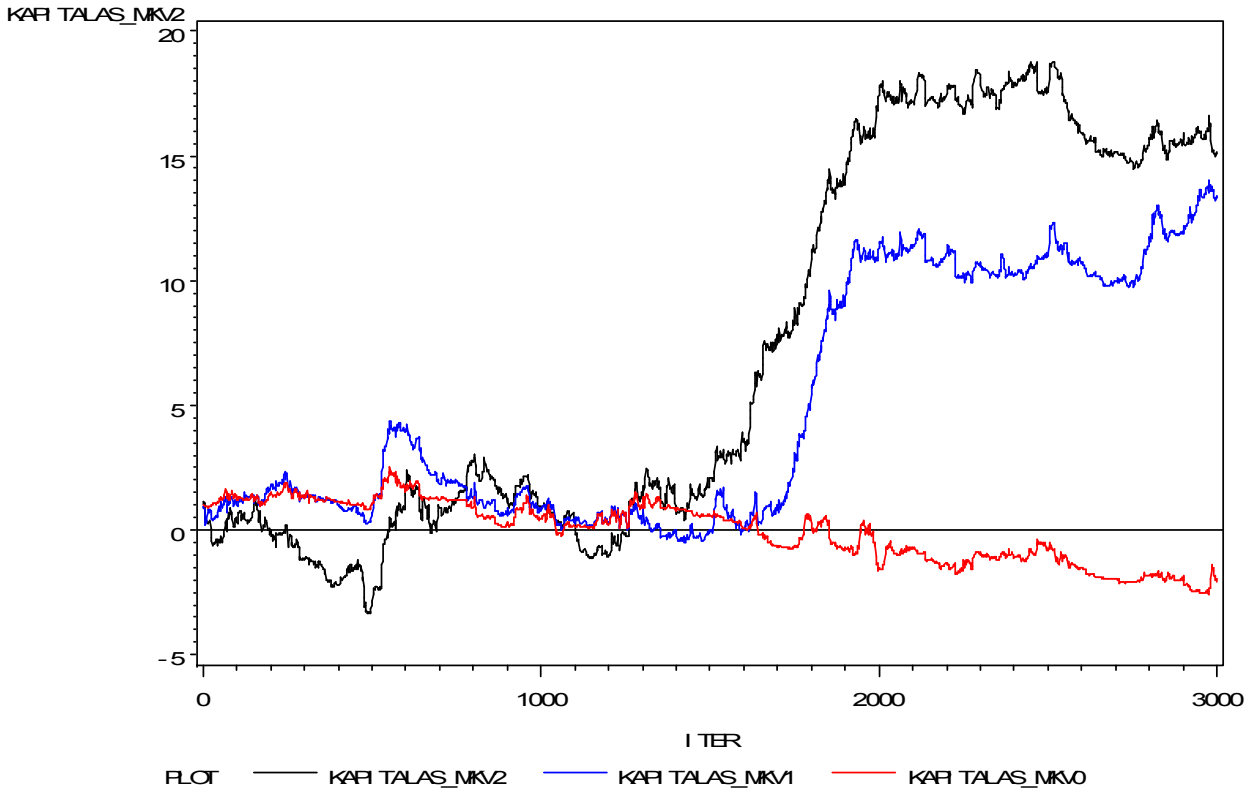
Rasta, kad MKV0 geriausius rezultatus rodo 146 periodų slenkančiam langui, MKV1 – 148, MKV2 – 192. Šių Markovo strategijų rezultatai pateikiami 6 lentelėje.

6 lentelė. EURUSD neuroninio tinklo su daugiamačiai įėjimais rezultatai.

	MKV0 (146)	**MKV1 (148)**	**MKV2 (192)**
Tikslumas	50,27 %	52,43 %	52,63 %
Kapitalas	-2,11	13,33	15,14

Vidutinis kapitalas	-0,02	5,02	7,28
Serijų testas	Z=6.61, p<0.001	Z=1.59, p=0.11	Z=-0.32, p=0.75
Proporcijos hipotezė	Z=0.29, p=0.39	Z=2.67, p=0.004	Z=2.89, p=0.002

Matyti, kad MKV2 strategija šiuo atveju yra geriausia. Nors nuo jos nelabai daug atsilieka ir MKV1 strategija. Abiejų strategijų tikslumas statistiškai skiriasi (viršija) nuo 50% ribos. Nekaip sekasi MKV0 strategijai, kuri generuoja neigiamą kapitalą, o tikslumas statistiškai nesiskiria nuo 50% ribos. Šiai strategijai serijų testas rodo, kad teisingi ir neteisingi spėjimai yra labai susimaišę. Taip pat pateikiamas šių trijų strategijų kapitalo kitimo grafikas (8 pav.). Matyti, kad MKV2 strategijai iš pradžių sekasi blogiausiai, bet po kurio laiko ji aplenkia kitas dvi strategijas. Taip pat galima pastebėti, jog MKV2 ir MKV1 strategijų svyravimų dinamika yra gana panaši.



8 pav. MKV2, MKV1 ir MKV0 strategijų kapitalo dinamika tiriamu periodu.

Palyginę neuroninio tinklo (7:2:8) (1 lentelė) ir MKV2 (192) (5 lentelė) metodus matyti, kad neuroninis tinklas rodo geresnius rezultatus tiek tikslumo, tiek kapitalo atveju. Žinoma, dėl kapitalo viskas aišku, nes, kaip minėta, neuroninis tinklas dažniausiai lošia iš viso kapitalo, o Markovo strategija ne. Žinoma, MKV2 veikia geriau už neuroninį tinklą su daugiamačiais įėjimais, bet jie smulkiau nenagrinėjami.

2.8. Branduolių metodo analizė

Šiame poskyryje trumpai aprašoma branduolių metodo analizė. Bet šis metodas prognozuoja ne pačios valiutos kainos pokytį, o jos kryptis. Kadangi daromos tik prognozės, todėl kapitalas ir vidutinis kapitalas nėra skaičiuojamas.

Teorinėje dalyje taip pat užsiminta, kad reikia riboti slenkančio lango dydį, nes tada labai išauga skaičiavimo kaštai (laikas). Bandant įvairias kovariančių kombinacijas, geriausio slenkančio lango dydžio ieškoma diapazone nuo 5 iki 100. Be to, tikslumui naudojamas proporcijų testas, o teisingų ir neteisingų atsakymų (1 ir 0) laiko eilutei patikrinamas serijų testą. Visi gauti rezultatai pateikiami 7 lentelėje.

7 lentelė. Branduolių metodo rezultatai esant įvairioms kovariančių kombinacijoms.

Kovariantės	**Langas**	**Tikslumas**	**Serijų testas**	**Proporcijos hipotezė**
OPEN	74	52,74 %	Z=2.89, p=0.004	Z=3.00, p=0.001
RSI	40	51,87 %	Z=2.58, p=0.01	Z=2.05, p=0.02
OPEN, RSI	81	51,81 %	Z=1.77, p=0.08	Z=1.98, p=0.02
OPEN, RSI, MACD_H	52	51,77 %	Z=0.97, p=0.33	Z=1.94, p=0.03
RSI, MACD_M MACD_S	89	51,64 %	Z=2.05, p=0.04	Z=1.80, p=0.04
SMA_10, SMA_34, SMA_55, EMA_10, EMA_34	64	51,44 %	Z=5.04, p<0.001	Z=1.58, p=0.06
SMA_10, SMA_34, SMA_55, EMA_10, EMA_34, RSI, MACD_M MACD_S	71	51,64 %	Z=0.38, p=0.71	Z=1.80, p=0.04
OPEN, SMA_10, SMA_34, SMA_55, EMA_10, EMA_34, RSI, MACD_M MACD_S, MACD_H	99	51,74 %	Z=0.52, p=0.60	Z=1.91, p=0.03

Iš lentelės matyti, kad kovariančių skaičius neigiamai veikia prognozių tikslumą, jis po truputį krenta. Galima numanyti, kad kai yra daugiau kovariančių, metodas tiesiog persimoko. Šiuo atveju tiksliausia, kai naudojame vieną kovariantę – atidarymo kainą (atidarymo kainos pokytį). Toks rezultatas atrodo visai logiškas. Taip pat iš lentelės matyti, kad esant mažiau konvariančių, teisingų ir neteisingų prognozių sekos linkusios kiek labiau klasterizuotis.

Taigi iš šio poskyrio gautą tiksliausią metodą dar pritaikysime sekančiame poskyryje.

2.9. Jungtinės strategijos ir metodai

Šiame skyrelyje aptariami jungtiniai metodai. Jungiami geriausius rezultatus parodę neuroninis tinklas (7:2:8), Markovo strategija (MKV2, 192) ir branduolių metodas (OPEN, 74). Kadangi neuroninis tinklas yra šio darbo „ašis“, todėl prie jo derinsimi kiti metodai. Principas yra toks, kad laikoma, jog parinktos strategijos ir metodai veikia tuo pačiu metu ir tuo pačiu metu pateikia prognozes. Tada, jei prognozės apie kainos kryptį sutampa, lošiamą iš neuroninio tinklo kapitalo santykio, o jei skiriasi - atsisakoma nuo sprendimo ir tą raundą nieko nedaroma.

Iš pradžių nagrinėjamas neuroninio tinklo ir Markovo strategijos junginys. Šio junginio rezultatai pateikiami 8 lentelėje.

8 lentelė. Neuroninio tinklo ir Markovo stategijos rezultatai

	NT (7:2:8) + MKV2 (192)
Tikslumas	55,04 %
Kapitalas	30,26
Vidutinis kapitalas	16,49
Sutapo	1993 (66,43 %)
Proporcijos hipotezė	$Z=4.50$, $p<0.001$

Gaunamas šiek tiek geresnis tikslumas nei naudojant abi strategijas atskirai. Kapitalas gaunamas mažesnis, nei vien neuroninio tinklo atveju, bet tai natūralu, nes daugmaž kas trečiame raunde atsisakoma priimti sprendimą. Nors šis kapitalas ir yra mažesnis už vien neuroninio tinklo kapitalą, jis nesumažėja trečdaliu, ko būtų galima tikėtis nelošiant trečdalyje raundų vien neuroninio tinklo strategija. Taip pat padidėjęs tikslumas neleidžia taip labai sumažėti kapitalui. Verta atkreipti dėmesį į tai, kad net apie 67 % atvejų neuroninio tinklo ir MKV metodo prognozės apie kainos kryptį sutampa.

Toliau aptariamas neuroninio tinklo ir branduolių metodo (BM) junginys. Tokio junginio rezultatai pateikiami 9 lentelėje.

9 lentelė. Neuroninio tinklo ir branduolių metodo rezultatai

	NT (7:2:8) + BM (OPEN, 74)
Tikslumas	56,73 %
Kapitalas	25,46
Vidutinis kapitalas	13,64
Sutapo	1479 (49,3 %)
Proporcijos hipotezė	$Z=5.17$, $p<0.001$

Dabar gaunamas dar geresnis tikslumas, nei neuroninio tinklo ir Markovo strategijos junginio. Bet kapitalas šiuo atveju mažesnis, nes daugiau nei pusė raundų yra atsisakoma nuo spendimo priėmimo. Tas taip pat reiškia, kad daugmaž kas antra abiejų metodų prognozė sutampa

Taip pat verta panagrinėti visų trijų strategijų ir metodų junginį. Šio junginio rezultatai pateikiami 10 lentelėje.

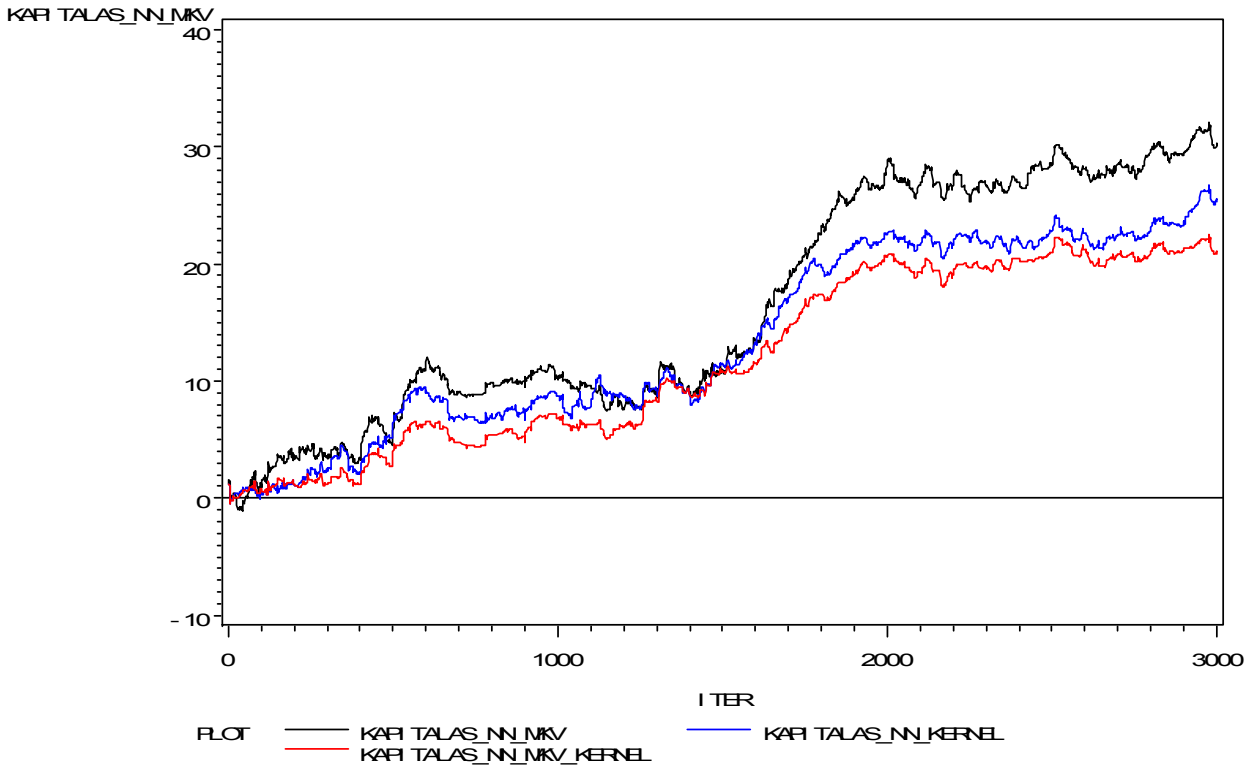
10 lentelė. Neuroninio tinklo, Markovo strategijos ir branduolių metodo rezultatai

	NT (7:2:8) + MKV2 (192) + BM (OPEN, 74)
Tikslumas	57,46 %
Kapitalas	21,00
Vidutinis kapitalas	11,91
Sutapo	959 (32 %)
Proporcijos hipotezė	$Z=4.62$, $p<0.001$

Šiuo atveju gaunami geriausi tikslumo rezultatai. Nors pats tikslumas pagerėja, toks jis gaunamas lošiant tik trečdalyje raundų. Bet vėlgi pats kapitalas nėra drastiškai mažas. Kadangi lošiama tik trečdalyje raundų, reiškia visų trijų strategijų ir metodų prognozės sutapo vidutiniškai tik kas trečiame raunde. Tai neturėtų labai stebinti, nes patys metodai yra pakankamai skirtingi ir veikia skirtingomis technikomis.

Toliau pateikiamas aptartų junginių grafikas (9 pav.), kuriame matyti kapitalo kitimo tendencijos visais trim atvejais.

Grafikas atvaizduoja jau ankščiau aptartus dalykus. Matyti, kad neuroninio tinklo ir MKV2 metodo kapitalo dinamika rodo geriausius rezultatus. Visais atvejais grafikų dinamika yra labai panaši, nes jie visi atspindi vieno neuroninio tinklo kapitalo kitimo dinamiką, tik su skirtingais (kartais ir tokiais pat) sprendimo atsisakymo momentais. Taip pat iš grafiko matyti, kad daugmaž ties 1300-1400 iteracijomis, visų trijų junginių kapitalai buvo lygūs, bet po to neuroninio tinklo ir MKV2 junginys išsiveržia į priekį, o su mažiausiu kapitalo lieka visų trijų strategijų ir metodų junginys.



9 pav. Neuroninio tinklo, Markovo strategijos ir branduolių metodo skirtingų junginių kapitalo kitimo dinamikos.

Apibendrinant galima pasakyti, kad aptartų strategijų ir metodų tikslumas yra nelabai didelis, bet kadangi jis yra kiek didesnis už 50 %, todėl ilguoju laikotarpiu yra generuojamas teigiamas kapitalas. Taip pat iš rezultatų išplaukia gyvenimiška taisyklė, kad didėjant patikimumui, tuo pačiu mažėja rizika, o tai lemia, kad mažėja ir kapitalas. Kitaip sakant, kur didesnė pinigai, ten didesni ir rizika. Todėl kiekvienas investuotojas turėtų pasirinkti jam priimtiną rizikos lygį, o nuo to priklausys kokią iš naudojamų strategijų rinktis.

IŠVADOS

Nagrinėjant neuroninį tinklą su vienmačiais įėjimais, gauta, kad duomenims geriausiai tinka 7:2:8 struktūra (apmokomojo lango dydis : neuronų skaičius įėjimo sluoksnyje : neuronų skaičius paslėptame sluoksnyje). Ši strategija generuoja 34,11 dydžio kapitalą, o prognozių tikslumas yra 53,73 %. Bet tokia strategija dažniausiai lošia iš viso kapitalo. Kad to išvengti, neuroniniam tinklui pritaikoma reguliarizacija. To pasėkoje gaunama mažesnio tikslumo strategija (50,2 %), kuri taip pat generuoja mažesnį kapitalo prieaugį (17,42). Verta paminėti, kad vertinant abiejų strategijų kapitalo kitimo dinamiką (7 pav.), stebimas daugiau mažiau pastovus kapitalo augimas.

Nagrinėjant neuroninius tinklus su daugiamačiais įėjimais, gauti ne itin džiuginantys rezultatai, be to, pastebimas šioks toks nepastovumas. Kartais nereguliarizuotas neuroninis tinklas rodo geresnius rezultatus, kartais reguliarizuotas. Pastebėta bendra tendencija, kad kuo daugiau papildomų kintamųjų dalyvauja įėjime, tuo prastesni rezultatai gaunami. Be to, tokių prognozių teisingų ir neteisingų prognozių serijose pastebima klasterizacija. Galima daryti išvadą, kad tokie tinklai tiesiog persimoko ir gaunami ne kokie rezultatai, todėl norint naudoti papildomą informaciją reikėtų bandyti naudoti kitokį neuroninį tinklą.

Lyginant Markovo strategijas tarpusavyje, geriausi rezultatai gaunami naudojant MKV2 strategiją. Nuo jos nedaug atsilieka MKV1, o MKV0 rezultatai gana nuviliantys. MKV2 ir MKV1 tiek tikslumai (52,63 % ir 52,43 %), tiek generuojama kapitalas (15,14 ir 13,33) yra gana panašūs. Vis dėlto, šioje vietoje geriausiu išrenkamas MKV2 metodas. Verta atkreipti dėmesį į tai, kad šioms Markovo strategijoms reikia pakankamai didelio apmokomojo lango, kuris priklausomai nuo periodo kinta nuo 150 iki beveik 200.

Kadangi darbe nagrinėjamos ir lyginamos tik dvi strategijos, tai palyginę neuroninį tinklą su vienmačiais įėjimais (7:2:8) ir Markovo strategiją MKV2 (192), geriau atrodo neuroninis tinklas, kurio tikslumas geresnis daugiau nei 1 % (kas šiuo atveju yra gana daug). O jo kapitalas daugiau nei per pusę didesnis. Žinoma, nereikėtų kapitalo skirtumo itin sureikšminti, nes neuroninis tinklas dažniausiai lošia iš viso kapitalo.

Analizuojant branduolių metodą, gauta, kad kuo mažiau kovariančių įeina į modelį, tuo dažniausiai gaunami geresni rezultatai. Bet tokiu atveju dažniau teisingos ir neteisingos prognozės klasterizuojasi. Žinoma, tokia taisyklė tinka tik šiems duomenims, kitiems duomenims gali būti visiškai kitaip. Taigi, gauta, kad tiksliausiai prognozuojamas uždarymo kainos pokytis pasinaudojant atidarymo kainos pokyčiu (52,74 %).

Nagrinėjant strategijų ir metodų junginius, gauti gana įdomūs rezultatai. Apjungime pagrindine strategija lieka neuroninis tinklas, ir jei jungiamų strategijų ir metodų prognozės

sutampa, lošiama iš neuroninio tinklo prognozuoto kapitalo dydžio (santykio). Apjungus neuroninio tinklo (7:2:8) ir MK2 (192) strategijas, prognozių tikslumas išauga iki 55,04 %, o kapitalas sumažėja iki 30,26, bet tai yra paaiškinama, nes lošiamų raundų skaičius sumažėja trečdaliu. Dar geresni rezultatai gaunami, apjungus neuroninio tinklo (7:2:8) strategiją ir branduolių metodą. Tada tikslumas pakyla iki 56,73 %. Šiuo atveju abu metodai vienodai prognozuoja kainos pokyčio kryptį daugmaž kas antrame raunde. Apjungus visas tris strategijas ir metodus, gaunamas pats geriausias tikslumas – 57,46 %. Bet šiuo atveju kapitalas nukrenta iki 21. Tai natūralu, nes metodai vienodas prognozes teikia beveik tik kas trečiame raunde, todėl ir kapitalas nespėja tiek daug užaugti.

Apibendrinant galima pasakyti, kad strategijų ir metodų junginiai yra verti dėmesio. Pasirinkimas priklauso nuo žmogaus, nes esant didesnei rizikai, gaunama ir didesnis pelnas. Nors prognozių tikslumai tikrai nėra dideli, bet jie viršija 50 % ir todėl ilguoju laikotarpiu yra generuojami teigiami pelnai.

LITERATŪRA

1. AK DHAMIJA, VK BHALLA. Financial Time Series Forecasting: Comparision of Neural Networks and ARCH models. *International Research Journal of Finance and Economics*. 2010 m., 49 leidinys, p. 185-202.
2. ARACHI R., TAKEMURA A. *Sequential optimazing investing strategy with neural networks*. 2010 m., p. 1-9.
3. ČEKANAVIČIUS V., MURAUSKAS G. *Statistika ir jos taikymai I*. 2000 m., TEV, p. 159-161.
4. ČEKANAVIČIUS V., MURAUSKAS G. *Statistika ir jos taikymai II*. 2000 m., TEV, p. 12-15.
5. DICKS J. *FOREX Made Easy*. The McGraw-Hill Companies, 2004 m.
6. KANCEREVYČIUS G., *Finansai ir investicijos* (III atnaujintas leidimas). 2009 m., „Smaltija“.
7. KNIGHT T. *Chart Your Way to Profits* (II leidimas). John Wiley & Sons, 2010 m.
8. KRUOPIS J. *Matematinė statistika*. „Mokslas“, 1993 m., p. 289.
9. LEVULIENĖ R. *Statistikos taikymai naudojant SAS®*. VU, 2009 m.
10. MENDELSOHN L. B. *Forex Trading Using Intermarket Analysis*. Market Technologies, 2006 m. p. 4-6.
11. PESTMAN W.R. *Mathematical Statistics*. WB-Druck Gmbh & Co, Berlin, 1998 m., p. 238-241.
12. SAS Online Help, http://support.sas.com/documentation
13. SHAFER G., VOVK V. *Probability and Finance: It's Only a Game!*. Wiley, New York, 2001 m., p. 61-75.
14. TAKEUCHI K., KUMON M., TAKEMURA A. *Mulstistep Bayesian strategy in coin-tossing games and its application to asset trading games in continuous time*. 2008 m., p. 9-11.
15. VAN VAERENBEGH S., VÍA J., SANTAMARIA I. Nonlinear System Identification Using a New Sliding-Window Kernel RLS Algorithm. *Journal of Communications*. 2007 m. 2 tomas, 2 numeris.
16. VERIKAS A., GELŽINIS A. *Neuroniniai tinklai ir neuroniniai skaičiavimai*. Kauno technologijos universitetas, 2008 m..
17. WALLACE M.P. Neural Networks and Their Application to Finance. *Business Intelligence Journal*. 2008 m., 1 tomas, 1 numeris, p. 67-76.

18. WIJNHOVEN R. Fast Training of Object Detection using Stochastic Gradient Descent. *International Conference on Pattern Recognition*, 2010 m.
19. ZHDANOV F., KALNISHKAN Y. *An Identity for Kernel Ridge Regression*. 2010 m.

SANTRAUKA

Šio darbo tikslas yra išnagrinėti neuroniniu tinklu paremtą prekybos strategiją, Markovo strategiją ir branduolių metodą, taip pat sukurti sudėtingesnį neuroninį tinklą ir visa tai pritaikyti valiutų rinkos duomenims.

Analizei naudoti euro ir JAV dolerio valiutų poros kaina, kuri apima 2011 lapkričio 1 d. – 2012 gegužės 15 d. periodą. Nagrinėjama šios valiutų poros uždarymo kaina valandiniu periodu.

Atlikus minėtą analizę, gauta, kad geriausius prognozių tikslumo rezultatus rodo neuroninio tinklo strategija, po to seka branduolio metodas ir galiausiai lieka Markovo strategija. O sukurto sudėtingesnio neuroninio tinklo rezultatai nėra labai geri. Analizuojant jungtinius metodus, gauta, kad tikslumas daugiausiai išauga jungiant visus tris metodus. Bet toks junginys generuoja mažiausią kapitalą. Šitai sąlygoja, kad lošiama tik kas trečiame raunde, o visais kitais atvejais yra atsisakoma nuo sprendimo priėmimo.

Išnagrinėti metodai yra verti dėmesio, o kurį rinktis, priklauso nuo paties žmogaus, nes kur didesnis kapitalas, ten didesnė ir rizika.

SUMMARY

The main aim of this work is to analyze a strategy based on the neural networks, Markovian strategies and kernel method, also to create more complex neural network and apply it all for the data of foreign exchange market.

Euro and U.S dollar currency pair price was analyzed during the 1 November 2011 - 15 May 2012 period. The closing price of the hourly period was analyzed.

The analysis revealed that the best forecasting accuracy shows neural network strategy, followed by the kernel method and end up with Markovian strategy. The results of created more complex neural network are not very good. The analysis of the joint methods revealed that the best accuracy obtained by combining all three methods. But it generates smaller capital than others. It happens, because bets are made only in the third of rounds, and in all other cases there is refusal to make decision.

The methods are worth of attention, and which one to choice depends on the person, because bigger capital leads to higher risk.

PRIEDAI

1 priedas. MQL4 programa

Pateikiama duomenų eksportavimo iš elektroninės prekybos platformos „MetaTrader 4“ programa, parašyta „MetaQuote Language 4“ programavimo kalba.

```
//+------------------------------------------------------------------+
int start() // Spec. funkcija start()
{
//+------------------------------------------------------------------+
   int n=4000;     // Kiek stulpeliu nuskaitysime
   int Handle;
   string File_Name="Duom_"+Symbol()+"_M"+Period()+".xls"; // Duomenu failo vardas
   string T[4000,3]; // Laiko masyvas
   string D[4000,12]; // Duomenu masyvas
   Handle=FileOpen(File_Name,FILE_CSV|FILE_WRITE,";");// Atidarome faila
   if(Handle==-1) // Jei nepavyko atidaryti failo
      {
      Alert("Klaida atidarinejant faila!");
      return; // Baigiama programa
      }
//+------------------------------------------------------------------+
   FileWrite(Handle,"DATE","TIME","DAY","OPEN","HIGH","LOW","CLOSE",
   "SMA_10","SMA_34","SMA_55","EMA_10","EMA_34","RSI","MACD_M","MACD_S");
   for(int i=n-1; i>=0; i--)
      {
      T[i,0]=TimeToStr(Time[i],TIME_DATE);
      T[i,1]=TimeToStr(Time[i],TIME_MINUTES);
      T[i,2]=TimeDayOfWeek(Time[i]);
      D[i,0]=DoubleToStr(Open[i],5);
      D[i,1]=DoubleToStr(High[i],5);
      D[i,2]=DoubleToStr(Low[i],5);
      D[i,3]=DoubleToStr(Close[i],5);
      D[i,4]=iMA(NULL,0,10,0,MODE_SMA,PRICE_OPEN,i);
      D[i,5]=iMA(NULL,0,34,0,MODE_SMA,PRICE_OPEN,i);
      D[i,6]=iMA(NULL,0,55,0,MODE_SMA,PRICE_OPEN,i);
      D[i,7]=iMA(NULL,0,10,0,MODE_EMA,PRICE_OPEN,i);
      D[i,8]=iMA(NULL,0,34,0,MODE_EMA,PRICE_OPEN,i);
      D[i,9]=iRSI(NULL,0,14,PRICE_OPEN,i);
      D[i,10]=iMACD(NULL,0,12,26,9,PRICE_OPEN,MODE_MAIN,i);
      D[i,11]=iMACD(NULL,0,12,26,9,PRICE_OPEN,MODE_SIGNAL,i);
      FileWrite(Handle,
      T[i,0],T[i,1],T[i,2],
      D[i,0],D[i,1],D[i,2],D[i,3],
      D[i,4],D[i,5],D[i,6],D[i,7],
      D[i,8],D[i,9],D[i,10],D[i,11]);
      }
//+------------------------------------------------------------------+
   FileClose( Handle ); // Uzdarome faila.
   Alert("The ",File_Name," file created.");
   return; // Baigiama programa
}
```

2 priedas. SAS programa

Pateikiama skaičiavimams naudota SAS programa. Skaičiavimai, priklausomai nuo kompiuterio, gali trukti nuo 2 iki 5 minučių.

```
%MACRO PROGRAMA(PATH=, Viso=, Initial=, Beta=, Apm=, Input=, Hidden=,
Lambda=, Kint=, Pap_kint=, Apm_MKV=, MKV=, SLW_K=, Pap_kint_kernel=);

/* Duomenu ikelimas i SAS*/
PROC IMPORT DATAFILE="&PATH" OUT=DUOMENYS
REPLACE;
GETNAMES=YES;
RUN;
DATA DUOMENYS; SET DUOMENYS;
MACD_H=MACD_M-MACD_S;
RUN;
DATA DUOM;
RETAIN &Kint &Pap_kint;
SET DUOMENYS;
KEEP &Kint &Pap_kint;
RUN;
PROC IML;/* Kainu skirtumu transformacija ir lenteles sukurimas*/
USE DUOM;
READ ALL INTO X;
N=NROW(X)-1;
M=NCOL(X);
Y=J(N,M,.);
DO i=1 TO N;
   Y[i,]=X[i+1,] - X[i,]; /* Kainu skirtumai */
END;
MIN=Y[><,];
MAX=Y[<>,];
SKIRT=MAX-MIN;
NORM=J(1,M,10);
DAUG=NORM/SKIRT;
DO i=1 TO N;
   DO j=1 TO M;
      Y[i,j]=(1/(1+exp(-1*DAUG[j]*Y[i,j])))*2-1; /* Kainu skirtumu
transformacija */
   END;
END;
CREATE KAINOS FROM Y;
APPEND FROM Y;
CLOSE KAINOS;
QUIT;
DATA KAINOS; SET KAINOS;
RENAME COL1=KAINA;
RUN;

/***  SOSNN SKAICIAVIMAS  ***/
PROC IML;
USE KAINOS;
READ ALL INTO X;
CLOSE KAINOS;
C1=&Apm-&Input; *<- Kiek iteraciju cikle C1;
Pradzia=&Initial-&Apm;
Sk=NCOL(X); *<- Kiek viso duomenu stulpeliu;

/*SUGENERUOJAM PRADINIUS SVORIUS*/
```

```
W12=J(&Input,&Hidden*Sk,.); /* &Input - eilutes, &Hidden*Sk - stulpeliai*/
W23=J(1,&Hidden*Sk,.);  /* 1 eilute, &Hidden*Sk stulpeliu */
/* W12 */
DO i=1 TO &Input;
   DO j=1 TO &Hidden*Sk;
      W12[i,j]=(RANUNI(12)/5 - 0.1);
   END;
END;
/* W23 */
DO j=1 TO &Hidden*Sk;
   W23[1,j]=(RANUNI(23)/5 - 0.1);
END;
/* SKAICIAVIMAI */
Duom=J(&Viso,6,.); /* Cia issaugomos prognozes, kapitalo kitimas ir pan. */

DO s=1 TO &Viso;
   N2=X[Pradzia+s-1:Pradzia+&Apm+s-2,];
   mW=10;
   Iter=0;
   DO WHILE((Iter<10000) || (mW>0.0001));
      I2=J(C1,&Hidden*Sk,.);
      Y2=J(C1,&Hidden*Sk,.);
      I3=J(C1,1,.);
      Y3=J(C1,1,.);
      DO k=1 TO C1;
         N1=N2[k:k+&Input-1,];
         /* I2 */
         DO u=1 TO &Hidden;
            I2_tarpine=N1#W12[,Sk*(u-1)+1:u*Sk];
            I2[k,sk*(u-1)+1:u*sk]=I2_tarpine[+,];
         END;
         /* Y2 */
         DO k1=1 TO &Hidden*Sk;
            IF I2[k,k1]>20 THEN I2[k,k1]=20; /* Apribojimai */
            IF I2[k,k1]<-20 THEN I2[k,k1]=-20; /* Apribojimai */
            Y2[k,k1]=(exp(2*I2[k,k1]) - 1) / (exp(2*I2[k,k1]) + 1);
         END;
         /* I3 */
         I3[k]=Y2[k,]*W23`;
         /* Y3 */
         IF I3[k]>20 THEN I3[k]=20; /* Apribojimai */
         IF I3[k]<-20 THEN I3[k]=-20; /* Apribojimai */
         Y3[k]=(exp(2*I3[k]) - 1) / (exp(2*I3[k]) + 1);
      END;
      /* Delta1 */
      Delta1=J(C1,1,.);
      DO k2=1 TO C1;
         Delta1[k2]=N2[k2+&Input]*(1/(1.000001 + Y3[k2]*N2[k2+&Input]))*(1 -
(Y3[k2])##2);
      END;
      /* Atnaujinam W23 */
      DeltaW23=&Beta*(Y2`*Delta1 + &Lambda*W23`);
      W23=W23 + DeltaW23`;
      /* Delta2 */
      Delta2=J(C1,&Hidden*Sk,.);
      DO k3=1 TO C1;
         DO j3=1 TO &Hidden*Sk;
            Delta2[k3,j3]=Delta1[k3]*W23[j3]*(1-(Y2[k3,j3])##2);
         END;
      END;
      /* Atnaujinam W12 */
      isW12=J(&Input,&Hidden*Sk,.);
      DO i4=1 TO &Input;
         DO j4=1 TO &Hidden*Sk;
```

```
            isW12_tarpine=N2[i4:C1+i4-1]#delta2[,j4];
            isW12[i4,j4]=isW12_tarpine[+,];
         END;
      END;
      DeltaW12=(isW12 + &Lambda*W12)*&Beta;
      W12=W12+DeltaW12;
      Iter=Iter+1;
      mW12=MAX(ABS(DeltaW12));
      mW23=MAX(ABS(DeltaW23));
      mW=MAX(mW12,mW23);
   END;

   /* Prognozavimas */
   N1=X[&Initial-&Input+s-1:&Initial+s-2,];
   /* I2 */
   I2=J(1,&Hidden*Sk,.);
   DO j4=1 TO &Hidden;
      I2_tarpine=N1#W12[,Sk*(j4-1)+1:j4*Sk];
      I2[,Sk*(j4-1)+1:j4*Sk]=I2_tarpine[+,];
   END;
   /* Y2 */
   Y2=J(1,&Hidden*Sk,.);
   DO k4=1 TO &Hidden*Sk;
      IF I2[k4]>20 THEN I2[k4]=20; /* Apribojimai */
      IF I2[k4]<-20 THEN I2[k4]=-20; /* Apribojimai */
   Y2[k4]=(exp(2*I2[k4]) - 1) / (exp(2*I2[k4]) + 1);
   END;
   /* I3 */
   I3=Y2*W23`;
   /* Y3 */
   IF I3>20 THEN I3=20; /* Apribojimai */
   IF I3<-20 THEN I3=-20; /* Apribojimai */
   Y3=(exp(2*I3) - 1) / (exp(2*I3) + 1);

   DUOM[s,1]=s;
   DUOM[s,2]=X[&Initial+s-1,1];
   DUOM[s,3]=Y3;
   IF s=1 THEN DUOM[s,4]=1+DUOM[s,2]*DUOM[s,3];
   ELSE DUOM[s,4]=DUOM[s-1,4]+DUOM[s,2]*DUOM[s,3];
   DUOM[s,5]=DUOM[s,2]*DUOM[s,3];
   IF (DUOM[s,5]>0) THEN DUOM[s,6]=1;
   ELSE DUOM[s,6]=0;
END;
PROC=DUOM[:,6]*100;
VID KAP=DUOM[:,4];
KAP=DUOM[&Viso,4];

/* Rezultatus sudedame i viena vieta*/
REZ=J(1,6,.);
REZ[1]=&Apm;
REZ[2]=&Input;
REZ[3]=&Hidden;
REZ[4]=PROC;
REZ[5]=KAP;
REZ[6]=VID_KAP;

/*** PROPORCIJOS TESTAS ***/
Z=2*(DUOM[:,6]-0.5)*SQRT(&Viso);
PVALUE=(1-probnorm(Z)); /* Vienpuse hipoteze*/
PROP=J(1,3,.);
PROP[1,1]=DUOM[:,6];
PROP[1,2]=Z;
PROP[1,3]=PVALUE;
```

```
CREATE NN_REZ FROM REZ; /* Sukuriame rezultatu lentele NN_REZ */
APPEND FROM REZ;
CLOSE NN_REZ;
CREATE NN_DUOM FROM DUOM; /* Sukuriame eigos lentele NN_DUOM */
APPEND FROM DUOM;
CLOSE NN_DUOM;
CREATE NN_PROP_T FROM PROP; /* Sukuriame proporciju testo rezultatu lentele
NN_PROP_T */
APPEND FROM PROP;
CLOSE NN_PROP_T;
QUIT;
DATA NN_REZ; SET NN_REZ;
RENAME
COL1=APM
COL2=INPUT
COL3=HIDDEN
COL4=PROC
COL5=KAP
COL6=VID_KAP;
RUN;
DATA NN_DUOM; SET NN_DUOM;
RENAME
COL1=ITER
COL2=KAINA
COL3=PROGNOZE
COL4=KAPITALAS
COL5=POKYTIS
COL6=SPEJIMAS;
RUN;
DATA NN_PROP_T; SET NN_PROP_T;
RENAME
COL1=PROC
COL2=Z
COL3=P_VALUE;
RUN;

/*** SERIJU TESTAS. Analizuojama lenteles NN_DUOM stulpelis SPEJIMAS ***/
PROC IML;
USE NN_DUOM;
READ ALL VAR {SPEJIMAS}INTO X;
CLOSE SS DUOM;
MAS=J(&Viso,2,0);
MAS[,1]=X;
MAS[1,2]=1;
DO i=1 TO &Viso-1;
   IF MAS[i+1,1]^=MAS[i,1] THEN MAS[i+1,2]=1;
END;
RUNS=MAS[+,2];
N1=MAS[+,1];
N0=&Viso-N1;
N=&Viso;
MIU=(2*N0*N1)/N + 1; /* Vidurkis */
SIGMA_2=(MIU-1)*(MIU-2)/(N-1); /* Dispersija */
SIGMA=SQRT(SIGMA_2);
IF (N0>=20) && (N1>=20) THEN Z=(RUNS-MIU)/SIGMA;
ELSE IF RUNS-MIU<0 THEN Z=(RUNS+0.5-MIU)/SIGMA;
ELSE Z=(RUNS-0.5-MIU)/SIGMA;
PVALUE=2*(1-probnorm(abs(Z)));
NN_RUNS_T=J(1,2,.);
NN_RUNS_T[1]=Z;
NN_RUNS_T[2]=PVALUE;
CREATE NN_RUN FROM NN_RUNS_T; /* Lentele su seriju testo rezultatais NN_DUOM
*/
APPEND FROM NN_RUNS_T;
```

```
CLOSE NN_RUN;
QUIT;
DATA NN_RUN; SET NN_RUN;
RENAME
COL1=Z
COL2=P_VALUE;
RUN;

/***  MARKOVO STRATEGIJA  ***/
PROC IML;
USE KAINOS;
READ ALL VAR {KAINA} INTO X;
CLOSE KAINOS;
MKV="&MKV";
IF MKV="MKV0" THEN MKV_SK=0;
ELSE IF MKV="MKV1" THEN MKV_SK=1;
ELSE IF MKV="MKV2" THEN MKV_SK=2;
Pradzia=&Initial-&Apm_MKV;
Apm=&Apm_MKV;
/* SKAICIAVIMAI */
Duom=J(&Viso,6,.);
Alpha = {0}; /* Pradinis paieskos taskas */
Optn = {1}; /* Ieskome maximumo, papildomos info nespausdiname */
Con = {-1,1}; /* Apribojimai */
DO s=1 TO &Viso;
   IF MKV="MKV0" THEN DO; /* MKV0 */
      Y=X[Pradzia+s-1-MKV_SK:Pradzia+&Apm_MKV+s-2];
      START MKV0(Alpha) GLOBAL(Y,Apm);
      SUMA = 0;
      DO i = 1 TO Apm;
         SUMA = SUMA + log(1+Alpha*Y[i]);
      END;
      F = SUMA;
      RETURN(F);
      FINISH MKV0;
      CALL NLPTR(Rc,Alpha_iv,"MKV0",Alpha,Optn,Con);
   END;
   IF MKV="MKV1" THEN DO; /* MKV1 */
      PLIUS=J(&Apm_MKV,1,.);
      MINUS=J(&Apm_MKV,1,.);
      Y=X[Pradzia+s-1-MKV_SK:Pradzia+&Apm_MKV+s-2];
      DO i=1 TO &Apm_MKV;
         IF Y[i]>=0 THEN PLIUS[i]=Y[i+1];
         IF Y[i]<0 THEN MINUS[i]=Y[i+1];
      END;
      PLIUS = PLIUS[LOC(PLIUS^=.),]; /* Matrica PLIUS be praleistu reiksmiu */
      MINUS = MINUS[LOC(MINUS^=.),]; /* Matrica MINUS be praleistu reiksmiu */
      N_plius=NROW(PLIUS);
      N_minus=NROW(MINUS);
      /* PLIUS */
      START MKV1_PLIUS(Alpha) GLOBAL(PLIUS,N_plius);
         SUMA = 0;
         DO i = 1 TO N_plius;
            SUMA = SUMA + log(1+Alpha*PLIUS[i]);
         END;
         F = SUMA;
         RETURN(F);
      FINISH MKV1_PLIUS;
      /* MINUS */
      START MKV1_MINUS(Alpha) GLOBAL(MINUS,N_minus);
         SUMA = 0;
         DO i = 1 TO N_minus;
            SUMA = SUMA + log(1+Alpha*MINUS[i]);
         END;
```

```
      F = SUMA;
      RETURN(F);
   FINISH MKV1_MINUS;
END;
IF MKV="MKV2" THEN DO; /* MKV2 */
   PLIUS_PLIUS=J(&Apm_MKV,1,.);
   PLIUS_MINUS=J(&Apm_MKV,1,.);
   MINUS_PLIUS=J(&Apm_MKV,1,.);
   MINUS_MINUS=J(&Apm_MKV,1,.);
   Y=X[Pradzia+s-1-MKV_SK:Pradzia+&Apm_MKV+s-2];
   DO i=1 TO &Apm_MKV;
      IF Y[i+1]>=0 && Y[i]>=0 THEN PLIUS_PLIUS[i]=Y[i+2];
      IF Y[i+1]>=0 && Y[i]<=0 THEN PLIUS_MINUS[i]=Y[i+2];
      IF Y[i+1]<=0 && Y[i]>=0 THEN MINUS_PLIUS[i]=Y[i+2];
      IF Y[i+1]<0 && Y[i]<0 THEN MINUS_MINUS[i]=Y[i+2];
   END;

   PLIUS_PLIUS = PLIUS_PLIUS[LOC(PLIUS_PLIUS^=.),]; /* Matrica PLIUS_PLIUS
be praleistu reiksmiu */
   PLIUS_MINUS = PLIUS_MINUS[LOC(PLIUS_MINUS^=.),]; /* Matrica PLIUS_MINUS
be praleistu reiksmiu */
   MINUS_PLIUS = MINUS_PLIUS[LOC(MINUS_PLIUS^=.),]; /* Matrica MINUS_PLIUS
be praleistu reiksmiu */
   MINUS_MINUS = MINUS_MINUS[LOC(MINUS_MINUS^=.),]; /* Matrica MINUS_MINUS
be praleistu reiksmiu */
   N_pl_pl=NROW(PLIUS_PLIUS);
   N_pl_mn=NROW(PLIUS_MINUS);
   N_mn_pl=NROW(MINUS_PLIUS);
   N_mn_mn=NROW(MINUS_MINUS);

   /* PLIUS_PLIUS */
   START MKV2_PL_PL(Alpha) GLOBAL(PLIUS_PLIUS,N_pl_pl);
      SUMA = 0;
      DO i = 1 TO N_pl_pl;
         SUMA = SUMA + log(1+Alpha*PLIUS_PLIUS[i]);
      END;
      F = SUMA;
      RETURN(F);
   FINISH MKV2_PL_PL;

   /* PLIUS_MINUS */
   START MKV2_PL_MN(Alpha) GLOBAL(PLIUS_MINUS,N_pl_mn);
      SUMA = 0;
      DO i = 1 TO N pl mn;
         SUMA = SUMA + log(1+Alpha*PLIUS_MINUS[i]);
      END;
      F = SUMA;
      RETURN(F);
   FINISH MKV2_PL_MN;

   /* MINUS_PLIUS */
   START MKV2 MN_PL(Alpha) GLOBAL(MINUS_PLIUS,N_mn_pl);
      SUMA = 0;
      DO i = 1 TO N_mn_pl;
         SUMA = SUMA + log(1+Alpha*MINUS_PLIUS[i]);
      END;
      F = SUMA;
      RETURN(F);
   FINISH MKV2_MN_PL;

   /* MINUS_MINUS */
   START MKV2_MN_MN(Alpha) GLOBAL(MINUS_MINUS,N_mn_mn);
      SUMA = 0;
      DO i = 1 TO N_mn_mn;
```

```
            SUMA = SUMA + log(1+Alpha*MINUS_MINUS[i]);
         END;
         F = SUMA;
         RETURN(F);
      FINISH MKV2_MN_MN;
   END;

   /* Prognozavimas */

   DUOM[s,1]=s;
   DUOM[s,2]=X[&Initial+s-1,1];
   IF MKV="MKV0" THEN DUOM[s,3]=alpha_iv;
   IF MKV="MKV1" THEN DO;
      IF X[&Initial+s-2,1]>=0 THEN DO;
         CALL NLPTR(Rc,Alpha_pl,"MKV1_PLIUS",Alpha,Optn,Con);
         DUOM[s,3]=alpha_pl;
      END;
      ELSE DO;
         CALL NLPTR(Rc,Alpha_mn,"MKV1_MINUS",Alpha,Optn,Con);
         DUOM[s,3]=alpha_mn;
      END;
   END;
   IF MKV="MKV2" THEN DO;
      IF X[&Initial+s-2,1]>=0 && X[&Initial+s-3,1]>=0 THEN DO;
         CALL NLPTR(Rc,Alpha pl_pl,"MKV2_PL_PL",Alpha,Optn,Con);
         DUOM[s,3]=alpha_pl_pl;
      END;
      IF X[&Initial+s-2,1]>=0 && X[&Initial+s-3,1]<=0 THEN DO;
         CALL NLPTR(Rc,Alpha pl_mn,"MKV2_PL_MN",Alpha,Optn,Con);
         DUOM[s,3]=alpha_pl_mn;
      END;
      IF X[&Initial+s-2,1]<=0 && X[&Initial+s-3,1]>=0 THEN DO;
         CALL NLPTR(Rc,Alpha mn_pl,"MKV2_MN_PL",Alpha,Optn,Con);
         DUOM[s,3]=alpha_mn_pl;
      END;
      IF X[&Initial+s-2,1]<0 && X[&Initial+s-3,1]<0 THEN DO;
         CALL NLPTR(Rc,Alpha_mn_mn,"MKV2_MN_MN",Alpha,Optn,Con);
         DUOM[s,3]=alpha_mn_mn;
      END;
   END;
   IF s=1 THEN DUOM[s,4]=1+DUOM[s,2]*DUOM[s,3];
   ELSE DUOM[s,4]=DUOM[s-1,4]+DUOM[s,2]*DUOM[s,3];
   DUOM[s,5]=DUOM[s,2]*DUOM[s,3];
   IF (DUOM[s,5]>0) THEN DUOM[s,6]=1;
   ELSE DUOM[s,6]=0;
END;
PROC=DUOM[:,6]*100;
VID KAP=DUOM[:,4];
KAP=DUOM[&Viso,4];

/* Rezultatus sudedame i viena vieta*/
REZ=J(1,5,.);
REZ[1]=&Apm MKV;
REZ[2]=MKV_SK;
REZ[3]=PROC;
REZ[4]=KAP;
REZ[5]=VID_KAP;

/* Proporcijos testas */
Z=2*(DUOM[:,6]-0.5)*SQRT(&Viso);
PVALUE=(1-probnorm(Z)); /* Vienpuse hipoteze*/
PROP=J(1,3,.);
PROP[1,1]=DUOM[:,6];
PROP[1,2]=Z;
```

```
PROP[1,3]=PVALUE;

CREATE MKV_REZ FROM REZ;    /* Sukuriame rezultatu lentele MKV_REZ */
APPEND FROM REZ;
CLOSE MKV_REZ;
CREATE MKV_DUOM FROM DUOM;  /* Sukuriame eigos lentele MKV_DUOM */
APPEND FROM DUOM;
CLOSE MKV_DUOM;
CREATE MKV_PROP_T FROM PROP; /* Sukuriame proporciju testo rezultatu lentele
MKV_PROP_T */
APPEND FROM PROP;
CLOSE MKV_PROP_T;
QUIT;
DATA MKV_REZ; SET MKV_REZ;
RENAME
COL1=APM_MKV
COL2=MKV
COL3=PROC_MKV
COL4=KAP_MKV
COL5=VID_KAP_MKV;
RUN;
DATA MKV_DUOM; SET MKV_DUOM;
RENAME
COL1=ITER
COL2=KAINA
COL3=PROGNOZE_MKV
COL4=KAPITALAS_MKV
COL5=POKYTIS_MKV
COL6=SPEJIMAS_MKV;
RUN;
DATA MKV_PROP_T; SET MKV_PROP_T;
RENAME
COL1=PROC
COL2=Z
COL3=P_VALUE;
RUN;

/*** SERIJU TESTAS. Analizuojama lenteles MKV_DUOM stulpelis SPEJIMAS_MKV
***/
PROC IML;
USE MKV_DUOM;
READ ALL VAR {SPEJIMAS_MKV}INTO X;
CLOSE MKV DUOM;
MAS=J(&Viso,2,0);
MAS[,1]=X;
MAS[1,2]=1;
DO i=1 TO &Viso-1;
   IF MAS[i+1,1]^=MAS[i,1] THEN MAS[i+1,2]=1;
END;
RUNS=MAS[+,2];
N1=MAS[+,1];
N0=&Viso-N1;
N=&Viso;
MIU=(2*N0*N1)/N + 1; /* Vidurkis */
SIGMA_2=(MIU-1)*(MIU-2)/(N-1); /* Dispersija */
SIGMA=SQRT(SIGMA 2);
IF (N0>=20) && (N1>=20) THEN Z=(RUNS-MIU)/SIGMA;
ELSE IF RUNS-MIU<0 THEN Z=(RUNS+0.5-MIU)/SIGMA;
ELSE Z=(RUNS-0.5-MIU)/SIGMA;
PVALUE=2*(1-probnorm(abs(Z)));
MKV_RUNS_T=J(1,2,.);
MKV_RUNS_T[1]=Z;
MKV_RUNS_T[2]=PVALUE;
CREATE MKV_RUN FROM MKV_RUNS_T; /* Lentele su seriju testo rezultatais */
```

```
APPEND FROM MKV_RUNS_T;
CLOSE MKV_RUN;
QUIT;
DATA MKV_RUN; SET MKV_RUN;
RENAME
COL1=Z_MKV
COL2=P_VALUE_MKV;
RUN;

/* Duomenys branduoliu metodui */
DATA DUOM2;
RETAIN &Kint &Pap_kint_kernel;
SET DUOMENYS;
KEEP &Kint &Pap_kint_kernel;
RUN;
PROC IML;/* Kainu skirtumu transformacija ir lenteles sukurimas*/
USE DUOM2;
READ ALL INTO X;
N=NROW(X)-1;
M=NCOL(X);
Y=J(N,M,.);
DO i=1 TO N;
   Y[i,]=X[i+1,] - X[i,]; /* Kainu skirtumai */
END;
MIN=Y[><,];
MAX=Y[<>,];
SKIRT=MAX-MIN;
NORM=J(1,M,10);
DAUG=NORM/SKIRT;
DO i=1 TO N;
   DO j=1 TO M;
      Y[i,j]=(1/(1+exp(-1*DAUG[j]*Y[i,j])))*2-1; /* Kainu skirtumu
transformacija */
   END;
END;
CREATE KAINOS2 FROM Y;
APPEND FROM Y;
CLOSE KAINOS2;
QUIT;
DATA KAINOS2; SET KAINOS2;
RENAME COL1=KAINA;
RUN;

/*** BRANDUOLIU METODAS ***/
PROC IML;
USE KAINOS2;
READ ALL INTO X;
CLOSE KAINOS2;
SK=NCOL(X);
C=0.5;
Sigma=1;
ALL=&Viso+2*&SLW_K;
Riba=0;
Pradzia=&Initial-&SLW_K;
Y=J(ALL,sk,0);
Y[&SLW_K+1:ALL,]=X[Pradzia:&Viso+&Initial-1,];
K=J(&SLW_K,&SLW_K,.);
N2=Y[1:&SLW_K,];
N=N2[,2:SK];
DO i=1 TO &SLW_K;
   DO j=1 TO &SLW_K;
      KR=N[i,]-N[j,];
      KR2=KR*KR`;
      K[i,j]=exp(-1*KR2/(2*Sigma));
```

```
   END;
END;
K=K+I(&SLW_K)*C;
K_INV=INV(K);
MAS=J(ALL,SK+3,.);
MAS[1:ALL,2:SK+1]=Y;
DO s=1 TO &Viso+&SLW_K-1;
   /* Sumazinam matrica */
   K_MAZ=K[2:&SLW_K,2:&SLW_K];
   /* Sumazintos matricos atvirkstinë */
   K_MAZ_INV=K_INV[2:&SLW_K,2:&SLW_K]-
K_INV[2:&SLW_K,1]*K_INV[2:&SLW_K,1]`/K_INV[1,1];
   /* Padidinam matrica */
   N2=Y[s+1:s+&SLW_K,];
   N=N2[,2:SK];
   DO i=1 TO &SLW_K;
      DO j=1 TO &SLW_K;
         KR=N[i,]-N[j,];
         KR2=KR*KR`;
         K[i,j]=exp(-1*KR2/(2*Sigma));
      END;
   END;
   /* Apjungiam */
   K[1:&SLW_K-1,1:&SLW_K-1]=K_MAZ;
   K[&SLW_K,&SLW_K]=K[&SLW_K,&SLW_K]+C;
   /* Padidintos matricos atvirkstine */
   g=INV(K[&SLW_K,&SLW_K]-K[&SLW_K,1:&SLW_K-1]*K_MAZ_INV*K[&SLW_K,1:&SLW_K-
1]`);
   K_INV[1:&SLW_K-1,1:&SLW_K-1]=K_MAZ_INV*(I(&SLW_K-1)+K[1:&SLW_K-
1,&SLW_K]*K[1:&SLW_K-1,&SLW_K]`*K_MAZ_INV*g);
   K_INV[&SLW_K,1:&SLW_K-1]=-1*(K_MAZ_INV*K[1:&SLW_K-1,&SLW_K])`*g;
   K_INV[1:&SLW_K-1,&SLW_K]=-1*(K_MAZ_INV*K[1:&SLW_K-1,&SLW_K])*g;
   K_INV[&SLW_K,&SLW_K]=g;
   Alpha=K_INV*N2[,1];
   N3=Y[s+1:s+&SLW_K+1,2:SK];
   IF s>=&SLW_K THEN DO;
      BRAND=J(&SLW_K,1,.);
      DO u=1 TO &SLW_K;
         KR=N3[&SLW_K+1,]-N3[u,];
         KR2=KR*KR`;
         BRAND[u]=exp(-1*KR2/(2*Sigma));
      END;
      PROG=Alpha`*BRAND;
      MAS[s+&SLW_K+1,1]=s-&SLW_K+1;
      MAS[s+&SLW_K+1,SK+2]=PROG;
   END;
END;
DO i=1 TO ALL;
   IF (MAS[i,1]^=.) && (MAS[i,SK+1]^=.) THEN DO;
      IF (MAS[i,2]>0) && (MAS[i,SK+2]>Riba) THEN MAS[i,SK+3]=1;
      IF (MAS[i,2]>0) && (MAS[i,SK+2]<Riba) THEN MAS[i,SK+3]=0;
      IF (MAS[i,2]<0) && (MAS[i,SK+2]<Riba) THEN MAS[i,SK+3]=1;
      IF (MAS[i,2]<0) && (MAS[i,SK+2]>Riba) THEN MAS[i,SK+3]=0;
   END;
END;
PROC=MAS[:,SK+3]*100;
DUOM=J(ALL,4,.);
DUOM[,1:2]=MAS[,1:2];
DUOM[,3:4]=MAS[,SK+2:SK+3];
/* Rezultatus sudedame i viena vieta*/
REZ=J(1,2,.);
REZ[1]=&SLW_K;
REZ[2]=PROC;
```

```
/* Proporcijos testas */
Z=2*(DUOM[:,4]-0.5)*SQRT(&Viso);
PVALUE=(1-probnorm(Z)); /* Vienpuse hipoteze*/
PROP=J(1,3,.);
PROP[1,1]=DUOM[:,4];
PROP[1,2]=Z;
PROP[1,3]=PVALUE;

CREATE KERNEL_REZ FROM REZ;    /* Sukuriame rezultatu lentele KERNEL_REZ */
APPEND FROM REZ;
CLOSE KERNEL_REZ;
CREATE KERNEL_DUOM FROM DUOM; /* Sukuriame eigos lentele KERNEL_DUOM */
APPEND FROM DUOM;
CLOSE KERNEL_DUOM;
CREATE KERNEL_PROP_T FROM PROP; /* Sukuriame proporciju testo rezultatu
lentele KERNEL_PROP_T */
APPEND FROM PROP;
CLOSE KERNEL_PROP_T;
QUIT;
DATA KERNEL_REZ; SET KERNEL_REZ;
RENAME
COL1=SLW_K
COL2=PROC_KERNEL;
RUN;
DATA KERNEL_DUOM; SET KERNEL_DUOM;
IF COL1>0;
RENAME
COL1=ITER
COL2=KAINA
COL3=PROGNOZE_KERNEL
COL4=SPEJIMAS_KERNEL;
RUN;
DATA KERNEL_PROP_T; SET KERNEL_PROP_T;
RENAME
COL1=PROC
COL2=Z
COL3=P_VALUE;
RUN;

/*** SERIJU TESTAS. Analizuojama lenteles KERNEL_DUOM stulpelis
SPEJIMAS_KERNEL ***/
PROC IML;
USE KERNEL_DUOM;
READ ALL VAR {SPEJIMAS_KERNEL}INTO X;
CLOSE KERNEL DUOM;
MAS=J(&Viso,2,0);
MAS[,1]=X;
MAS[1,2]=1;
DO i=1 TO &Viso-1;
   IF MAS[i+1,1]^=MAS[i,1] THEN MAS[i+1,2]=1;
END;
RUNS=MAS[+,2];
N1=MAS[+,1];
N0=&Viso-N1;
N=&Viso;
MIU=(2*N0*N1)/N + 1; /* Vidurkis */
SIGMA_2=(MIU-1)*(MIU-2)/(N-1); /* Dispersija */
SIGMA=SQRT(SIGMA_2);
IF (N0>=20) && (N1>=20) THEN Z=(RUNS-MIU)/SIGMA;
ELSE IF RUNS-MIU<0 THEN Z=(RUNS+0.5-MIU)/SIGMA;
ELSE Z=(RUNS-0.5-MIU)/SIGMA;
PVALUE=2*(1-probnorm(abs(Z)));
KERNEL_RUNS_T=J(1,2,.);
KERNEL_RUNS_T[1]=Z;
```

```
KERNEL_RUNS_T[2]=PVALUE;
CREATE KERNEL_RUN FROM KERNEL_RUNS_T; /* Lentele su seriju testo rezultatais
*/
APPEND FROM KERNEL_RUNS_T;
CLOSE KERNEL_RUN;
QUIT;
DATA KERNEL_RUN; SET KERNEL_RUN;
RENAME
COL1=Z_KERNEL
COL2=P_VALUE_KERNEL;
RUN;

/* BENDRA VISU METODU LENTELE */
DATA VISI_NN_MKV_KERNEL; MERGE NN_DUOM MKV_DUOM KERNEL_DUOM; BY ITER; RUN;

/*****  BENDRI METODAI   *****/
DATA BENDRA; SET VISI_NN_MKV_KERNEL;
KEEP ITER PROGNOZE POKYTIS PROGNOZE_MKV PROGNOZE_KERNEL;
RUN;
PROC IML;
USE BENDRA;
READ ALL INTO X;
CLOSE B_NN_MKV;
N=NROW(X);
M=NCOL(X);
Y=J(N,M+9,.);
Y[,1:M]=X;
NN1=0;
NN2=0;
NN3=0;
DO i=1 TO N;
   /* NN + MKV */
   IF (Y[i,2]>0 && Y[i,4]>0) THEN Y[i,M+1]=Y[i,3];
      ELSE IF (Y[i,2]<0 && Y[i,4]<0) THEN Y[i,M+1]=Y[i,3];
      ELSE Y[i,M+1]=0;
   IF i=1 THEN Y[i,M+2]=1+Y[i,M+1];
      ELSE Y[i,M+2]=Y[i-1,M+2]+Y[i,M+1];
   IF Y[i,M+1]>0 THEN Y[i,M+3]=1;
   IF Y[i,M+1]<0 THEN Y[i,M+3]=0;
   IF Y[i,M+1]=0 THEN Y[i,M+3]=.;
   IF Y[i,M+3]^=. THEN NN1=NN1+1;
   /* NN + KERNEL */
   IF (Y[i,2]>0 && Y[i,5]>0) THEN Y[i,M+4]=Y[i,3];
      ELSE IF (Y[i,2]<0 && Y[i,5]<0) THEN Y[i,M+4]=Y[i,3];
      ELSE Y[i,M+4]=0;
   IF i=1 THEN Y[i,M+5]=1+Y[i,M+4];
      ELSE Y[i,M+5]=Y[i-1,M+5]+Y[i,M+4];
   IF Y[i,M+4]>0 THEN Y[i,M+6]=1;
   IF Y[i,M+4]<0 THEN Y[i,M+6]=0;
   IF Y[i,M+4]=0 THEN Y[i,M+6]=.;
   IF Y[i,M+6]^=. THEN NN2=NN2+1;
   /* NN + MKV + KERNEL */
   IF (Y[i,2]>0 && Y[i,4]>0 && Y[i,5]>0) THEN Y[i,M+7]=Y[i,3];
      ELSE IF (Y[i,2]<0 && Y[i,4]<0 && Y[i,5]<0) THEN Y[i,M+7]=Y[i,3];
      ELSE Y[i,M+7]=0;
   IF i=1 THEN Y[i,M+8]=1+Y[i,M+7];
      ELSE Y[i,M+8]=Y[i-1,M+8]+Y[i,M+7];
   IF Y[i,M+7]>0 THEN Y[i,M+9]=1;
   IF Y[i,M+7]<0 THEN Y[i,M+9]=0;
   IF Y[i,M+7]=0 THEN Y[i,M+9]=.;
   IF Y[i,M+9]^=. THEN NN3=NN3+1;
END;
DUOM=J(N,10,.);
DUOM[,1]=Y[,1];
```

```
DUOM[,2:10]=Y[,M+1:M+9];
PROC1=DUOM[:,4]*100;
VID_KAP1=DUOM[:,3];
KAP1=DUOM[&Viso,3];
PROC2=DUOM[:,7]*100;
VID_KAP2=DUOM[:,6];
KAP2=DUOM[&Viso,6];
PROC3=DUOM[:,10]*100;
VID_KAP3=DUOM[:,9];
KAP3=DUOM[&Viso,9];
/* Rezultatus sudedame i viena vieta*/
REZ=J(1,12,.);
REZ[1]=PROC1;
REZ[2]=KAP1;
REZ[3]=VID_KAP1;
REZ[4]=NN1;
REZ[5]=PROC2;
REZ[6]=KAP2;
REZ[7]=VID_KAP2;
REZ[8]=NN2;
REZ[9]=PROC3;
REZ[10]=KAP3;
REZ[11]=VID_KAP3;
REZ[12]=NN3;
/* Proporcijos testas */
Z1=2*(DUOM[:,4]-0.5)*SQRT(NN1);
PVALUE1=(1-probnorm(Z1));
Z2=2*(DUOM[:,7]-0.5)*SQRT(NN2);
PVALUE2=(1-probnorm(Z2));
Z3=2*(DUOM[:,10]-0.5)*SQRT(NN3);
PVALUE3=(1-probnorm(Z3));
PROP=J(1,12,.);
PROP[1,1]=DUOM[:,4];
PROP[1,2]=Z1;
PROP[1,3]=PVALUE1;
PROP[1,4]=NN1;
PROP[1,5]=DUOM[:,7];
PROP[1,6]=Z2;
PROP[1,7]=PVALUE2;
PROP[1,8]=NN2;
PROP[1,9]=DUOM[:,10];
PROP[1,10]=Z3;
PROP[1,11]=PVALUE3;
PROP[1,12]=NN3;
CREATE BENDRA_REZZ FROM REZ;    /* Sukuriame rezultatu lentele BENDRA_REZ */
APPEND FROM REZ;
CLOSE BENDRA_REZ;
CREATE BENDRA_DUOM FROM DUOM;  /* Sukuriame eigos lentele BENDRA_DUOM */
APPEND FROM DUOM;
CLOSE BENDRA_DUOM;
CREATE BENDRA_PROP_T FROM PROP; /* Sukuriame proporciju testo rezultatu
lentele BENDRA_PROP_T */
APPEND FROM PROP;
CLOSE BENDRA_PROP_T;
QUIT;
DATA BENDRA_REZZ; SET BENDRA_REZZ;
RENAME
COL1=PROC_NN_MKV
COL2=KAP_NN_MKV
COL3=VID_KAP_NN_MKV
COL4=N_NN_MKV
COL5=PROC_NN_KERNEL
COL6=KAP_NN_KERNEL
COL7=VID_KAP_NN_KERNEL
```

```
COL8=N_NN_KERNEL
COL9=PROC_NN_MKV_KERNEL
COL10=KAP_NN_MKV_KERNEL
COL11=VID_KAP_NN_MKV_KERNEL
COL12=N_NN_MKV_KERNEL;
RUN;
DATA BENDRA_DUOM; SET BENDRA_DUOM;
RENAME
COL1=ITER
COL2=POKYTIS_NN_MKV
COL3=KAPITALAS_NN_MKV
COL4=SPEJIMAS_NN_MKV
COL5=POKYTIS_NN_KERNEL
COL6=KAPITALAS_NN_KERNEL
COL7=SPEJIMAS_NN_KERNEL
COL8=POKYTIS_NN_MKV_KERNEL
COL9=KAPITALAS_NN_MKV_KERNEL
COL10=SPEJIMAS_NN_MKV_KERNEL;
RUN;
DATA BENDRA_PROP_T; SET BENDRA_PROP_T;
RENAME
COL1=PROC_NN_MKV
COL2=Z_NN_MKV
COL3=P_VALUE_NN_MKV
COL4=N_NN_MKV
COL5=PROC_NN_KERNEL
COL6=Z_NN_KERNEL
COL7=P_VALUE_NN_KERNEL
COL8=N_NN_KERNEL
COL9=PROC_NN_MKV_KERNEL
COL10=Z_NN_MKV_KERNEL
COL11=P_VALUE_NN_MKV_KERNEL
COL12=N_NN_MKV_KERNEL;
RUN;
SYMBOL1 I=J C=BLACK W=1;
SYMBOL2 I=J C=BLUE W=1;
SYMBOL3 I=J C=RED W=1;
PROC GPLOT DATA=VISI NN MKV KERNEL;
PLOT KAPITALAS*ITER=1 KAPITALAS_MKV*ITER=2/ VREF=0 LEGEND overlay;
RUN;
QUIT;
PROC GPLOT DATA=BENDRA_DUOM;
PLOT KAPITALAS_NN_MKV*ITER=1 KAPITALAS_NN_KERNEL*ITER=2
KAPITALAS_NN_MKV_KERNEL*ITER=3/ VREF=0 LEGEND overlay;
RUN;
QUIT;
%MEND PROGRAMA;
%PROGRAMA(
PATH=,
Viso=,
Initial=,
Beta=,
Apm=,
Input=,
Hidden=,
Lambda=,
Kint=,
Pap_kint=,
Apm_MKV=,
MKV=,
SLW_K=,
Pap_kint_kernel=
);
```