KAUNO TECHNOLOGIJOS UNIVERSITETAS
MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

Paulius Čepulionis

SKRUZDŽIŲ KOLONIJOS ALGORITMAS CHAOTINIŲ LAIKO EILUČIŲ REKONSTRAVIMO UŽDAVINIUI OPTIMIZUOTI

Baigiamasis magistro projektas

Vadovas
Doc. dr. Kristina Lukoševičiūtė

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS
MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

SKRUZDŽIŲ KOLONIJOS ALGORITMAS CHAOTINIŲ LAIKO EILUČIŲ REKONSTRAVIMO UŽDAVINIUI OPTIMIZUOTI

Baigiamasis magistro projektas
Taikomoji matematika (kodas 621G10003)

Vadovas
Doc. dr. Kristina Lukoševičiūtė
2016.05.30

Recenzentas
Prof. dr. Jonas Valantinas

Projektą atliko
Paulius Čepulionis
2016.05.30

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Matematikos ir gamtos mokslų fakultetas
(Fakultetas)

Paulius Čepulionis
(Studento vardas, pavardė)

Taikomoji matematika, 621G10003
(Studijų programos pavadinimas, kodas)

„Baigiamojo projekto pavadinimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 16 m. gegužės 30 d.
Kaunas

Patvirtinu, kad mano, **Pauliaus Čepulionio**, baigiamasis projektas tema „Skruzdžių kolonijos algoritmas chaotinių laiko eilučių rekonstravimo uždaviniui optimizuoti“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka) *(parašas)*

TURINYS

Čepulionis, Paulius. Skruzdžių Kolonijos Algoritmas Chaotinių Laiko Eilučių Rekonstravimo Uždaviniui Rekonstruoti. Magistro baigiamasis projektas / vadovas doc. dr. Kristina Lukoševičiūtė; Kauno technologijos universitetas, matematikos ir gamtos mokslų fakultetas.

Mokslo kryptis ir sritis: Fiziniai mokslai, matematika

Reikšminiai žodžiai: skruzdžių kolonijos optimizavimo algoritmas, nereguliarus rekonstravimas, chaotinės laiko eilutės prognozavimas

Kaunas, 2016. 54 p.

SANTRAUKA

Laiko eilučių analizės tikslas yra nustatyti modelį, kuris apibūdina laiko eilutės dinamiką ir vėliau panaudoti šį modelį laiko eilučių prognozavimui. Nors yra labai daug efektyviai veikiančių laiko eilučių prognozavimo metodų, tačiau geriausio prognozavimo metodo nėra. Vienas iš dažniausiai naudojamų laiko eilučių prognozavimo metodų yra autoregresinis integruotas slenkančio vidurkio modelis (*ARIMA(p,d,q)*). Šis metodas bus naudojamas mūsų laiko eilučių palyginimui su mūsų siūlomu metodu.

Darbo tikslas yra sudaryti laiko eilučių, kurios yra grįstos nereguliariu rekonstravimu į laiko vėlinimų erdvę, dinaminį modelį. Rekonstravimo dimensijos nustatymui yra naudojamas klaidingų artimiausių kaimynų metodas. Darbe naudojami nereguliarūs laiko vėlinimai, kurių paieškai pritaikytas skruzdžių kolonijos optimizavimo algoritmas. Skruzdžių feromonų pasiskirstymui skaičiuoti yra naudojamas ruletės rato pasirinkimo algoritmas. Optimizavimo tikslo funkcija yra visų atraktorių vidutinis plotas. Modelis yra pritaikomas prognozei, kai surandamos tinkamiausios laiko vėlinimų reikšmės. Laiko eilučių prognozei yra naudojamas radialinių bazinių funkcijų neuroninis tinklas. Pusė laiko eilutės yra naudojama neuroninio tinklo apmokymui, o kita pusė – prognozei ir prognozės paklaidų įvertinimui.

Darbe siūlomo modelio įvertinimui prognozių rezultatai yra lyginami su kitais laiko eilučių prognozavimo metodais naudojantis prognozės paklaidų metrikomis. Darbe tikrinama hipotezė, jog mūsų siūlomas modelis geba geriau prognozuoti įvairias laiko eilutes lyginant su kitais metodais bei kitų autorių tyrimais.

Tyrime yra pateikiama Mackey-Glass chaotinės laiko eilutės prognozė ir kitos realaus pasaulio laiko eilutės. Mackey-Glass laiko eilutė buvo prognozuojama su mažomis paklaidomis. Dauguma kitų modelių prognozavo prasčiau nei mūsų siūlomas modelis. Tačiau straipsniuose galime rasti kitų metodų, kurie prognozavo šią laiko eilutę su žymiai mažesnėmis paklaidomis. Realaus pasaulio laiko eilučių prognozės paklaidos buvo lyginamos su ARIMA modeliu. Mūsų siūlomas modelis visur prognozavo tiksliau nei ARIMA modelis. Tyrimai parodė, jog mūsų siūlomas modelis sugeba prognozuoti toli į ateiti su mažomis paklaidomis.

Čepulionis, Paulius. Ant Colony Optimization Algorithm For Time Series Embedding Optimization: Master‘s thesis in applied mathematics / supervisor doc. dr. Kristina Lukoševičiūtė. The Faculty of mathematics and natural Sciences, Kaunas University of Technology.

Research area and field: physical sciences, mathematics

Key words: ant colony optimization, chaotic time series, non-uniform embedding

Kaunas, 2016. 54 p.

SUMMARY

Time series analysis purpose is to establish a model that describes the dynamics of the time series and then use this model for time series forecasting. Although there are a lot of effective operating time series forecasting methods, but not the best forecasting method. All methods have their own advantages and disadvantages. One of the most commonly used time-series forecasting methods are autoregressive integrated moving average model (*ARIMA(p,d,q)*). This method will be used in our time series to compare with our proposed method.

The aim of this work is to create the time series dynamic model, which is based on non-uniform embedding in the phase-space. False nearest neighbor method is used for determine the embedding dimension. Ant colony optimization algorithm is used for non-uniform time delay search. Ants’ pheromone is used to calculate the distribution of the roulette wheel selection algorithm. Optimization objective function is the average area of all attractors. The model is applied for forecasting with the best the time delay values. Finally, radial basis function neural network is used to forecast values. In the beginning half of the time series is used for training the neural network, and the other half - prognosis and the prediction error evaluation.

Results of forecast are compared with other time series methods using the prediction error metrics. The paper examined the hypothesis that our proposed model is better compared with other methods, and the other authors of the studies.

The study is presented in Mackey-Glass chaotic time series prediction, and other real-world time series. Mackey-Glass time series were predicted with small tolerances. Most other models predicted worse than our proposed model. However, there are methods with much smaller prediction tolerances. Real-world time series prediction error was compared with the ARIMA model. Our proposed model everywhere predicted more accurately than ARIMA model. Studies have shown that our proposed model is able to predict far into the future with small tolerances.

Ženklų, simbolių ir santrumpų sąrašas

d – laiko vėlinimų erdvės dimensija

τ – laiko vėlinimas

ACO – skruzdėlių kolonijos optimizavimo algoritmas

AI – dirbtinis intelektas

AR – autoregresinis modelis

ANN – dirbtiniai neuroniniai tinklai

AR(p) – autoregresinis modelis

ARIMA(p,d,q) – autoregresinis integruotas slenkančio vidurkio modelis su laipsniais p, d ir q

BSPO – dalelių spiečiaus algoritmu

ECG – elektrokardiograma

FNN – klaidingų artimiausių kaimynų metodas

MA – slenkančio vidurkio modelis

MAPE – vidutinė absoliučioji procentinė paklaida

MDL – minimalaus aprašymo ilgio modelis

NAR – netiesiniais autoregresiniais neuroniniais tinklais

RBF – radialinių bazinių funkcijų neuroniniai tinklai

RMSE – vidutinės kvadratinės paklaidos kvadratinė šaknis

RWS – ruletės pasirinkimo algoritmas

ĮVADAS

Laiko eilutė yra tam tikrais laiko momentais fiksuoti stebėjimų dydžiai. Laiko eilutės naudojamos priimant dabarties sprendimus pagrįstus stebėjimų istorija. Jų tikslas yra nustatyti modelį, kuris apibūdina laiko eilutės dinamiką ir, panaudojant tą modelį, prognozuoti laiko eilutės reikšmes į ateitį. [1, 2]

Chaosas buvo tiriamas ilgą laiką. Paprastai yra manoma, kad Poincaré yra pirmasis, kuris studijavo chaosą. XIX amžiuje Poincaré tyrė ribotą trijų kūnų problemą (angl. Three-body problem), kai vienas kūnas yra nereikšmingai mažas lyginant su kitais dviem. Poincaré nustatė, kad šios sistemos sprendinys yra labai sudėtingas ir negali būti tiksliai apskaičiuotas. Chaoso tyrimų pradininku laikomas Lorenz, kai 1963 metais, studijuodamas orų prognozę, atskleidė drugelio efektą (angl. butterfly effect). [3, 4]

Laiko raidos sistemų prognozavimas yra svarbi problema daugelyje mokslo sričių, pavyzdžiui, ekonomikos ar orų prognozių. Daugybė įrankių buvo sukurta būtent šiam tikslui. Svarbiausią metodą pasiūlė *Packard (1980),* kurio veikimo principu stengiamasi rekonstruojanti laiko eilutę į fazinę erdvę naudojantis laiko vėlinimo technika. Šios technikos pagalba yra konstruojamas laiko vėlinimų vektorius. Vėliau teoremą matematiškai įrodė *Takens (1981)* kaip įtvirtinimo teoremą (angl. embedding theorem), o ją patvirtino ir papildė *Sauer (1991).* Chaoso teorijos atradimas suteikė galimybę netiesinėmis deterministinėmis lygtimis atpažinti sunkiai nuspėjamo dėsningumo rezultatus. [4, 5, 6]

Darbo tikslas – sudaryti laiko eilučių, kurios yra grįstos rekonstravimu į laiko vėlinimų erdvę, dinaminį modelį paremtą geometrinėmis atraktoriaus savybėmis. Geometrinėms atraktoriaus savybėms tirti analizuojamas atraktorių užimamas plotas. Darbe naudoti nereguliarius laiko vėlinimus, kurių paieškai pritaikytas skruzdžių kolonijos optimizavimo algoritmas. Suradus tinkamiausiais laiko vėlinimų reikšmes, pritaikyti modelį eilutės reikšmių prognozavimui. Sudarytas prognozavimo modelis turi prognozuoti chaotines laiko eilutes.

Darbe siūlomo modelio įvertinimui palyginti su kitais laiko eilučių prognozavimo metodais naudojantis prognozės paklaidų metrikomis. Darbe tikrinama hipotezė, jog mūsų siūlomas modelis geba geriau prognozuoti įvairias laiko eilutes lyginant su kitais metodais bei kitų autorių tyrimais.

Šio darbo tema buvo paruoštas straipsnis tema „Electrocardiography time series forecasting and optimization using ant colony optimization algorithm“ recenzuojame tarptautiniame žurnale „Journal of Mathematical Models in Engineering“ ir skaityti pranešimai konferencijose „Matematika ir matematikos dėstymas - 2016“ bei „Matematika ir gamtos mokslai: teorija ir taikymai“.

1. LITERATŪROS APŽVALGA

Laiko eilučių prognozavimas yra labai svarbus uždavinys dėl didžiulio šių laikų pritaikomumo. Prognozavimo uždavinius galima spręsti su daugybe metodų – neuroninių tinklų, autoregresinių modelių, neraiškios logikos tinklų, eksponentinio glodinimo, stochastinių diferencialinių lygčių ar slenkančio vidurkio metodais. Nors yra labai daug efektyviai veikiančių laiko eilučių prognozavimo metodų, tačiau geriausio prognozavimo metodo nėra ir visi metodai turi savų privalumų ir trūkumų.[1, 2, 3]

Vienas iš dažniausiai naudojamų laiko eilučių prognozavimo metodų yra autoregresinis integruotas slenkančio vidurkio modelis, dar kitaip žinomas kaip *ARIMA(p,d,q)*. Jį pirmą kartą pristatė Box ir Jenkins 1967 metais sujungus autoregresijos (*AR(p)*) ir slenkančio vidurkio (*MA(q)*) modelius. *d* – modelio integruotumas, nes ARIMA modelis turi būti stacionarus procesas, kur *d* eilės pokyčiai yra stacionarūs procesai, o *d-1* eilės pokyčiai nėra stacionarūs. *ARIMA(p,d,q,)* modelis apibrėžiamas lygybe

$$P(L)(1-L)^d\widehat{\xi_t} = Q(L)\epsilon_t, \qquad (1)$$

čia *P(z)* ir *Q(z)* yra atitinkami p ir q eilės autoregresiniai *AR(p)* ir slenkančio vidurkio *MA(q)* eilės polinomai, o ϵ_t – balto triukšmo procesas.[1, 2, 7]

Dirbtiniai neuroniniai tinklai (ANN) taip pat yra naudojami laiko eilučių prognozei. Nors ARIMA modeliai naudojami tik prognozavimui, ANN metodai yra sėkmingai taikomi ir kitose srityse kaip klasterizavimas ar mašinų apmokymas. Taip pat, ANN modeliai dažnu atveju parodo geresnius rezultatus nei kiti standartiniai laiko eilučių prognozavimo metodai. Tokiu atveju uždaviniai dažniausiai pasižymi papildomu triukšmu bei netiesiškumu. Toks modelio efektyvumas argumentuojamas, jog neuronų tinklai imituoja gyvų organizmų biologinę nervų sistemą, kuri pasižymimi asociatyvine atmintimi, apsimokymu bei adaptyviu duomenų valdymu. ANN metodai pirmiausia apsimoko iš turimų duomenų, bando nustatyti funkcines priklausomybes tarp duomenų ir vėliau prognozuoja jomis besivadovaudama. Dirbtinių neuroninių tinklų metodai yra apmokomi kol apmokymo paklaidos mažėja arba kol pasiekia apmokymo paklaidų tikslą.[4, 8, 9]

Realaus pasaulio laiko eilučių modeliai dažniausiai pasižymi nenuspėjamumu ir permainingumu, nes eilučių kintamųjų skaičiai nėra žinomi. Šiose dinaminėse sistemose, pavyzdžiui degalų sąnaudų ar elektrokardiogramos signalo sistemose, sudėtinga atlikti prognozes, o jų paklaidos dažnu atveju būna didžiulės. Vienas efektyviausių metodu tokių laiko eilučių analizei yra metodas, kurio pagalba rekonstruojamos laiko eilutės sistemos į laiko vėlinimų erdvę. Šiame metode naudojantis laiko eilutės duomenimis nustatomos pilnos sistemos dinaminės charakteristikos. Sistemos rekonstravimo metodu sukuriama *d-matė* laiko vėlinimų

erdvė, kurioje d – laiko vėlinimų erdvės dimensija, o τ – laiko vėlinimai. Šių parametrų nustatymui buvo pasiūlyta daug metodų: klaidingo artimiausio kaimyno metodas, Liapunovo eksponenčių metodas, apibendrintos dimensijos metodas, taškų sąrašų metodas, Cao metodas ar tarpusavio informacijos algoritmas. Taip pat, literatūroje ir straipsniuose yra pateikta daugybė rekonstravimo pavyzdžių su reguliariais laiko vėlinimais, tačiau naudojantis nereguliariais laiko vėlinimais gaunamas geresnis rekonstravimo atsakymas.[10, 11, 12, 13]

Kombinatorinio optimizavimo uždaviniai yra dažnai taikomi įvairiose srityse dėl greito beveik optimalaus rezultato radimo. Evoliuciniai algoritmai yra dažniausiai naudojami laiko eilučių rekonstravimui dėl galimybės vis gerinti ir adaptuoti paieškos sistemą. Dažnu atveju yra pasirenkamas genetinis algoritmas, tačiau straipsniuose galima rasti ir kitų evoliucinių algoritmų pritaikytų rekonstravimo uždaviniui spręsti, pavyzdžiui dalelių spiečiaus ar skruzdžių kolonijos optimizavimo algoritmus. Šių kombinatorinio optimizavimo algoritmų tikslo funkcijos būna labai įvairios: didžiausios Liapunovo eksponenčių reikšmė, minimalių kaimynų atstumų paieška, minimalių aprašymų ilgio (MDL) reikšmė ir kiti metodai.[4, 13, 14, 15]

Meie Shen, Wei-neng Chen ir kiti bendraautoriai savo darbe nagrinėja optimalaus fazinės erdvės rekonstravimo dimensijos ir nereguliarių laiko vėlinimų paiešką naudojantis skruzdžių kolonijos optimizavimo algoritmu. Straipsnyje skruzdės ieško abiejų parametrų – dimensijos ir laiko vėlinimų. Tikslo funkcijai ir kokybės įvertinimui yra naudojami klaidingų artimiausių kaimynų, vidutinio keitimosi informacijos ir MDL metodai. Būtina pabrėžti, jog šiame darbe kiekvienos iteracijos metu pradžioj pasirenkama norima dimensija, o tik vėliau laiko vėlinimai. Darbo rezultatai buvo įvertinti naudojantis tikslo funkcijų reikšmėmis lyginant su genetinio algoritmo rezultatais. Išvadose pateikta, kad skruzdžių kolonijos algoritmas suteikia lankstų, mažesnių parametrų reikalaujantį ir greitą būdą, norint įvertinti dimensiją ir laiko vėlinimus, tačiau genetinis algoritmas duoda geresnius optimizavimo rezultatus.[4]

K. Mezeiová ir A. Krakovská nagrinėjo klaidingų artimiausių kaimynų (FNN) metodą ir palygino jį su simbolinių stebėjimų laipsnių metodu (angl. Symbolic observability degrees). Metodų rezultatai lyginami rekonstruojant Rössler, Lorenz, Sprott F ir hyperchaotinę Rössler sistemas. Autoriai atkreipia dėmesį, jog FNN rezultatai ne visur sutapo su simbolinių stebėjimų laipsnių metodu, tačiau FNN rezultatą nežymiai įtakoja pradiniai parametrai. Išvadose teigiama, kad klaidingų artimiausių kaimynų metodas yra efektyvus algoritmas norint rekonstruoti fazinę erdvę.[16]

Xiaoxiao Cui ir Mingyan Jiang savo darbe prognozavo chaotines laiko eilutes su dalelių spiečiaus algoritmu (BPSO) ir klaidingų artimiausių kaimynų metodu. Straipsnyje BPSO yra pritaikytas fazinės erdvės rekonstrukcijos parametrų paieškai, kai tikslo funkcija yra klaidingų artimiausių kaimynų algoritmas. Toliau prognozei naudotas prisitaikantis svertinis branduolių

sintezės algoritmas (angl. adaptive weight fusion algorithm). Tyrimų rezultatuose paminėta, kad algoritmas nesugebėjo prognozuoti Lorenz laiko eilutės, tačiau puikiai prognozavo tradicines tiesines laiko eilutes.[17]

Jianping Wang ir kiti kolegos savo straipsnyje prognozavo saulės radiacijos stiprumą. Autoriai negalėjo naudotis tradiciniais prognozavimo metodais, kadangi signalas yra stiprus stochastinis procesas, pasižymintis netiesiškumu ir nestacionarumu. Tyrime laiko eilutės fazines erdvės rekonstravimas atliktas autokoreliacijos (angl. Autocorrelation) ir keitimosi informacijos (angl. Mutual information) algoritmais, o prognozė atlikta bangelių neuroninių tinklų modeliu (angl. Wavelet neural network model). Prognozės paklaidos buvo žemos ir išvadose autoriai pabrežia, jog siūlomas neuroninis tinklas yra greitesnis ir atlieka tikslų prognozavimą nei kiti tradiciniai algoritmai.[18]

Taip pat Khalil Benmouiza ir Ali Cheknane nagirnėjo saulės radiacijos laiko eilutę. Savo darbe jie parametrų paieškai taiko klaidingų artimiausių kaimynų ir keitimosi informacijos algoritmus, tačiau toliau duomenys buvo klasterizuojami k-vidurkių metodu ir klasteriai prognozuojami netiesiniais autoregresiniais neuroniniais tinklais (NAR). Vėliau gautų klasterių prognozė naudojama saulės radiacijos laiko eilutės prognozei. Išvadose autoriai teigia, jog klasterizavimas padeda neuroniniams tinklams interpretuoti laiko eilučių tendencijas ir, to pasėkoje, gaunama tiksli laiko eilutės prognozė.[10]

Li Han ir kiti autoriai prognozavo vėjo energiją, kuri pasižymi stochastinėmis ir netiesinėmis charakteristikomis. Fazinės erdvės rekonstravimo dimensijai ir laiko vėlinimų paieškai buvo pritaikyta pagrindinių komponenčių analizė, prognozei – išteklių pasiskirstymo neuroninis tinklas (angl. Resource allocating neural network). Modelio adekvatumas yra įvertinamas naudojantis Mackey-Glass chaotinę laiko eilutę, o vėjo energijos prognozė palyginama su kitais 3 metodais. Išvadose rašoma, kad pagrindinių komponenčių analizė gali susidoroti su dimensijos ir laiko vėlinimų atranka, tačiau neuroninių tinklų prognozė neparodė mažiausių prognozavimo paklaidų lyginant su kitais prognozavimo modeliais.[11]

Atlikus literatūros apžvalgą pastebėta, kad įvairiuose moksliniuose straipsniuose galima rasti gaunamų gerų prognozavimo rezultatų naudojantis fazinių erdvių rekonstravimo metodu, tačiau vis dar dažnai yra taikomi reguliarių laiko vėlinimų rekonstravimo algoritmai. Literatūroje minima, jog nereguliarūs laiko vėlinimai pateikia geresnius rezultatus, nes laiko eilutėje įžvelgiamos skirtingos tendencijos. Bet optimalių nereguliarių laiko vėlinimų paieška yra sudėtingas uždavinys ir dažniausiai reikalauja daug laiko.

2. METODOLOGINĖ DALIS

2.1. Chaotinės laiko eilutės ir jų prognozavimas

Laiko eilutė yra laike kintantis duomenų taškų rinkinys, kuriame matavimai dažniausiai atliekami pastoviais laiko intervalais. Šie rinkiniai yra naudojami duomenų tendencijų analizei bei prognozavimo problemoms spręsti. Laiko eilučių metodų tikslas yra nustatyti laiko eilutės dinamiką naudojantis turima informacija, ir vėliau pritaikyti tas pačias tendencijas ateityje. Tokios problemos dažniausiai yra sunkiai išsprendžiamos, nes realaus pasaulio laiko eilutės yra labai kompleksiškos. Laiko eilutė apibrėžiama formule

$$X(t) = \{X(t) \in R \mid t = 1, 2, 3 \dots N\}; \quad (2)$$

čia t yra laiko indeksas ir N yra stebėjimų skaičius. Laiko eilučių duomenis galima analizuoti naudojantis įvairiais modeliais kaip eksponentinio glodinimo modelis ar ARIMA, tačiau dažniausiai šie modeliai nepasižymi gera kompleksiškų laiko eilučių prognoze. To pasėkoje, yra svarbu nagrinėti šias kompleksiškas eilutes, kurios dar kitaip yra vadinamos chaotinėmis laiko eilutėmis.[2, 7]

Chaosas yra kompleksinis dinaminis reiškinys, egzistuojantis įvairiose sistemose, pavyzdžiui, signaluose, atmosferoje, ekonomikoje ar biologijoje. Daugeliu realaus pasaulio atvejų laiko eilutės pasižymi netiesiškumu bei turi chaoso fenomeną. Taigi chaotinių laiko eilučių prognozavimas tapo iššūkiu daugelyje mokslo sričių.[1, 4]

2.2. Chaotinės laiko eilutės rekonstravimo problema

Chaoso identifikavimo technikai yra būtinas laiko eilutės rekonstravimas fazinėje erdvėje. Chaotinės dinaminės sistemos išdėlioja trajektorijas jų fazių erdvėje, kurios ir sudaro atraktorių. Toks rekonstravimas naudojamas tam, kad vieno kintamojo eilutę būtų galima pavaizduoti daugiamatėje erdvėje ir būtų aiškus jo dinaminis išsidėstymas. Tad fazinių erdvių rekonstrukcija yra pirmas žingsnis netiesinių laiko eilučių analizei ir prognozavimui. Packard pirmasis 1980 metais atsakė į klausimą, kaip rekonstruoti fazinę erdvę. Jis parodė, kad įmanoma rekonstruoti didelių dimensijų fazinių erdvių vektorius naudojant laiko vėlinimą. Pagal pateiktą Packard teoremą skaliarinės laiko eilutės dinamika yra integruojama į *d-osios* dimensijos vėlinimų erdvę, kurioje yra parenkami laiko vėlinimai τ tarp laiko eilutės taškų. Atsižvelgiant į laiko vėlinimo techniką, vėlinimų erdvė yra išreikšta:

$$x(t) = \big(x(t), x(t+\tau), x(t+2\tau), \dots, x(t-(d-1)\tau)\big); \quad (3)$$

čia τ yra laiko vėlinimas, o d yra rekonstravimo dimensija. Laiko vėlinimo ir rekonstravimo dimensija yra svarbūs parametrai rekonstruojant erdvę. Taigi, *Packard* pasiūlė metodą, o *Takens* pateikė įtvirtinimo teoremą.[4, 5, 17]

Įtvirtinimo teorema (angl. embedding theorem) teigia, kad fizinės sistemos dinamika yra registruojama kaip laiko eilutė $x(t_1), x(t_2), x(t_3), \dots, x(t_k), \dots, x(t_n)$ gali būti gaunama arba integruota į m dimensijų fazinę erdvę. Jei atraktoriaus dimensija yra d, o duotas bet koks laiko vėlinimas yra τ_0, tai yra reikalingas $m \geq 2d + 1$ rekonstravimo dimensija (angl. embedding dimension). *Whitney* (1936) pateikė rekonstravimo dimensijos ir rekonstravimo problemą, kuri teigė, jog d-dimensinė erdvė gali būti patalpinta į $2d + 1$ koordinačių sistemą. [4, 14]

Šis metodas dar yra vadinamas reguliariu rekonstravimu ir yra plačiai taikomas atraktorių rekonstravimo problemai sprendžiant modelio atpažinimo uždavinius. Tiesą sakant, renkantis τ, trumpas laiko vėlinimas gali būti optimalus aukšto dažnio dinamikos rekonstravimui, o ilgas laiko vėlinimas gali geriau rekonstruoti žemo dažnio dinamikas. Realaus pasaulio laiko eilutės dažnai pasižymi nereguliariais dažniais, t.y. laiko eilutėje pastebima tiek žemo, tiek aukšto dažnio dinamikos. Ši dinamikos problema yra labai aktuali sudėtingų ir nestacionarių signalų rekonstravimui, pavyzdžiui, biologinių procesų laiko eilutėms. Turint tai omenyje, nereguliarus laiko rekonstravimas gali susidoroti su tokiomis laiko eilutės dinamikomis, pakeičiant vieną laiko vėlinimą į laiko vėlinimų vektorių τ_j. Tačiau laiko vėlinimų vektoriaus paieška gali būti NP-sunki problema rekonstruojant didelės dimensijos d laiko eilutes su galimai didžiuliais laiko vėlinimais, t.y. geriausio sprendinio paieška užtruktų ilgai. [5, 14, 19]

Tarkime turime laiko eilutę $x(t)$, rekonstravimo dimensiją m ir laiko vėlinimų vektorių τ_j. Tuomet gauname fazinės erdvės trajektorijas:

$$Y = \begin{bmatrix} x(1) & x(1+\tau_1) & \dots & x(1+\tau_{m-1}) \\ x(2) & x(2+\tau_1) & \dots & x(2+\tau_{m-1}) \\ \vdots & \vdots & \ddots & \vdots \\ x(i) & x(2+\tau_1) & \dots & x(2+\tau_{m-1}) \\ \vdots & \vdots & \ddots & \vdots \\ x(M) & x(M+\tau_1) & \dots & x(M+\tau_{m-1}) \end{bmatrix}; \quad (4)$$

čia M yra fazinės erdvės taškai po fazinės erdvės rekonstrukcijos su sąlyga $M = N - \tau_{m-1}$. Y yra pateikiamas kaip $Y = [x(1), x(2), \dots, x(M)]^T$, kur $x(i)$ rekonstruoti vektoriai.

Optimaliai rekonstruotas atraktorius su tinkamai parinktais laiko vėlinimais gali atskleisti naujus dinaminės sistemos dėsningumus, kurių išsamesnė analizė padėtų tobulinant sistemos matematinius modelius bei atliekant procesų prognozavimą. [8, 14]

2.3. Rekonstravimo dimensijų kiekio pasirinkimas

Fazinių erdvių rekonstravimui privaloma pasirinkti tinkamą dimensiją. Kennel ir kiti pasiūlė klaidingų artimiausių kaimynų metodą norint nustatyti minimalią dimensiją m fazinės erdvės rekonstravimui. Šis metodas yra pagrįstas prielaida, kad du taškai, kurie yra arti vienas nuo kito turėtų išlikti artimi didėjant dimensijai. Tarkime turime dimensiją d ir atstumą iki artimiausio r-tojo kaimyno $y^r(n)$. Apskaičiuojame Euklidinį atstumo kvardratą tarp $y(n)$ ir $y^r(n)$:

$$R_d^2(n,r) = \sum_{j=0}^{d-1}[x(n+jT) - x^r(n+jT)]^2. \quad (5)$$

Skaičiuojant $d+1$ dimensiją, apskaičiuojamas atstumas tarp $y(n)$ ir to pačio artimiausio kaimyno $y^r(n)$ naudojantis formule:

$$R_{d+1}^2(n,r) = R_d^2(n,r) + [x(n+dT) - x^r(n+dT)]^2. \quad (6)$$

Šiame metode yra tiriamas atstumo skirtumas, kai pereinama iš dimensijos d į $d+1$. Atstumo skirtumas gali būti apskaičiuotas pasinaudojus aukščiau pateiktomis formulėmis (5,6). Klaidingas kaimynas priskiriamas bet kuriam kaimynui naudodamasis formule:

$$\sqrt[2]{\frac{R_{d+1}^2(n,r) - R_d^2(n,r)}{R_d^2(n,r)}} = \frac{|x(n+dT) - x^r(n+dT)|}{R_d^2(n,r)} > R_{tol}; \quad (7)$$

kai rekomenduoja $R_{tol} = 15$. Antras kriterijus apriboja duomenų rinkinio dydžio klausimą. Jei $y(n)$ kaimynas nėra arti ir jis yra „klaidingas artimiausias kaimynas, tai $R_{d+1}(n) \approx 2R_A$.

$$R_A = \frac{1}{N}\sum_{n=1}^{N}[x(n) - \bar{x}]^2. \quad (8)$$

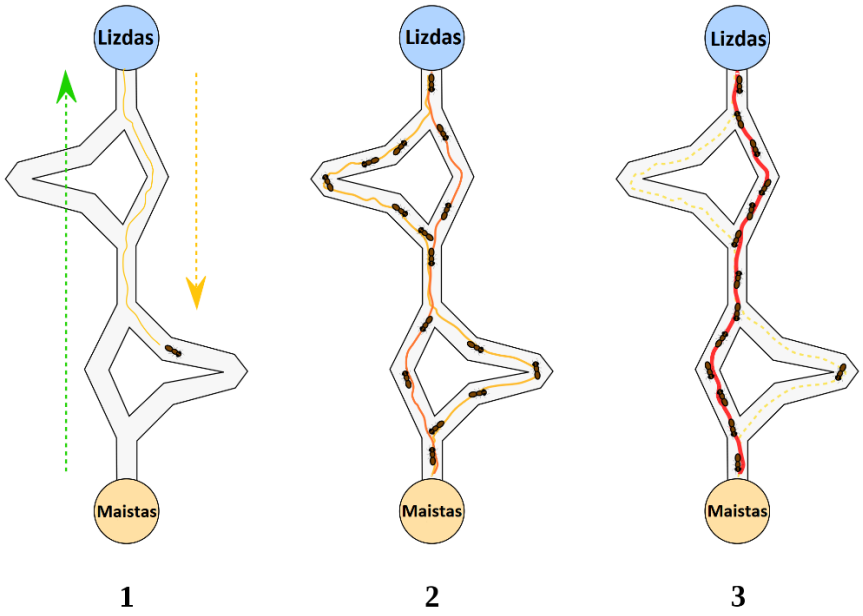
Antro kriterijaus formulė:

$$\frac{R_{d+1}(n)}{R_A} > A_{tol}; \quad (9)$$

kur rekomenduojama $A_{tol} = 2$.[4, 10, 16, 20]

2.4. Optimizavimo algoritmas rekonstravimo uždaviniui spręsti

2.4.1. Skruzdžių kolonijos algoritmas



1 pav. Skruzdžių kolonijos veikimo principas

Kombinatorinio optimizavimo metodai yra labai aktuali matematikos mokslo sritis dėl didelio praktinio pritaikomumo. Daug algoritmų buvo sukurta stengiantis išspręsti įvarius kombinatorinius uždavinius. Šie algoritmai gali būti skirstomi į dvejas rūšis: tikslius ir apytikslius algoritmus. Tikslūs algoritmai visada randa geriausią sprendinį, o apytiksliai algoritmai randa optimalų sprendinį per ribotą laiką. Visgi, kombinatoriniai uždaviniai yra NP-sunkūs ir todėl tikslūs algoritmai yra neoptimalūs laiko atžvilgiu. O apytiksliai algoritmai suranda optimaliausius sprendinius per trumpą laiką, t.y. aukojama geriausio atsakymo paieška vardan sutaupyto laiko.

Skruzdžių kolonijos optimizavimas (ACO) (angl. ant colony optimization) yra vienas iš naujausių metodų, skirtų apytiksliam optimizavimui. Skruzdžių kolonijų charakteristika išnaudojama ACO algoritmu, siekiant išspręsti kombinatorinius uždavinius. Marco Dorigo ir kolegos pristatė ACO algoritmą 1992-aisiais metais. Šio algoritmo sukūrimas buvo įkvėptas skruzdžių kolonijų stebėjimui. Skruzdės yra bendruomeniniai vabzdžiai. Jie gyvena kolonijomis ir jų elgesiai paremti kolonijos išgyvenimo tikslui, o ne į individų išlikimą. Šis algoritmas buvo atrastas remiantis skruzdžių maisto paieškos procesu. Šio proceso esmė yra netiesioginis bendravimas tarp skruzdėlių naudojant cheminius feromonų takus, kurie leidžia jiems rasti trumpus kelius tarp jų kolonijos ir maisto šaltinių. Pradžioj skruzdėlės ieško trumpiausių kelių tarp maisto šaltinių ir jų lizdo. Ieškodami maisto, skruzdės iš pradžių atsitiktine tvarka ištiria plotą aplink savo lizdą ir vaikščiodamos palieka feromonų cheminį taką ant žemės, o vėliau

skruzdėlės gali užuosti feromonus. Renkantis savo kelią, jos linkusios pasirinkti stiprių feromonų koncentracija pažymėtus takus. Kai tik skruzdės randa maisto šaltinį, įvertina maisto kiekį, kokybę ir grįžta atgal į lizdą. grįžimo metu, skruzdės paliktas feromonų kiekis gali priklausyti nuo maisto kiekio ir kokybės. Vėliau feromonų takai ves kitas skruzdėles prie maisto šaltinio. Iliustruotas pavyzdys pateiktas 1 paveikslėlyje.

Šiame darbe skruzdžių kolonijos algoritmas bus naudojamas laiko vėlinimų reikšmių paieškai. Tinkamai parinktos nereguliarių laiko vėlinimų aibės pagalba rekonstruojamo atraktoriaus savybės gali atskleisti tiriamos dinaminės sistemos dėsningumus fazinėje erdvėje. Šių dėsningumų analizė padėtų neuroniniui tinklui atlikti laiko eilutės prognozę.[4, 14]

2.4.2. Ruletės rato pasirinkimo algoritmas

Ruletės rato parinkimo algoritmas, dar vadinamas atsitiktinė atranka su pakeitimu, yra stochastinis algoritmas norint sudaryti atrinkimo schemą. James Baker pristatė šį algortimą 1987 metais. Mūsų tyrime jis bus naudojamas padėti skruzdėlėms pasirinkti laiko vėlinimus τ.

Tarkime skruzdėlės gali pasirinkti bet kokį laiko vėlinimą τ, kai $\tau \in \{1,2,\dots,Max\mathrm{T}\}$. Pirmiausiai pasirinkimo variantai gauna vienodas tikimybes būti pasirinktam, t.y. jei yra $Max\mathrm{T}$ vienodų pasirinkimo galimybių, visos pasirinkimo galimybės turi tikimybę $p = \frac{1}{Max\mathrm{T}}$ būti pasirinktas. Tada atsitiktinai pasirenkamas vienas iš laiko vėlinimų τ_j. Jei optimizavimo algoritmas pateikia geresnį tikslo funkcijos rezultatą su reikšme τ_j nei buvo, skruzdėlė palieka savo feromonus ant būtent šio laiko vėlinimų reikšmės $P(\tau_j) = P(\tau_j) + ph$, kur ph yra skruzdėlės feromonai, ir padidina pasirinkimo tikimybę vėliau.

Tolimesnėse algoritmo iteracijose $P(\tau_j)$ yra įvertinama remiantis skruzdžių feromonais ir euristinėmis reikšmėmis:

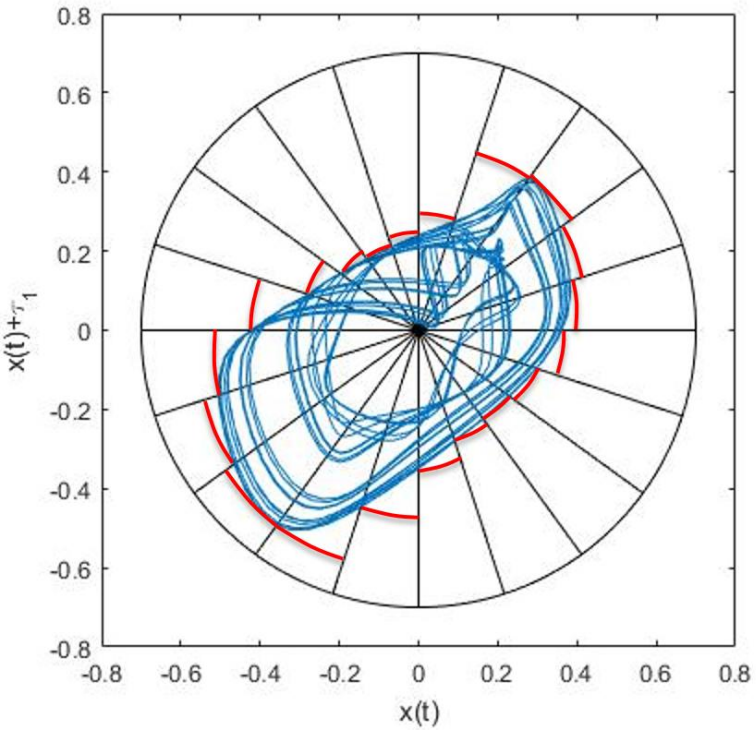
$$P(\tau_i) = \begin{cases} P(\tau_i) = \frac{P(\tau_i)}{\sum_{k=1}^{Max\mathrm{T}} P(\tau_k)}, \ kai\ \tau \in \{1,2,\dots,Max\mathrm{T}\} \\ 0, \ kitais\ atvėjais \end{cases}. \quad (10)$$

Skruzdėlių pasirinkimo galimybės yra atnaujinamos kas kartą radus geresnę tikslo funkciją. Didžiulis skruzdėlių feromonų kiekis tam tikroje laiko vėlinimų reikšmėje τ_j gerokai padidina tikimybę pasirinkti šį laiko vėlinimą, tačiau tikimybė pasirinkti kitus laiko vėlinimus niekada neišnyksta. To pasėkoje, skruzdėlės turi tikimybę pasirinkti kitus laiko vėlinimus ir ieškoti geresnės tikslo funkcijos rezultato.[4, 14]

2.4.3. Skruzdžių kolonijos optimizavimo algoritmo tikslo funkcija

Skruzdžių kolonijos optimizavimo algoritmo pagalba randamos optimaliausias laiko vėlinimų reikšmes τ_i, kai $i \in 1,2,\dots,d-1$. Norint surasti šias reikšmes bus ieškomas didžiausias vidutinis atraktorių plotas. Kiekvieno atraktoriaus ploto skaičiavimui išdalinome atraktoriaus

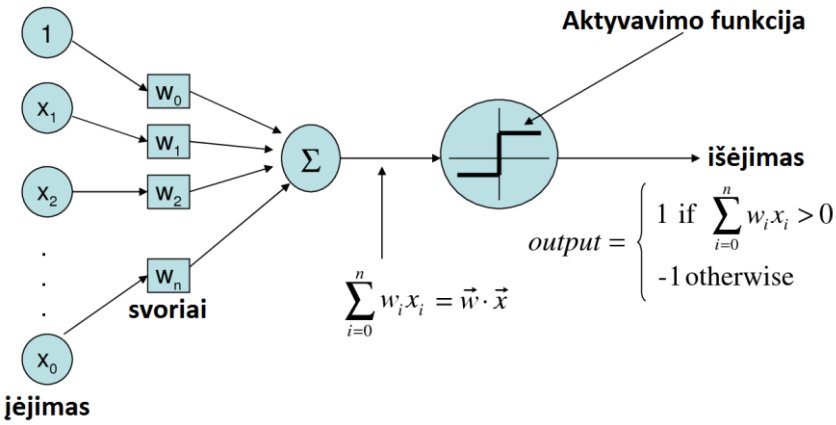
plokštumą į n skritulio išpjovų (žr. 2 pav), suskaičiavome kiekvienos išpjovos užimamą plotą, kai r yra didžiausias taško nuotolis nuo atraktoriaus centro tam tikroje išpjovoje, ir sudėjome visų išpjovų plotus.



2 pav. Atraktorius skritulyje su 20 išpjovų

2.5. Dirbtiniai neuroniniai tinklai

Dirbtiniai neuroniniai tinklai (ANN) yra sistema, kuri remiasi biologinių neuronų tinklų operacijomis ir todėl gali būti apibrėžiami kaip biologinių neuroninių sistemų emuliacijos. ANN yra kompiuterinės sistemos, skirtos sukurti arba bent jau imituoti protingą elgesį. Skirtingai nuo klasikinių dirbtinių intelekto (AI) sistemų, kurios yra skirtos tiesiogiai priimti racionalius ir loginius sprendimus, neuroniniais tinklais siekiama atkurti mechanizmus, kurie apmokyti galėtų nuolat spręsti naujai besiformuojančias sudėtingas sistemas. Neuroniniai tinklai buvo sėkmingai sukurti ir pritaikyti sprendžiant modelių atpažinimo, pajėgumų planavimo, robotikos problemas. Informacinių technologijų mokslo srityje neuroniniai tinklai buvo sėkmingai pritaikyti prognozavimo, duomenų analizės ar duomenų gavybos uždaviniuose.[9, 21]



3 pav. Perceptrono sandaros schema

Viena iš ANN sistemų yra pagrįsta moduliu, vadinamu perceptronu (3 pav.). Į perceptroną paduodamas realių reikšmių įėjimo vektorius. Perceptronas apskaičiuoja tiesinę šių įėjimų kombinaciją, tada išveda 1, jei rezultatas yra didesnis nei nustatytas slenkstis, ir išveda -1 priešingu atveju. Apibrėžiant tiksliau, įėjimams $x_1, \dots, x_n$, išėjimas $f(x_1, \dots, x_n)$ bus apskaičiuojamas taip

$$o(x_1, \dots, x_n) = \begin{cases} 1 & jei\ \ w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n > 0 \\ -1 & priešingu\ atveju \end{cases}; \tag{11}$$

čia kiekvienas w_i yra realaus tipo konstanta arba svoris, kuris apibrėžia įėjimo x_i indėlį perceptrono išėjimui. Dydis (w_0) nusako slenkstį, kurį svorinė įėjimų kombinacija $w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$ turi viršyti norint, kad perceptronas išėjime duotų 1.

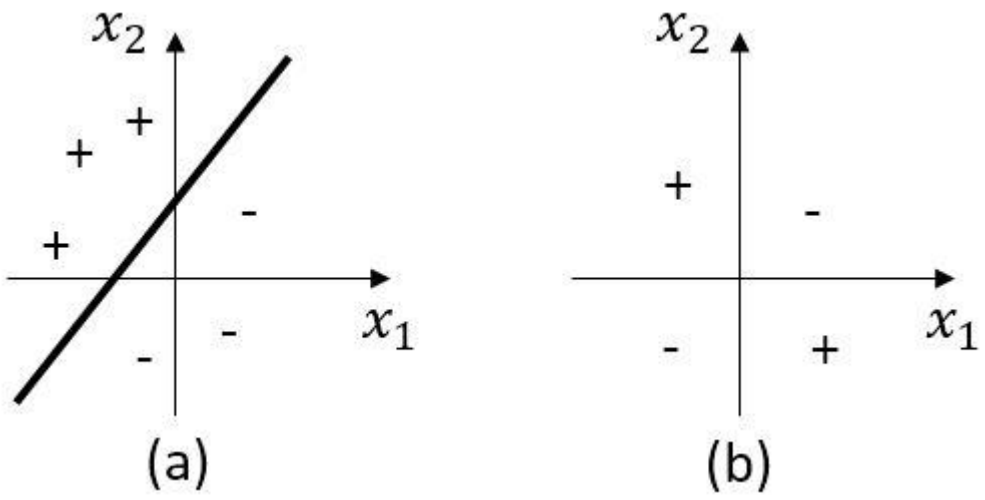
Norėdami supaprastinti išraišką įsivaizduojame, kad lygtyje (11) $x_0 = 1$, tada galime užrašyti $\sum_{i=0}^{n} w_ix_i > 0$ arba vektoriniame pavidale $\vec{w} \times \vec{x} > 0$. Trumpiau, perceptrono funkcija kartais užrašoma ir kaip

$$o(\vec{x}) = r(\vec{w} \times \vec{x}),\ kur\ \ r(y) = \begin{cases} 1 & jei\ \ y > 0 \\ -1 & priešingu\ atveju \end{cases}. \tag{12}$$

Mokydamasis perceptronas turi parinkti reikšmes svoriams $w_0, \dots, w_n$. Hipotezių erdvė H yra visų įmanomų realaus tipo svorių vektorių aibė

$$H = \{\vec{w} | \vec{w} \in R^{n+1}\}. \tag{13}$$

Į perceptroną galime žvelgti kaip į hiperplokštuminį sprendimų priėmimo paviršių n-matėje erdvėje. Perceptronas išėjime išveda 1 egzemplioriams gulintiems vienoje hiperplokštumos pusėje ir -1 egzemplioriams kitoje hiperplokštumos pusėje (4 pav.).[9, 22]



3 pav. Tiesiškai perceptronu atskiriamo (a) ir tiesiškai neatskiriamo (b) duomenų rinkinio pavyzdžiai

2.5.1. Perceptrono mokymosi taisyklė

Perceptrono svoriai atnaujinami kiekviename žingsnyje, pagal tokią taisyklę

$$w_i \leftarrow w_i + \Delta w_i,\ kur\ \Delta w_i = \eta(t - o)x_i; \tag{14}$$

čia t – nagrinėjamo mokymo imties egzemplioriaus išėjimas pagal tikslo funkciją, f – perceptrono apskaičiuotas išėjimas, η – teigiama konstanta, vadinama mokymosi greičiu. Mokymosi greitis nusako kokiu lygiu atnaujinami svoriai. Dažniausiai η parenkama maža vertė, kartais daromas konstantos mažinimas didėjant iteracijų skaičiui. Ši taisyklė konverguoja tiesiškiems duomenų rinkiniams. Konvergavimas tiesiškai neatskiriamų duomenų rinkinių atveju nėra užtikrintas.[9, 21]

2.5.2. Atgalinio sklidimo algoritmas

Atgalinio sklidimo algoritmas pateiktam ANN tinklui su fiksuotu modulių ir sąryšių skaičiumi išmoksta daugiasluoksnio tinklo svorius. Kvadratinė paklaida tarp tinklo išėjimo ir tikslo funkcijos reikšmės minimizuojama gradiento mažėjimo būdu. Aukščiau pateiktą kvadratinės paklaidos skaičiavimo formulę perrašome taip, kad apimtų visus tinklo išėjimo modulius

$$E(\vec{w}) = \frac{1}{2}\sum_{d\in D}\sum_{k\in išėjimai}(t_{kd} - o_{kd})^2; \tag{15}$$

čia *išėjimai* –visų tinklo išėjimo modulių aibė; t_{kd} ir f_{kd} – tikslo ir išėjimo reikšmės, susietos su k-tuoju išėjimo moduliu ir mokymo pavyzdžiu d.

Toliau pateikiamas algoritmo variantas ANN tiesioginio apėjimo (angl. feed-forward) tinklui su dvejais sigmoidinių modulių sluoksniais.[9, 21]

2.5.3. Dirbtinio neuroninio tinklo metodo eiga

- Sukuriamas tiesioginio apėjimo tinklas su n_{in} įėjimų moduliais, n_{hidden} paslėptais moduliais ir n_{out} išėjimų moduliais.
- Inicijuojami visi tinklo svoriai. Atsitiktinai parenkamos reikšmės iš pakankamai mažų reikšmių intervalo.
- Kol bus įvykdyta sustojimo sąlyga atliekame žingsnius:

1. Tinklui paskleidžiami įėjime turimi duomenys – tinklui paduodame egzempliorių $\vec{x}$, apskaičiuojame kiekvieno tinklo modulio u išėjimą o_u.
2. Atgalios tinklui paskleidžiama paklaidas – Kiekvienam tinklo išėjimui k apskaičiuojama jo paklaidos apibrėžtis δ_k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k). \tag{16}$$

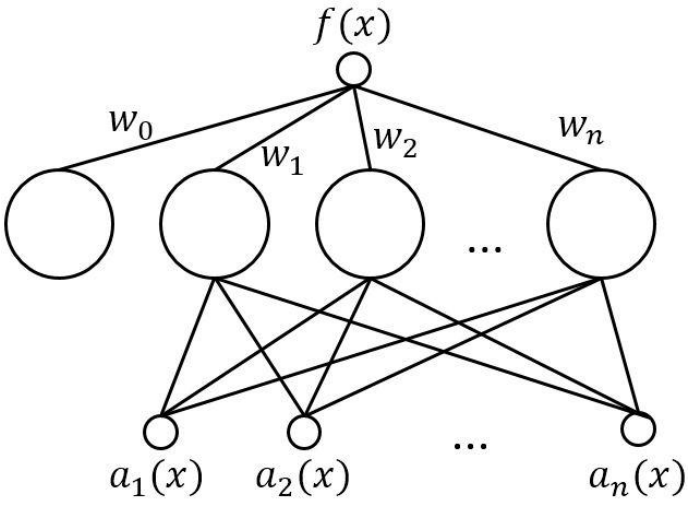
3. Kiekvienam paslėptajam tinklo moduliui h apskaičiuojama jo paklaidos apibrėžtis δ_h

$$\delta_h \leftarrow o_h(1 - o_k)\sum_{k \in išėjimai} w_{hk}\delta_k. \tag{17}$$

4. Atnaujinamas kiekvienas tinklo svoris w_{ji}[21, 22]

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji},\ kur\ \Delta w_{ji} = \eta\delta_j x_{ji}. \tag{18}$$

2.5.4. Radialinės bazės funkcijos neuroninis tinklas



5 pav. RBF tinklo schema

Radialinės bazės funkcijos neuroniniai tinklai (RBF) vadovaujasi daugiamačių interpoliacijos modelių metodika. RBF gali būti naudojami tiesiniams arba netiesiniams modeliams ir vieno arba kelių sluoksnių tinklui. Šių funkcijų aproksimacijai pritaikant ANN tinklams išmokstama hipotezė turi pavidalą

$$\hat{f}(x) = w_0 + \sum_{u=1}^{k} w_u K_u(d(x_u, x)); \quad (19)$$

čia kiekvienas x_u yra iš aibės X, $K_u(d(x_u, x))$ – branduolio funkcija mažėja, kai atstumas $d(x_u, x)$ didėja. k yra vartotojo apibrėžta konstanta, kuri nusako kiek branduolio funkcijų turi būti įtraukta į mokymąsi.

Nors $\hat{f}(x)$ yra globali $f(x)$ aproksimacija, kiekvieno $K_u(d(x_u, x))$ indėlis yra lokalizuotas taško x_u kaiminystės aplinkoje. Dažniausiai kiekviena iš $K_u(d(x_u, x))$ yra Gauso funkcija su vidurkiu x_u (centruota taške x_u) ir dispersija σ^2.

$$K_u\big(d(x_u, x)\big) = e^{\frac{1}{2\sigma_u^2} d^2(x_u, x)}. \quad (20)$$

19 lygtis aproksimuoja bet kurią funkciją su sąlygiškai maža paklaida, kai yra pateiktas pakankamai didelis Gauso branduolių skaičius k tokių, kad branduolio plotis σ^2 gali būti nustatomas pasirinktinai. 19 lygtį galime įsivaizduoti kaip dviejų sluoksnių tinklą, kuriame pirmasis modulių sluoksnis apskaičiuoja skirtingų $K_u(d(x_u, x))$ reikšmes, o antrasis apskaičiuoja tiesinę pirmojo sluoksnio reikšmių kombinaciją.

RBF tinklas mokomas dvejais etapais:

1. Nustatomas paslėptų modulių skaičius k, o kiekvienas paslėptas modulis u apibrėžiamas pasirenkant x_u ir σ_u^2 reikšmes, kurios apibrėžia jo branduolio funkciją $K_u(d(x_u, x))$.
2. Svoriai w_u mokomi maksimizuoti tinklo atitikimą mokymo duomenims. Tam naudojamas globalios klaidos kriterijus, apibrėžtas formule

$$E = \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2. \quad (21)$$

RBF tinklai yra globali tikslo funkcijos aproksimacija, apibrėžta daugelio lokalių branduolio funkcijų tiesine kombinacija. Pagrindinis RBF tinklų privalumas yra jų mokymosi efektyvumo pranašumas prieš tiesioginio apėjimo (feed-forward) tinklus, kurie mokomi atgalinio sklidimo algoritmu. Šis pranašumas pasiekiamas įėjimo ir išėjimo sluoksnių RBF tinkluose atskiro mokymo dėka.[9, 21, 22, 23, 24]

2.6. Prognozės paklaidų tikrinimas

Svarbus klausimas skaičiuojant prognozavimo metodus yra prognozavimo tikslumas, nes kuo mažesnės prognozavimo paklaidos, tuo prognozavimo patikimumas yra didesnis. Dažniausiai naudojamos paklaidos yra mažiausia vidutinė kvadratinė paklaida (Mean squared error (MSE), Root mean squared error (RMSE)), kurios yra matuojamos atsižvelgiant į paklaidų dispersiją ir standartinį nuokrypį. [8, 11]

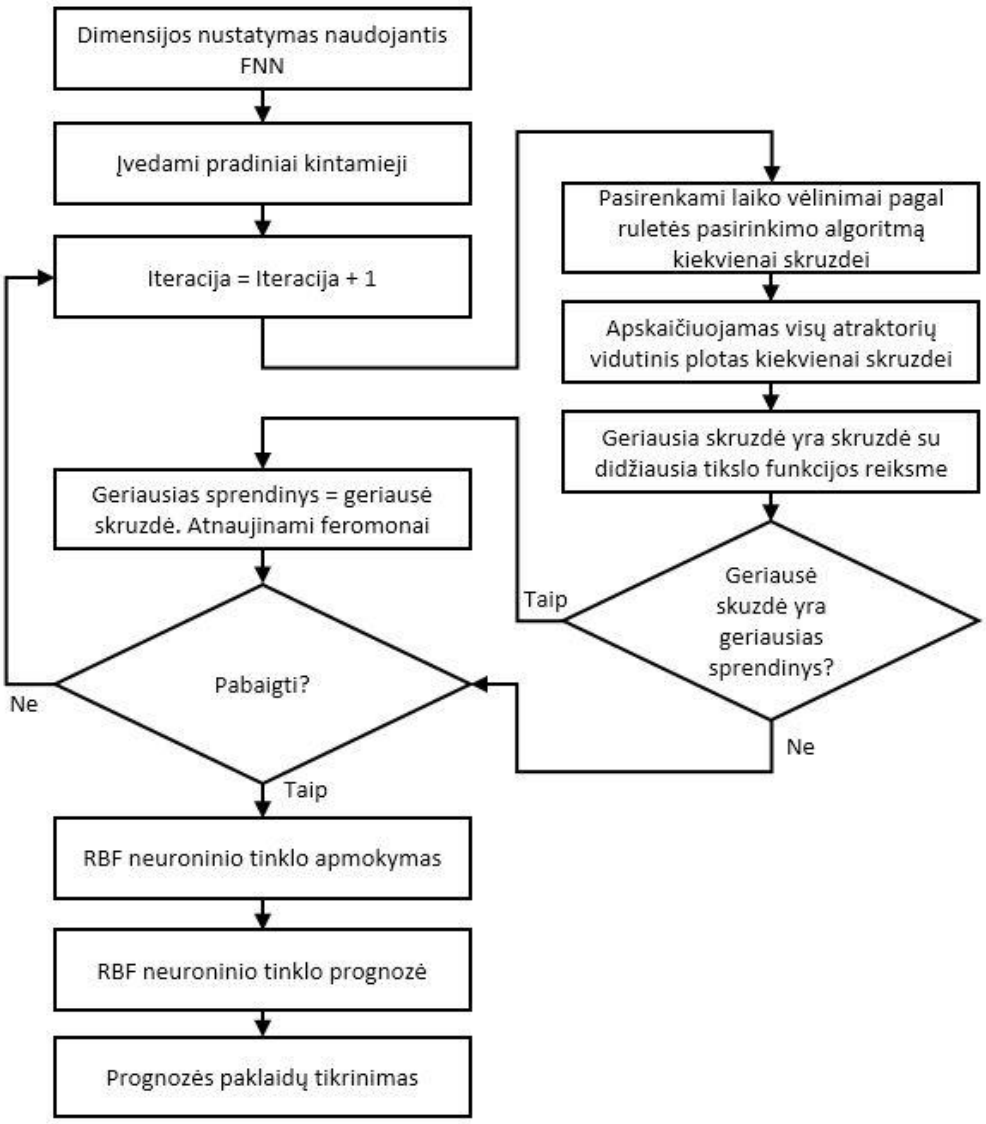
$$RMSE = \sqrt{\frac{\sum_{t=1}^{n} \widehat{x_t} - x_t}{n}} \quad (22)$$

čia $\widehat{x_t}$ ir x_t y yra laiko eilutės tikroji reikšmė ir prognozės reikšmė laiko momentu t. n yra prognozuojamų duomenų kiekis.

Kita tyrime naudojama paklaidų metrika yra vidutinė absoliučioji procentinė paklaida (angl. Mean Absolute Percentage Error):[11, 14]

$$MAPE = \frac{1}{N} \sum_{t=1}^{N} \frac{|\widehat{x_t} - x_t|}{x_t} \quad (23)$$

2.7. Modelio struktūra



6 pav. Modelio struktūra

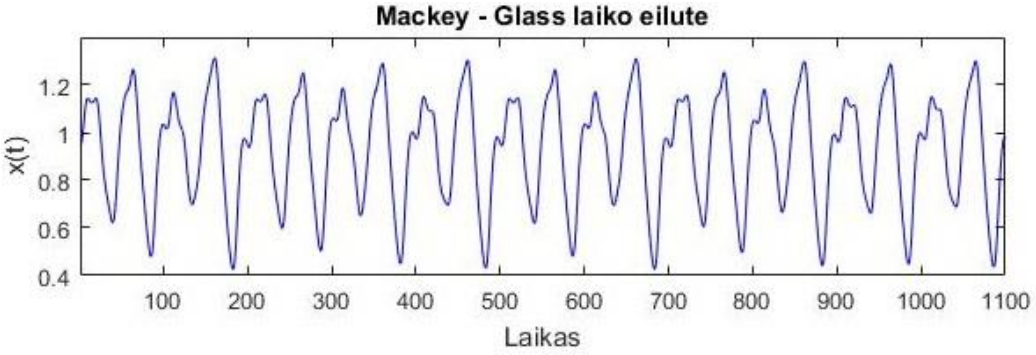
6 paveiksle yra pateikta modelio struktūra. Naudojantis šia struktūra yra prognozuojama kiekviena chaotinė laiko eilutė. Modelio eiga paeiliui:

1. Naudojantis klaidingų artimiausių kaimynų metodu yra nustatoma minimali fazinės erdvės *d* dimensija;
2. Modelyje nustatomi reikalingi kintamieji: skruzdžių kiekis, pradinis ir vienodas feromonų pasiskirstymas tarp skruzdžių, fazinės erdvės dimensija, maksimalus vėlinimų skaičius, iteracijų skaičius, maksimalus skruzdžių iteracijų skaičius;
3. Kiekviena skruzdė pasirenka *d-1* laiko vėlinimus τ naudojantis ruletės pasirinkimo metodu (laiko vėlinimų reikšmės rinkinyje negali kartotis);
4. Kiekvienai skruzdei yra skaičiuojama tikslo funkcija – vidutinis visų atraktorių plotas;
5. Tikrinama, ar kuri nors skruzdė surado didesnę tikslo funkcijos reikšmę. Jei surandama didesnė tikslo funkcijos reikšmė, atnaujinami skruzdžių feromonų takai ir išsaugoma geriausios skruzdės duomenys – laiko vėlinimai;
6. Tikrinama ar iteracijų kiekis pasiekė maksimalų skruzdžių iteracijų kiekį. Jei nepasiekė, grįžtam į 3 žingsnį;
7. Naudodamiesi geriausios skruzdėlės duomenimis (laiko vėlinimais), apmokomas neuroninis tinklas;
8. Prognozuojamas tolimesnės laiko eilutės reikšmės naudojantis apmokytų neuroninių tinklų;
9. Įvertinamos prognozės paklaidos.

3. TIRIAMOJI DALIS

Visos tyrimui pasirinktos laiko eilutės duomenų šaltiniuose buvo apibrėžiamos kaip chaotinės laiko eilutės arba turinčios chaoso fenomeną. Tiriamojoje dalyje buvo naudojami vienodi pradiniai kintamieji nagrinėjant visas aptariamas laiko eilutes: skruzdėlių kiekis – 30, paliktas feromono dydis radus didžiausią tikslo funkcija – 0.01, atraktorius buvo dalinamas į 50 išpjovų skaičiuojant jo plotą, maksimalių iteracijų skaičius ieškant optimalių laiko vėlinimų ACO algoritmu – 500. Maksimalus laiko vėlinimas buvo pasirenkamas atsižvelgiant į laiko eilutės sezoniškumą arba naudojant ne daugiau kaip 20% chaotinės laiko apmokymo duomenų. Radialinių bazinių funkcijų neuroninis tinklas naudojo 2 sluoksnius, maksimalus neuronų skaičius galėjo būti iki 1000 ir neuroninis tinklas buvo apmokamos kol apmokymo paklaida mažėjo.

3.5. Mackey-Glass chaotinės laiko eilutės prognozavimas



7 pav. Chaotinė Mackey-Glass laiko eilutė

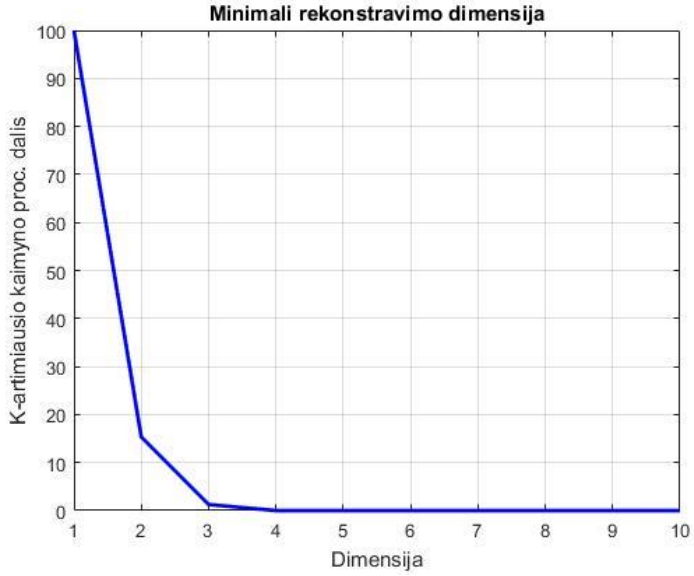
Mackey-Glass chaotinę eilutę atrado Leonas Glass 1977 metais McGill universitete. Glass gali būti geriausiai žinomas už matematiškai pagrįstus įrodymus, jog tam tikri fiziologiniai sutrikimai gali būti laikomi ligomis. Šioms ligoms būdingi staigūs dinamikos pokyčiai žmogaus fiziologinės kontrolės mechanizme, kurie sukelia ligas. Būtent šios savybės yra atvaizduojamos Mackey-Glass lygtyje:

$$\frac{dx}{dt} = \beta \frac{x_\tau}{1+{x_\tau}^{10}} - \gamma x, \quad \gamma, \beta, n > 0, \tag{24}$$

čia γ, β, τ, n yra realūs skaičiai ir x_τ yra kintamojo x vertė laiku $(t-\tau)$. Laiko eilutė yra sugeneruojama kai $\beta = 0.2$, $\gamma = 0.1$ ir laiko vėlinimai $\tau = 17$ (žr. 7 pav.).

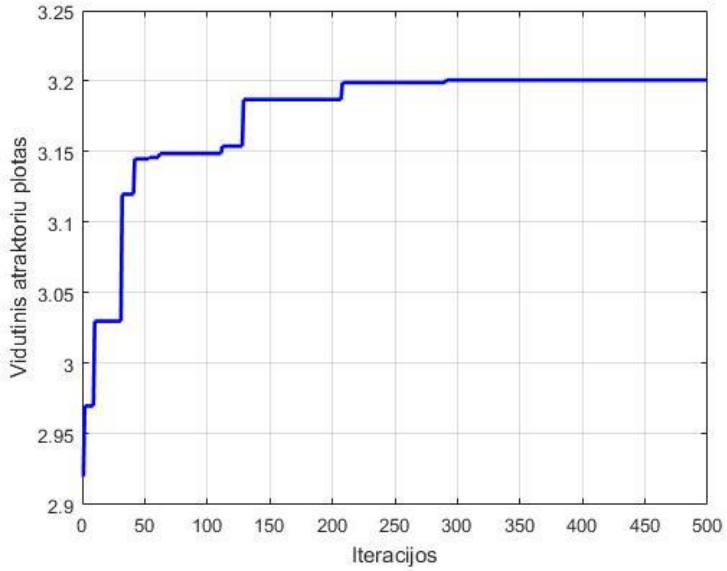
3.5.1. Mackey-Glass laiko eilutės prognozė kai $d = 4$

Mackey – Glass laiko eilutę rekonstruosime į laiko vėlinimų erdvę su nereguliariais laiko vėlinimais. Norint nustatyti į kokią laiko vėlinimų dimensiją rekonstruosime laiko eilutę, naudojame „klaidingo artimiausio kaimyno“ algoritmą (žr. 8 pav.).



8 pav. Mackey-Glass laiko eilutės minimalios rekonstravimo dimensijos paieška

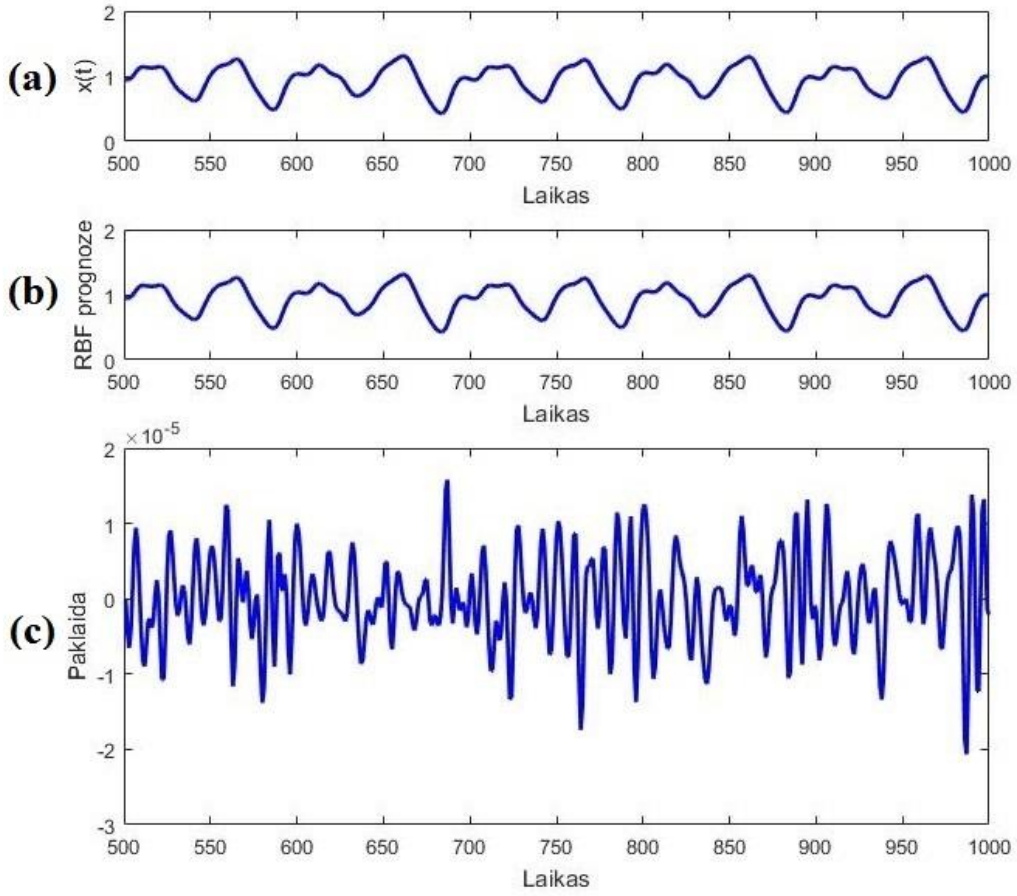
Galime teigti, jog optimaliam rekonstravimui reikalinga minimali dimensija yra $d = 4$. Turėdami minimalią rekonstravimo dimensiją, galime ieškoti optimalių laiko vėlinimų τ_i, kai $i = 1, \dots, d-1$. Laiko vėlinimų paieškai naudojame skrudzdžių kolonijos algoritmą, kuris ieško didžiausio vidutinio atraktorių ploto tarp visų dimensijų, laiko intervale nuo 1 iki 100.



9 pav. Mackey-Glass laiko eilutės optimalių laiko vėlinimų paieškos tikslo funkcija

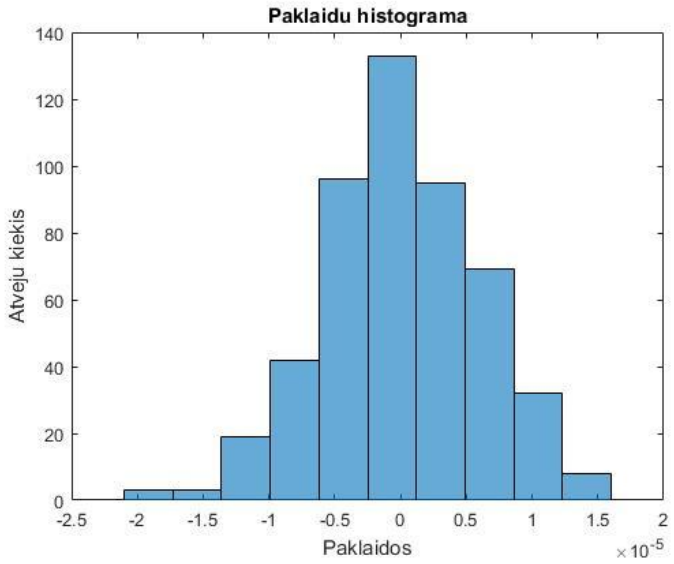
Skruzdžių kolonijos optimizavimo algoritmu randama optimali laiko vėlinimų aibė $\{8; 91; 97\}$, su kuria vidutinis atraktorių plotas yra $F(8{,}91{,}97) = 3{,}2008$ (žr 9 pav.). Prognozuojama $x(t+6)$ reikšmė iš praeities duomenų $x(t-92)$, $x(t-84)$ ir $x(t)$.

Prognozavimui naudojamas radialinės bazės funkcijų neuroninis tinklas. Pirmiausia yra surandami RBF neuroninio tinklo paplitimas (angl. spread) – 2.3. Vėliau neuroninis tinklas yra apmokamas naudojantis 500 Mackey-Glass laiko eilutės elementų, o likę 500 eilutės narių yra naudojami prognozavimui (žr. 10 pav.). RBF neuroninis tinklas apmokamas, kol apmokymo paklaidos gerėja. Neuroninio tinklo apmokymų epochų kiekis – 61, o pasiekta apmokymų paklaida – 3.48×10^{-11}.



10 pav. (a) – tikras Mackey-Glass signalas; (b) – RBF prognozė; (c) – prognozės paklaidos.

Modelio patikrai stebimos prognozės skirtuminės paklaidos (žr 10 apatinį pav.). Stjudento-t kriterijus patvirtino, kad paklaidos yra pasiskirsčiusios palei nulį ($p = 0$), Kolmogrovo-Smirnovo testas paneigė hipotezę, jog paklaidos yra priklausomos nuo duomenų ($p = 1$) (žr. 11 pav.). Modelio RMSE paklaida – 5.8821×10^{-6}, MAPE paklaida – $5.5664 \times 10^{-4}\%$.



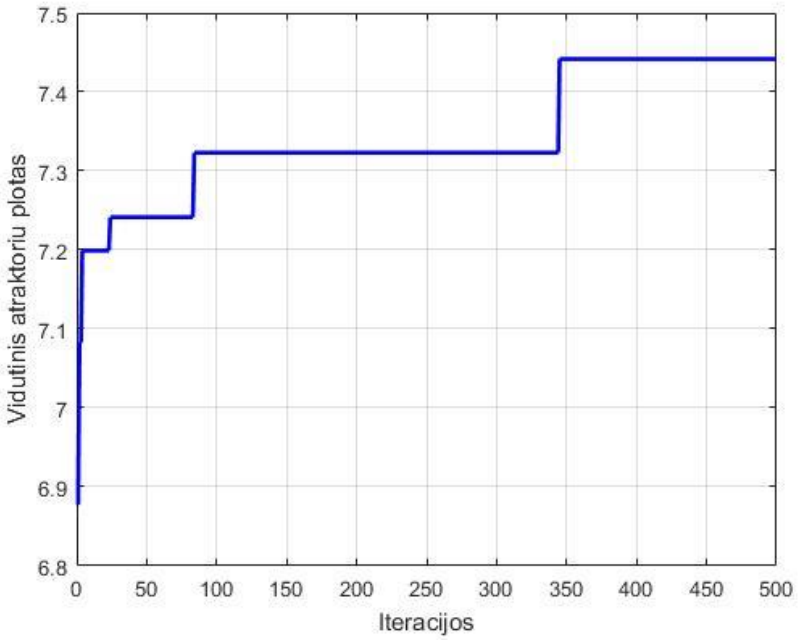
11 pav. Mackey-Glass laiko eilutės prognozės paklaidų histograma, kai $d = 4$.

3.5.2. Mackey-Glass laiko eilutės prognozė kai $d = 6$

Siekiant įvertinti sukonstruoto modelio rezultatus, reikia palyginti prognozavimo paklaidas su kitomis tos pačios laiko eilutės prognozės paklaidomis, kur buvo naudojama 50% duomenų apmokymui, 50% duomenų testavimui ir laiko eilutės prognozės rezultatų kokybei vertinimui naudojama ta pati metrika. Šiam tikslui yra panaudotas šaltinis su 8 skirtingomis Mackey-Glass laiko eilutės prognozėmis.[8]

Taip pat, prieš sulygindami mūsų siūlomo algoritmo paklaidas perskaičiuosime Mackey-Glass laiko eilutės prognozę, kai $d = 6$. Tai yra atliekama, nes modelio palyginimui naudojamas šaltinis naudojo 6 dimensijų laiko vėlinimų erdvę ir klaidingų artimiausių kaimynų metodas pateikia minimalią dimensiją optimalią laiko eilutės rekonstravimui, bet ne optimaliausią prognozavimui.

Analogiškai ieškome optimalių laiko vėlinimų τ_i, kai $i = 1, \dots, d-1$. Laiko vėlinimų paieškai naudojame skrudzdžių kolonijos algoritmą laiko intervale nuo 1 iki 100.

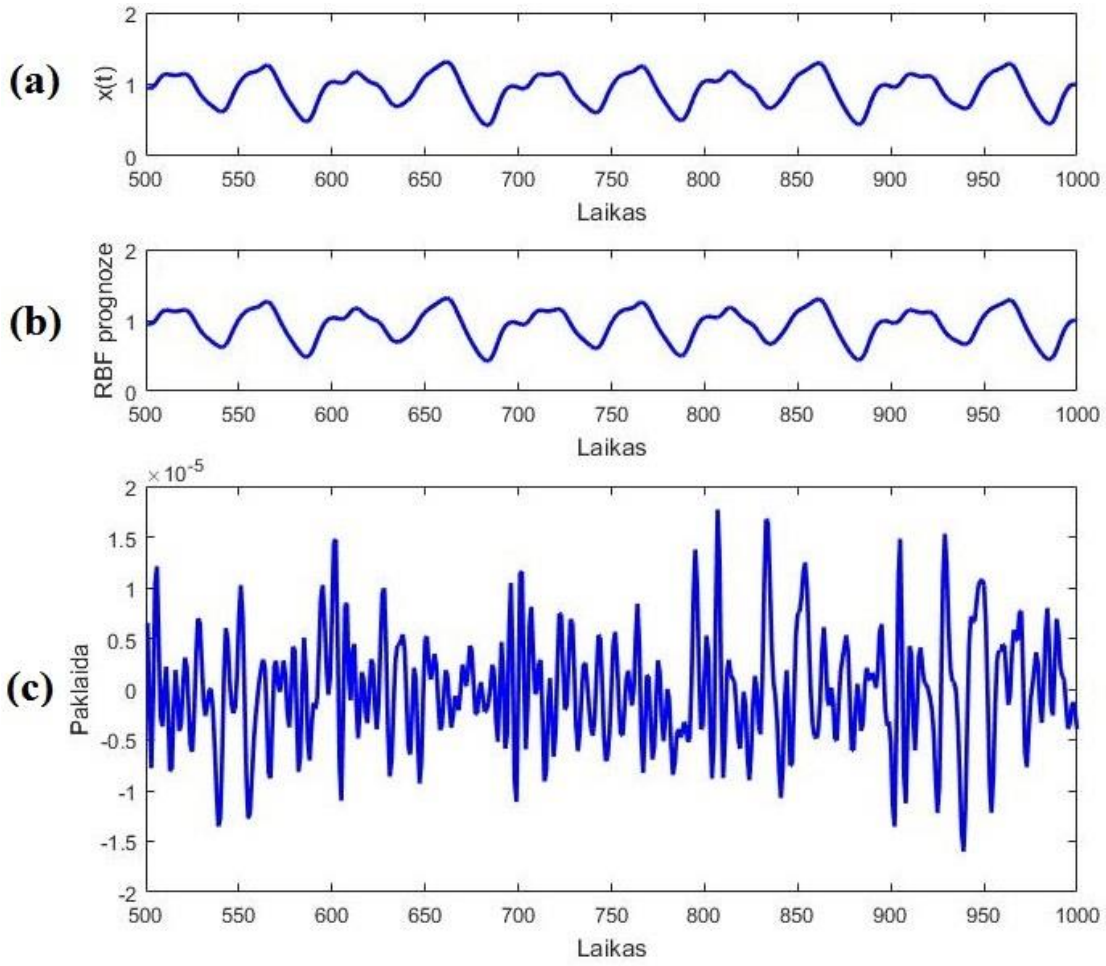


12 pav. Mackey-Glass laiko eilutės optimalių laiko vėlinimų paieškos tikslo funkcija

Skruzdžių kolonijos optimizavimo algoritmu randama optimali laiko vėlinimų aibė $\{7; 15; 85; 94; 99\}$ randama, su kuria tikslo funkcija yra $F(7; 15; 85; 94; 99) = 7.4417$ (žr 12 pav.). Prognozuojama $x(t+5)$ reikšmė iš praeities duomenų $x(t-94)$, $x(t-87)$, $x(t-79)$, $x(t-9)$ ir $x(t)$.

Prognozavimui naudojamas RBF neuroninis tinklas. RBF neuroninio tinklo paplitimas (angl. spread) – 3.01. Vėliau neuroninis tinklas yra apmokamas naudojantis 500 Mackey-Glass laiko eilutės elementų, o likę 500 eilutės narių yra naudojami prognozavimui. RBF neuroninis

tinklas apmokamas, kol apmokymo paklaidos gerėja. Neuroninio tinklo apmokymų epochų kiekis – 103, o pasiekta apmokymų paklaida – 2.936×10^{-11}.



13 pav. (a) – tikras Mackey-Glass signalas; (b) – RBF prognozė, kai *d=6*; (c) – prognozės paklaidos.

Modelio adekvatumui ir tikslumui nustatyti stebimos prognozės skirtuminės paklaidos (žr 13 apatinį pav.). Stjudento-t kriterijus patvirtino, kad paklaidos yra pasiskirsčiusios palei nulį ($p = 0$), Kolmogrovo-Smirnovo testas paneigė hipotezę, jog paklaidos yra priklausomos nuo duomenų ($p = 1$). Modelio RMSE paklaida – 5.553×10^{-6}, MAPE paklaida – $4.9321 \times 10^{-4}\%$.

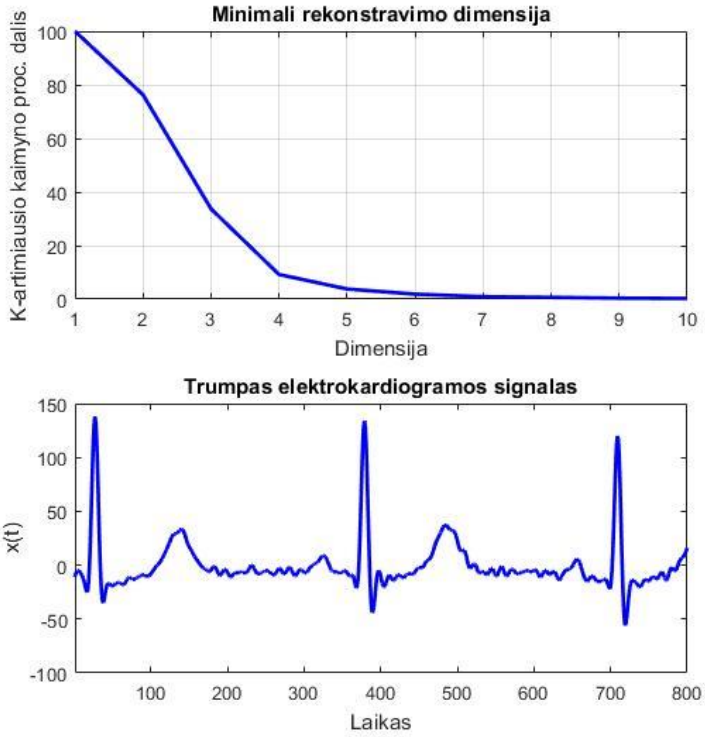
Mūsų siūlomo modelio palyginimui su kitomis prognozėmis naudojama RMSE metrika. 1 lentelėje yra pateiktos skirtingų modelių prognozavimo RMSE paklaidos. Joje matosi, jog mažiausios paklaidos gautos prognozuojant naudojantis kitu modeliu. Tačiau šis modelis yra 3 modelių hibridas, kuris buvo apmokytas naudojantis 6 milijonus iteracijų ir buvo reikalingos 40140 globalios iteracijos (bandymai) norint pasiekti šį geriausią rezultatą.[25] Lyginant su panašiais tyrimais, mūsų siūlomo modelio paklaidos yra 63 kartus mažesnės už šaltinyje pateiktą geriausią prognozavimo paklaidą.[8]

1 lentelė. Mackey-Glass laiko eilutės paklaidų palyginimas

Prognozės modelis	RMSE
Lanksčių beta funkcijų nervų medis naudojantis išplėstu genetiniu algoritmu ir hibridinių bičių kolonijos algoritmu	$1{,}3534 \times 10^{-10}$
Mūsų siūlomas modelis, kai d=6	$5{,}553 \times 10^{-6}$
Mūsų siūlomas modelis, kai d=4	$5{,}8821 \times 10^{-6}$
ANFIS su nereguliariu rekonstravimu	$3{,}497 \times 10^{-4}$
ANFIS su savaiminiu apsimokymu	$5{,}5 \times 10^{-4}$
ANFIS su reguliariu rekonstravimu	$7{,}284 \times 10^{-4}$
Evoliucionuojantis RBF su įėjimų atranka	$8{,}1 \times 10^{-4}$
Hibridinis ANN ir ARMA	$2{,}5 \times 10^{-3}$
Klasikinis RBF (su 23 neuronais)	$1{,}14 \times 10^{-2}$
Atgalinio sklidimo NN	2×10^{-2}
Autoregresinis modelis	$1{,}9 \times 10^{-1}$

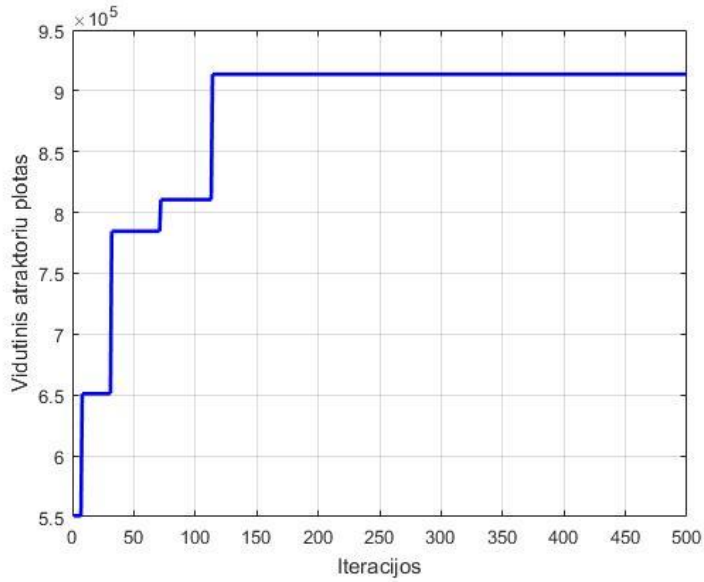
3.6. Elektrokardiogramos prognozavimas

Mūsų siūlomas modelis taikomas elektrokardiogramos (ECG) signalui (žr. 14 pav.). Ši prognozė yra naudojama aukštos kokybės sistemose norint suteikti pacientams perspėjimus prieš artėjančius infarktus ar sustiprėjusius prieširdžių virpėjimus.[26] Šiam tyrimui buvo specialiai pasirinktas paciento elektrokardiogramos įrašas, kuriam yra diagnozuotas prieširdžių virpėjimas. Prieširdžių virpėjimas yra širdies ritmo sutrikimas, pasireiškiantis nereguliariu ir labai greitu širdies dažniu (plakimu), todėl ši laiko eilutė pasižymi papildomu nestacionarumu bei triukšmu. Duomenys yra gauti iš „Northwestern“ universiteto Čikagoje, kuriuos viešai patalpino Masačiusetso technologijų universitetas tarptautiniam konkursui.[27] ECG laiko eilutę sudaro 8340 taškų (50 sekundžių), kurių pusę panaudosime neuroninių tinklų apmokymui, o kitą pusę – prognozei.



14 pav. Elektrokardiogramos signalo dalis ir minimalios dimensijos paieška rekonstravimui

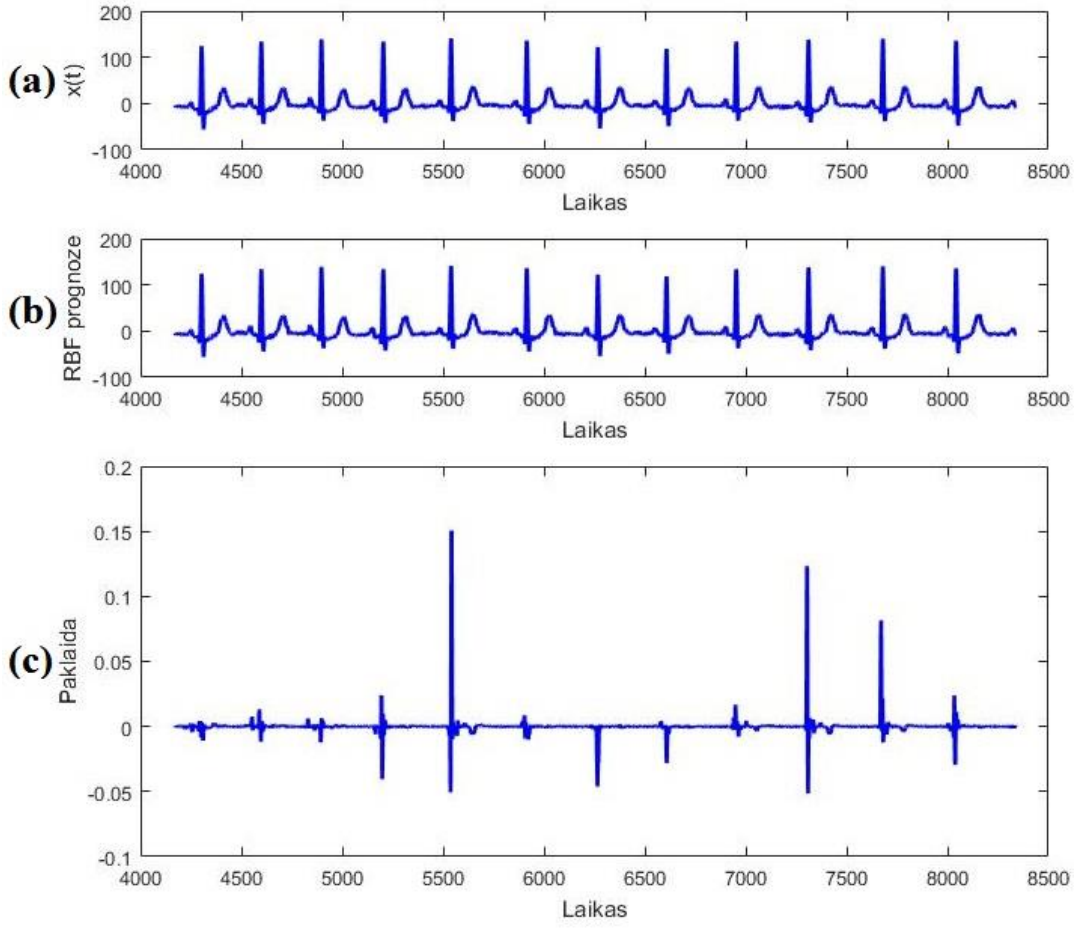
Pagal anksčiau pateiktą metodiką, reikia nustatyti į kokią laiko vėlinimų dimensiją rekonstruosime ECG laiko eilutę. Naudojantis klaidingų artimiausių kaimynų metodu nustatoma minimali dimensija optimaliam rekonstravimui $d = 9$ (žr. 14 pav.).



15 pav. ECG laiko eilutės optimalių laiko vėlinimų paieškos tikslo funkcija

Žinant reikiamą dimensiją, laiko eilutė rekonstruojama į vėlinimų erdvę nereguliariais laiko vėlinimais. SKO algoritmas yra naudojamas norint rasti optimalius laiko vėlinimus rinkinį laiko intervale nuo 1 iki 400, t.y. viename signalo periode.

Skruzdžių kolonijos optimizavimo algoritmo pagalba randami laiko vėlinimai $\{1; 4; 5; 309; 337; 343; 344; 374\}$ randama, su kuria tikslo funkcija yra $F(1; 4; 5; 309; 337; 343; 344; 374) = 9.1365 \times 10^5$ (žr. 15 pav.). Prognozuojama $x(t + 30)$ reikšmė iš praeities duomenų $x(t - 344)$, $x(t - 343)$, $x(t - 340)$, $x(t - 339)$, $x(t - 7)$, $x(t - 1)$ ir $x(t)$. 4170 duomenų (25 sekundės) naudojami RBF tinklo apmokymui, o likusi pusė – prognozei. RBF neuroninio tinklo paplitimas (angl. spread) – 927.5, neuroninio tinklo apmokymų epochų kiekis – 137, o pasiekta apmokymų paklaida – 5.391×10^{-7}.



16 pav. (a) – tikras ECG signalas; (b) – RBF prognozė; (c) – prognozės paklaidos.

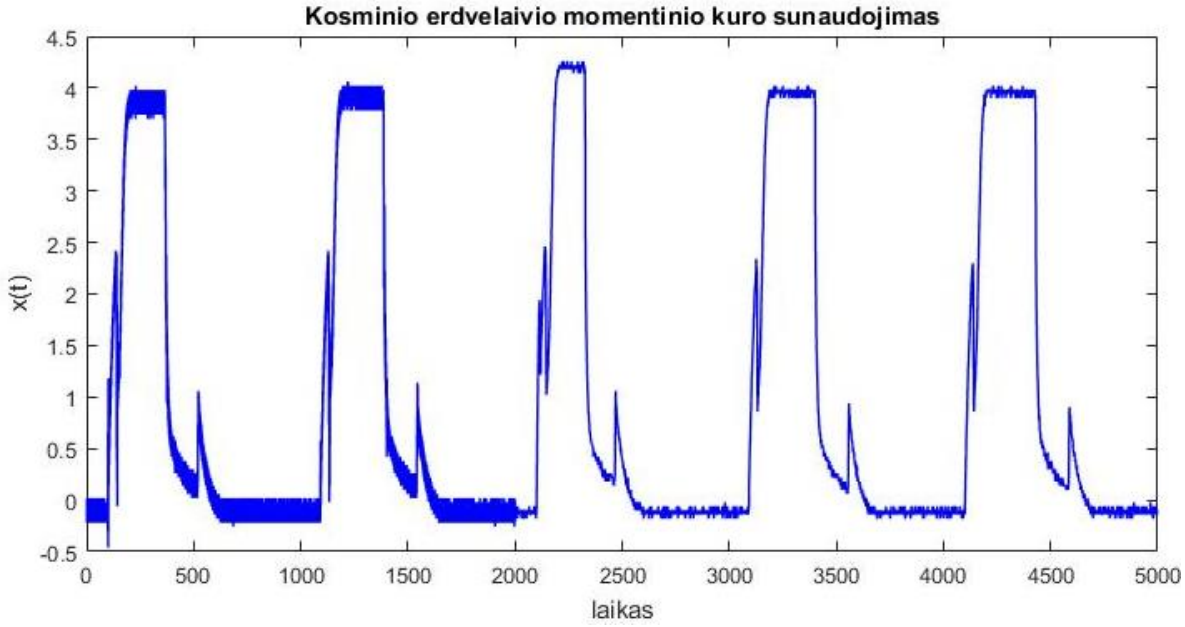
Modelio skirtuminės paklaidos pateiktos 16 apatiniame paveiksle. Stjudento-t kriterijus paneigė, kad paklaidos yra pasiskirsčiusios palei nulį ($p = 0$), o Kolmogrovo-Smirnovo testas patvirtino hipotezę, jog paklaidos yra priklausomos nuo duomenų ($p = 0.003$). Modelio paklaidų palyginimas su ARIMA modeliu pateiktas 2 lentelėje.

2 lentelė. ECG signalo prognozės paklaidų palyginimas

Modelis	RMSE
Mūsų siūlomas modelis	$6{,}6 \times 10^{-3}$
ARIMA(4,0,4)	0,7557

3.7. Kosminio erdvėlaivio momentinio kuro sunaudojimo signalo prognozavimas

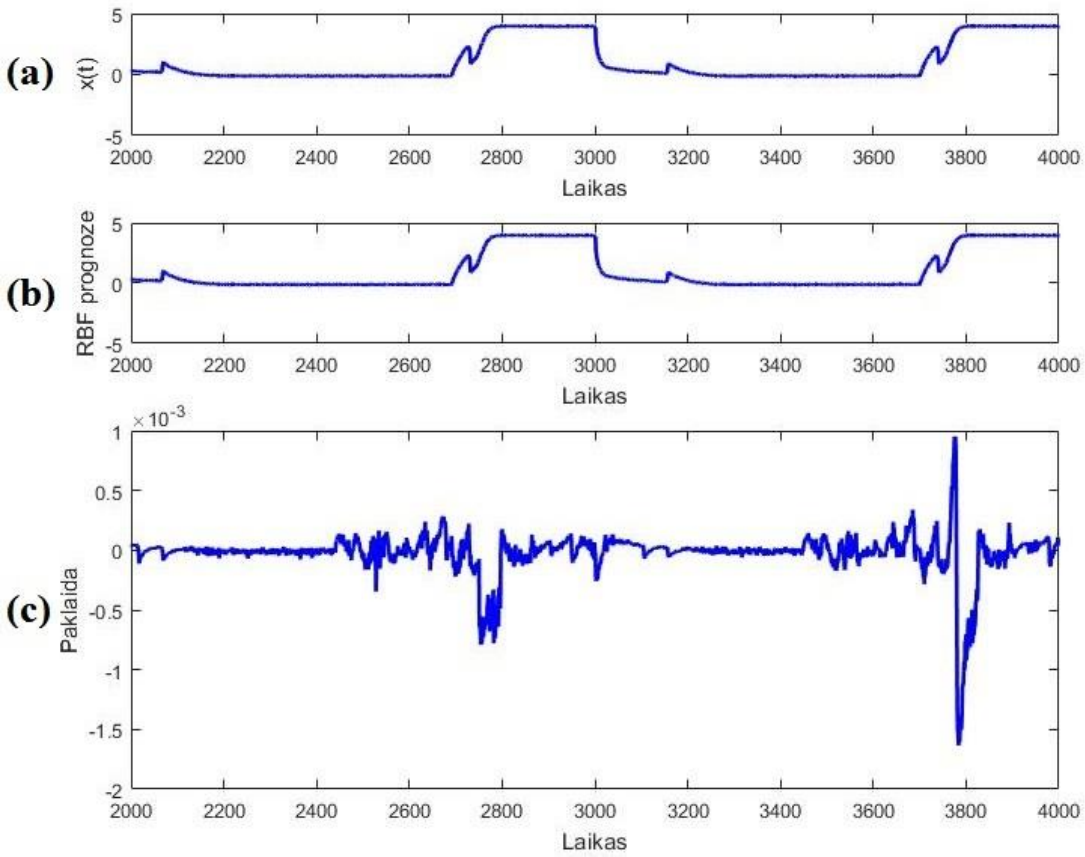
Laiko eilučių duomenys yra momentinio kuro sunaudojamo elektromagnetiniai matavimai, kurie naudojami kosminiams erdvėlaiviams JAV „Space Shuttle“ programoje. Šie vožtuvai yra naudojami siekiant kontroliuoti degalų srautą. Duomenis prognozėms paviešino Kenedžio kosmoso centro darbuotojai Bob Ferrell, Steven Santuro ir kiti kolegos. Šį projektą remia NASA (NAS10-02044).[28] Šią laiko eilutę sudaro 5000 taškų, kurių 2000 taškų panaudosime neuroninių tinklų apmokymui, o kitus 2000 taškų – prognozei.



17 pav. Kosminio erdvėlaivio momentinio kuro sunaudojimo signalas

Klaidingų artimiausių kaimynų metodu yra nustatoma minimali dimensija optimaliam rekonstravimui $d = 5$. Žinant reikiamą dimensiją, laiko eilutė rekonstruojama į vėlinimų erdvę nereguliariais laiko vėlinimais. ACO algoritmas yra naudojamas norint rasti optimalius laiko vėlinimus rinkinį laiko intervale nuo 1 iki 600.

Skruzdžių kolonijos optimizavimo algoritmo pagalba randami laiko vėlinimai $\{52; 138; 205; 251\}$ randama, su kuria tikslo funkcija yra $F(52; 138; 205; 251) = 267.07$. Prognozuojama $x(t + 46)$ reikšmė iš praeities duomenų $x(t - 205)$, $x(t - 153)$, $x(t - 67)$ ir $x(t)$. RBF neuroninio tinklo paplitimas (angl. spread) – 33.98, neuroninio tinklo apmokymų epochų kiekis – 197, o pasiekta apmokymų paklaida – 9.30589×10^{-9}.



18 pav. (a) – kuro sunaudojimo signalas; (b) – RBF prognozė; (c) – prognozės paklaidos

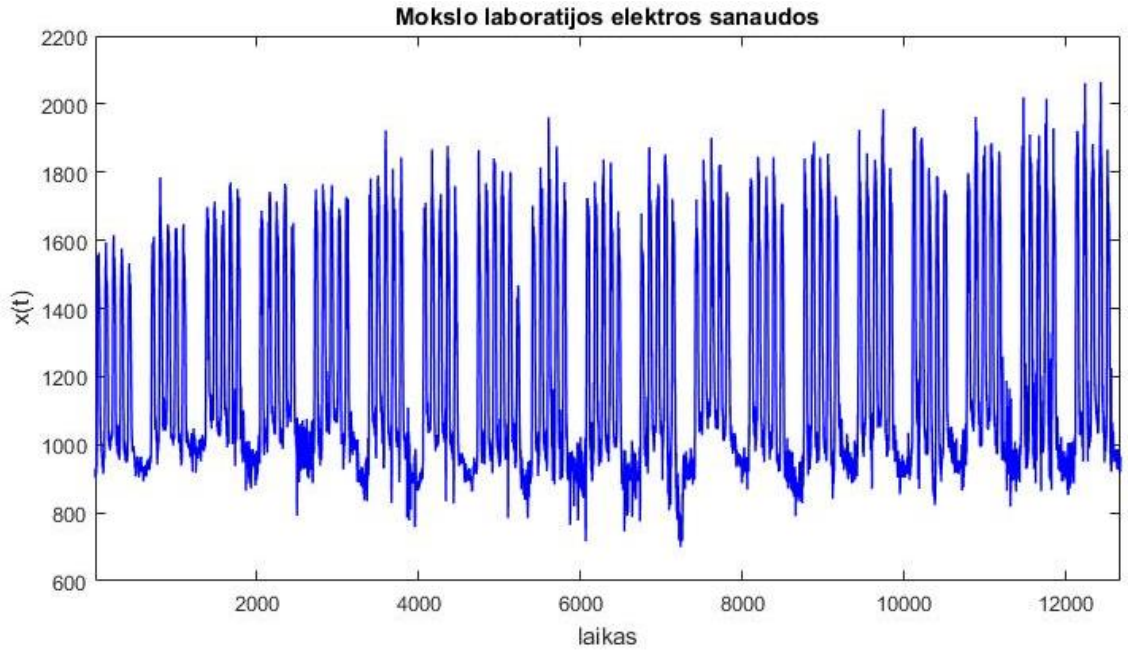
Modelio skirtuminės paklaidos pateiktos 18 apatiniame paveiksle. Stjudento-t kriterijus patvirtino, kad paklaidos yra pasiskirsčiusios palei nulį ($p = 0$), tačiau Kolmogrovo-Smirnovo testas patvirtino hipotezę, jog paklaidos yra priklausomos nuo duomenų ($p = 0$). Modelio RMSE paklaida – 1.9322×10^{-4}, MAPE paklaida – 0.0258%. Modelio paklaidų palyginimas su ARIMA modeliu pateiktas 3 lentelėje

3 lentelė. Momentinių kuro sąnaudų prognozių palyginimas

Modelis	RMSE
Mūsų siūlomas modelis	$1{,}9322 \times 10^{-4}$
ARIMA(2,1,3)	$8{,}92 \times 10^{-2}$

3.8. Elektros sąnaudų prognozavimas

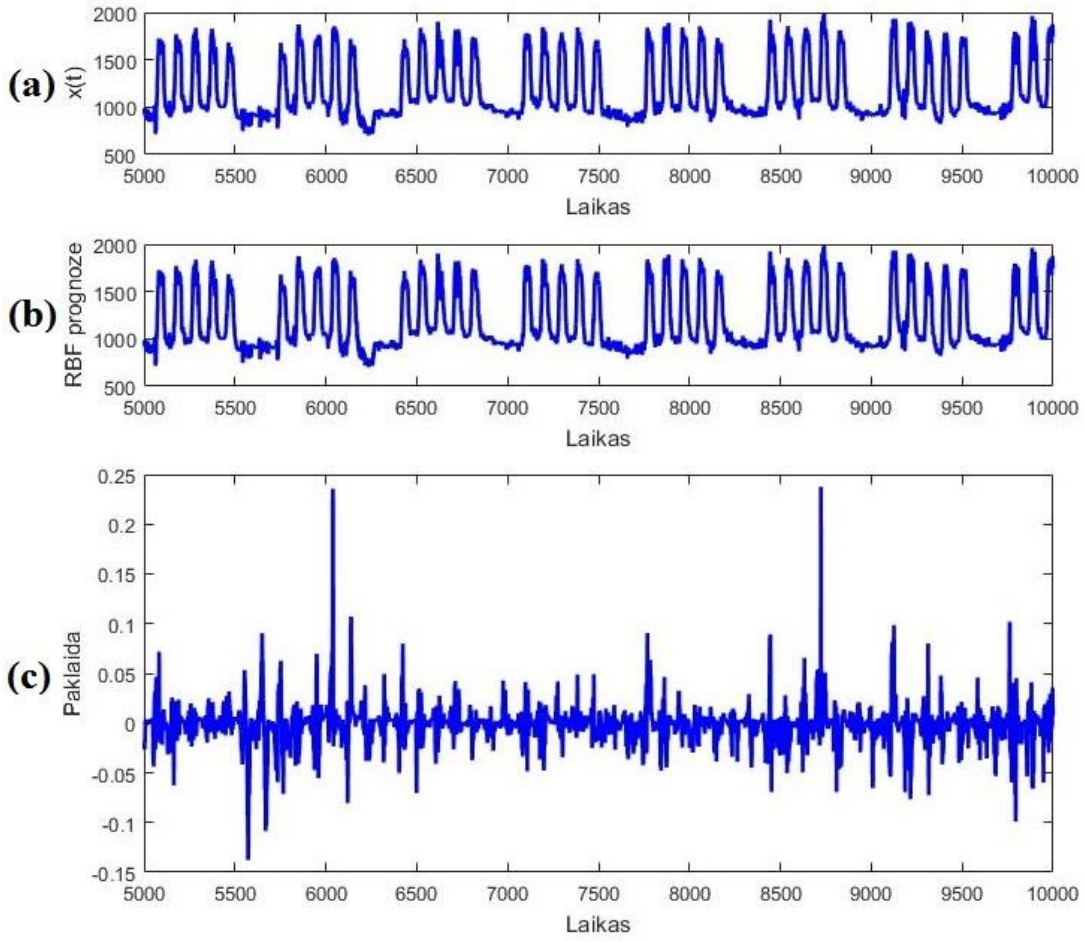
Laiko eilučių duomenys yra Nyderlandų mokslo laboratorijos elektros sąnaudos (19 pav.). Naudosime 4 mėnesių ilgio elektros sąnaudų laiko eilutę, kurioje elektros sunaudojamas pateikiamas kas 15 minučių, t.y. 96 matavimai per parą.[29]



19 pav. Elektros sąnaudų signalas

Klaidingų artimiausių kaimynų metodu yra nustatoma minimali dimensija optimaliam rekonstravimui $d = 8$. Žinant reikiamą dimensiją, laiko eilutė rekonstruojama į vėlinimų erdvę nereguliariais laiko vėlinimais. ACO algoritmas yra naudojamas norint rasti optimalius laiko vėlinimus rinkinį laiko intervale nuo 1 iki 700, t.y. vienoje savaitėje.

Skruzdžių kolonijos optimizavimo algoritmo pagalba randami laiko vėlinimai $\{117; 199; 298; 413; 505; 583; 684\}$ randama, su kuria tikslo funkcija yra $F(117; 199; 298; 413; 505; 583; 684) = 3.926 \times 10^7$. Prognozuojama $x(t + 101)$ reikšmė iš praeities duomenų $x(t - 583)$, $x(t - 466)$, $x(t - 384)$, $x(t - 285)$, $x(t - 170)$, $x(t - 78)$ ir $x(t)$. RBF neuroninio tinklo paplitimas (angl. spread) – 1.2422×10^4, neuroninio tinklo apmokymų epochų kiekis – 167, o pasiekta apmokymų paklaida – 9.82545×10^{-5}.



20 pav. (a) – energijos sunaudojimo laiko eilutė; (b) – RBF prognozė; (c) – prognozės paklaidos

Modelio skirtuminės paklaidos pateiktos 20 apatiniame paveiksle. Stjudento-t kriterijus patvirtino, kad paklaidos yra pasiskirsčiusios palei nulį ($p = 0$), o Kolmogrovo-Smirnovo testas paneigė hipotezę, jog paklaidos yra priklausomos nuo duomenų ($p = 1$). Modelio RMSE paklaida – 0.0188, MAPE paklaida – $9.53 \times 10^{-4}\%$. Modelio paklaidų palyginimas su ARIMA modeliu pateiktas 4 lentelėje.

4 lentelė. Elektros sąnaudų prognozavimo paklaidų palyginimas

Modelis	RMSE
Mūsų siūlomas modelis	0.0188
ARIMA(4,1,2)	0,3378

4. IŠVADOS

1. Sukonstruotas naujas laiko eilučių nereguliaraus rekonstravimo į laiko vėlinimų erdvę modelis, skirtas optimalių laiko vėlinimų rinkiniui ieškoti. Darbe pasiūlyta laiko eilučių prognozavimo metodika, pagrįsta skruzdžių kolonijos optimizavimo algoritmu, radialinių bazinių funkcijų neuroniniais tinklais ir nereguliariu atraktoriaus rekonstravimu į vėlinimų erdvę. Šiame darbe pasiūlyta metodika padeda tiksliau prognozuoti laiko eilutes palyginus modelio paklaidas su kitų autorių darbais ir kitais prognozavimo metodais.
2. Ieškant optimalaus laiko vėlinimų rinkinio su skruzdžių kolonijos optimizavimo algoritmu, pasiūlyta nauja tikslo funkcija – visų atraktorių vidutinis plotas. Ploto vidurkio skaičiavimas padeda plačiau išpūsti plotų vidurkius, ko pasėkoje geriausiai atsiskleidžia atraktoriaus savybės Tikslo funkcija yra greita sprendžiant mažų dimensijų rekonstravimo uždavinius ir rado optimalias laiko vėlinimų reikšmes.
3. Darbe rekonstruojant tradicinę Mackey - Glass chaotinę laiko eilutę nereguliariais laiko vėlinimais naudojamas skruzdžių kolonijos optimizavimo algoritmas. Minimali rekonstravimo dimensija – 4. Šiuo atveju optimizavimo tikslo funkcijos reikšmė $F(8{,}91{,}97) = 3{,}2008$, gaunama kai laiko vėlinimų rinkinys $\{8; 91; 97\}$. Atitinkamai kitoms laiko eilutėms buvo gaunami tokie laiko vėlinimai ir tikslo funkcijos: elektrokardiogramos – $F(1; 4; 5; 309; 337; 343; 344; 374) = 9.1365 \times 10^5$, elektros sąnaudų – $F(117; 199; 298; 413; 505; 583; 684) = 3.926 \times 10^7$, momentinių kuro sąnaudų – $F(52; 138; 205; 251) = 267.07$;
4. Minėtos laiko eilutės toliau yra prognozuotos naudojant RBF neuroniniu tinklu. Šio tinklo įėjimai yra laiko eilutės rekonstruoti vektoriai tarp kurių yra rasti laiko vėlinimai. Prognozuojant Mackey – Glass eilutę neuroninis tinklas apmokyme atliko 61 epochas, o prognozavimo RMSE paklaida – 5.8821×10^{-6}. Siūlomos metodikos 25 sekundžių ECG laiko eilutės prognozės RMSE paklaida – 6.6×10^{-3}, momentinių kuro sąnaudų – 1.9322×10^{-4}, o 2 mėnesių elektros sąnaudų – 0.0188.
5. Darbe prognozavimo paklaidos palyginamos su kitų metodų paklaidomis. Mackey – Glass mums pavyko pagerinti 168 kartus lyginant su „ANFIS su nereguliariu rekonstravimu“ metodu. Darbe pasiūlyto modelio 2 mėnesių elektros sąnaudų prognozės paklaidos buvo mažesnės 18 kartų, 25 sekundžių ECG laiko eilutės paklaidos buvo mažesnės 114 kartų, o momentinių kuro sąnaudų – 461 kartą nei ARIMA modelio. Geresnės RBF tinklo tikslumo priežastys yra tai, kad RBF tinklas yra netiesinis metodas ir apmokomas iki minimalios mokymosi paklaidos.

PADĖKA

Gerb. doc. dr. Kristinai Lukoševičiūtei

Nuoširdžiai dėkoju savo magistrantūros projekto vadovei Kristinai Lukoševičiūtei už kantrybę, skirtą brangų laiką, konsultacijas, skatinimą dirbti ir pagalbą rašant baigiamąjį magistrantūros baigiamąjį darbą. Jos idėjos ir pagalba padėjo pasiekti aukštus tyrimo rezultatus. Ypatingai dėkoju vadovei, jog supažindino su man nauja sritimi ir sudomino tolimesniais šios srities pritaikymais.

Su pagarba,
Paulius Čepulionis

5. LITERATŪRA

1. BOWERMAN, B.L. and O'CONNELL, R.T. *Time Series and Forecasting.* Duxbury Press North Scituate, Massachusetts, 1979.

2. BROCKWELL, P.J. and DAVIS, R.A. *Introduction to Time Series and Forecasting.* Springer Science & Business Media, 2006.

3. LIU, Z. Chaotic Time Series Analysis. *Mathematical Problems in Engineering*, 2010, vol. 2010.

4. CHEN, W. and ZHANG, J. *Ant Colony Optimization for Determining the Optimal Dimension and Delays in Phase Space Reconstruction.* ACM, 2011.

5. SAUER, T., YORKE, J.A. and CASDAGLI, M. Embedology. *Journal of Statistical Physics*, 1991, vol. 65, no. 3-4. pp. 579-616.

6. ASTUDILLO, H., BOROTTO, F. and ABARCA-DEL-RIO, R. Embedding Reconstruction Methodology for Short Time Series-Application to Large El Nino Events. *Nonlinear Processes in Geophysics*, 2010, vol. 17, no. 6. pp. 753-764.

7. RAO, T.S., RAO, S.S. and RAO, C.R. *Handbook of Statistics: Time Series Analysis: Methods and Applications.* Elsevier, 2012.

8. LUKOŠEVIČIŪTĖ, K. *Chaotinių Procesų Rekonstravimo Bei Algebrinių Sekų Modeliai Laiko Eilučių Prognozavime.* Kaunas: Kaunas University of Technology, 2012.

9. ZHANG, G., PATUWO, B.E. and HU, M.Y. Forecasting with Artificial Neural Networks:: The State of the Art. *International Journal of Forecasting*, 1998, vol. 14, no. 1. pp. 35-62.

10. BENMOUIZA, K. and CHEKNANE, A. Forecasting Hourly Global Solar Radiation using Hybrid K-Means and Nonlinear Autoregressive Neural Network Models. *Energy Conversion and Management*, 2013, vol. 75. pp. 561-569.

11. HAN, L., ROMERO, C.E. and YAO, Z. Wind Power Forecasting Based on Principle Component Phase Space Reconstruction. *Renewable Energy*, 2015, vol. 81. pp. 737-744.

12. CAO, L. Practical Method for Determining the Minimum Embedding Dimension of a Scalar Time Series. *Physica D: Nonlinear Phenomena*, 1997, vol. 110, no. 1. pp. 43-50.

13. WU, Y., SU, J., TANG, H. and TIANFIELD, H. Analysis of the Emergence in Swarm Model Based on Largest Lyapunov Exponent. *Mathematical Problems in Engineering*, 2011, vol. 2011.

14. SHEN, M., et al. Optimal Selection of Parameters for Nonuniform Embedding of Chaotic Time Series using Ant Colony Optimization. *Cybernetics, IEEE Transactions On*, 2013, vol. 43, no. 2. pp. 790-802.

15. WANG, J., ZHOU, B., ZHOU, S. and SHENG, Z. Forecasting Nonlinear Chaotic Time Series with Function Expression Method Based on an Improved Genetic-Simulated Annealing Algorithm. *Computational Intelligence and Neuroscience*, 2015, vol. 2015. pp. 42.

16. MEZEIOVÁ, K. and KRAKOVSKÁ, A. Choice of Measurement for Phase-Space Reconstruction: Decision Based on False Nearest Neighbors Method.

17. CUI, X. and JIANG, M. Chaotic Time Series Prediction Based on Binary Particle Swarm Optimization. *AASRI Procedia*, 2012, vol. 1. pp. 377-383.

18. WANG, J., XIE, Y. and ZHU, C. Solar Radiation Prediction Based on Phase Space Reconstruction of Wavelet Neural Network. *Procedia Engineering*, 2011, vol. 15. pp. 4603-4607.

19. KLIKOVÁ, B. and RAIDL, A. *Reconstruction of Phase Space of Dynamical Systems using Method of Time Delay*.

20. RHODES, C. and MORARI, M. False-Nearest-Neighbors Algorithm and Noise-Corrupted Time Series. *Physical Review E*, 1997, vol. 55, no. 5. pp. 6162.

21. MICHALSKI, R.S., CARBONELL, J.G. and MITCHELL, T.M. *Machine Learning: An Artificial Intelligence Approach*. Springer Science & Business Media, 2013.

22. MITCHELL, T.M. Machine Learning. *Machine Learning*, 1997.

23. BISHOP, C. Improving the Generalization Properties of Radial Basis Function Neural Networks. *Neural Computation*, 1991, vol. 3, no. 4. pp. 579-588.

24. ORR, M.J. *Introduction to Radial Basis Function Networks*, 1996.

25. BOUAZIZ, S., DHAHRI, H., ALIMI, A.M. and ABRAHAM, A. Evolving Flexible Beta Basis Function Neural Tree using Extended Genetic Programming & Hybrid Artificial Bee Colony. *Applied Soft Computing*, 2016.

26. TANG, X. Novel Remote ECG Real-Time Monitoring System, 2009.

27. MOODY, G., GOLDBERGER, A., MCCLENNEN, S. and SWIRYN, S. *Predicting the Onset of Paroxysmal Atrial Fibrillation: The Computers in Cardiology Challenge 2001*. IEEE, 2001.

28. FERRELL, B. and SANTURO, S. *NASA Shuttle Valve Data*, 2005.

29. VAN WIJK, J.J. and VAN SELOW, E.R. *Cluster and Calendar Based Visualization of Time Series Data*. IEEE, 1999.

PRIEDAI

1 PRIEDAS. KLAIDINGŲ ARTIMIAUSIŲ KAIMYNŲ ALGORITMAS.

```
function [FNN] = knn_deneme(x,tao,mmax,rtol,atol)
clc;
clear all;
workspace;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load Space;
x = Space;
plot(1:length(x),x,'b','LineWidth',1);
xlabel('laikas');
ylabel('x(t)');
title('Kosminio erdvelaivio momentinio kuro sunaudojimas')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
time = (1:length(x)-1);
mmax = 10;
tao = 20;
rtol = 15;
atol=2;
fprintf('nuskaityta')
N=length(x);
Ra=std(x,1);
fprintf('1')
for m=1:mmax
    M=N-m*tao;
    fprintf('2')
    Y=psr_deneme(x,m,tao,M);
    FNN(m,1)=0;
    for n=1:M
        y0=ones(M,1)*Y(n,:);
        distance=sqrt(sum((Y-y0).^2,2));
        [neardis nearpos]=sort(distance);

        D=abs(x(n+m*tao)-x(nearpos(2)+m*tao));
        R=sqrt(D.^2+neardis(2).^2);
        if D/neardis(2) > rtol || R/Ra > atol
             FNN(m,1)=FNN(m,1)+1;
        end
    end
    fprintf('iteracija #%d\n', m)
end

FNN=(FNN./FNN(1,1))*100;

figure(1)
plot(1:length(FNN),FNN,'b','LineWidth',2)
grid on;
title('Minimali rekonstravimo dimensija')
xlabel('Dimensija')
ylabel('K-artimiausio kaimyno proc. dalis')
subplot(2,1,2)
plot(1:length(FNN),FNN,'b','LineWidth',2)
grid on;
title('Minimali rekonstravimo dimensija')
xlabel('Dimensija')
ylabel('K-artimiausio kaimyno proc. dalis')
subplot(2,1,2)
plot(1:length(x)x,'k','LineWidth',2);
xlabel('Time');
ylabel('x(t)');
xlim([1 800])
```

```
title('Small part of the ECG signal')
function Y=psr_deneme(x,m,tao,npoint)
N=length(x);
if nargin == 4
    M=npoint;
else
    M=N-(m-1)*tao;
end

Y=zeros(M,m);

for i=1:m
    Y(:,i)=x((1:M)+(i-1)*tao)';
end
```

2 PRIEDAS. SKRUZDELIŲ KOLONIJOS OPTIMIZAVIMO ALGORITMAS.

```
clc;
clear;
close all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load Power;
XT = Power;
XT = XT - mean(XT);
T=1:length(XT);
XT = transpose(XT);
n=numel(XT);
T = transpose(T);
dim=7;
phermone = 0.01;
imtis=500;
Maksiter=500;
dal = 50;
total_ants=20;
P(1:imtis)=1/imtis;
Skruzde(dim,total_ants)=0;
SkruDidz = 0;
Didz(1:Maksiter) = 0;
sprendinys(1:dim)=0;
Matrica(n-imtis,dim)=0;
Geriausias = 0;
for iter=1:Maksiter
    for i=1:total_ants
        Skruzde(:,i)=Velinimas(P,dim);
        Matrica(:,1)=XT(1:size(Matrica,1));
        for x=2:dim+1
            indeksas = Skruzde(x-1,i);
            Matrica(:,x)=XT(1+indeksas:size(Matrica,1)+indeksas);
        end
        SkruDidz = Plotukas(Matrica,dal);
        if SkruDidz > Geriausias
            Geriausias = SkruDidz;
            Didz(iter) = SkruDidz;
            sprendinys=Skruzde(:,i);
        else
            %Geriausias = Geriausias;
            Didz(iter) = Geriausias;
        end
    end
    if Didz(iter) > Geriausias
        Geriausias = Didz(iter);
        sprendinys=Skruzde(:,i);
        for h=0:dim
```

```
            P(Skruzde(h,i))=P(Skruzde(h,i))+phermone;
        end
    end
    fprintf('iteracija #%d\n', iter)
end
figure(1)
plot(Didz,'b','LineWidth',2)
grid on;
xlabel('Iteracijos')
ylabel('Vidutinis atraktoriu plotas')
figure(2)
plot(Didz,'k','LineWidth',2)
grid on;
xlabel('Iterations')
ylabel('Average area of attractors')
```

3 PRIEDAS. RULETĖS RATO PASIRINKIMO FUNKCIJA LAIKO VĖLINIMAMS

```
function [Geras] = Velinimas(P,dimensija)
k=0;
Ats = zeros(1,dimensija);
Atsakymas = zeros(1,dimensija);
X=1:numel(P);
while k < dimensija
    p = cumsum([0; P(1:end-1).'; 1+1e3*eps]);
    [a a] = histc(rand,p);
    F = X(a);
    if any(F==Ats)==1
        k = k;
    else
        Atsakymas(k+1)=F;
        k = k + 1;
    end
end
Geras = Atsakymas;
end
```

4 PRIEDAS. ATRAKTORIAUS PLOTO SKAIČIAVIMO FUNKCIJA

```
function [Vis_plotas] = Plotukas(A,dal)
[~,stulpeliu] = size(A);
comb = nchoosek(1:stulpeliu,2);
[Ceil,~] = size(comb);
Plo(1:dal) = 0;
as=0;
for i=1:Ceil
    k=comb(i,1);
    x=comb(i,2);
    B1 = A(:,k);
    B2 = A(:,x);
    S = Dalinimas(B1,B2,dal);
    for y=1:dal
        Plo(y)=(S(y)*S(y)*pi)/dal;
    end
    asd = sum(Plo);
    as=as+asd;
end
Vis_plotas=as;
end
```

5 PRIEDAS. ATRAKTORIAUS DALINIMO FUNKCIJA Į IŠPJOVAS

```
function [S] = Dalinimas(XXT,YYT,dal)
k = dal/2;
pra = k+1;
S(1:dal) =0;
n = length(XXT);
BAM = [0:pi/k:pi];
BAM1 = [-pi:pi/k:0];
for i=1:n
    ilgis = sqrt(XXT(i)*XXT(i) + YYT(i)*YYT(i));
    if (atan2(YYT(i),XXT(i)) >= 0)
        for j=1:k
            if BAM(j)<=atan2(YYT(i),XXT(i))
                if BAM(j+1)>atan2(YYT(i),XXT(i))
                    if S(j)<ilgis;
                        S(j)=ilgis;
                    end
                end
            end
        end
    else
        for j=1:k
            if BAM1(j)<= atan2(YYT(i),XXT(i))
                if BAM1(j+1)>atan2(YYT(i),XXT(i))
                    if S(k+j)<ilgis;
                        S(k+j)=ilgis;
                    end
                end
            end
        end
    end
end
end
```

6 PRIEDAS. RBF NEURONINIO TINKLO FUNKCIJA

```
clc;
workspace;
clear all;
tic

N=500;
load mgdata.dat
a = mgdata;
duom = a(:,2);
tau1 = 8;
tau2 = 84;
tau3 = 6;
P = duom';
Apmok = N/5;
Start1 = Apmok+1;
Start2 = Start1+N;
Pabaiga=2*N;

P1=zeros(3,N);
Start=Start1;
P1(1,:)=P(Start:Start+N-1);
Start=Start+tau1;
P1(2,:)=P(Start:Start+N-1);
Start=Start+tau2;
P1(3,:)=P(Start:Start+N-1);
Start=Start+tau3;
```

```
P1(4,:)=P(Start:Start+N-1);
% Start=Start+tau4;
% P1(5,:)=P(Start:Start+N-1);
% Start=Start+tau5;
% P1(6,:)=P(Start:Start+N-1);
% Start=Start+tau6;
% P1(7,:)=P(Start:Start+N-1);
% Start=Start+tau7;
% P1(8,:)=P(Start:Start+N-1);
% Start=Start+tau8;
% P1(9,:)=P(Start:Start+N-1);

T = P(Start1:N+Apmok);
mn = 1000;
df = 2;
y = mean(T);
y00 = repmat(mean(T,2),1,size(T,2));
e00 = T-y00;
SSE00 = sse(e00);
SSEgoal = SSE00/100;
spread = 0.5*mean(median(dist(P1,P1'))) % issibarstymo konstanta
MSE0 = (N-1)*mean(var(T'))/N;
MSEi = (N-1)*var(T)/N;
size(T)
[ O Ntrain ] = size(T);
Ytrain00 = repmat(mean(T,2),1,Ntrain);
MSEtrain00 = mse(T-Ytrain00);
MSEgoal = MSEtrain00/2000 % R2train >= 0.995; % sum-squared error goal
MSE00 = mean(var(T'));
net = newrb1(P1,T,1000000,spread,mn,df);
ytrain = net(P1);
Pt2=zeros(3,1);
Start = Start2;
Pt2(1,:)=P(Start:Start+1-1);
Start=Start+tau1;
Pt2(2,:)=P(Start:Start+1-1);
Start=Start+tau2;
Pt2(3,:)=P(Start:Start+1-1);
Start=Start+tau3;
Pt2(4,:)=P(Start:Start+1-1);
% Start=Start+tau4;
% Pt2(5,:)=P(Start:Start+1-1);
% Start=Start+tau5;
% Pt2(6,:)=P(Start:Start+1-1);
% Start=Start+tau6;
% Pt2(7,:)=P(Start:Start+1-1);
% Start=Start+tau7;
% Pt2(8,:)=P(Start:Start+1-1);
% Start=Start+tau8;
% Pt2(9,:)=P(Start:Start+1-1);
for i = 0:N
    if i~=0
        Start=Start2+i;
        %tau=9;
        Pt2(1,:)=P(Start:Start+1-1);
        Start=Start+tau1;
        Pt2(2,:)=P(Start:Start+1-1);
        Start=Start+tau2;
        Pt2(3,:)=P(Start:Start+1-1);
        Start=Start+tau3;
        Pt2(4,:)=P(Start:Start+1-1);
%         Start=Start+tau4;
%         Pt2(5,:)=P(Start:Start+1-1);
%         Start=Start+tau5;
```

```
%          Pt2(6,:)=P(Start:Start+1-1);
%          Start=Start+tau6;
%          Pt2(7,:)=P(Start:Start+1-1);
%          Start=Start+tau7;
%          Pt2(8,:)=P(Start:Start+1-1);
%          Start=Start+tau8;
%          Pt2(9,:)=P(Start:Start+1-1);
         Pt3(i)=yPred;
    end
    yPred = net(Pt2);
end
Tt=P(N+Start1:2*N+Apmok);
Yt = Pt3;
train=duom(Start1:N+Apmok)';
e=Tt-Yt;
ytrainn=ytrain*(max(duom)-min(duom))+min(duom);
index=Start1+N:2*N+Apmok;
figure(1);
subplot(2,1,1)
plot(Tt);
xlabel('time(index)'); ylabel('x(t)');
subplot(2,1,2)
plot(Yt);
xlabel('time(index)'); ylabel('radial basis');
figure(2);
plot(N+1:2*N,e);
xlabel('Time');
ylabel('Error between the Objects and Outputs');
title('Simulation Error Analysis')
figure(3);
plot(1:N,T,'b-',N+1:2*N,Tt,'r-')
grid on
hold on
plot(1:N,ytrain,'g+',N+1:2*N,Yt,'k+');
hold off
index=101:1101;
figure(5);

subplot(2,1,1)
plot(1:N,T,N+1:2*N,Tt);
xlabel('time(index)'); ylabel('x(t)');
subplot(2,1,2)
plot(1:N,ytrain,'b-',N+1:2*N,Yt,'r-');
xlabel('time(index)'); ylabel('radial basis');
% Isvedami svoriai ir poslinkiai
% W1=net.IW{1,1};
% b1=net.b{1,1};
% W2=net.LW{2,1};
% b2=net.b{2,1};
RMSE = sqrt(mean((Tt-Yt).^2))
toc
figure(6)
histogram(Tt-Yt,10)
xlabel('Paklaidos');
ylabel('Atveju kiekis');
title('Paklaidu histograma')
mape = mean(abs(e./Tt))*100
figure(10);
subplot(4,1,1)
plot(N+1:2*N,Tt,'b','LineWidth',2);
xlabel('Laikas'); ylabel('x(t)');
subplot(4,1,2)
plot(N+1:2*N,Yt,'b','LineWidth',2);
xlabel('Laikas'); ylabel('RBF prognoze');
```

```
subplot(4,1,[3:4])
plot(N+1:2*N,e,'b','LineWidth',2);
xlabel('Laikas');
ylabel('Paklaida');
Studento1=ttest(e);
[kazkas,Studento] = ttest2(Tt,e);
Studento
[kazkas,KolSmirnov]=kstest2(Tt,Yt);
KolSmirnov
```

6 PRIEDAS. RBF NEURONINIO TINKLO APMOKYMO FUNKCIJA

```
function [out1,out2] = newrb1(varargin)
%NEWRB Design a radial basis network.
%
%  Radial basis networks can be used to approximate functions.  <a
href="matlab:doc newrb">newrb</a>
%  adds neurons to the hidden layer of a radial basis network until it
%  meets the specified mean squared error goal.
%
%  <a href="matlab:doc newrb">newrb</a>(X,T,GOAL,SPREAD,MN,DF) takes
these arguments,
%    X      - RxQ matrix of Q input vectors.
%    T      - SxQ matrix of Q target class vectors.
%    GOAL   - Mean squared error goal, default = 0.0.
%    SPREAD - Spread of radial basis functions, default = 1.0.
%    MN     - Maximum number of neurons, default is Q.
%    DF     - Number of neurons to add between displays, default = 25.
%  and returns a new radial basis network.
%
%  The larger that SPREAD is the smoother the function approximation
%  will be.  Too large a spread means a lot of neurons will be
%  required to fit a fast changing function.  Too small a spread
%  means many neurons will be required to fit a smooth function,
%  and the network may not generalize well.  Call NEWRB with
%  different spreads to find the best value for a given problem.
%
%  Here we design a radial basis network given inputs X and targets T.
%
%    X = [1 2 3];
%    T = [2.0 4.1 5.9];
%    net = <a href="matlab:doc newrb">newrb</a>(X,T);
%    Y = net(X)
%
%  See also SIM, NEWRBE, NEWGRNN, NEWPNN.

% Mark Beale, 11-31-97
% Copyright 1992-2011 The MathWorks, Inc.
% $Revision: 1.1.10.6 $ $Date: 2013/10/09 06:34:28 $

%% =======================================================
%  BOILERPLATE_START
%  This code is the same for all Network Functions.

  persistent INFO;
  if isempty(INFO), INFO = get_info; end
  if (nargin > 0) && ischar(varargin{1}) ...
      && ~strcmpi(varargin{1},'hardlim') &&
~strcmpi(varargin{1},'hardlims')
    code = varargin{1};
    switch code
      case 'info',
```

```
        out1 = INFO;
      case 'check_param'
        err = check_param(varargin{2});
        if ~isempty(err), nnerr.throw('Args',err); end
        out1 = err;
      case 'create'
        if nargin < 2, error(message('nnet:Args:NotEnough')); end
        param = varargin{2};
        err = nntest.param(INFO.parameters,param);
        if ~isempty(err), nnerr.throw('Args',err); end
        out1 = create_network(param);
        out1.name = INFO.name;
      otherwise,
        % Quick info field access
        try
          out1 = eval(['INFO.' code]);
        catch %#ok<CTCH>
          nnerr.throw(['Unrecognized argument: ''' code '''']);
        end
    end
  else
    [args,param] = nnparam.extract_param(varargin,INFO.defaultParam);
    [param,err] = INFO.overrideStructure(param,args);
    if ~isempty(err), nnerr.throw('Args',err,'Parameters'); end
    [net,tr] = create_network(param);
    net.name = INFO.name;
    out1 = net;
    out2 = tr;
  end
end

function v = fcnversion
  v = 7;
end

%  BOILERPLATE_END
%% ===========================================================

function info = get_info
  info = nnfcnNetwork(mfilename,'Radial Basis Network',fcnversion, ...
    [ ...
    nnetParamInfo('inputs','Input Data','nntype.data',{0},...
    'Input data.'), ...
    nnetParamInfo('targets','Target Data','nntype.data',{0},...
    'Target output data.'), ...
    nnetParamInfo('goal','Performance Goal','nntype.pos_scalar',0,...
    'Performance goal.'), ...
    nnetParamInfo('spread','Radial basis
spread','nntype.strict_pos_scalar',1,...
    'Distance from radial basis center to 0.5 output.'), ...
    nnetParamInfo('maxNeurons','Maximum number of
neurons','nntype.pos_int_inf_scalar',inf,...
    'Maximum number of neurons to add to network.'), ...
    nnetParamInfo('displayFreq','Display
Frequency','nntype.strict_pos_int_scalar',50,...
    'Number of added neurons between displaying progress at command
line.'), ...
    ]);
end

function err = check_param(param)
  err = '';
end
```

```
function [net,tr] = create_network(param)

  % Data
  p = param.inputs;
  t = param.targets;
  if iscell(p), p = cell2mat(p); end
  if iscell(t), t = cell2mat(t); end

  % Max Neurons
  Q = size(p,2);
  mn = param.maxNeurons;
  if (mn > Q), mn = Q; end


  % Dimensions
  R = size(p,1);
  S2 = size(t,1);

  % Architecture
  net = network(1,2,[1;1],[1; 0],[0 0;1 0],[0 1]);

  % Simulation
  net.inputs{1}.size = R;
  net.layers{1}.size = 0;
  net.inputWeights{1,1}.weightFcn = 'dist';
  net.layers{1}.netInputFcn = 'netprod';
  net.layers{1}.transferFcn = 'radbas';
  net.layers{2}.size = S2;
  net.outputs{2}.exampleOutput = t;

  % Performance
  net.performFcn = 'mse';

  % Design Weights and Bias Values
  warn1 = warning('off','MATLAB:rankDeficientMatrix');
  warn2 = warning('off','MATLAB:nearlySingularMatrix');
  [w1,b1,w2,b2,tr] =
designrb(p,t,param.goal,param.spread,mn,param.displayFreq);
  warning(warn1.state,warn1.identifier);
  warning(warn2.state,warn2.identifier);

  net.layers{1}.size = length(b1);
  net.b{1} = b1;
  net.iw{1,1} = w1;
  net.b{2} = b2;
  net.lw{2,1} = w2;
end

%=========================================================
function [w1,b1,w2,b2,tr] = designrb(p,t,eg,sp,mn,df)

  [r,q] = size(p);
  [s2,q] = size(t);
  b = sqrt(-log(.5))/sp;

  % RADIAL BASIS LAYER OUTPUTS
  P = radbas(dist(p',p)*b);
  PP = sum(P.*P)';
  d = t';
  dd = sum(d.*d)';
```

```
% CALCULATE "ERRORS" ASSOCIATED WITH VECTORS
e = ((P' * d)' .^ 2) ./ (dd * PP');

% PICK VECTOR WITH MOST "ERROR"
pick = findLargeColumn(e);
used = [];
left = 1:q;
W = P(:,pick);
P(:,pick) = []; PP(pick,:) = [];
e(:,pick) = [];
used = [used left(pick)];
left(pick) = [];

% CALCULATE ACTUAL ERROR
w1 = p(:,used)';
a1 = radbas(dist(w1,p)*b);
[w2,b2] = solvelin2(a1,t);
a2 = w2*a1 + b2*ones(1,q);
MSE = mse(t-a2);

% Start
tr = newtr(mn,'perf');
tr.perf(1) = mse(t-repmat(mean(t,2),1,q));
tr.perf(2) = MSE;
if isfinite(df)
  fprintf('NEWRB, neurons = 0, MSE = %g\n',tr.perf(1));
end
flag_stop=plotperfrb(tr,eg,'NEWRB',0);

iterations = min(mn,q);
for k = 2:iterations

  % CALCULATE "ERRORS" ASSOCIATED WITH VECTORS
  wj = W(:,k-1);
  a = wj' * P / (wj'*wj);
  P = P - wj * a;
  PP = sum(P.*P)';
  e = ((P' * d)' .^ 2) ./ (dd * PP');

  % PICK VECTOR WITH MOST "ERROR"
  pick = findLargeColumn(e);
  W = [W, P(:,pick)];
  P(:,pick) = []; PP(pick,:) = [];
  e(:,pick) = [];
  used = [used left(pick)];
  left(pick) = [];

  % CALCULATE ACTUAL ERROR
  w1 = p(:,used)';
  a1 = radbas(dist(w1,p)*b);
  [w2,b2] = solvelin2(a1,t);
  a2 = w2*a1 + b2*ones(1,q);
  MSE = mse(t-a2);

  % PROGRESS
  tr.perf(k+1) = MSE;

  % DISPLAY
  if isfinite(df) & (~rem(k,df))
    fprintf('NEWRB, neurons = %g, MSE = %g\n',k,MSE);
    flag_stop=plotperfrb(tr,eg,'NEWRB',k);
```

```
    end

    % CHECK ERROR
    %if (MSE < eg), break, end
    if (MSE > eg), break, end
    if (MSE < eg)
        eg=MSE;
    end

    if (flag_stop), break, end
    %MSE=eg;

  end

  [S1,R] = size(w1);
  b1 = ones(S1,1)*b;

  % Finish
  if isempty(k), k = 1; end
  tr = cliptr(tr,k);
end

%=========================================================

function i = findLargeColumn(m)
  replace = find(isnan(m));
  m(replace) = zeros(size(replace));
  m = sum(m .^ 2,1);
  i = find(m == max(m));
  i = i(1);
end

%=========================================================

function [w,b] = solvelin2(p,t)
  if nargout <= 1
    w= t/p;
  else
    [pr,pc] = size(p);
    x = t/[p; ones(1,pc)];
    w = x(:,1:pr);
    b = x(:,pr+1);
  end
end

%=========================================================

function stop=plotperfrb(tr,goal,name,epoch)

  % Error check: must be at least one argument
  if nargin < 1, error(message('nnet:Args:NotEnough')); end

  % NNT 5.1 Backward compatibility
  if (nargin == 1) && ischar(tr)
    stop = 1;
    return
  end

  % Defaults
  if nargin < 2, goal = NaN; end
  if nargin < 3, name = 'Training Record'; end
```

```
if nargin < 4, epoch = length(tr.epoch)-1; end

% Special case 2: Delete plot if zero epochs
if (epoch == 0) || isnan(tr.perf(1))
  fig = find_existing_figure;
  if ~isempty(fig), delete(fig); end
  if (nargout), stop = 0; end
  return
end

% Special case 3: No plot if performance is NaN
if (epoch == 0) || isnan(tr.perf(1))
  if (nargout) stop = 0; end
  return
end

% GET FIGURE AND USER DATA
% ========================

% Get existing/new figure
fig2 = find_existing_figure;
if isempty(fig2), fig2 = new_figure(name); end
figure(fig2);

% Get existing/new userdata
ud=get(fig2,'userdata');
if isempty(ud)
  createNewPlot(fig2);
  ud = get(fig2,'userdata');
end

% UPDATE PLOTTING DATA
% ====================

% Epoch indices and initial y-limits
ind = 1:(epoch+1);
ymax=1e-20;
ymin=1e20;

% Update performance plot and ylimits
set(ud.TrainLine(2),...
    'Xdata',tr.epoch(ind),...
    'Ydata',tr.perf(ind),...
    'linewidth',2,'color','b');
ymax=(max([ymax tr.perf(ind)]));
ymin=(min([ymin tr.perf(ind)]));

% Update performance goal plot and y-limits (if required)
% plot goal only if > 0, or if 0 and ymin is also 0
plotGoal = isfinite(goal) & ((goal > 0) | (ymin == 0));
if plotGoal
  set(ud.TrainLine(1),...
      'Xdata',tr.epoch(ind),...
      'Ydata',goal+zeros(1,epoch+1),...
      'linewidth',2,'color','k');
  ymax=(max([ymax goal]));
  ymin=(min([ymin goal]));
end

% Update axis scale and rounded y-limits
if (ymin > 0)
  yscale = 'log';
```

```
    ymax=10^ceil(log10(ymax));
    ymin=10^fix(log10(ymin)-1);
  else
    yscale = 'linear';
    ymax=10^ceil(log10(ymax));
    ymin=0;
  end
  set(ud.TrainAxes,'xlim',[0 epoch],'ylim',[ymin ymax]);
  set(ud.TrainAxes,'yscale',yscale);

  % UPDATE FIGURE TITLE, NAME, AND AXIS LABLES
  % ==========================

  % Update figure title
  tstring = sprintf('Performance is %g',tr.perf(epoch+1));
  if isfinite(goal)
    tstring = [tstring ', ' sprintf('Goal is %g',goal)];
  end
  set(ud.TrainTitle,'string',tstring);

  % Update figure name
  if isempty(name)
    set(fig2,'name',['Training with '
upper(tr.trainFcn)],'numbertitle','off');
  end

  % Update axis x-label
  if epoch == 0
     set(ud.TrainXlabel,'string','Zero Epochs');
  elseif epoch == 1
     set(ud.TrainXlabel,'string','One Epoch');
  else
     set(ud.TrainXlabel,'string',[num2str(epoch) ' Epochs']);
  end

  % Update axis y-lable
  set(ud.TrainYlabel,'string','Performance');

  % FINISH
  % ======

  % Make changes now
  drawnow;

  % Return stop flag if required
  if (nargout), stop = 0; end
end

%=========================================================

% Find pre-existing figure, if any
function fig = find_existing_figure
  % Initially assume figure does not exist
  fig = [];
  % Search children of root...
  for child=get(0,'children')'
    % ...for objects whose type is figure...
    if strcmp(get(child,'type'),'figure')
      % ...whose tag is 'train'
      if strcmp(get(child,'tag'),'train')
         % ...and stop search if found.
         fig = child;
```

```
      break
    end
   end
  end
  % Not sure if/why this is necessary
  if isempty(get(fig,'children'))
    fig = [];
  end
end

%=======================================================

% New figure
function fig = new_figure(name)
  fig = figure(...
      'Units',          'pixel',...
      'Name',           name,...
      'Tag',            'train',...
      'NumberTitle',    'off',...
      'IntegerHandle',  'off',...
      'Toolbar',        'none');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create new plot in figure

function createNewPlot(fig)
  % Delete all children from figure
  z = get(fig,'children');
  for i=1:length(z)
      delete (z(i));
  end

  % Create axis
  ud.TrainAxes     = axes('Parent',fig);
  ud.TrainLine     = plot(0,0,0,0,0,0,0,0,'Parent',ud.TrainAxes);
  ud.TrainXlabel   = xlabel('X Axis','Parent',ud.TrainAxes);
  ud.TrainYlabel   = ylabel('Y Axis','Parent',ud.TrainAxes);
  ud.TrainTitle    = get(ud.TrainAxes,'Title');
  set(ud.TrainAxes,'yscale','log');
  ud.XData      = [];
  ud.YData      = [];
  ud.Y2Data     = [];
  set(fig,'UserData',ud,'menubar','none','toolbar','none');

  legend(ud.TrainLine(2),'Train');

  % Bring figure to front
  figure(fig);
end

function tr=newtr(epochs,varargin)
  names = varargin;
  tr.epoch = 0:epochs;
  blank = zeros(1,epochs+1)+NaN;
  for i=1:length(names)
    eval(['tr.' names{i} '=blank;']);
  end
end
%=======================================================

function tr=cliptr(tr,epochs)
```

```
  indices = 1:(epochs+1);
  names = fieldnames(tr);
  for i=1:length(names)
    name = names{i};
    value = tr.(name);
    if isnumeric(value) && (numel(value) > epochs)
      tr.(name) = value(:,indices);
    end
  end
end

%=========================================================
```