VILNIAUS PEDAGOGINIS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

TEMA:

KLAIDOS SKLEIDIMO ATGAL ALGORITMO TYRIMAI

Magistro darbas

Darbo vadovas: prof. habil. dr. Gintautas Dzemyda

Atliko: Kęstas Sargelis

Vilnius 2009

Turinys

ĮVADAS **3**
Neuroniniai tinklai **4**
Neuroniniai tinklai – kas tai? 4
Biologinis neurono modelis 6
Neuroninių tinklų elementai 7
Vienasluoksnis perceptronas 10
Daugiasluoksnis tinklas. Neuroninių tinklų klasifikacija 11
Dirbtinio neuroninio tinklo mokymo tipai 13
Klaidos skilidimo atgal algoritmas 14
Neuroninių tinklų konstravimas Matlab sistemoje 17
Tyrimams naudoti duomenys 20
Dirbtinio neuroninio tinklo apmokymo realizacija Matlab 7.1 sistemoje **21**
Bandymas su IRISŲ duomenimis 21
Bandymas su NO_2 duomenimis 24
Dirbtinio neuroninio tinklo apmokymo realizacija Visual Studio Web Developer 2008 sistemoje **27**
IRISŲ duomenys 27
NO_2 duomenys 29
Tyrimai **31**
Su IRISŲ duomenimis atlikti tyrimai 31
Su NO_2 atlikti tyrimai 37
Irisų ir NO_2 palyginimai tarp Matlab ir VS Web Developer sistemų 43
Tinklo testavimas su fiksuotais, apmokyto tinklo, svoriais **47**
IRISŲ duomenys. Tyrimai su fiksuotais, apmokyto tinklo, svoriais 49
NO_2 duomenys. Tyrimai su fiksuotais, apmokyto tinklo, svoriais 55
Palyginama klaida tarp irisų ir NO_2 duomenų VS Web Developer 2008 programoje: 60
IŠVADOS **62**
LITERATŪRA **63**
SANTRAUKA **64**
SUMMARY **65**
PRIEDAI **66**
Priedas 1. Irisų duomenų bazė. 66
Priedas 2. Oro taršos duomenų bazė (NO_2) 70
Priedas 3. Gauti rezultatai. 1 palyginimas 82
Priedas 4. Gauti rezultatai. 2 palyginimas 88
Priedas 5. Sistemos Matlab 7.1 programos kodas 94
Priedas 6. Visual Studio Web Developer 2008 programos kodas 96

ĮVADAS

Paskutiniaisiais metais mokslinėms ir praktinėms problemoms spęsti sėkmingai naudojami dirbtiniai neuroniniai tinklai (DNT). DNT pradėti nagrinėti XX a. šeštajame dešimtmetyje, tačiau iki devintojo dešimtmečio vidurio jie nebuvo plačiai naudojami. Tik išradus greitus ir galingus mokymo mechanizmus, DNT galėjo spręsti realius uždavinius. Šiuo metu DNT naudojami tokiose žmogaus veiklos srityse kaip signalų analizė, triukšmo eliminavimas, duomenų klasifikavimas, sistemų modeliavimas ir identifikavimas, prognozė bei kontrolė. DNT – tai informacijos apdorojimo struktūros, netiksliai imituojančios kai kuriuos gyvųjų organizmų smegenyse vykstančius informacijos apdorojimo procesus.

Šiame darbe analizuojamas dirbtinio neuroninio tinklo mokymo algoritmas – klaidos skleidimo atgal algoritmas, bei jo realizacija programoje Visual Studio Web Developer 2008.

Darbo tikslas - išanalizuoti klaidos skleidimo atgal mokymo algoritmą ir atlikti tyrimus.

Uždavinai tikslo įgyvendinimui:

- Susipažinti su klaidos sklidimo atgal algoritmo teorija;
- Susipažinti su Matlab sistemoje siūlomais klaidos sklidimo atgal algoritmo mokymo įrankiais;
- Sukurti programą klaidos sklidimo atgal algoritmo realizavimui keliais tinklo apmokymo metodais ir ištirti jų efektyvumą;
- Palyginti Matlab sistemoje gautus rezultatus su programos rezultatais.
- Palyginti Visual Studio Web Developer 2008 programoje gautus rezultatus.

Neuroniniai tinklai

Neuroniniai tinklai – kas tai?

Pastaraisiais metais vis svarbesnės tampa intelektinės sistemos, kurios plačiąja prasme remiasi programiniais skaičiavimais. Tradicinio aparatinio skaičiavimo operacijoms būdingas tikslumas ir apibrėžtumas, tuo tarpu programiškai skaičiuojama su tam tikrais tikslumo nuostoliais. Pastaraisiais metais vis svarbesnės tampa intelektinės sistemos, kurios plačiąja prasme remiasi programiniais skaičiavimais (*soft computing*). Tradicinio aparatinio skaičiavimo (*hard computing*) operacijoms būdingas tikslumas ir apibrėžtumas. Programiniai skaičiavimai imituoja žmogaus suvokimą ir sąmoningumą. Tokios sistemos geba mokytis iš patirties, todėl gali būti taikomos net tose srityse, apie kurias dar nesukaupta tiesioginių žinių. Be to, pasitelkusios lygiagrečias skaičiavimo architektūras, modeliuojančias biologinius procesus, jos gautus įvesties signalus gali susieti su išvesties signalais daug greičiau, nei taikant nuoseklius analitinius metodus. Dažniausiai tokios sistemos remiasi neraiškia logika, skaičiavimais neuroniniais tinklais, bendriniais algoritmais ir tikimybiniu pagrindimu. Taigi metodologijos prasme jos yra hibridinės. Daug dešimtmečių mokslininkai siekė sukurti mašinas, sudarytas iš daugybės paprastų komponentų. Užuominų šia tema galima rasti net XIX amžiaus mokslinėje literatūroje. Stengdamiesi atkartoti žmogaus smegenų veiklą, praėjusio amžiaus ketvirtojo dešimtmečio tyrinėtojai sukūrė paprastą techninę (vėliau – ir programinę) įrangą biologiniams neuronams ir jų sąveikai modeliuoti. Šeštajame dešimtmetyje, apibendrinus biologinių ir fiziologinių neuronų sampratą, buvo sukurtas pirmasis dirbtinis neuroninis tinklas. Iš pradžių tai buvo elektroninė schema, o vėliau neuroninis tinklas perkeltas į lengviau manipuliuojamą kompiuterinio modeliavimo lygmenį.

Dirbtinių neuroninių tinklų (artificial neural networks) idėja yra kilusi iš neurobiologijos. Jų pagalba yra konstruojamos lygiagrečių jungčių sistemos, sudarytos iš didelio skaičiaus paprastų elementų. Sudėtingos dirbtinio intelekto ir atpažinimo problemos yra srendžiamos, tinkamu būdu parenkant jungčių (svorių) stiprumą. Neuroniniai tinklai yra įdomūs daugeliu aspektų, pavyzdžiui, jų pagalba galima aprašyti aukštosios nervinės veiklos funkcijas (suvokimas ir mastymas), neuropsichologai gali tirti sensorines ir motorines sistemų bei atminties mechanizmus, inžinieriams jie įdomūs kaip įrankis duomenų ir signalų apdorojimo uždaviniams spręsti, fizikai domisi jais modeliuodami saviorganizacijos reškinius, o informatikai pasitelkia neuroninius tinklus, kurdami save mokančias informacijos apdorojimo sistemas. Daugelis neuroninių tinklų yra įtakoti fizikos, statistikos ir optimizavimo teorijos idėjų. Be abejonės, neurininiai tinklai yra tarpdalykinė tyrimų sritis. Vis didėjantį neuroninių tinklų populiarumą sąlygojo gausybė pavyzdžių, kada sudėtingos dirbtinio intelekto ir vaizdų atpažinimo problemos buvo sėkmingai sprendžiamos.

Teoriniu požiūriu neoroniniai tinklai yra elegantiškas būdas modeliuoti sistemas, kurių įėjimo-išėjimo priklausomybė yra nežinoma. Labai dažnai praktikoje yra sutinkamas atvejis, kai neturime pakankamai informacijos apie šią priklausomybę ir/arba negalime pakankamai tiksliai išmatuoti stebimos priklausomybės charakteristikų arba iš viso nėra patenkinamos teorijos, galinčios paaiškinti šią nežinomą priklausomybę. Neuroninių tinklų pagalba nežinoma priklausomybė yra „išmokstama“ iš pavyzdžių aibės, kuri apibūdina šią priklausomybę.

Neuroninių tinklų teoriniai tyrimai iš esmės yra nukreipti dviem kryptimis. Viena iš jų ezistencijos problema: ar tam tikrai neuroninio tinklo architektūrai egzistuoja tokie svoriai, jog neuroninis tinklas galėtų modeliuoti nežinomą priklausomybę norimu tikslumu? Antroji kryptis gvildena neuroninių tinklų apmokymo problemą: tarus, jog optimalus neuroninis tinklas egzistuoja , ar yra įmanoma surasti tą optimalų sprendinį tam tikru tikslumu, panaudojus ribotą (bet fiziškai įmanomą) kompiuterinių resursų, mokymo laiko ir mokymo duomenų kiekį. Ši problema yra dvejopa: ar įmanoma sukonstruoti efektyvų apmokymo metodą sprendiniui rasti ir kaip tinkamai parinkti neuroninio tinklo dydį beimokymo duomenų kiekį, jog problema būtų išspręsta norimu tikslumu. Pastaroji problema yra kertinis klausimas neuroninių tinklų teorijoje. Akivaizdu, jog neuroninio tinklo padidėjimas (t.y. parametrų skaičiaus padidėjimas) paprastai sumažina mokymo paklaidą, bet paklaida naujų (neapmokytų) pavyzdžių atžvilgiu gali padidėti. Taigi, mažos paklaidos žinomiems pavyzdžiams konkuruoja su tiksliu išėjimo numatymu nežinomų pavyzdžių atžvilgiu. Kuriant ir modeliuojant save apmokančias sistemas, be galo svarbu yra žinoti šiuos ypatumus.

Tačiau vėliau dėl daugelio priežasčių vietoj neuroninių tinklų pradėtas naudoti simbolius apdorojantis Von Neumanno kompiuterio tipas. Ir nors septintajame dešimtmetyje vis dar buvo tyrinėjami dirbtiniai neuroniniai tinklai, jiems buvo skiriama per mažai dėmesio. Pastaraisiais metais padaugėjo darbų, kuriuose aprašomas daugiasluoksnių tinklų mokymas, taip pat buvo sukurta matematinė teorija, padedanti suprasti svarbios neuroninių tinklų klasės dinamiką. Neuroniniais tinklais susidomėta dar ir dėl to, kad dabartiniai kompiuteriai daug spartesni nei penktajame ir šeštajame dešimtmečiuose.

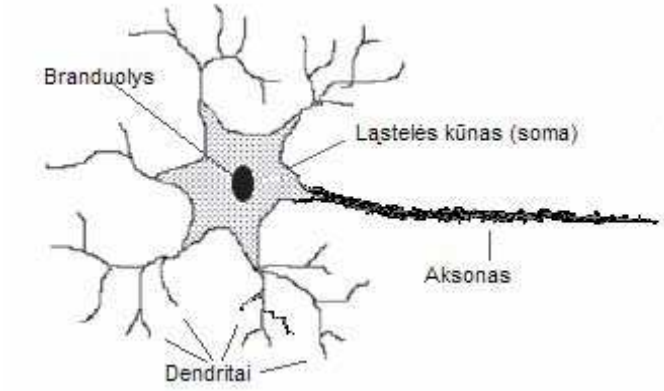
Tyrėjus neuroniniai tinklai domina gebėjimu pamėgdžioti žmogaus smegenų veiklą ir galimybe mokytis bei reaguoti. Prisitaikymas arba mokymasis – pagrindinis neuroninių tinklų tyrimų objektas. Užduotims atlikti neuroniniai tinklai išmoksta įvesčių ir išvesčių rinkinį, o paskui pritaiko savo žinias aproksimuodami arba prognozuodami įvesčių ir išvesčių priklausomybę. Praktinis neuroninių tinklų tyrinėjimas buvo galimas tik atsiradus kompiuteriams, kada teorija galėjo būti patikrinta praktiškai. Pirmas dirbtinis neuroninis tinklas buvo realizuotas 1954 metais Farley ir Clark ant skaitmeninio kompiuterio M.I.T. Jų tinklas susidėjo iš atsitiktinių mazgų, kurie sumuodavo ateinančius signalus, ir svorių. Jeigu suma neviršydavo norimo slenksčio, tai visi svoriai

padidinami, jei suma viršydavo slenkstį, tai visi svoriai sumažinami. Toks tinklas galėjo suskirstyti duomenis į dvi grupes. Paprasčiausios architektūros neuroninio tinklo (perceptrono) strūktūra buvo pasiūlyta 1958 metais Frank‘o Rosenblat‘o (Frank Rosenblatt). Daugelį metų mokslininkai tyrinėjo tik vienasluoknį neuroninį tinklą, tačiau 1982 metais Doug Reilly, Leon Cooper ir Charles Elbaum pristatė dviejų sluoksnių neuroninį tiklą. Atsiradus daugiasluoksniams neuroniniams tinklams, iškilo problema, kaip apmokyti tokį tinklą. 1986 metais trys nepriklausomos tyrėjų grupės pasiūlė tą patį sprendimą – gautą klaidą padalinti visiems neuronams. Toks mokymo būdas buvo pavadintas atgaliniu klaidos sklidimo algoritmu.

Dirbtinių neuroninių tinklų tyrimai vyksta ir šiandien – ieškoma būdų, kaip pagerinti, paspartinti dirbtinio neuroninio tinklo apsimokymą.

Biologinis neurono modelis

Žmogaus nervų sistema – labai sudėtingas neuronų tinklas. Pagrindinis šios sistemos elementas – smegenys, sudarytos iš be galo daug biologinių neuronų, tarpusavyje sujungtų potinkliais. Neuronai (gr. neuron – nervas) – tai ląstelės, kurių dydis ir forma yra įvairūs, tačiau jos visos turi tris dalis: ląstelės branduolį, vieną aksoną ir daugybę dendritų. Dendritas priima signalus iš kitų neuronų. Aksoną galima įsivaizduoti kaip ilgą vamzdelį su atšakomis. Maži tarpai tarp išsišakojusių aksono galų ir dendritų vadinami sinapse.



1 pav. Biologinio neurono sandara

Vieno neurono aksonas sudaro sinapsinius ryšius su daugeliu kitų neuronų. Atsižvelgiant į neurono tipą, sinapsinių ryšių skaičius svyruoja nuo kelių šimtų iki dešimties tūkstančių. Ląstelės

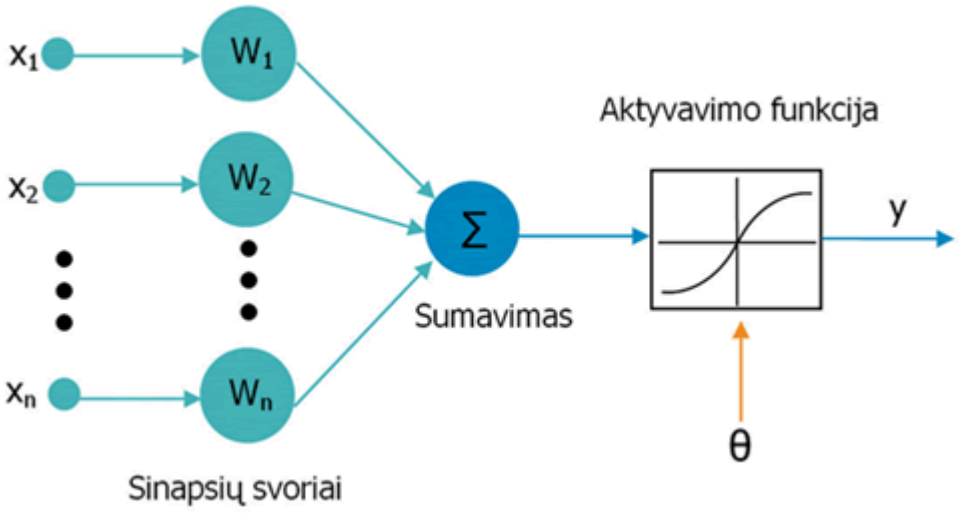
branduolys sumuoja signalus, gautus iš dendritų ir daugybės sinapsių. Neuronas, gavęs pakankamai įvesties signalų, stimuliuojančių neuroną iki slenkstinio (threshold) lygio, išsiunčia impulsą savo aksonui. Tačiau jei įvesties signalai nepasiekia reikiamo slenkstinio lygio, jie greitai nuslopsta taip ir nesukėlę jokių veiksmų.

Anglų mokslininkas Čarlsas Šeringtonas (Charles Sherrington) pastebėjo du nervinio impulso ypatumus. Pirma, impulsas, sklindantis iš vienos nervinės ląstelės į kitą, visada juda tik viena kryptimi. Antra, nervinio impulso perdavimas iš vieno neurono į kitą užtrunka labai mažai. Remiantis šiuo pastebėjimu, iškelta hipotezė, kad tarp neuronų yra labai mažas tarpas. [458 psl. Sylvia S. Mader]

Neuroninių tinklų elementai

Remiantis biologinio neurono modeliu buvo sukonstruotas dirbtinis neuronas (2 pav.). Dirbtinis neuronas yra svarbiausias neuroninio tinklo elementas. Jį sudaro trys pagrindiniai komponentai: svoriai, slenksčiai ir viena aktyvavimo funkcija. W = [W_1, W_2, ..., W_n] yra svorio koeficientai, rodantys stiprumus atskirų įvesčių, aprašytų vektoriumi X = [x_1 x_2 x_3, ..., x_n]. Kiekvienos įvesties signalas dauginamas iš svorio koeficiento. Tokiu būdu gaunama neuroninė jungtis XW. Jei svorio koeficientas teigiamas, XW sužadina signalą išvestyje y, o jei neigiamas – XW slopina išvesties signalą. Vidinis neurono slenkstis veikia neurono y išvesties aktyvavimą tokiu būdu:

$$y = f(\sum_{i=1}^{n} XW - \theta)$$

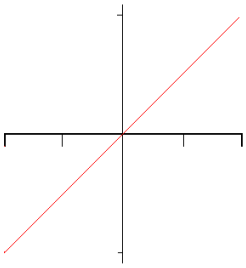


2 pav. Dirbtinio neurono sandara

Aktyvavimo funkcija – tai matematinės operacijos su išvesties signalu. Kokio sudėtingumo aktyvavimo funkcija taikoma, priklauso nuo neuroninio tinklo sprendžiamo uždavinio. Populiariausios – tiesinė, slenksčio, loginis sigmoidas, tangento sigmoidas ir Gauso aktyvavimo funkcijos. Aktyvavimo funkcijos:

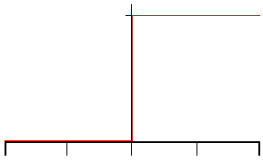
Tiesinė

$$f(x) = x$$



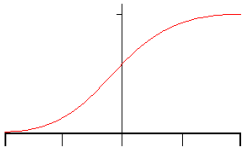
Slenksčio

$$f(x) = \begin{cases} 0, jei\_x < 0 \\ 1, jei\_x \geq 0 \end{cases}$$



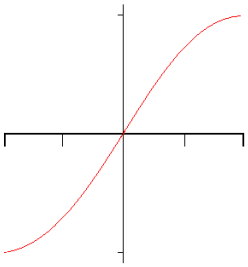
Loginis sigmoidas

$$f(x) = \frac{1}{1+e^{-x}}$$



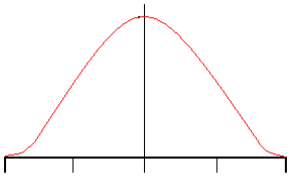
Tangento sigmoidas

$$f(x) = \frac{2}{1+e^{-2x}} - 1$$



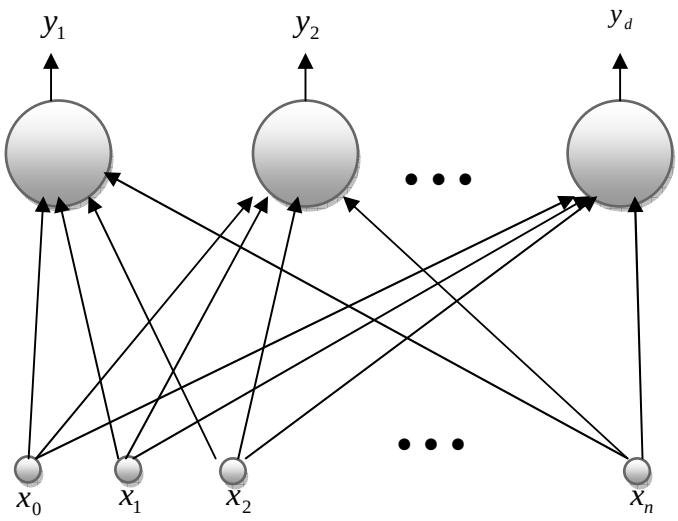
Gauso

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$



Vienasluoksnis perceptronas

Perceptronas – tai vienasluoksnis dirbtinis neuroninis tinklas, kuris sprendžia klasifikavimo uždavinius. Šis neuroninis tinklas susideda iš vieno sluoksnio *d* neuronų, sujungtų su *n* įėjimais. Neuronų skaičius *d* lygus išėjimų skaičiui. [6 sk. 1 psl. G. Dzemyda]



Pav. 3 Perceptrono modelis

Kiekvieno neurono išėjimas y_i yra apskaičiuojamas pagal įėjimo vektoriaus $X = (x_1, x_2, ..., x_n)$ funkciją, kuri skaičiuojama pagal formulę:

$$y_i = f(a_i) = f(\sum_{k=1}^{n} w_{ik} x_k + w_0), \; i = 1, ..., d \qquad (2)$$

Čia w_{ik} jungties iš *k*-tosios įėjimo vektoriaus komponentės į *i*-tąjį išėjimą svoris. Slenksčio reikšmė – w_0.

Tarkime, kad $x_0 = 1$, tada iš (2) formulės gautume:

$$y_i = f(a_i) = f(\sum_{k=0}^{n} w_{ik} x_k), \; i = 1, ..., d \qquad (3)$$

Dirbtinių neuroninių tinklų (tame tarpe ir perceptrono) mokymo metu yra sprendžiamas uždavinys – kaip minimizuoti neuroninio tinklo daromą klaidą. Klaida gali būti parenkama pagal sprendžiamą uždavinį ir neuroninio tinklo tipą. [36 psl. A. Verikas] Dažniausiai yra naudojama suminė kvadratinė klaida, kuri paskaičiuojama pagal formulę:

$$E(W) = \frac{1}{2}\sum_{i=1}^{d}(y_i - t_i)^2 \quad (4)$$

Čia t_i – trokštamos neuroninio tinklo išėjimo reikšmės.

Jeigu paklaidos funkcija *E(W)* yra diferencijuojama pagal svorius w_{ik}, jos minimumas gali būti randamas gradientiniais optimizavimo metodais. [6 sk. 1 psl. G. Dzemyda]

Prieš pradedant apmokyti dirbtinį neuroninį tinklą, atsitiktinai parenkamas pradinių svorių vektorius. Vėliau gradientinio nusileidimo metodu svorių vektorius keičiamas pastumiant jį svorių erdvėje mažu žingsniu ta kryptimi, kuria klaida *E* mažėja greičiausiai. [37 psl. A. Verikas] Svorių reikšmės keičiamos pagal iteracines formules:

$$w_{ik}(m'+1) = w_{ik}(m') + \Delta w_{ik}(m') \quad (5)$$

$$\Delta w_{ik}(m') = -\eta\frac{\partial E(m')}{\partial w_{ik}} = -\eta\sum_{j=1}^{m}\frac{\partial E^j(m')}{\partial w_{ik}} = \sum_{j=1}^{m}\Delta w_{ik}^j(m') \quad (6)$$

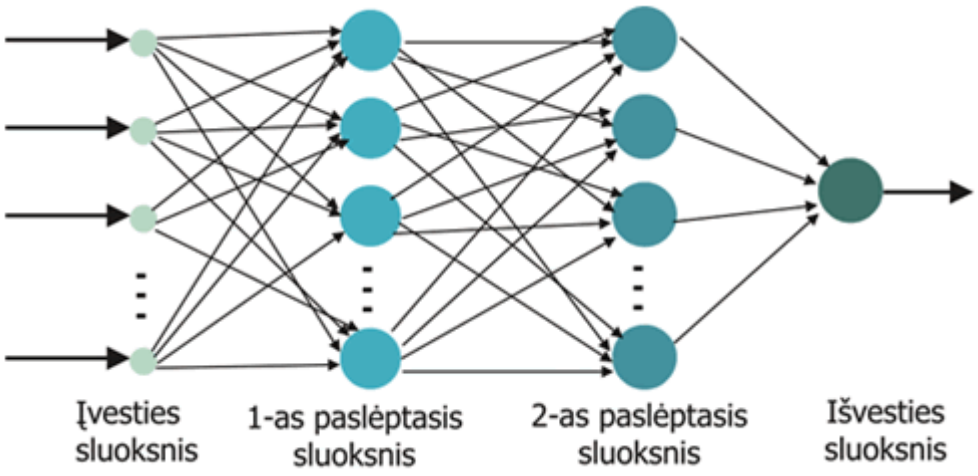
$$\Delta w_{ik}^j(m') = -\eta\frac{\partial E^j(m')}{\partial w_{ik}} \quad (7)$$

Čia η yra mažas teigiamas skaičius vadinamas mokymo greičiu, m‘ – iteracijos numeris. [Medžiaga paimta iš L. Verbylaitės magistrinio darbo „Atgalinio klaidos sklidimo neuroninio tinklo realizavimo problemos ir taikymai“].

Daugiasluoksnis tinklas. Neuroninių tinklų klasifikacija

Skiriami vienasluoksnių ir daugiasluoksnių perceptronų neuroniniai tinklai. Dažniausiai naudojamą daugiasluoksnių perceptronų tipo neuroninį tinklą (4 pav.) sudaro:

- Įvesties sluoksnis –neuronai, priimantys informaciją iš išorinių šaltinių ir siunčiantys ją apdoroti tinklui. Tai gali būti arba jutiklių įvestys, arba tinklo išorėje esančių sistemų siunčiami signalai.
- Paslėptasis sluoksnis –neuronai, priimantys informaciją iš įvesties sluoksnio ir apdorojantys ją tik jiems žinomu būdu. Šis sluoksnis tiesiogiai nesusijęs su išoriniu pasauliu, t.y. jungiasi tik su kitais neuroninio tinklo sluoksniais.
- Išvesties sluoksnis – neuronai, gaunantys apdorotą informaciją ir siunčiantys ją iš neuroninio tinklo.

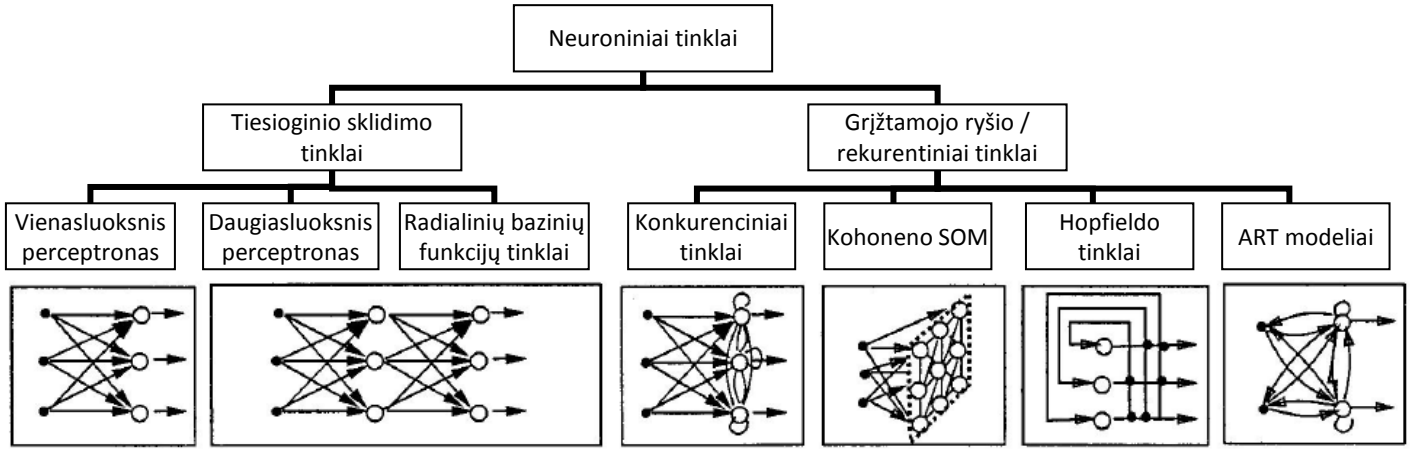


4 pav. Daugiasluoksnių perceptronų tipo neuroninis tinklas

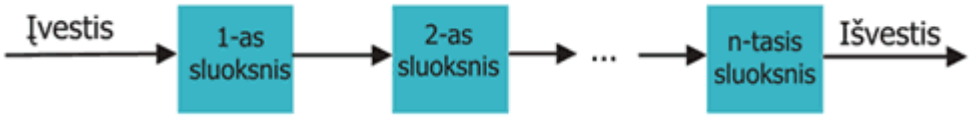
Atsižvelgiant į tai, kokia kryptimi siunčiami signalai, skiriami į dvi grupes:

1. Tiesioginio sklidimo tinklai
2. Grįžtamojo ryšio (arba rekurentiniai) tinklai

Žemiau pateiktas paveikslas, kuriame vaizdžiai parodyta dirbtinių neuroninių tinklų klasifikacija:

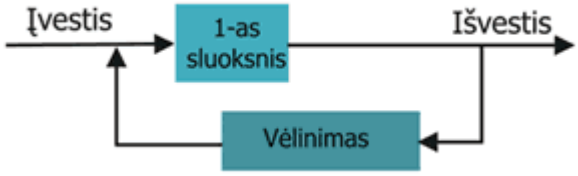


Vienkrypčio ryšio neuroniniuose tinkluose vieno sluoksnio išvestys gali jungtis tik su kito sluoksnio įvestimis. Neegzistuoja ryšiai tarp vieno sluoksnio išvesčių ir to paties ar prieš tai einančio sluoksnio įvesčių. 5 pav. pateiktas vienkrypčio tinklo pavyzdys. Vieno sluoksnio išvestys susijusios su paskui einančio sluoksnio įvestimis. Jeigu šakos svoris lygus nuliui, laikoma, kad tarp šakos jungiamųjų neuronų ryšio nėra. Paskutinio sluoksnio išvestys laikomos tinklo išvestimis.



5 pav. Vienkrypčio ryšio neuroninis tinklas

Grįžtamojo ryšio neuroninio tinklo (6 pav.) įvestis sudaro išorinės įvestys ir paties tinklo išvestis, kuriai būdingas tam tikras vėlinimas. Puikūs grįžtamojo ryšio architektūros pavyzdžiai – Hopfieldo tinklas ir Boltzmano mašina.



6 pav. Grįžtamojo ryšio neuroninis tinklas

Dirbtinio neuroninio tinklo mokymo tipai

Skiriami trys neuroninių tinklų mokymo tipai – mokymas su mokytoju, mokymas be mokytojo ir hibridinis.

Mokymui su mokytoju reikalingas išorinis mokytojas, valdantis mokymosi procesą ir teikiantis informaciją. Tai gali būti mokyti skirtų duomenų rinkinys arba stebėtojas, vertinantis neuroninio tinklo našumą. Prižiūrimo mokymo algoritmams skiriami mažiausio kvadratinio vidurkio, atgalinio skleidimo bei radialinės bazės funkcijos algoritmai. Prižiūrimo mokymo tikslas – priversti neuroninį tinklą pakeisti neuroninių jungčių svorius pagal pavyzdines įvestis ir išvestis. Mokymas baigiamas tinklui išmokus (galima minimali paklaida) sieti įvestis su išvestimis. Svarbus veiksnys – mokymo duomenų aibė, kuri turi būti suprantama ir privalo aprėpti visas praktines tinklo taikymo sritis. Taigi tinklas veiks gerai tik parinkus tinkamą mokymo aibę.

Mokymas be mokytojo neturi išorinio mokytojo. Remdamasi vidiniais kriterijais ir tinklo informacija, sistema pati save turi suderinti. Tokiems neuroniniams tinklams pateikiami tik įvesčių pavyzdžiai, o sistema pati pagal požymius turi suklasifikuoti įvestis. Neprižiūrimo mokymo pavyzdys – Kohoneno tinklai.

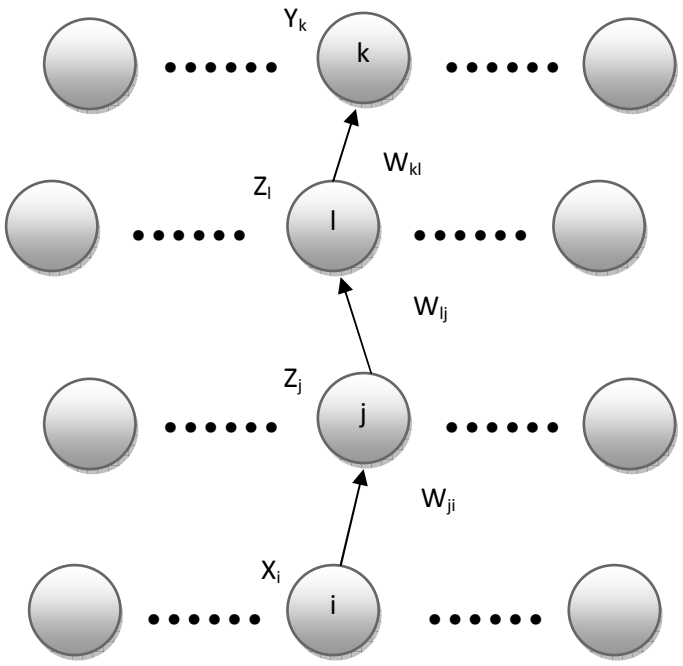
Hibridinis mokymas jungia mokymo su mokytoju ir be mokytojo tipus: dalis tinklo svorių nustatomi pagal mokymą su mokytoju, kita dalis gaunama iš mokymo be mokytojo.

Klaidos skilidimo atgal algoritmas

Nagrinėsime, kaip naudojant mokymo imtį apmokomas neuroninis tinklas. Mokymas grindžiamas parenkant klaidos funkciją ir minimizuojant ją neuroninio tinklo svorių atžvilgiu.
Daugiasluoksniame neuroniniame tinkle susiduriama su tokiomis problemomis:
1. Nėra būdų nustatyti paslėptų neuronų išėjimų norimas (trokštamas) reikšmes, t. y., suformuluoti užduotį, kad konkretus paslėptas neuronas turėtų konkrečią išėjimo reikšmę fiksuotam duomenų rinkiniui.
2. Jei neuroninio tinklo išėjime išduodamas neteisingas rezultatas, nėra būdų nustatyti, kuris iš paslėptų neuronų atsakingas už neteisingą atsakymą, t.y., nėra būdų nustatyti, kiek ir kurį svorį reikia keisti.
Reikia ieškoti būdų, kaip šias problemas spręsti.
Vertingi faktai:
1. Jei neuroninio tinklo neuronų perdavimo funkcijos diferencijuojamos, tai išėjimo sluoksnio neuronų perdavimo funkcijos *yi* yra diferencijuojamos įėjimo kintamųjų (*x*1, *x*2,..., *xn*) ir svorių *wik* atžvilgiu.
2. Jei klaidos funkciją *E* parinksime diferencijuojamą neuroninio tinklo išėjimo *yi* atžvilgiu, tai ši funkcija bus diferencijuojama ir svorių atžvilgiu (ir įėjimo kintamųjų atžvilgiu).
3. Tad galima skaičiuoti klaidos išvestines svorių atžvilgiu ir naudoti jas svorių, minimizuojančių klaidos funkciją, radimui (gradientiniu ar kitais optimizavimo metodais).

Klaidos skleidimo atgal algoritmas (*angl. back propagation error*) taip vadinasi todėl, kad jis „paskirsto“ klaidą tarp neuroninio tinklo elementų pradedant nuo viršaus ir tolyn į apačią.
Taigi, klaidos skleidimo atgal algoritmą sudaro du žingsniai:
1) įėjimų reikšmių „skleidimas pirmyn“ iš įėjimų į išėjimų sluoksnį;
2) paklaidos „skleidimas atgal“ iš išėjimų į įėjimų sluoksnį. [7.1sk. 1 psl. G. Dzemyda]

Klaidos sklidimo atgal algoritme naudojama mokymo su mokytoju strategija. Todėl pradinai duomenys yra mokymo reikšmių vektorius ir trokštamų reikšmių vektorius. Paimkime dalį neurininio tinklo:



Pav. 7 Daugiasluoksnio neuroninio tinklo fragmentas

Tiesioginio sklidimo metu kiekvienas elementas skaičiuoja svorinę įėjimo elementų sumą, pagal (1) formulę. Gauta suma transformuojama netiesine perdavimo funkcija:

1. Įėjimo kintamieji. Suma: $a_j = \sum_i w_{ji} x_i$ Funkcija: $z_j = f(a_j)$ (8), (9)
2. Paslėpti sluoksniai. Suma: $a_l = \sum_j w_{lj} z_j$ Funkcija $z_l = f(a_l)$ (11), (12)
3. Tinklo išėjimas. Suma: $a_k = \sum_l w_{kl} z_l$ Funkcija: $y_k = f(a_k)$ (13), (14)

Čia:

a_j, a_l, a_k - svorinė suma visų neuronų išėjimų;

x_i, z_j, z_l - įėjimo kintamieji arba žemesniam sluoksnyje esančio neurono išėjimo reikšmė;

w_{ji}, w_{lj}, w_{kl} - svoriai tarp nagrinėjamo sluoksnio ir žemesnio sluoksnio neuronų elementų;

z_j, z_l, y_k - neuronų išėjimo reikšmės.

Kaip jau anksčiau minėta dirbtinio neuroninio tinklo mokymo metu stengiamasi minimizuoti klaidą. Klaida apskaičiuojama pagal suminę kvadratinę klaidą (4).

Klaida E^s priklauso nuo svorio w_{ji} tik per svorinę įėjimų į *j-qjį* elementą sumą a_j. Dalinė klaidos išvestinė pagal svorį w_{ji} užrašoma taip:

$$\frac{\partial E^s}{\partial w_{ji}} = \frac{\partial E^s}{\partial a_j} \cdot \frac{\partial a_j}{\partial w_{ji}} \tag{15}$$

Pažymėkime:

$$\delta_j = \frac{\partial E^s}{\partial a_j} \tag{16}$$

Iš (8) formulės turime:

$$\frac{\partial a_j}{\partial w_{ji}} = \frac{\partial(\sum_i w_{ji} z_i)}{\partial w_{ji}} = z_i \tag{17}$$

Todėl:

$$\frac{\partial E^s}{\partial w_{ji}} = \delta_j z_i \tag{18}$$

Jei nagrinėsime apatinį sluoksnį, į kurio įėjimą paduodamas įėjimo reikšmių vektorius, tai

$$\frac{\partial E^s}{\partial w_{ji}} = \delta_j x_i \tag{19}$$

δ yra tam tikros rūšies išėjimo klaida. [7.1 sk. 3 psl. G. Dzemyda]

Dabar paanalizuokime atskirai δ išėjimo mazgams (pažymėkime tai δ_k) ir paslėptų sluoksnių neuronams (pažymėkime tai δ_j).

Žinodami, kad $y_k = f(a_k)$ pagal formulę galime apskaičiuoti klaidą išėjimo mazgams:

$$\delta_k = \frac{\partial E^s}{\partial a_k} = \frac{\partial E^s}{\partial y_k} \cdot \frac{\partial y_k}{\partial a_k} = f'(a_k) \frac{\partial E^s}{\partial y_k} \tag{20}$$

Paslėptiems mazgams:

$$\delta_j = \frac{\partial E^s}{\partial a_j} = \sum_k \frac{\partial E^s}{\partial a_k} \cdot \frac{\partial a_k}{\partial a_j}; \tag{21}$$

Įsistatę $a_k = \sum_j f(a_j) w_{ik}$ ir $\frac{\partial a_k}{\partial a_j} = f'(a_j) w_{kj}$ į (21) formulę gauname:

$$\delta_j = \sum_k \delta_k f'(a_j) w_{kj} = f'(a_j) \sum_k \delta_k w_{kj} \tag{22}$$

Čia s – sekančio sluoksnio (nei nagrinėjama mazgo j sluoksnis) neuronas.

Turėdami išvestines, paskaičiuojam bendrą klaidą pagal formulę:

$$\frac{\partial E}{\partial w_{ji}} = \sum_s \frac{\partial E^s}{\partial w_{ji}} \tag{23}$$

Dabar belieka atnaujinti svorius. Svoriai gali būti keičiami dviem būdais:

1. Po vieno vektoriaus pateikimo:

$$w_{ji}^{naujas} = w_{ji}^{senas} - \eta \frac{\partial E^s}{\partial w_{ji}} = w_{ji}^{senas} - \eta \delta_j z_i \tag{24}$$

2. Po visos imties pateikimo tinklui:

$$w_{ji}^{naujas} = w_{ji}^{senas} - \eta \frac{\partial E}{\partial w_{ji}} = w_{ji}^{senas} - \eta \sum_s \delta_j^s z_i^s \tag{25}$$

Čia η – koeficientas priklausantis intervalui [0;1].

Neuroninių tinklų konstravimas Matlab sistemoje

Neuroninių tinklų įrankinė tai funkcijų rinkinys Matlab programinėje aplinkoje, kuriuo galima kurti, analizuoti, modeliuoti ir redaguoti dirbtinius neuroninius tinklus. Neuroniniai tinklai yra puikus įrankis, kuris padeda išspręsti tokius uždavinius, kur formali analizė nepadeda.

Neuroninių tinklų taikymo pavyzdžiai - vaizdų atpažinimas, netiesinių sistemų identifikavimas, tokių sistemų valdymas ir t.t. Neuroninių tinklų paketas užtikrina daugelį pasiteisinusių neuroninių tinklų paradigmų, taip pat turi grafinę vartotojo sąsają padedančią greitai ir efektingai kurti ir valdyti dirbtinių neuronų tinklus. Grafinė vartotojo sąsaja neuroniniams tinklams kurti iškviečiama panaudojant komanda *nntool*.

Visų pirma norint apmokyti neuroninų tinklą, reikia turėti mokymo duomenis ir trokštamas reikšmes. Matlab sistema suteikia galimybę šiuos duomenis nuskaityti iš tekstinio dokumento, palengvindama duomenų įvedimo procesą. Pagrindinė Matlab sistemoje neuroninių tinklų apmokymo funkcija yra *newff*. Šiai funkcijai pirmasis paduodamas parametras, tai mokymo duomenų kiekvieno stulpelio mažiausia ir didžiausia reikšmės (duomenys iš tekstinio failo). Antrasis parametras nurodo, kiek yra paslėptų sluoksnių ir po kiek neuronų juose. Trečiasis parametras apibrėžia perdavimo funkcijas kiekviename sluoksnyje. Matlab sistemoje perdavimo funkcijos siūlomos trys: loginis sigmoidas (*logsig*), tangento sigmoidas (*tansig*) ir tiesinė (*purelin*). Paskutinysis parametras nurodo mokymo funkciją, bet jis nėra būtinas. Tinklo mokymui naudojama funkcija *train*. Ši funkcija tinklo mokymui naudoja klaidos sklidimo atgal algoritmą. Kreipiantis į funkciją reikia nurodyti savo sukurtą tinklą, mokymo duomenis ir trokštamas reišmes. Taip pat, prieš pradedant mokyti tinklą, galima nustatyti keletą mokymo parametrų (*trainParam*): epochų skaičių (*epoch*), kas kiek epochų parodyti rezultatus ekrane (*show*), norimą pasiekti klaidą (*goal*) ir mokymo konstantą (*lr – learning rate*).

Pavyzdys:

Sukurkime dirbtinį neuroninį tinklą su dviem įėjimų reikšmėmis, vienu paslėptu sluoksniu, penkiais neuronais jame ir išėjime vienas neuronas. Tarkim pirmoji įėjimo rekšmė yra iš intervalo [-3;2], antroji iš intervalo [1;6]. Aktyvavimo funkcijos paslėptuose sluoksniuose ir išėjimo sluoksnyje – tangento sigmoidas.

Aprašome mokymo reikšmes:

mok = [-3 -3 2 2; 1 1 6 6];

Aprašome trokštamas reikšmes:

trok = [-1 -1 1 1];

Sukuriame tinklą su pradiniais svoriais ir tinklo paklaida:

net = newff(minmax(mok), [5,1], {'tansig','tansig'},'traingd');

Apsirašome norimus tinklo apmokymo parametrus:

net.trainParam.epochs = 400;

net.trainParam.show = 25;

net.trainParam.lr = 0.1;

net.trainParam.goal = 0;

Paleidžiam programos vykdymą:

[net,tr] = train(net,mok,trok);

TRAINGD, Epoch 0/400, MSE 2.8257/0, Gradient 3.76011/1e-010

TRAINGD, Epoch 25/400, MSE 0.006579/0, Gradient 0.0524864/1e-010

TRAINGD, Epoch 50/400, MSE 0.00320379/0, Gradient 0.0253704/1e-010

TRAINGD, Epoch 75/400, MSE 0.00212976/0, Gradient 0.016801/1e-010

TRAINGD, Epoch 100/400, MSE 0.00159833/0, Gradient 0.0125815/1e-010

TRAINGD, Epoch 125/400, MSE 0.00128036/0, Gradient 0.0100651/1e-010

TRAINGD, Epoch 150/400, MSE 0.00106844/0, Gradient 0.00839183/1e-010

TRAINGD, Epoch 175/400, MSE 0.000916971/0, Gradient 0.00719786/1e-010

TRAINGD, Epoch 200/400, MSE 0.000803259/0, Gradient 0.00630262/1e-010

TRAINGD, Epoch 225/400, MSE 0.000714718/0, Gradient 0.00560624/1e-010

TRAINGD, Epoch 250/400, MSE 0.000643807/0, Gradient 0.00504893/1e-010

TRAINGD, Epoch 275/400, MSE 0.000585726/0, Gradient 0.00459274/1e-010

TRAINGD, Epoch 300/400, MSE 0.000537277/0, Gradient 0.00421238/1e-010

TRAINGD, Epoch 325/400, MSE 0.000496242/0, Gradient 0.00389037/1e-010

TRAINGD, Epoch 350/400, MSE 0.000461039/0, Gradient 0.00361421/1e-010

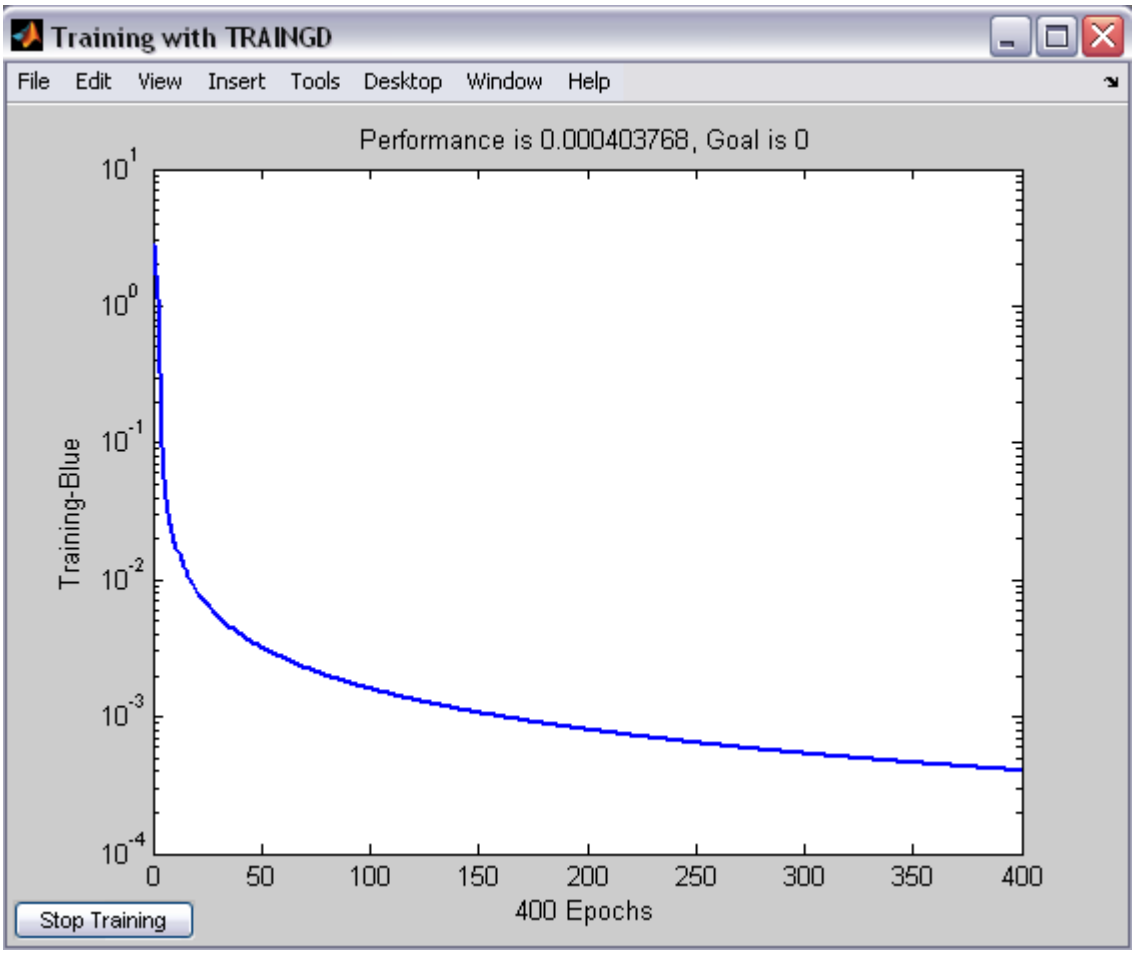
TRAINGD, Epoch 375/400, MSE 0.000430505/0, Gradient 0.00337474/1e-010

TRAINGD, Epoch 400/400, MSE 0.000403768/0, Gradient 0.00316509/1e-010

TRAINGD, Maximum epoch reached, performance goal was not met.

Paskutinėje eilutėje parašoma, dėl kokios priežasties buvo sustabdytas tinklo mokymas. Dirbtinio neuroninio tinklo mokymas gali būti nutrauktas dėl kelių priežasčių: pasiektas apsibrėžtas epochų skaičius, pasiekta norima klaida, gradientinio nusileidimo reikšmė yra mažesnė nei nustatyta minimali gradientinio nusileidimo reikšmė arba apmokymo laikas yra ilgesnis už nustatytąjį.

Kol vyksta tinklo apmokymas ekrane rodomas grafikas su klaidos sklidimu:



Pav. 8 Klaidos kitimo grafikas tinklo apmokymo metu

Norint patikrinti ar tinklas apsimokė Matlab sistema turi funkcija *sim*. Šiai funkcijai reikia nurodyti apmokytą tinklą ir mokymo duomenis, o ji išveda gautus rezultatus, kurie parodo ar tinklas apsimoke gerai ar ne.

A = sim(net,mok)

Tyrimams naudoti duomenys

Gėlių irisų duomenų bazė.

Irisų duomenys buvo surinkti 1988 m. mokslininko R. A. Fisher. Duomenų bazėje pateikiami trijų rūšių irisai – Setosa, Versicolour ir Virginica. Kiekvienos klasės yra po 50 duomenų vektorių, iš viso 150 irisų duomenų vektorių. Kiekvienas duomenų vektorius sudarytas iš keturių parametrų – taurėlapio ilgio, taurėlapio pločio, vainiklapio ilgio ir vainiklapio pločio. Ilgis ir plotis matuotas centimetrais.

Oro taršos keliuose tyrimo duomenų bazė.

500 atliktų stebėjimų duomenų vektoriai yra gauti po atlikto tyrimo kur oro užteršimas keliuose yra susijęs su transporto/eismo kiekiu ir atmosferiniais kintamaisiais, kuriuos gavo Norvegų viešųjų kelių administracija. Kintamasis (1 stulpelyje) susideda iš valandinės vertės koncentracijos logaritmo NO_2 (dalelių) , nustatyto Oslo mieste, Norvegijoje, spalio mėn. 2001 m. - rugpjūčio m. 2003 metų laikotarpiu. Kiti kintamieji (2 - 8 stulpeliuose) yra logaritmai - mašinų kiekis per valandą, temperatūros skaitiklis virš žemės paviršiaus (laipsniais C), vėjo stiprumas (metras/sekundę), temperatūros skirtumas (laipsniais C), vėjo kryptis (laipsniai tarp 0 ir 360), valandų per dieną ir dienų skaičius nuo 2001 metų spalio 1 dienos.

Dirbtinio neuroninio tinklo apmokymo realizacija Matlab 7.1 sistemoje

Kaip realizuojamas neuroninio tinklo apmokymas Matlab sistemoje? Vienas iš šio darbo uždavinių išnagrinėti Matlab sistemoje apmokyto neuroninio tinklo realizavimą. Taigi su atitinkamais duomeninimis buvo vykdomas tinklo apmokymo procesas.

Bandymas su IRISŲ duomenimis

Visų pirma sukuriamas M tipo failas (nurodoma priede) su jame aprašytu algoritmu neuroniniui tinklui apmokyti. Į programos algoritmą įeina: mokymo duomenų nuskaitymas, mokymo duomenų normavimas pagal formulę:

$$\frac{X^i - \min_{(x^1,x^2,...,x^m)}}{\max_{(x^1,x^2,...,x^m)} - \min_{(x^1,x^2,...,x^m)}} \qquad (26)$$

(normavimas būtinas, nes loginio sigmoido reikšmių aibė turi patekti į intervalą [0;1]), trokštamų reikšmių nuskaitymas, trokštamų reikšmių normavimas. Toliau aprašomas tinklo kūrimas su Matlab'o funkcija *newff* :

```
net=newff(minmax(p1),[y,x],{'logsig'});
```

Nustatomi norimi parametrai neuroninio tinklo apmokymui:

```
net.trainParam.epochs = 400;
net.trainParam.show = 25;
net.trainParam.lr = 0.1;
```

Toliau funkcijos *train* pagalba apmokomas tinklas:

```
[net,tr]=train(net,p1,t1);
```

Gautų rezultatų po tinklo apmokymo patikrinimui naudojama funkcija *sim.*

```
D = sim(net,p1);
```

Apmokius tinklą atnormuojami gauti duomenys pagal formule:

$$\frac{y - y_{\min}}{y_{\max} - y_{\min}} = \frac{\bar{y} - 0{,}1}{0{,}9 - 0{,}1} \Rightarrow y = \frac{\bar{y} - 0{,}1}{0{,}9 - 0{,}1}(y_{\max} - y_{\min}) + y_{\min} \qquad (27)$$

Čia $\bar{y}$ - reikšmė, kurią gauna apsimokęs tinklas

y_{min}, y_{max} – trokštamų reikšmių minimumas ir maksimumas

y – atnormuota reikšmė

Gauti rezultatai atvaizduojami grafiškai.

TRAINLM, Epoch 0/400, MSE 0.265176/0, Gradient 3.24456/1e-010

TRAINLM, Epoch 25/400, MSE 4.18345e-005/0, Gradient 0.0628174/1e-010

TRAINLM, Epoch 50/400, MSE 9.01748e-006/0, Gradient 0.00402495/1e-010

TRAINLM, Epoch 75/400, MSE 4.76939e-006/0, Gradient 0.00296603/1e-010

TRAINLM, Epoch 100/400, MSE 3.22763e-006/0, Gradient 0.00386681/1e-010

TRAINLM, Epoch 125/400, MSE 1.51075e-006/0, Gradient 0.00754886/1e-010

TRAINLM, Epoch 150/400, MSE 1.0542e-006/0, Gradient 0.00148469/1e-010

TRAINLM, Epoch 175/400, MSE 8.74359e-007/0, Gradient 0.001932/1e-010

TRAINLM, Epoch 200/400, MSE 7.28356e-007/0, Gradient 0.00450985/1e-010

TRAINLM, Epoch 225/400, MSE 6.02772e-007/0, Gradient 0.0033763/1e-010

TRAINLM, Epoch 250/400, MSE 5.32147e-007/0, Gradient 0.000906609/1e-010

TRAINLM, Epoch 275/400, MSE 4.91333e-007/0, Gradient 0.000543592/1e-010

TRAINLM, Epoch 300/400, MSE 4.64563e-007/0, Gradient 0.000501523/1e-010

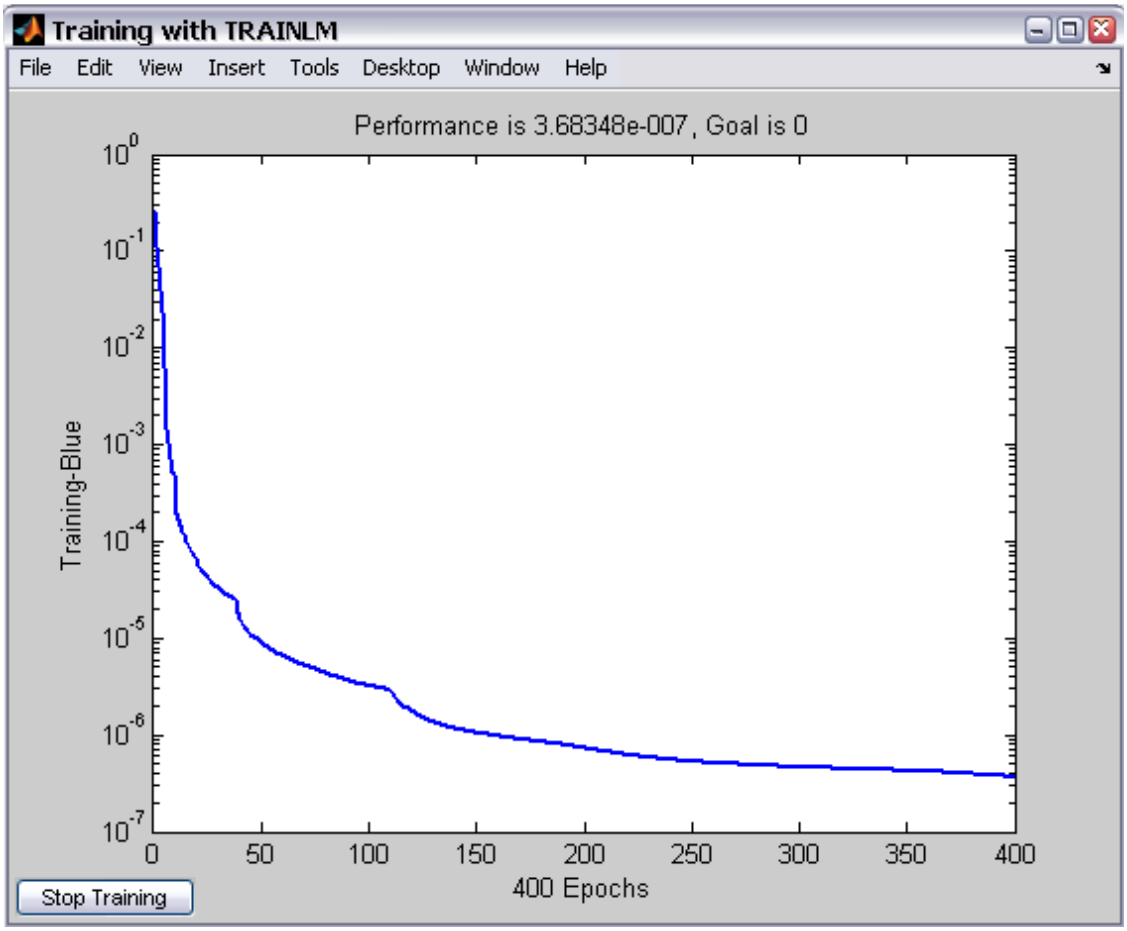
TRAINLM, Epoch 325/400, MSE 4.44562e-007/0, Gradient 0.000781585/1e-010

TRAINLM, Epoch 350/400, MSE 4.26233e-007/0, Gradient 0.00132354/1e-010

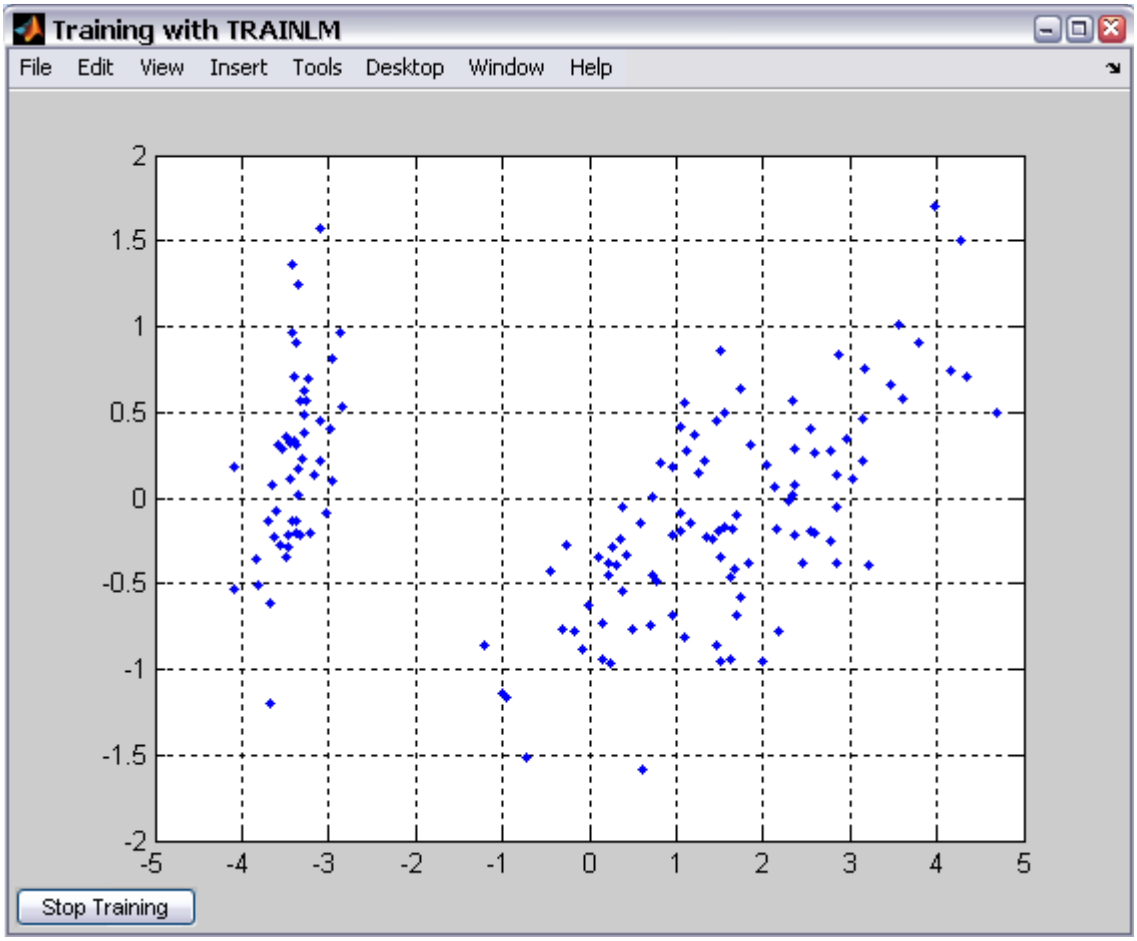
TRAINLM, Epoch 375/400, MSE 4.01009e-007/0, Gradient 0.00274997/1e-010

TRAINLM, Epoch 400/400, MSE 3.68348e-007/0, Gradient 0.0029916/1e-010

TRAINLM, Maximum epoch reached, performance goal was not met.

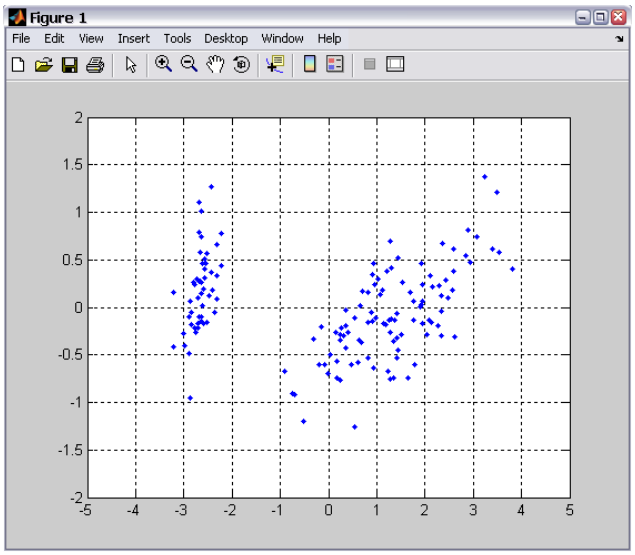


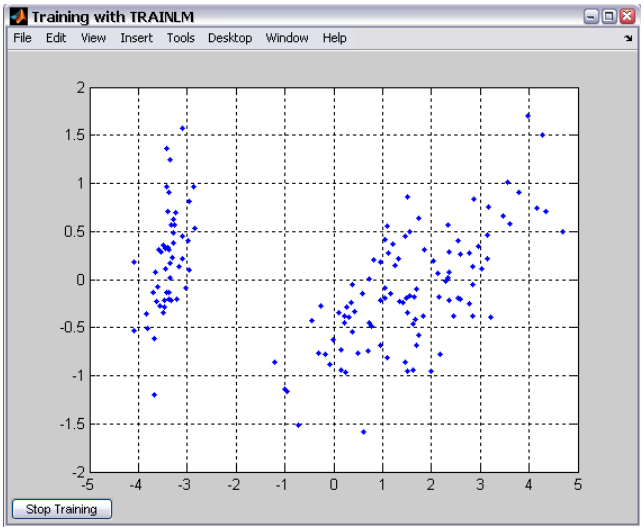
Pav 9. Klaidos kitimo grafikas



Pav. 10 Gautų duomenų grafikas

Atlikus vieną bandymą per 24.42 sek. su 400 epochų ir mokymo konstanta 0.1 matomas klaidos kitimo grafikas (9 pav.) ir kaip apsimokė neuroninis tinklas (10 pav.). 11 pav. Pateikiamas trokštamų ir gautų reikšmių grafikų palyginimas.





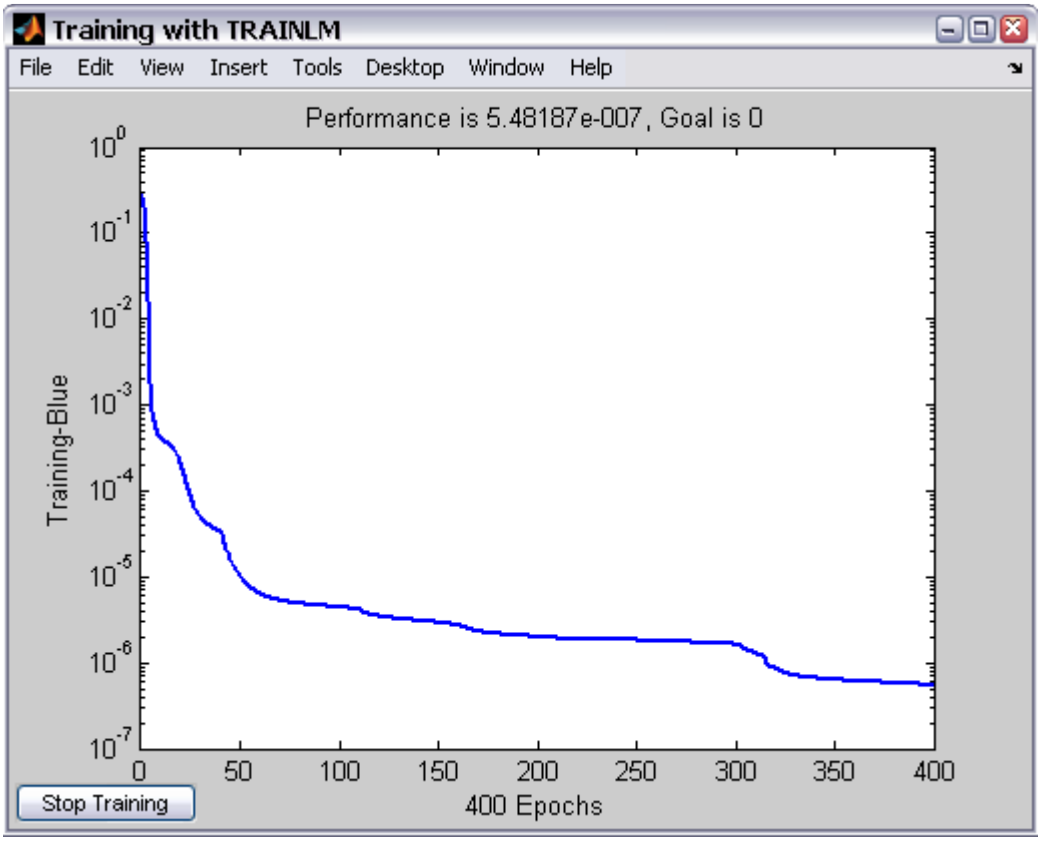
Trokštamų reikšmių grafikas **Reikšmės gautos po apmokymo**

11 pav.

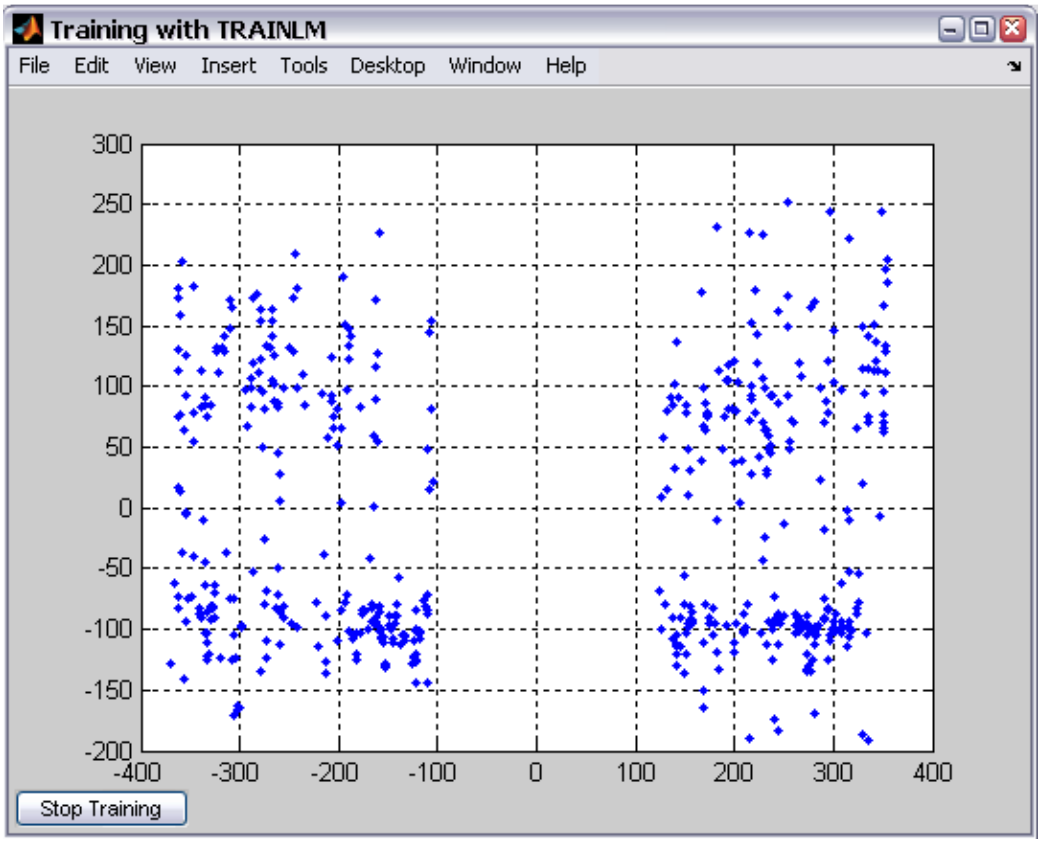
Bandymas su NO_2 duomenimis

Atlikime šį bandymą su kitais, didesniais duomenimis ir pažiūrėkime kaip apsimokys tinklas. Tame pačiame M faile aprašykime naujų duomenų parametrus, kad programa galėtų juos apdoroti. Tinklo apmokymui naudosime vieną paslėptą sluoksnį su jame paslėptais 15 neuronų, atliksime 400 iteracijų, konstanta pasirenkame 0,1. Gauti rezultatai vaizduojami grafiškai:

TRAINLM, Epoch 0/400, MSE 0.284572/0, Gradient 8.39167/1e-010
TRAINLM, Epoch 25/400, MSE 8.84412e-005/0, Gradient 0.270282/1e-010
TRAINLM, Epoch 50/400, MSE 1.02191e-005/0, Gradient 0.0316733/1e-010
TRAINLM, Epoch 75/400, MSE 5.03781e-006/0, Gradient 0.00657433/1e-010
TRAINLM, Epoch 100/400, MSE 4.41004e-006/0, Gradient 0.00354264/1e-010
TRAINLM, Epoch 125/400, MSE 3.33955e-006/0, Gradient 0.0102694/1e-010
TRAINLM, Epoch 150/400, MSE 2.93328e-006/0, Gradient 0.00292732/1e-010
TRAINLM, Epoch 175/400, MSE 2.21441e-006/0, Gradient 0.00269535/1e-010
TRAINLM, Epoch 200/400, MSE 1.99394e-006/0, Gradient 0.00819142/1e-010
TRAINLM, Epoch 225/400, MSE 1.87362e-006/0, Gradient 0.000559315/1e-010
TRAINLM, Epoch 250/400, MSE 1.8199e-006/0, Gradient 0.00112177/1e-010
TRAINLM, Epoch 275/400, MSE 1.73449e-006/0, Gradient 0.00131407/1e-010
TRAINLM, Epoch 300/400, MSE 1.62764e-006/0, Gradient 0.000805194/1e-010
TRAINLM, Epoch 325/400, MSE 7.66998e-007/0, Gradient 0.0280966/1e-010
TRAINLM, Epoch 350/400, MSE 6.31134e-007/0, Gradient 0.00340029/1e-010
TRAINLM, Epoch 375/400, MSE 5.85913e-007/0, Gradient 0.000891131/1e-010
TRAINLM, Epoch 400/400, MSE 5.48187e-007/0, Gradient 0.00293633/1e-010
TRAINLM, Maximum epoch reached, performance goal was not met.



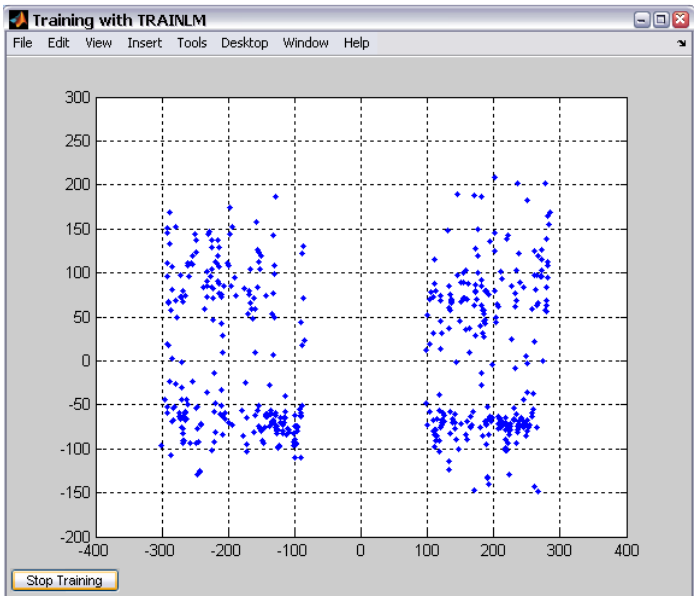
Pav 12. Klaidos kitimo grafikas



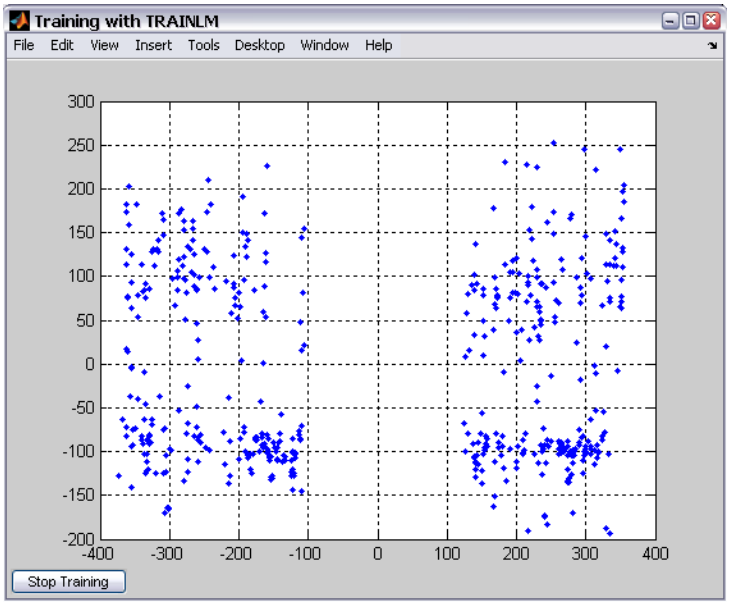
Pav. 13 Gautų duomenų grafikas

Atlikus skaičiavimus per 20.3438 sekundes su 400 epochų, konstanta 0,1, su vienu paslėptu neuronų sluoksniu ir 15 neuronų jame, gavome tokį reikšmių grafiką (13 pav.).

Trokštamų ir gautų reikšmių palyginimas pateikiamas 14 pav.



Trokštamų reikšmių grafikas



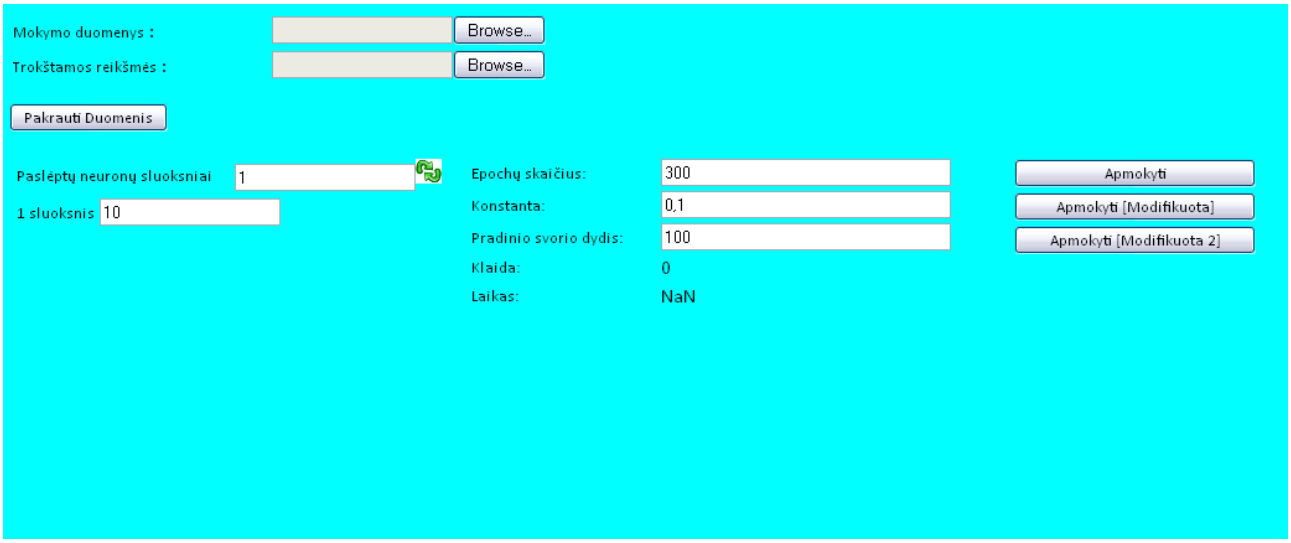
Gautų reikšmių grafikas

14 pav.

Dirbtinio neuroninio tinklo apmokymo realizacija Visual Studio Web Developer 2008 sistemoje

Išnagrinėjus klaidos sklidimo atgal algoritmą buvo sukurta programa dirbtiniui neuroniniui tinklui apmokyti. Dirbtinio neuroninio tinklo apmokymas Visual Studio Web Developer 2008 aplinkoje vyksta panašiai kaip ir Matlab sistemoje. Kaip ir Matlab sistemoje Web programoje nurodomi parametrai pagal kuriuos bus apmokomas tinklas: paslėptų sluoksnių skaičius, neuronų skaičius kiekviename paslėptame sluoksnyje, epochų (iteracijų) skaičius, konstanta, pradinio svorio dydis. Atlikus skaičiavimus programa apskaičiuoja apmokyto tinklo daromą klaidą, ir per kiek laiko tinklas apsimokė.

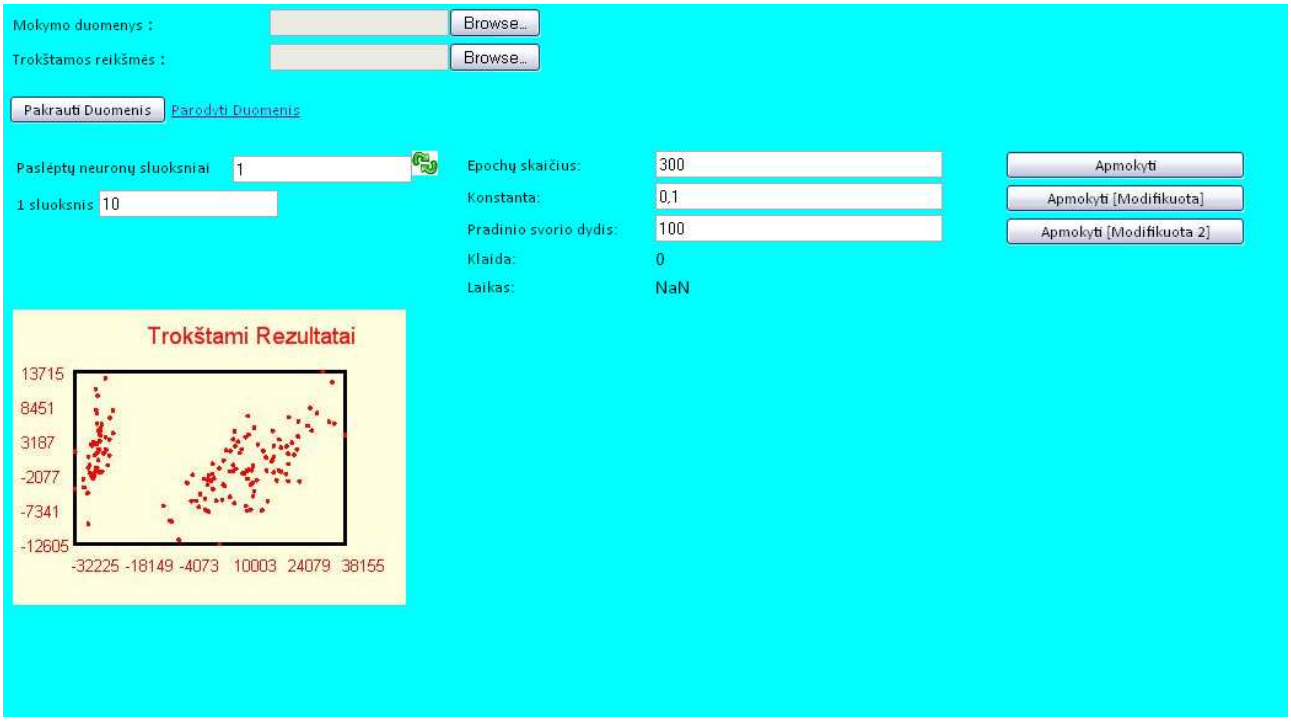
Programos pagrindinis vaizdas:



15 pav. Programos pagrindinis langas

IRISŲ duomenys

Skiltyje „Mokymo duomenys“ įkeliame duomenis, kuriuos apmokysime. Trokštamų reikšmių skiltyje įkeliame trokštamas reikšmes. Sekančiai nurodome norimą paslėptų sluoksnių skaičių, kiekviename jų neuronų skaičių, tada epochų skaičių, konstantą ir pradinio svorio dydį. Pavaizduokime konkretų atvejį su Iriso duomenimis. Kaip ir Matlab sistemoje įkelkime duomenis kartu su trokštamomis reikšmėmis. Nurodykime vieną paslėptą sluoksnį su jame esančiais 15 neuronų. Epochų skaičių nurodome 300, konstantą 0,3, pradinio svorio dydį paliekame 100. Programa mums išveda tokį vaizdą:



16 pav. Programos langas įkėlus Irisų duomenis

Kai visi duomenys įvesti paleidžiame tinklą apsimokyti.

17 pav. Programos langas apmokius tinklą

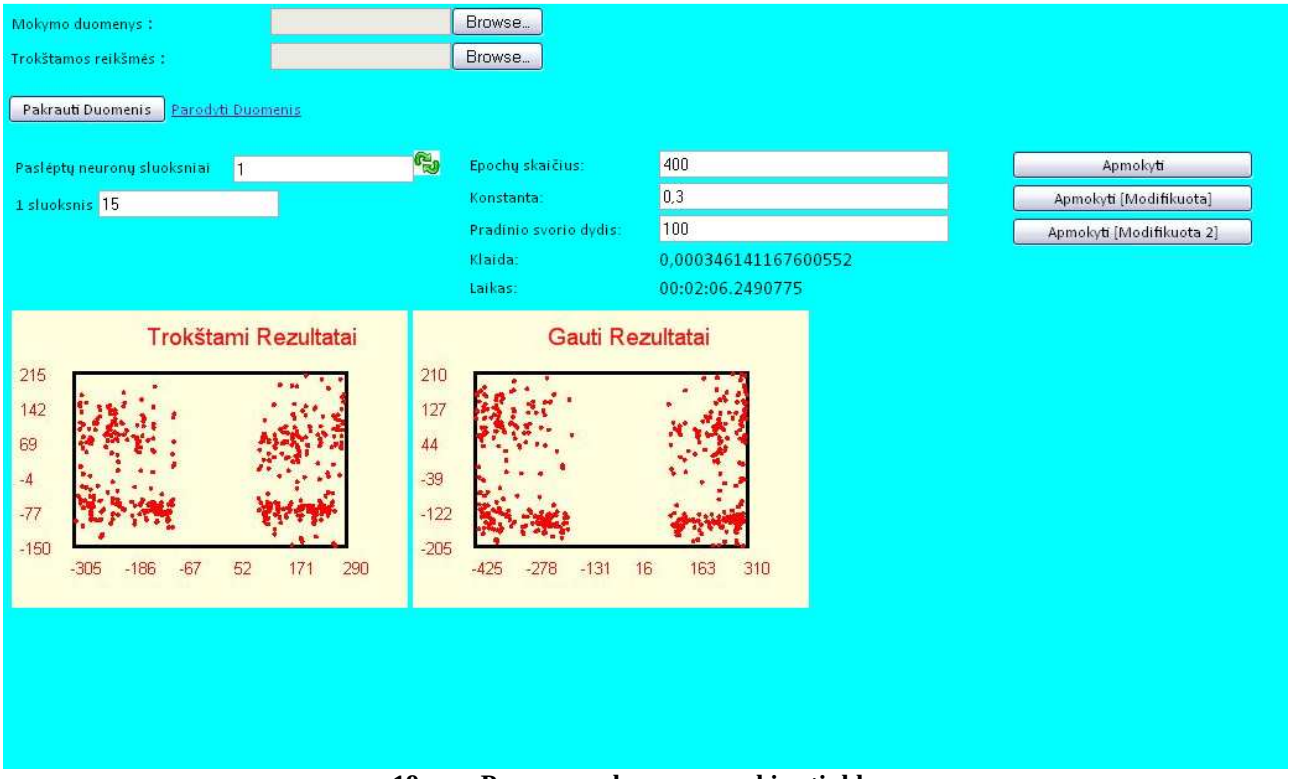
Atlikus 400 epochų, kai konstanta 0,3 per 36 sekundes gavome klaidą 0,0005309958162675. Iš grafiko matyti, kad tinklas apsimokė neblogai.

NO_2 duomenys

Taip pat kaip ir su irisų duomenimis pasirenkame atitinkamus parametrus ir įkeliame duomenis.

18 pav. Programos langas įkėlus Irisų duomenis

Vaizdas tinklui apsimokius.



19 pav. Programos langas apmokius tinklą

Atlikus 400 epochų, kai konstanta 0,3 matome, jog tinklas dėl didelio duomenų kiekio mokėsi kur kas ilgiau nei su irisų duomenimis. Tinklas apsimokė per 2min 6 sek., ir gavosi klaida lygi 0,000346141167600552.

Tyrimo metu buvo bandyta apmokyti tinklą su daugiau nei vienu paslėptu sluoksniu, tačiau eksperimento metu ši mokymo strategija nepasiteisino ir buvo gaunami kur kas blogesni rezultatai, nei buvo tikėtasi, tinklas apsimokydavo prastai. Todėl savo darbe tyrimui naudojau vieną paslėptą sluoksnį keisdamas epochų skaičių, konstantą bei neuronų skaičių paslėptuose sluoksniuose. Stebėjau pagal kuriuos kriterijus tinklas apsimoko geriausiai, t.y. laiko ir klaidos santykis.

Tyrimai

Su IRISŲ duomenimis atlikti tyrimai

Pradžioje pateikiami su irisų duomenų baze atlikti tyrimai, kuriu metu buvo stebima daroma klaida su Matlab sistema ir su VS Web Developer programa. Duomenys pateikti atsižvelgiant į neuronų skaičių, konstantą, daromą klaidą ir laiką abiejuose programose. Kuo mažesnė klaida, tuo tinklas apsimokė geriau, ir atvirkščiai. Trumpai apie kiekvieną tinklo apmokymo metodą:

- „matlab“ – metodas, kai tinklui paduodami vektoriai iš eilės po vieną. Skaičiuojama klaida ir siunčiamas vektorius atgal, atnaujinami svoriai ir paduodamas kitas, naujas vektorius.
- „web1“ – metodas veikia lygiai taip pat kaip ir „matlab“ metodas, tik VS Web Developer programos terpėje.
- „web2“ – metodas, kai paduodamas vienas vektorius, t.y. visų mokymo duomenų vidurkių vektorius.
- „web3“ – metodas, kai paduodamas vienas vektorius, t.y. visų mokymo duomenų sumos vektorius. Prieš sumuojant mokymo vektoriai dauginami iš atsitiktinių koeficientų.

Irisų duomenų palyginimui apžvelgsime mažiausiai daromą klaidą abiejuose programose. Lentelėse pateikiami geriausiai apmokytų tinklų rezultatai, kai svorių atnaujinimo konstanta kinta nuo 0,1 iki 0,5, o epochų skaičius 400. Lentelėse lyginama mažiausia klaida, daroma dviejų vienodų skaičiavimo metodų: „Matlab“ ir „Web1“.

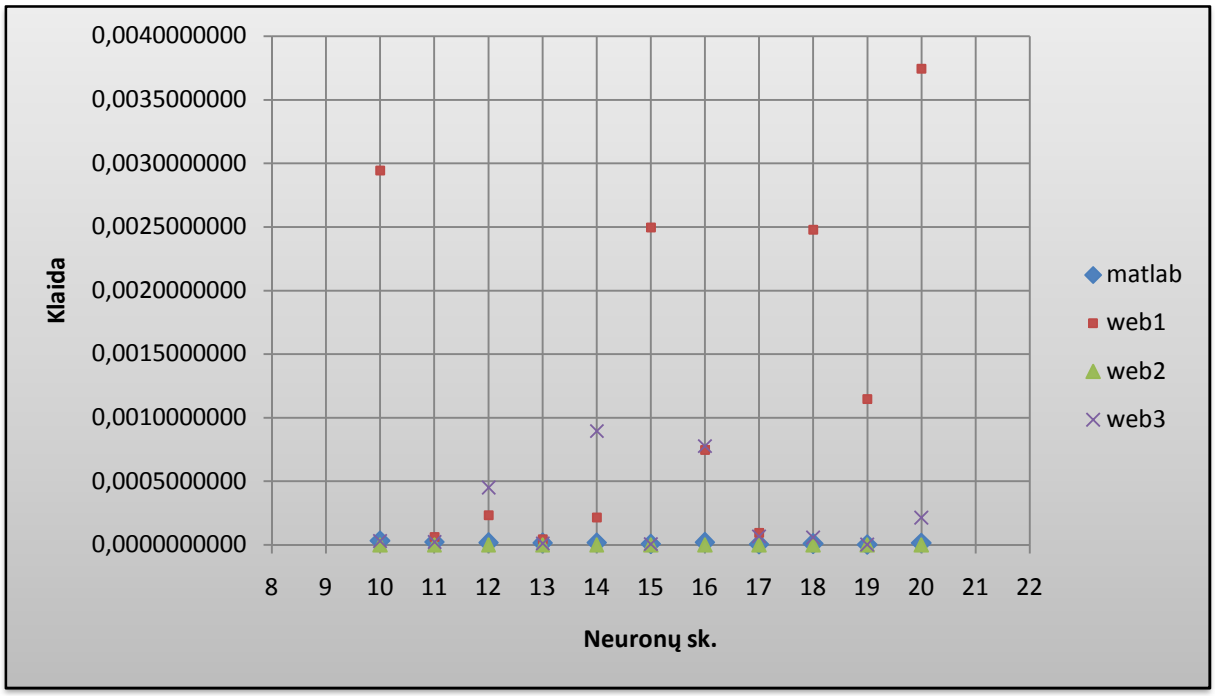
Paryškintuose langeliuose pateikiama mažiausia po apmokymo tinklo daroma klaida, t.y. geriausias tinklo apmokymas.

Kai konstanta 0,1.

Neuronų skaičius	Konstanta	Matlab	Laikas(s)	Web1	Laikas(s)
19	0.1	0,0000036999	11,78	0,0011471844	00:00:50
13		0,0000151249	5,75	0,0000461343	00:00:33

Lentelė 1. Mažiausia daroma klaida.

Grafiškai pateikiamas klaidos išsidėstymas pagal neuronų skaičių paslėptame sluoksnyje pagal visus metodus:



20 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,1 ir epochų skaičius 400.

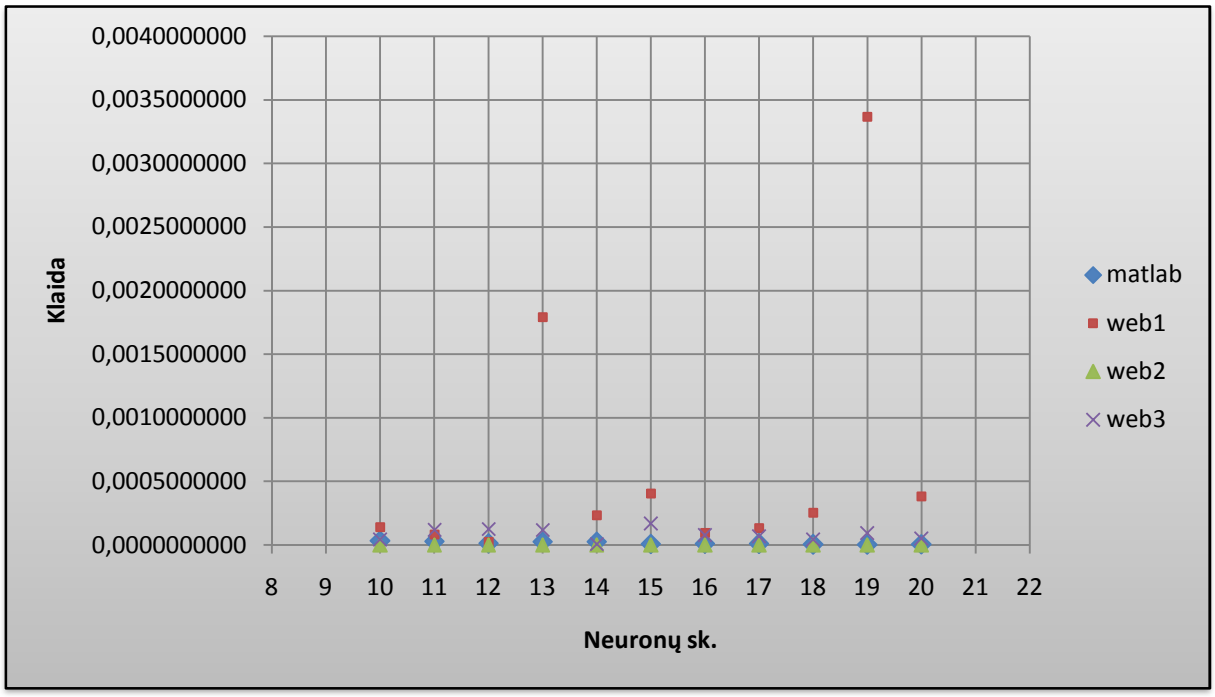
Iš 20 paveikslo matome, kad mažiausią klaidą tinklas daro, kai paslėptame sluoksnyje yra 19 arba 13 neuronų. Kai svorių perskaičiavimui naudojama konstanta 0.1, tai geriausia konstruoti tinklą su paslėptame sluoksnyje esančiais 11 neuronų.

Kai konstanta 0,2.

Neuronų skaičius	Konstanta	Matlab	Laikas(s)	Web1	Laikas(s)
19	0.2	0,0000036421	11,52	0,0033660696	00:00:44
12		0,0000141769	5,16	0,0000257733	00:00:29

Lentelė 2. Mažiausia daroma klaida.

Grafiškai pateikiamas klaidos išsidėstymas pagal neuronų skaičių paslėptame sluoksnyje pagal visus metodus:



21 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,2 ir epochų skaičius 400.

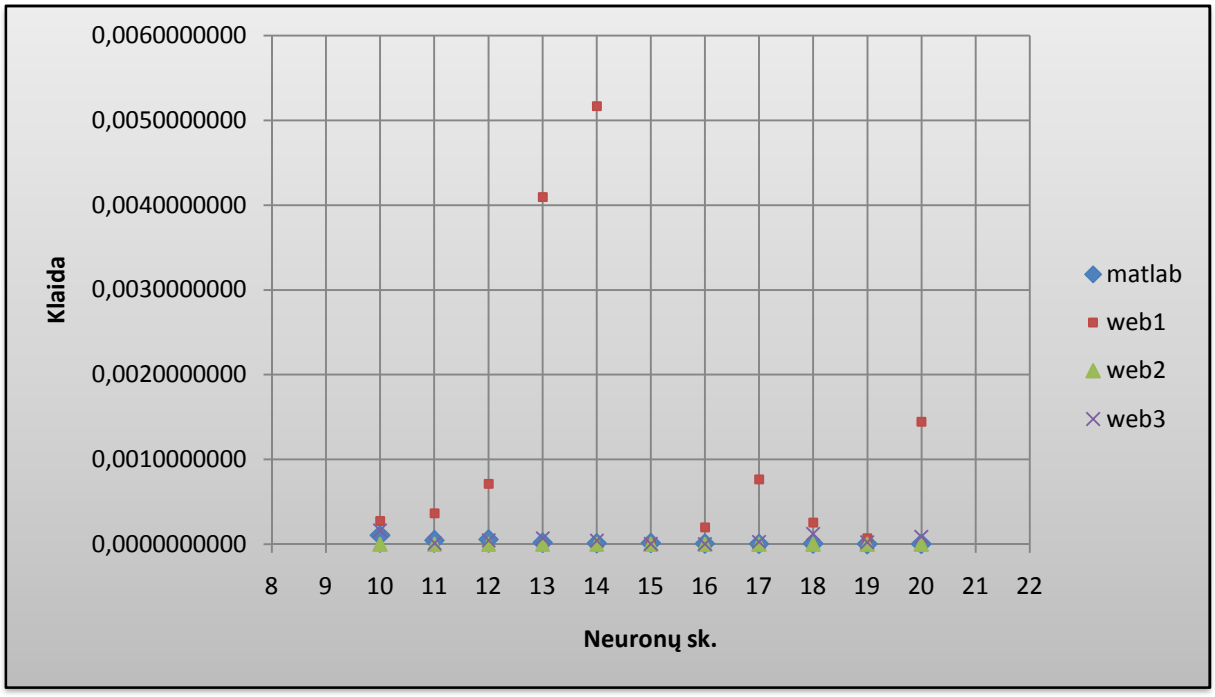
21 paveiksle matyti, jog tinklas geriausiai apsimoko esant konstantai 0,2, kai paslėptame sluoksnyje Matlab 7.1 sistemoje turi 19, o VS Web Developer 2008 – 12 neuronų. Laiko atžvilgiu Matlab 7.1 sistema apsimoko greičiau.

Kai konstanta 0,3.

Neuronų skaičius	Konstanta	Matlab	Laikas(s)	Web1	Laikas(s)
19	0.3	0,0000038777	7,77	0,0000737940	00:00:44
15		0,0000161723	6,50	0,0000112972	00:00:34

Lentelė 3. Mažiausia daroma klaida.

Grafiškai pateikiamas klaidos išsidėstymas pagal neuronų skaičių paslėptame sluoksnyje pagal visus metodus:



22 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,3 ir epochų skaičius 400.

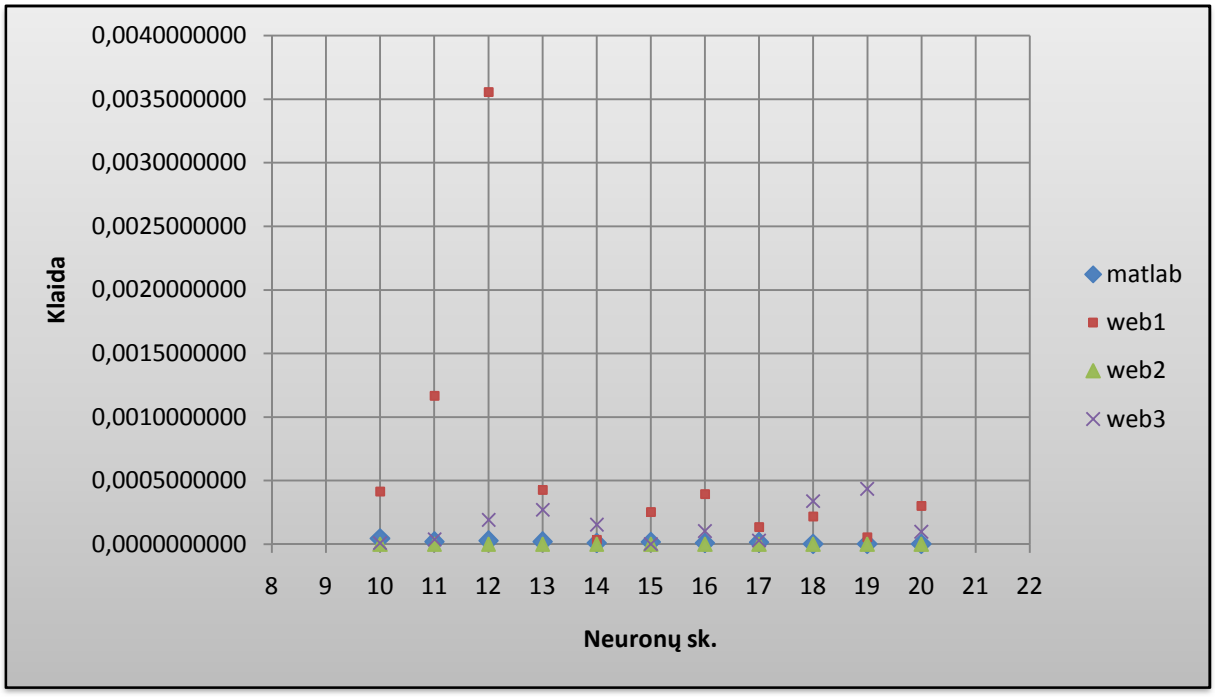
Iš 22 paveikslo matyti, kad geriausiai tinklas apsimoko, kai paslėptame sluoksnyje turi 19 ir 15 neuronų. Konstanta lygi 0,3.

Kai konstanta 0,4.

Neuronų skaičius	Konstanta	Matlab	Laikas(s)	Web1	Laikas(s)
20	0.4	0,0000032723	7,95	0,0003018581	00:00:50
14		0,0000122289	5,92	0,0000363462	00:00:37

Lentelė 4. Mažiausia daroma klaida.

Grafiškai pateikiamas klaidos išsidėstymas pagal neuronų skaičių paslėptame sluoksnyje pagal visus metodus:



23 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,4 ir epochų skaičius 400.

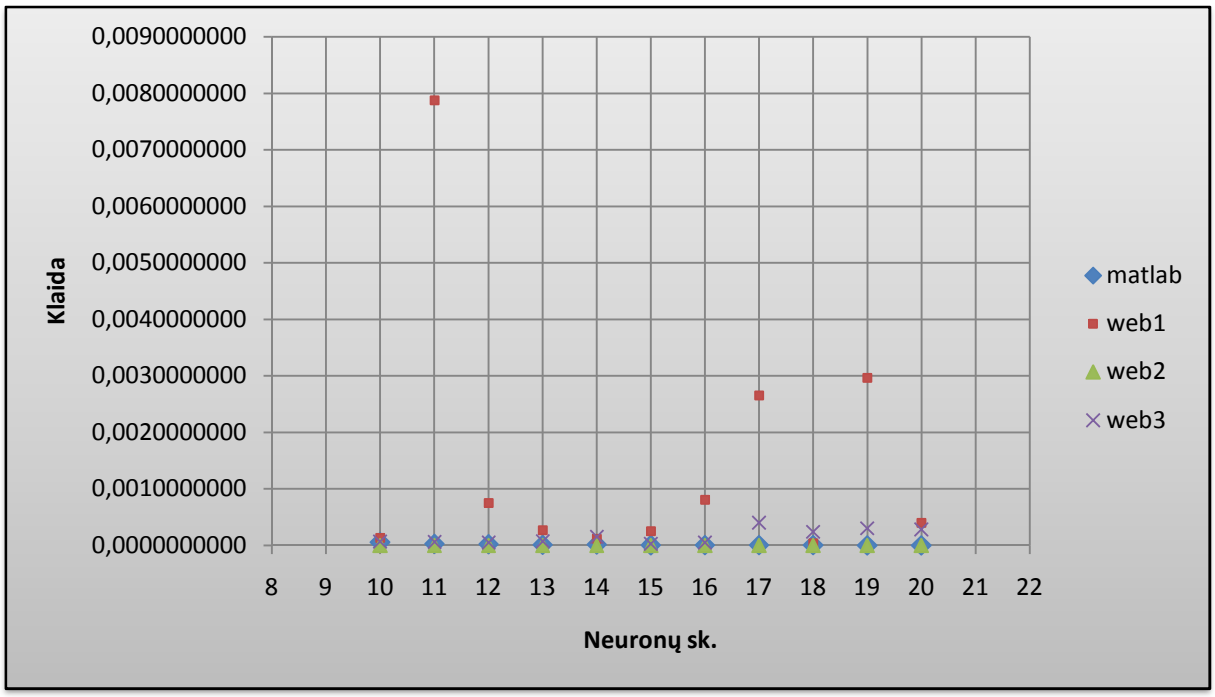
Iš 23 paveikslo ir 9 lentelės matome, kad mažiausią klaidą tinklas daro, kai paslėptame sluoksnyje yra 20 ir atitinkamai 14 neuronų. Tinklas greičiau apsimoko Matlab 7.1 sistemoje ir daro kur kas mažesnę klaidą nei VS Web Developer sistema.

Kai konstanta 0,5.

Neuronų skaičius	Konstanta	Matlab	Laikas(s)	Web1	Laikas(s)
19	0.5	0,0000036247	11,22	0,0029703289	00:00:46
18		0,0000071249	7,05	0,0000399745	00:00:44

Lentelė 5. Mažiausia daroma klaida.

Grafiškai pateikiamas klaidos išsidėstymas pagal neuronų skaičių paslėptame sluoksnyje pagal visus metodus:



24 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,5 ir epochų skaičius 400.

Iš 10 lentelėje pateiktų duomenų matyti, jog tinklas savo konstrukcijoje paslėptame sluoksnyje turintis 19 ir atitinkamai 18 neuronų, kai konstanta 0,5, apsimoko geriausiai. Vėl gi Matlab 7.1 sistema yra lankstesnė skaičiuojant klaidą.

Su NO_2 atlikti tyrimai

Dirbtinis neuroninis tinklas buvo apmokomas naudojant oro taršos duomenis. Lyginama mažiausiai daroma klaida tarp Matlab sistemos daromų klaidų ir Visual Studio Web Developer 2008 programos daromų klaidų. Tinklo apmokymo metodai tie patys:

- „matlab“ – metodas, kai tinklui paduodami vektoriai iš eilės po vieną. Skaičiuojama klaida ir siunčiamas vektorius atgal, atnaujinami svoriai ir paduodamas kitas, naujas vektorius.
- „web1“ – metodas veikia lygiai taip pat kaip ir „matlab“ matodas, tik VS Web Developer programos terpėje.
- „web2“ – metodas, kai paduodamas vienas vektorius, t.y. visų mokymo duomenų vidurkių vektorius.
- „web3“ – metodas, kai paduodamas vienas vektorius, t.y. visų mokymo duomenų sumos vektorius. Prieš sumuojant, mokymo vektoriai dauginami iš atsitiktinių koeficientų.

Lygiai taip pat apžvelgsime NO_2 duomenų mažiausiai daromą klaidą abejose programose. Toliau lentelėse pateikiami geriausiai apmokytų tinklų rezultatai, kai svorių atnaujinimo konstanta kinta nuo 0,1 iki 0,5, o epochų skaičius 400. Lentelėse lyginama mažiausia klaida, daroma dviejų vienodų skaičiavimo metodų: „Matlab“ ir „Web1“.

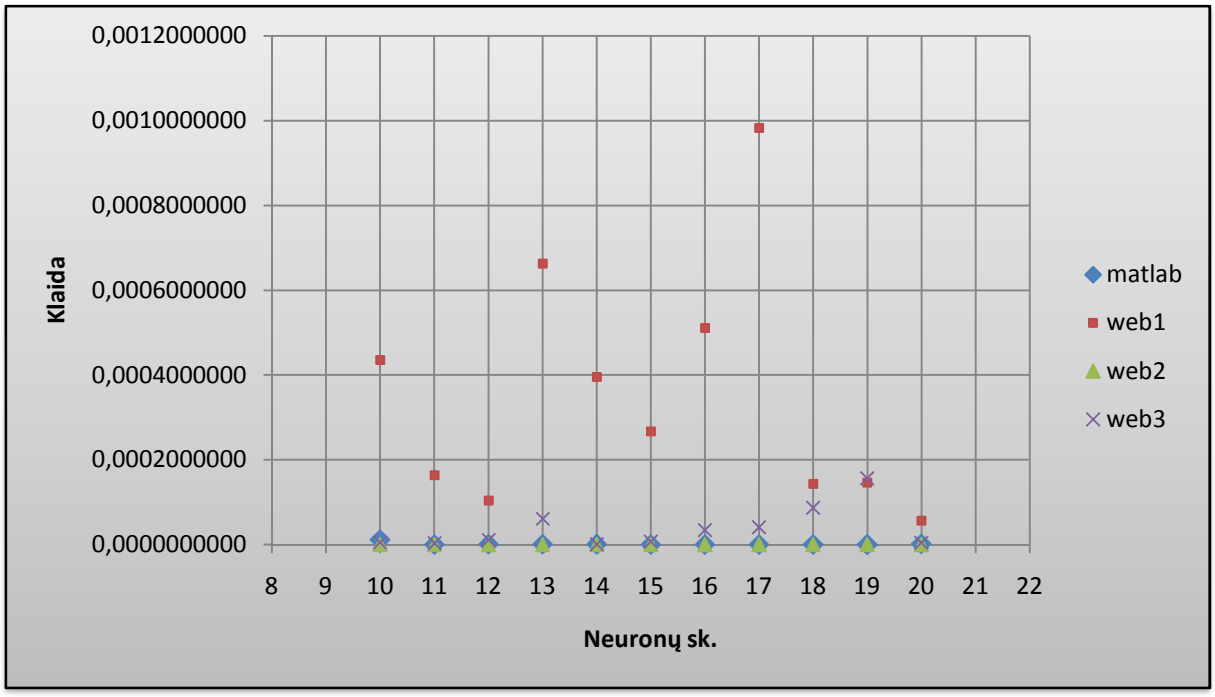
Paryškintuose langeliuose pateikiama mažiausia po apmokymo tinklo daroma klaida.

Kai konstanta 0,1.

Neuronų skaičius	Konstanta	Matlab	Laikas(s)	Web1	Laikas(s)
19	0,1	0,0000001490	27,27	0,000146873174259	00:03:12
20		0,0000020869	41,42	0,000057179769856	00:03:29

Lentelė 6. Mažiausia daroma klaida.

Grafiškai pateikiamas klaidos išsidėstymas pagal neuronų skaičių paslėptame sluoksnyje pagal visus metodus:



25 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,1 ir epochų skaičius 400.

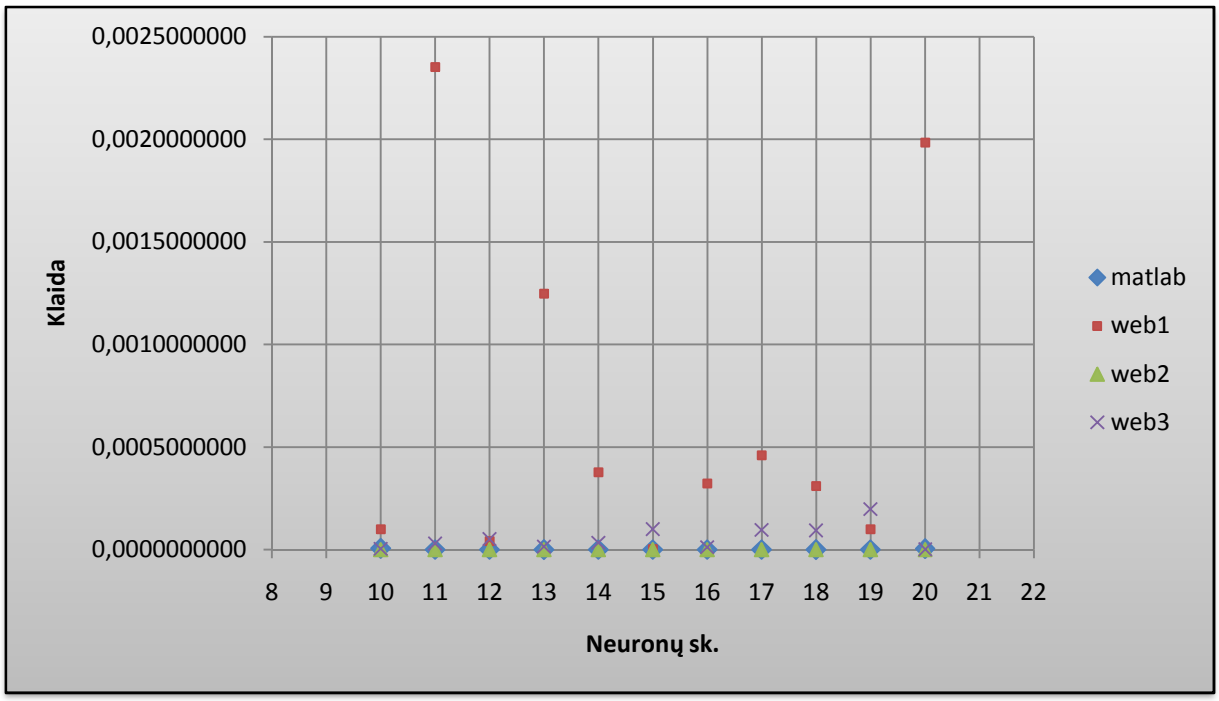
Iš pateiktų duomenų, kai konstanta 0,1, matyti, jog mažiausią klaidą daro tinklas paslėptame sluoksnyje turintis 19 ir 20 neuronų. Priešingai nei su irisų duomenimis tinklas apsimoko kur kas ilgiau, tačiau daroma klaida irgi žymiai mažesnė.

Kai konstanta 0,2.

Neuronų skaičius	Konstanta	Matlab	Laikas(s)	Web1	Laikas(s)
18	0,2	0,0000002009	28,44	0,000310849740278	00:03:44
15		0,0000008468	30,06	0,000004348970470	00:03:30

Lentelė 7. Mažiausia daroma klaida.

Grafiškai pateikiamas klaidos išsidėstymas pagal neuronų skaičių paslėptame sluoksnyje pagal visus metodus:



26 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,2 ir epochų skaičius 400.

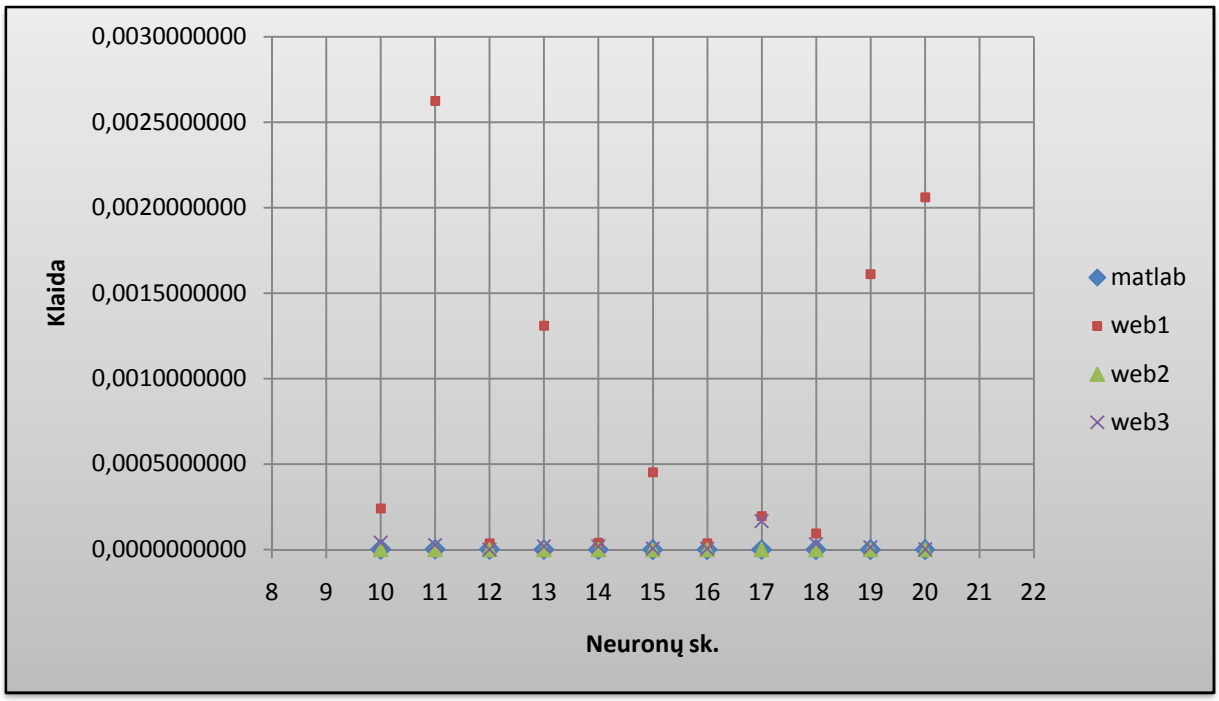
7 lentelėje pateikti duomenys rodo, kad tinklas geriausiai apsimoko, kai paslėptame sluoksnyje Matlab 7.1 sistemoje turi 18, VS Web Developer 2008 – 15 neuronų. Laiko atžvilgiu greičiau apsimoko Matlab 7.1 sistemoje.

Kai konstanta 0,3.

Neuronų skaičius	Konstanta	Matlab	Laikas(s)	Web1	Laikas(s)
15	0,3	0,0000002157	21,97	0,000454213846954	00:02:17
12		0,0000021843	15,61	0,000037443885572	00:01:51

Lentelė 8. Mažiausia daroma klaida.

Grafiškai pateikiamas klaidos išsidėstymas pagal neuronų skaičių paslėptame sluoksnyje pagal visus metodus:



27 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,3 ir epochų skaičius 400.

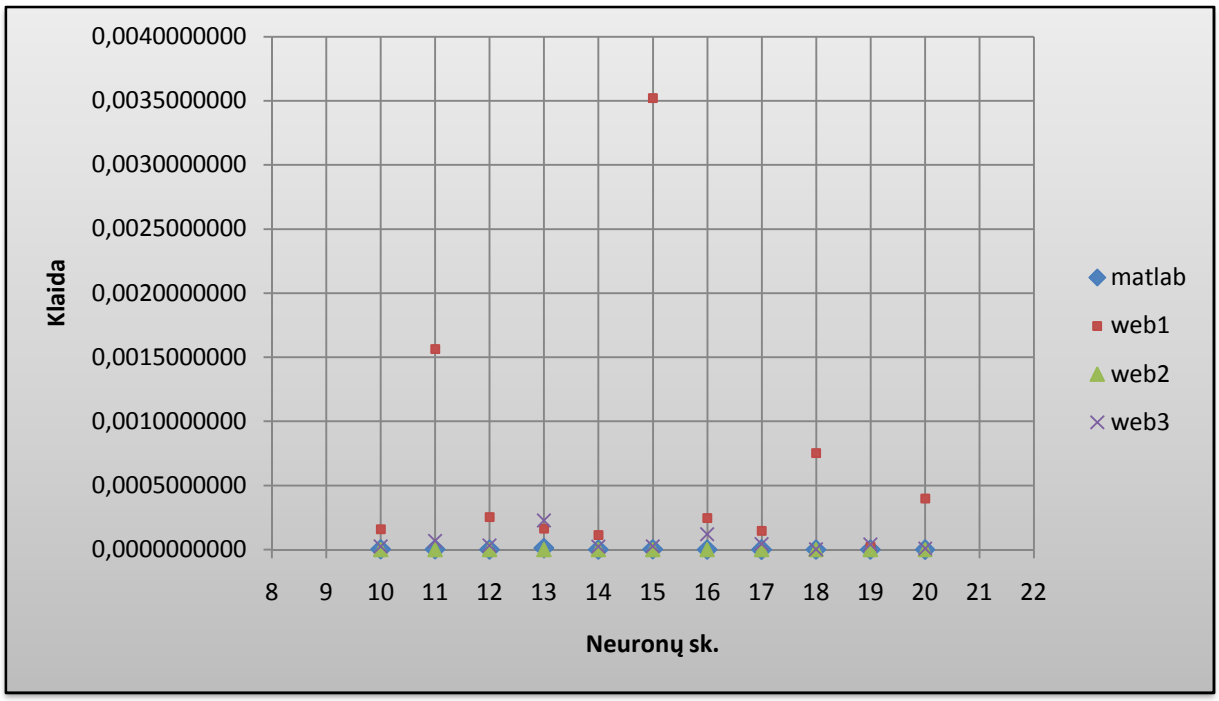
Pagal 18 lentelę ir 27 paveikslą matome, kad tinklas, paslėptame sluoksnyje turintis 15 neuronų, po apmokymo daro mažiausią klaidą, bei jo apmokymo laikas trumpiausias. Iš grafiko matyti, kad Matlab 7.1 sistemoje neuronų išsidėstymas pagal daromą klaidą labai artimas nuliui. Iš to galime daryti išvada, jog tinklas apsimoko gerai.

Kai konstanta 0,4.

Neuronų skaičius	Konstanta	Matlab	Laikas(s)	Web1	Laikas(s)
17	0,4	0,0000001268	26,44	0,000148825791476	00:02:38
19		0,0000008588	42,05	0,000024362146338	00:02:55

Lentelė 9. Mažiausia daroma klaida.

Grafiškai pateikiamas klaidos išsidėstymas pagal neuronų skaičių paslėptame sluoksnyje pagal visus metodus:



28 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,4 ir epochų skaičius 400.

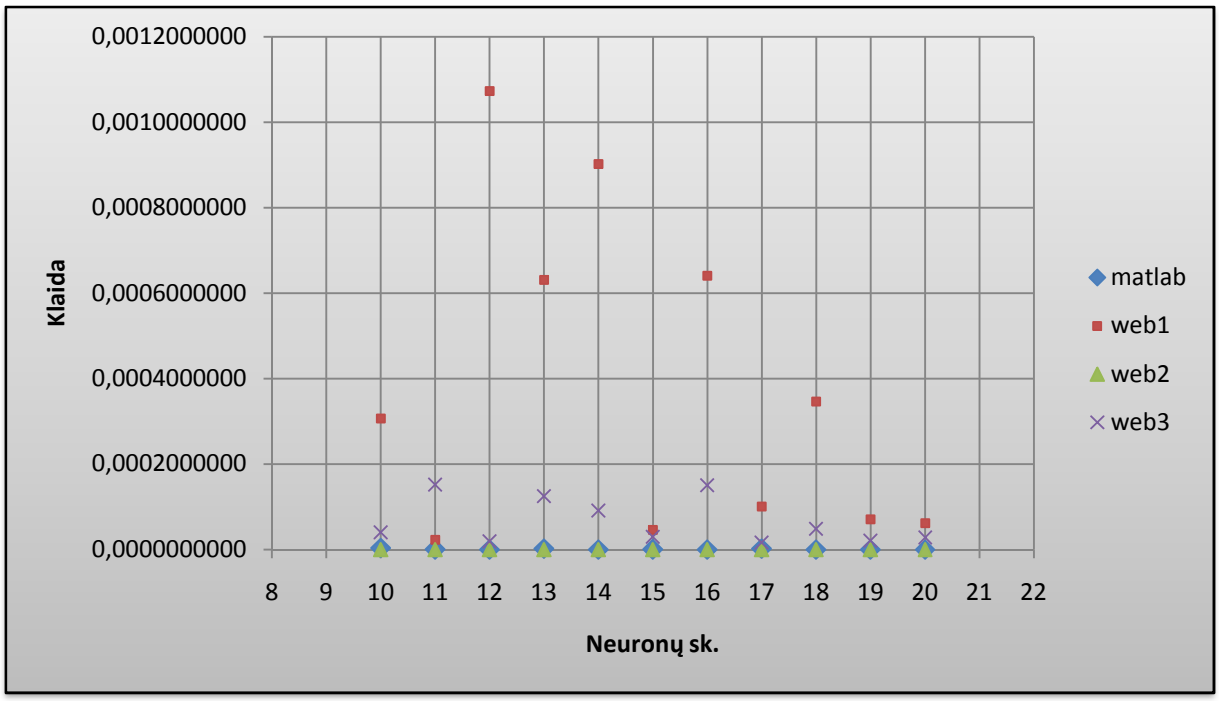
Iš 28 paveikslo matyti, jog beveik visi paslėptame sluoksnyje esantys neuronai, darantys mažą klaidą išsidėstę arčiau nulio. Geriausiai tinklas apsimoko, kai konstanta 0,4, paslėptame sluoksnyje turintis 17 ir atitinkamai 19 neuronų.

Kai konstanta 0,5.

Neuronų skaičius	Konstanta	Matlab	Laikas(s)	Web1	Laikas(s)
20	0,5	0,0000002199	42,73	0,000062463943492	00:03:01
11		0,0000008561	21,97	0,000023576705134	00:01:44

Lentelė 10. Mažiausia daroma klaida.

Grafiškai pateikiamas klaidos išsidėstymas pagal neuronų skaičių paslėptame sluoksnyje pagal visus metodus:



29 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,5 ir epochų skaičius 400.

Iš 29 paveikslo matyti, kad neuronų išsidėstymas paslėptame sluoksnyje, taikant „web1“ metodą, daro pakankamai dideles klaidas ir yra gerokai nutolę nuo nulio. Tinklas apsimoko prastai. Konstantai esant 0,5, geriausiai apsimoko tinklas savo konstrukcijoje paslėptame sluoksnyje turintis 20 ir 11 neuronų.

Irisų ir NO_2 palyginimai tarp Matlab ir VS Web Developer sistemų

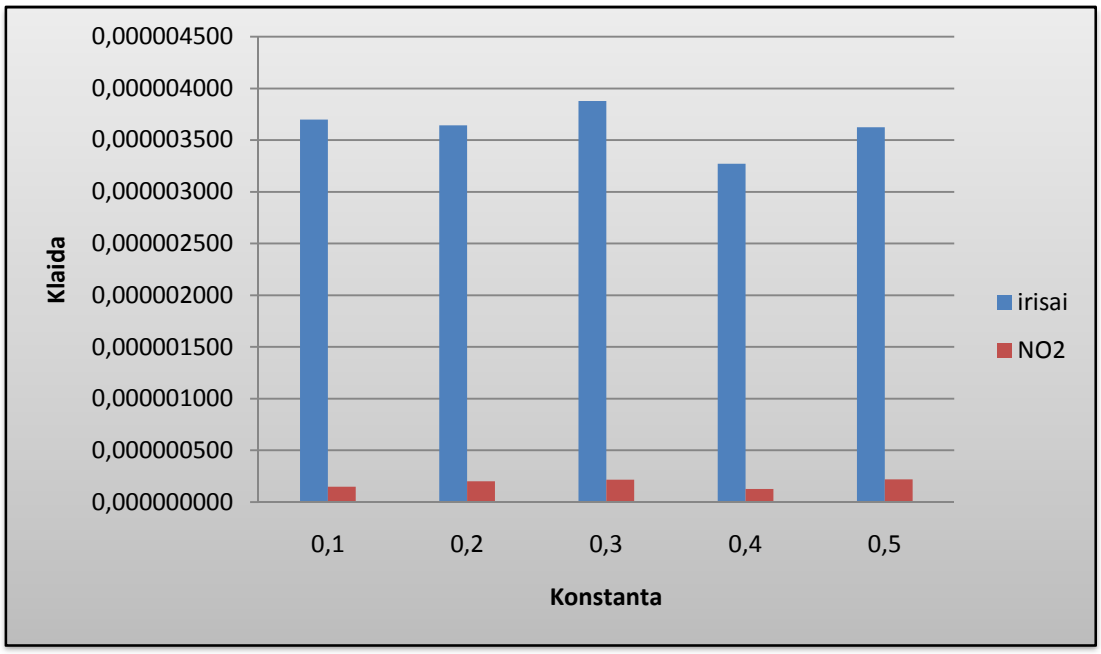
Palyginama klaida tarp irisų ir NO_2 duomenų *Matlab7.1* sistemoje:

Konstanta	**Matlab**	
	Irisai	NO_2
0,1	0,0000335695	0,0000119641
	0,0000234456	0,0000004606
	0,0000180477	0,0000009854
	0,0000151249	0,0000009889
	0,0000190615	0,0000006656
	0,0000092832	0,0000002174
	0,0000205810	0,0000004799
	0,0000056810	0,0000003297
	0,0000101730	0,0000003726
	0,0000036999	0,0000001490
	0,0000169221	0,0000020869
0,2	0,0000345261	0,0000061384
	0,0000298302	0,0000009369
	0,0000141769	0,0000010971
	0,0000269539	0,0000002171
	0,0000269248	0,0000011175
	0,0000087802	0,0000008468
	0,0000119088	0,0000006498
	0,0000102639	0,0000004426
	0,0000048286	0,0000002009
	0,0000036421	0,0000004113
	0,0000064856	0,0000047332
0,3	0,0001071110	0,0000017515
	0,0000450043	0,0000031432
	0,0000595714	0,0000021843
	0,0000205100	0,0000005403
	0,0000129798	0,0000012425
	0,0000161723	0,0000002157
	0,0000097673	0,0000002183
	0,0000063529	0,0000015516
	0,0000086118	0,0000002588
	0,0000038777	0,0000002311
	0,0000053593	0,0000002930
0,4	0,0000464872	0,0000033924
	0,0000198385	0,0000038397
	0,0000280697	0,0000015037
	0,0000205747	0,0000139323
	0,0000122289	0,0000009318
	0,0000184434	0,0000038533
	0,0000109922	0,0000005469
	0,0000153240	0,0000001268
	0,0000040061	0,0000010492
	0,0000038320	0,0000008588
	0,0000032723	0,0000003203

0,5	0,0000589003	0,0000041946
	0,0000305248	0,0000008561
	0,0000266767	0,0000003052
	0,0000134427	0,0000024563
	0,0000211009	0,0000004352
	0,0000060613	0,0000008762
	0,0000070683	0,0000005867
	0,0000076108	0,0000026594
	0,0000071249	0,0000005349
	0,0000036247	0,0000003053
	0,0000039089	0,0000002199

Lentelė 11. Matlab sistemos rezultatai

Mažiausios daromos klaidos palyginimas *Matlab 7.1* sistemoje grafiškai:



30 pav.

Iš 30 paveikslo matyti, kad kur kas mažesnė daroma klaida su NO_2 duomenimis, nors tarp parametrų jokio skirtumo nėra. Galime daryti išvadą, jog su daugiau duomenų tinklas apsimoko geriau, tačiau apmokymo laikas yra ilgesnis.

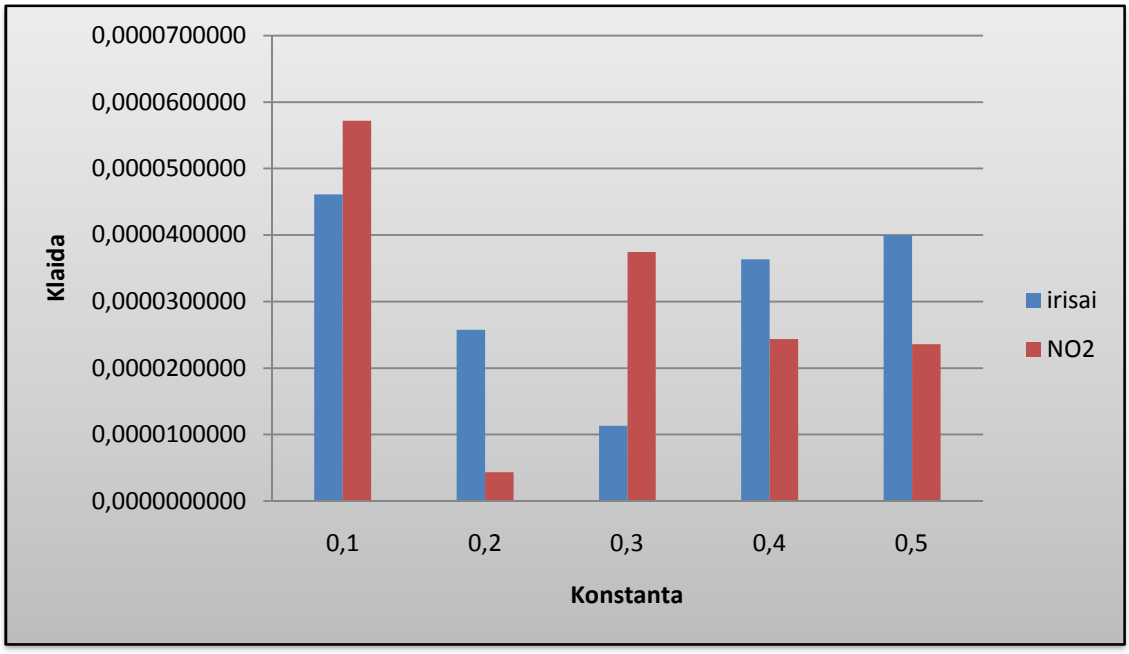
Palyginama klaida tarp irisų ir NO_2 duomenų *VS Web Developer2008* programoje:

Konstanta	**VS Web Developer**	
	Irisai	NO_2
0,1	0,0029445222	0,000435570695255
	0,0000641903	0,000163881237215
	0,0002350397	0,000104101904053
	0,0000461343	0,000662679421016
	0,0002178311	0,000396152350832
	0,0024971011	0,000267237822097
	0,0007486593	0,000511382889945
	0,0000975800	0,000983219758148
	0,0024798717	0,000143854124240
	0,0011471844	0,000146873174259
	0,0037438025	0,000057179769856
0,2	0,0001410049	0,000100560789571
	0,0000845609	0,002352519860059
	0,0000257733	0,000043445730557
	0,0017911422	0,001247719966643
	0,0002361592	0,000377662192955
	0,0004064291	0,000004348970470
	0,0000958183	0,000323726063234
	0,0001343199	0,000460984868299
	0,0002552366	0,000310849740278
	0,0033660696	0,000101088172669
	0,0003837633	0,001984390349164
0,3	0,0002771391	0,000242748447598
	0,0003669699	0,002623845532458
	0,0007137130	0,000037443885572
	0,0040973232	0,001311120075535
	0,0051668369	0,000042715423571
	0,0000112972	0,000454213846954
	0,0001999195	0,000038056285908
	0,0007678750	0,000196919267019
	0,0002569466	0,000096821315905
	0,0000737940	0,001612618894738
	0,0014473585	0,002062144613088
0,4	0,0004159412	0,000160036643803
	0,0011676555	0,001565328377148
	0,0035566063	0,000254816117465
	0,0004284525	0,000166463559148
	0,0000363462	0,000115646636495
	0,0002552369	0,003521163463894
	0,0003966251	0,000248163520672
	0,0001375169	0,000148825791476
	0,0002201188	0,000755379597726
	0,0000561830	0,000024362146338
	0,0003018581	0,000401447350485

0,5	0,0001375492	0,000307033058558
	0,0078785981	0,000023576705134
	0,0007528676	0,001073104174970
	0,0002744292	0,000631587099348
	0,0001186356	0,000902243232485
	0,0002545296	0,000046824719126
	0,0008123067	0,000641415966280
	0,0026583294	0,000101419897750
	0,0000399745	0,000346971957273
	0,0029703289	0,000071632653356
	0,0004043260	0,000062463943492

Lentelė 12. VS Web Developer 2008 sistemos rezultatai

Mažiausios daromos klaidos palyginimas *VS Web Developer 2008* programoje grafiškai:



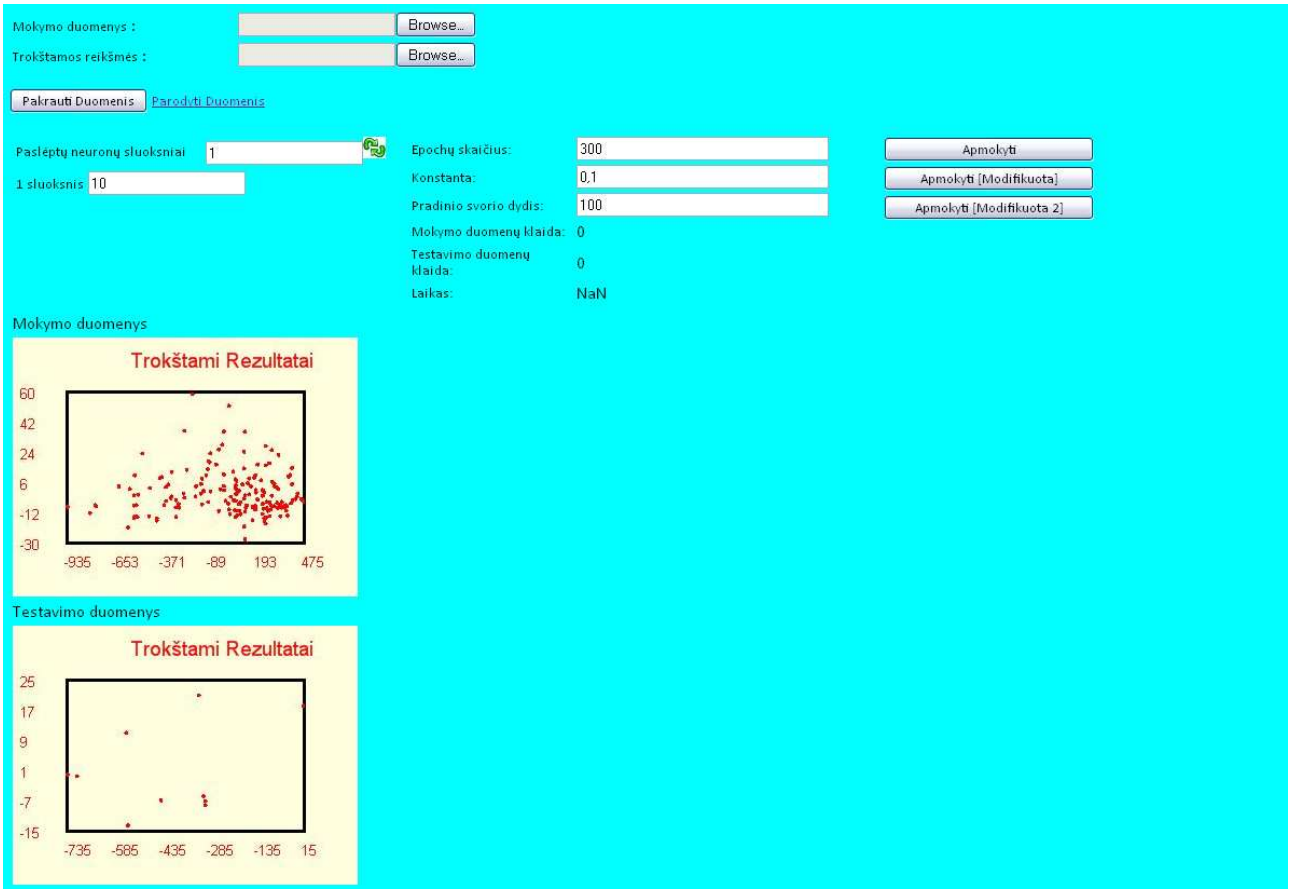
31 pav.

Iš 31 paveikslo matyti, kad klaida pasiskirsto įvairiai esant tiems patiems nustatytiems parametrams. Kai konstanta lygi 0,1, mažesnę klaidą daro irisų duomenys, kai 0,2 – NO_2 duomenys. Konstantai esant 0,3 tinklas apsimoko geriau su irisų duomenimis, tačiau kai konstanta 0,4 ir 0,5 - NO_2 tinklas apsimoko geriau. 3 iš 5 atvejų oro taršos duomenys daro mažesnę klaidą.

Tinklo testavimas su fiksuotais, apmokyto tinklo, svoriais

VS Web Developer 2008 programa buvo sukurtas tinklo testavimo metodas. Pagrindinis metodo tikslas: pratestuoti jau apmokytą tinklą su fiksuotais apmokyto tinklo svoriais, ir ištirti daromą klaidą.

Pagrindinis programos meniu langas:

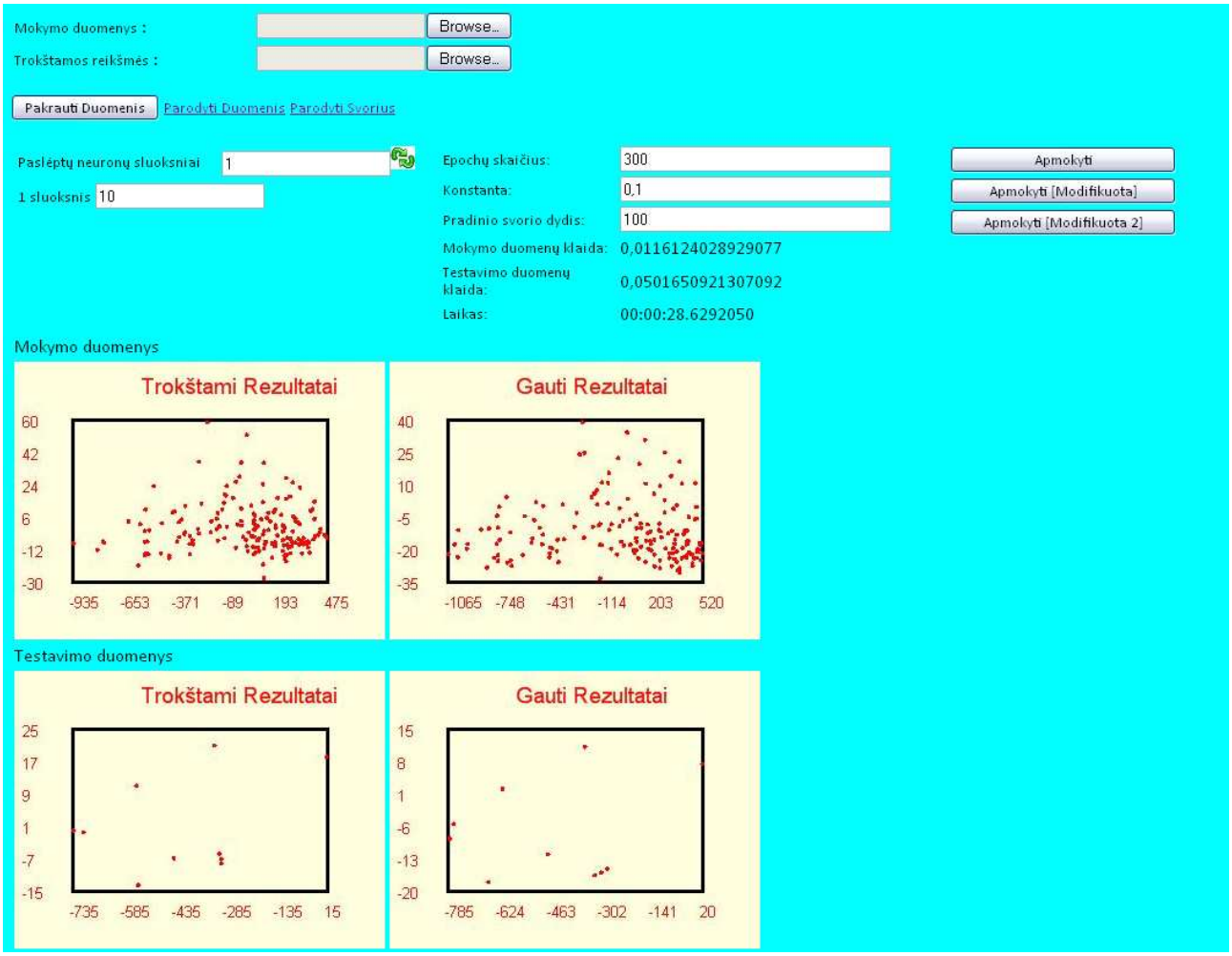


32 pav. Programos pagrindinis langas

Įkeliame norimus duomenis, nustatome atitinkamus parametrus ir paleidžiame tinklą apsimokyti.

Meniu lange matome „Testavimo duomenis“, kuriais testuosime apmokytą tinklą jau su fiksuotais svoriais. Testavimo duomenis sudaro atsitiktinai paimti 10 mokymo duomenų vektorių, kurie pavaizduoti grafiškai. Programa apmokydama tinklą skaičiuoja visų mokymo duomenų klaidų vidurkį, kurį išveda galutiniame rezultate: „Mokymo duomenų klaida“. „Testavimo duomenų klaida“ – tai testavimo duomenų klaidų vidurkis, gautas apmokius tinklą testavimo duomenimis su fiksuotais svoriais, gautais apmokius tinklą su mokymo duomenimis. Tikslas: palyginti abi gautas klaidas.

Meniu langas apmokius tinklą:



33 pav. Programos langas apmokius tinklą

Iš gautų rezultatų matyti, kad tinklas apsimokė neblogai. Jau su minėtais ir ankščiau tyrime naudotais parametrais atliksime bandymus.

Tyrimui naudosime irisų ir NO_2 duomenis. Tinklo apmokymui naudosime vieną paslėptą sluoksnį su jame paslėptais neuronais intervale [10; 20], atliksime 400 iteracijų, konstanta pasirenkame intervalą nuo 0,1 iki 0,3.

IRISŲ duomenys. Tyrimai su fiksuotais, apmokyto tinklo, svoriais

Tinklo apmokymo metodai su fiksuotais, apmokyto tinklo, svoriais:

- „web1“ – metodas, kai tinklui paduodami vektoriai iš eilės po vieną. Skaičiuojama klaida ir siunčiamas vektorius atgal, atnaujinami svoriai ir paduodamas kitas, naujas vektorius.
- „web2“ – metodas, kai paduodami keturi vektoriai. Mokymo duomenys dalinami į keturias dalis ir iš kiekvienos dalies skaičiuojamas vidurkis.
- „web3“ – metodas, kai paduodami keturi vektoriai. Mokymo duomenys dalinami į keturias dalis ir iš kiekvienos dalies skaičiuojamas sumos vektorius. Prieš sumuojant, mokymo vektoriai dauginami iš atsitiktinių koeficientų.

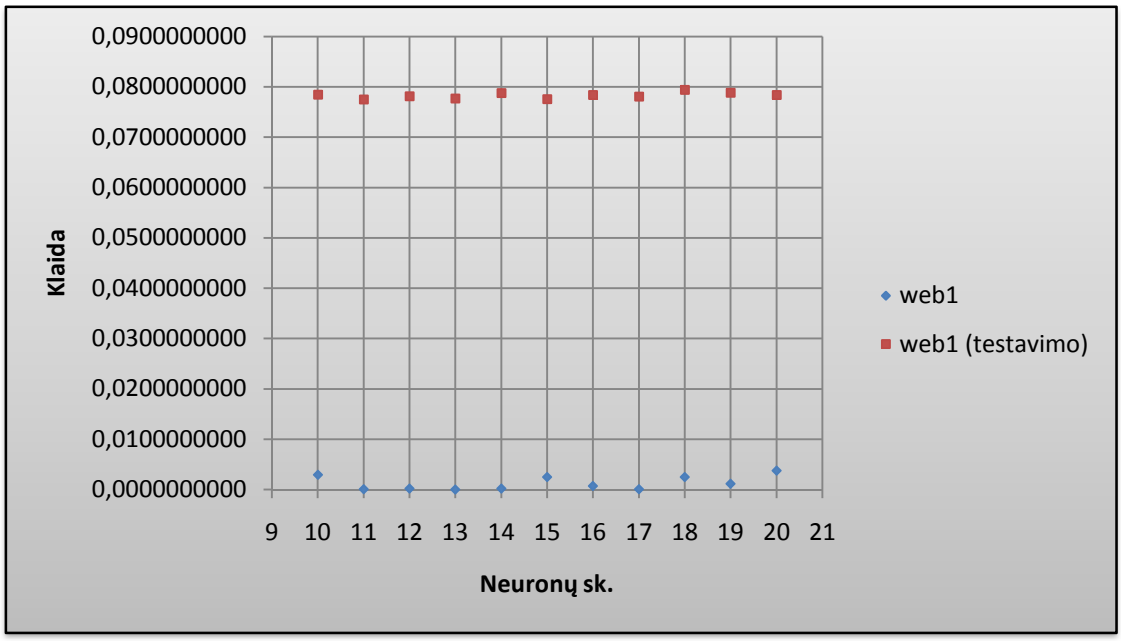
Toliau lentelėse pateikiami geriausiai apmokytų tinklų rezultatai tinklą apmokant iš karto, ir tinklą apmokant su fiksuotais svoriais.

Kai konstanta 0,1.

Konstanta	Metodas	Neuronų skaičius	Mažiausia daroma klaida	Laikas(s)
0,1	**Web1**	13	0,0000461343	00:00:33
	Web1 (Testavimo)	11	0,0775347137	00:00:25

Lentelė 13. Mažiausia daroma klaida.

Grafiškai pavaizduojami gauti rezultatai:



34 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,1 ir epochų skaičius 400

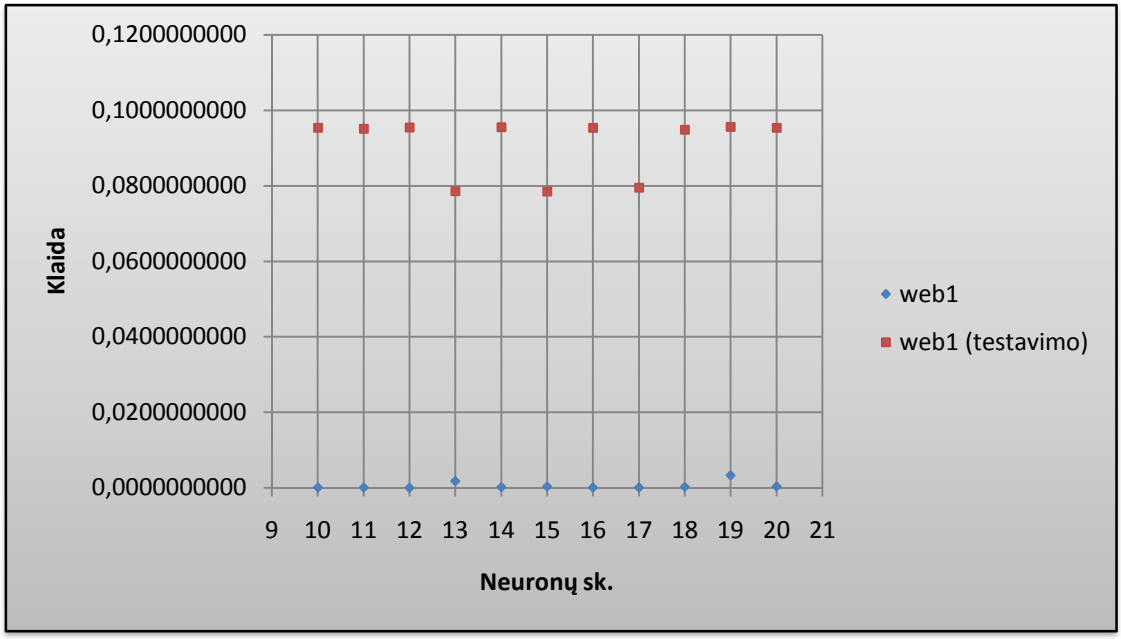
Iš 34 paveikslo matyti, kad neuronų išsidėstymas yra labai skirtingas. Taikant pirmąjį metodą tinklas mažiausią klaidą daro, kai paslėptame sluoksnyje turi 13 neuronų. Antruoju metodu – 11 neuronų. Taip yra todėl, kad antruoju metodu skaičiuojamas klaidų vidurkis nuo jau apmokyto tinklo su fiksuotais svoriais ir gaunamas pakankamai didelis skaičius, o pirmuoju paskutinio vektoriaus klaida. Laiko atžvilgiu greičiau apsimoko tinklas turintis 11 neuronų paslėptame sluoksnyje.

Kai konstanta 0,2.

Konstanta	Metodas	Neuronų skaičius	Mažiausia daroma klaida	Laikas(s)
0,2	**Web1**	12	0,0000257733	00:00:29
	Web1 (Testavimo)	15	0,0785402925	00:00:32

Lentelė 14. Mažiausia daroma klaida.

Grafiškai pavaizduojami gauti rezultatai:



35 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,2 ir epochų skaičius 400

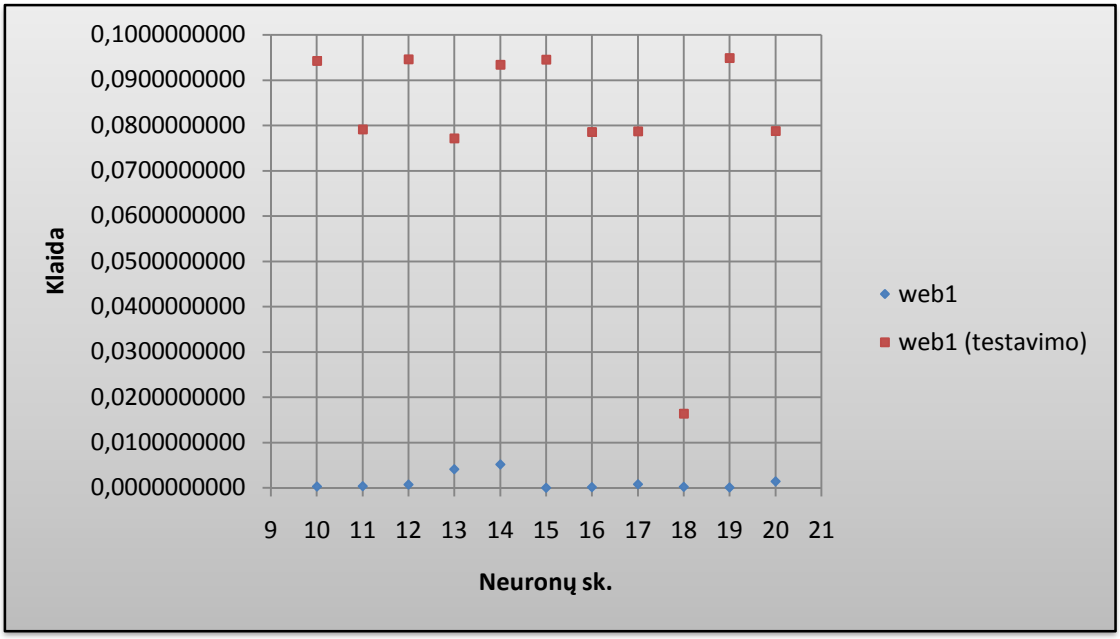
Iš pateiktų duomenų matyti, jog iš 11 atvejų 3 daro jau mažesnę klaidą skaičiuojant antruoju metodu. Geriausiai apsimoko tinklas turintis 12 neuronų paslėptame sluoksnyje kai konstanta lygi 0,2.

Kai konstanta 0,3.

Konstanta	Metodas	Neuronų skaičius	Mažiausia daroma klaida	Laikas(s)
0,3	**Web1**	15	0,0000112972	00:00:34
	Web1 (Testavimo)	18	0,0164267569	00:00:37

Lentelė 15. Mažiausia daroma klaida.

Grafiškai pavaizduojami gauti rezultatai:



36 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,3 ir epochų skaičius 400

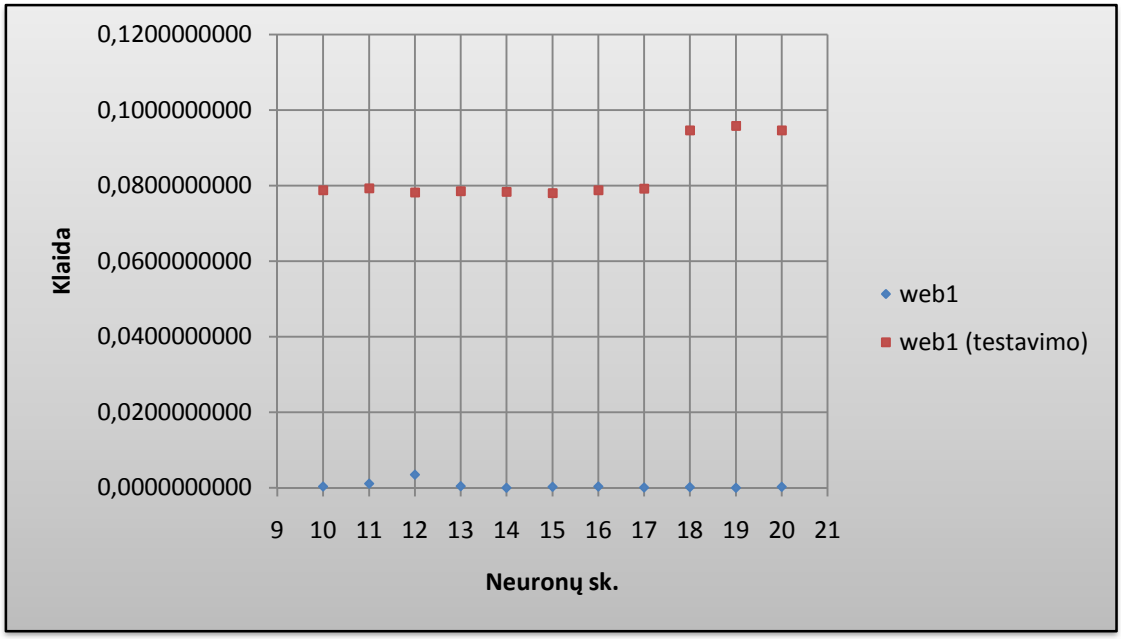
36 paveiksle matyti, kad antruoju atveju skaičiuojant klaidą, tinklas apsimoko vis geriau. Iš 11 atvejų jau 6 daro kur kas mažesnę klaidą. Geriausios konstrukcijos tinklas paslėptame sluoksnyje turintis 15 ir atitinkamai 18 neuronų.

Kai konstanta 0,4.

Konstanta	Metodas	Neuronų skaičius	Mažiausia daroma klaida	Laikas(s)
0,4	**Web1**	14	0,0000363462	00:00:37
	Web1 (Testavimo)	15	0,0780521222	00:00:32

Lentelė 16. Mažiausia daroma klaida.

Grafiškai pavaizduojami gauti rezultatai:



37 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,4 ir epochų skaičius 400

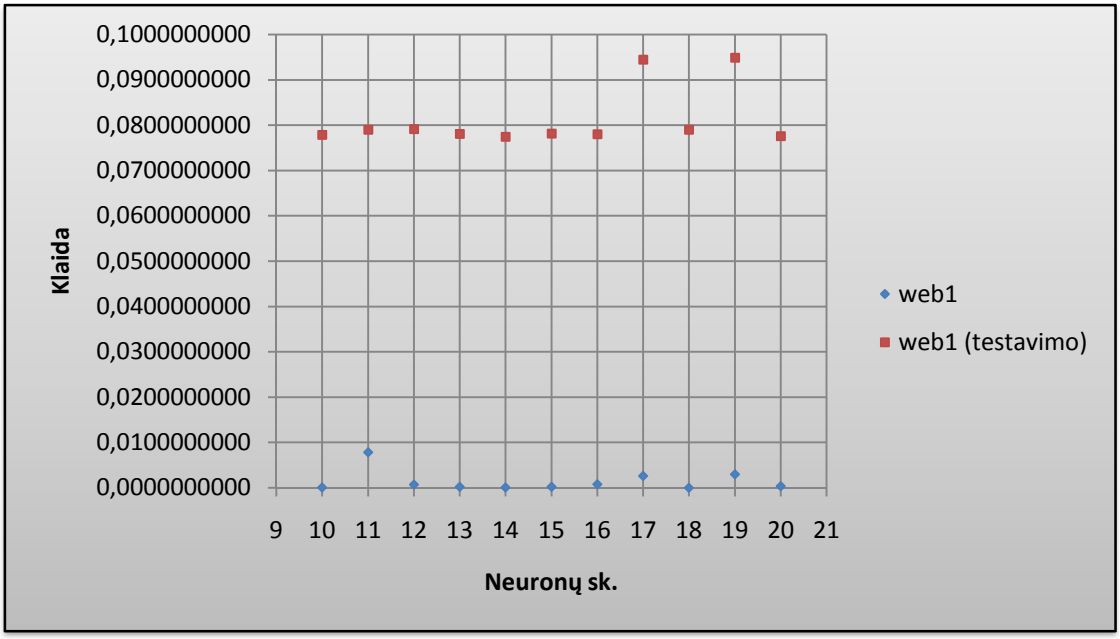
Iš 37 paveikslo pateiktų duomenų matyti, kad tinklas apmokomoas fiksuotais svoriais iš 11 klaidą mažesnę daro net 8 atvejai. Nors antruoju metodu tinklas apsimoko greičiau, tačiau jo daroma klaida yra didesnė kai konstanta lygi 0,4 ir iteracijų sakičius lygus 400.

Kai konstanta 0,5.

Konstanta	Metodas	Neuronų skaičius	Mažiausia daroma klaida	Laikas(s)
0,5	**Web1**	18	0,0000399745	00:00:44
	Web1 (Testavimo)	14	0,0775142885	00:00:37

Lentelė 17. Mažiausia daroma klaida.

Grafiškai pavaizduojami gauti rezultatai:



38 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,5 ir epochų skaičius 400

Geriausiai apsimoko tinklas paslėptame sluoksnyje turintis 18 neuronų. Negalime teigti, kad antruoju metodu apmokomas tinklas daro labai didelę klaidą. Priešingai, kuo gauti skaičiai panašesni, tuo tinklas apsimokė geriau. Jei klaidų vidurkį gauname pakankamai vienodą, vadinasi ir klaidos, iš kurių vedamas vidurkis, yra irgi labai panašios. O jei panašios, vadinasi, tinklas apsimoko gerai.

NO_2 duomenys. Tyrimai su fiksuotais, apmokyto tinklo, svoriais

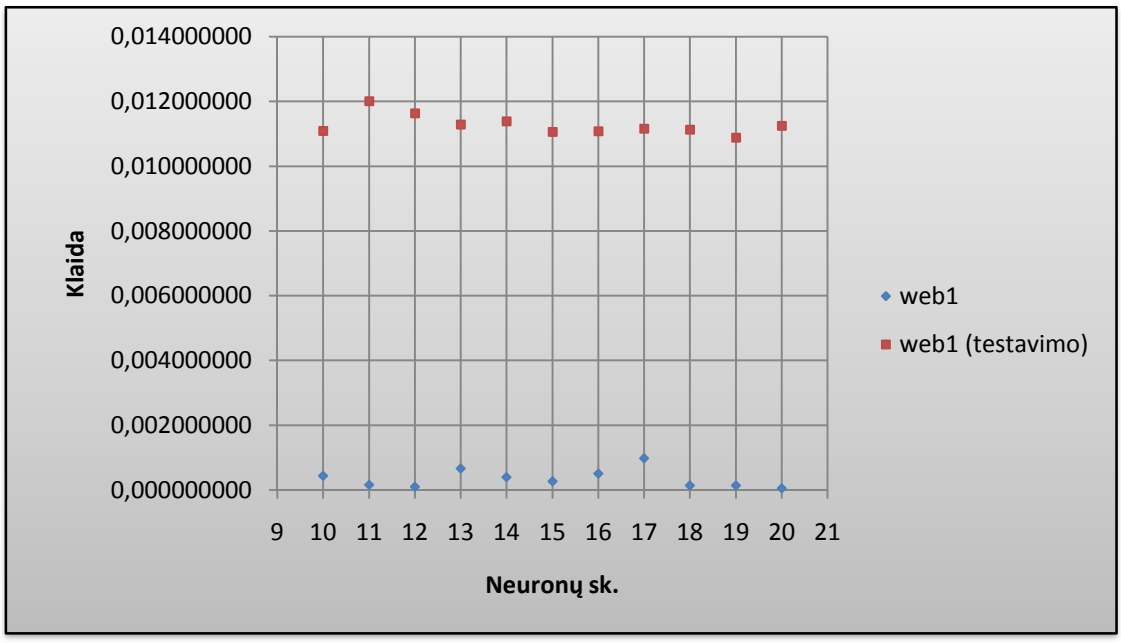
Geriausiai apmokytų tinklų rezultatai:

Kai konstanta 0,1.

Konstanta	Metodas	Neuronų skaičius	Mažiausia daroma klaida	Laikas(s)
0,1	**Web1**	20	0,000057180	00:03:29
	Web1 (Testavimo)	19	0,010890801	00:02:40

Lentelė 18. Mažiausia daroma klaida.

Grafiškai pavaizduojami gauti rezultatai:



39 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,1 ir epochų skaičius 400

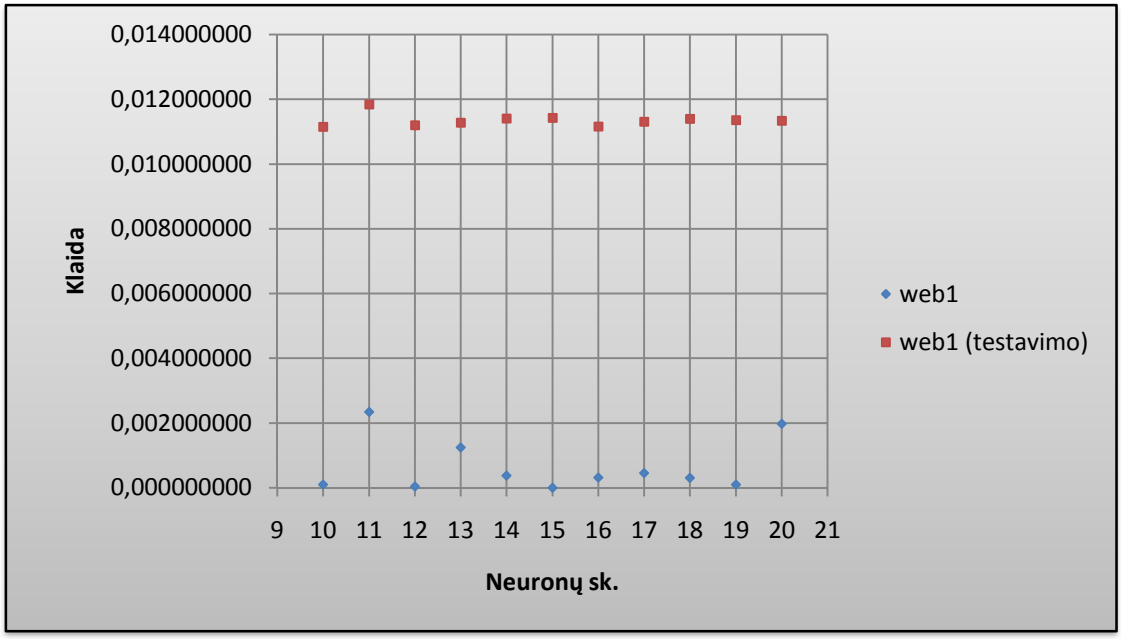
Kaip ir buvo tikėtasi su oro taršos duomenimis gautos klaidos yra kur kas mažesnės lyginant su irisų duomenimis. Iš pateiktų duomenų matyti, kad geriausiai apsimokė tinklas su 20 neuronų paslėptame sluoksnyje, kai konstanta 0,1 ir epochų skaičius lygus 400. Laiko atžvilgiu tinklas apsimoko ilgiau.

Kai konstanta 0,2.

Konstanta	Metodas	Neuronų skaičius	Mažiausia daroma klaida	Laikas(s)
0,2	**Web1**	15	0,000004349	00:03:30
	Web1 (Testavimo)	10	0,011153584	00:01:35

Lentelė 19. Mažiausia daroma klaida.

Grafiškai pavaizduojami gauti rezultatai:



40 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,2 ir epochų skaičius 400

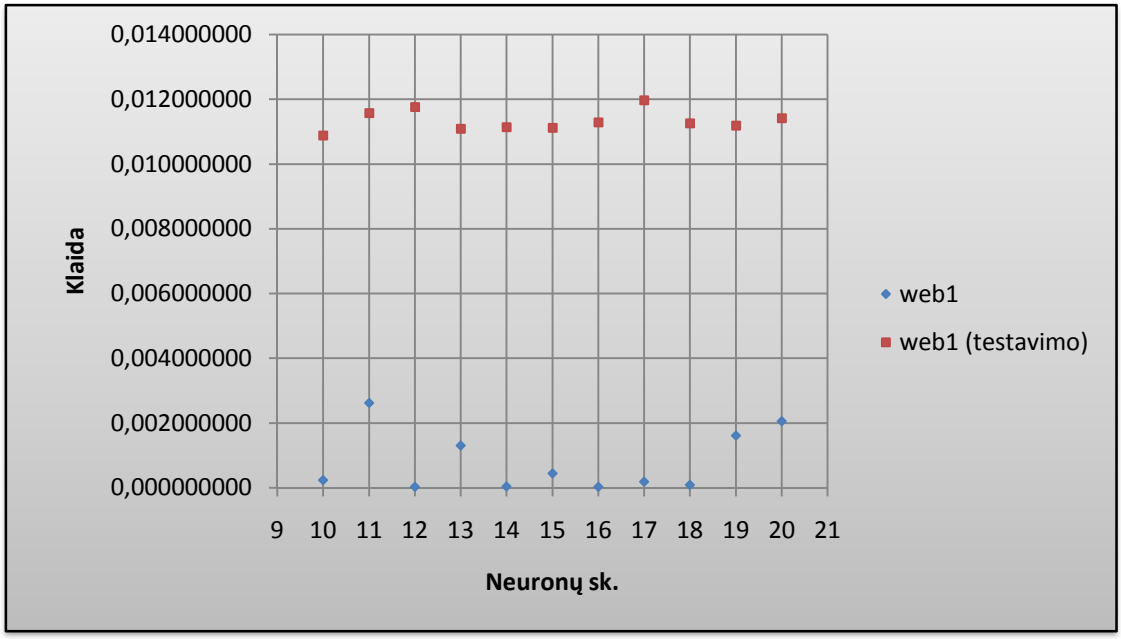
Vėlgi iš duomenų matyti, kad antruoju metodu daroma klaida yra daug maž vienoda, tai leidžia daryti išvada, jog tinklas apsimoko gerai. Šiuo atveju geriausiai apsimokė tinklas turintis 15 ir atitinkamai 10 neuronų paslėptame sluoksnyje.

Kai konstanta 0,3.

Konstanta	Metodas	Neuronų skaičius	Mažiausia daroma klaida	Laikas(s)
0,3	**Web1**	12	0,000037444	00:01:51
	Web1 (Testavimo)	10	0,010890316	00:01:35

Lentelė 20. Mažiausia daroma klaida.

Grafiškai pavaizduojami gauti rezultatai:



41 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,3 ir epochų skaičius 400

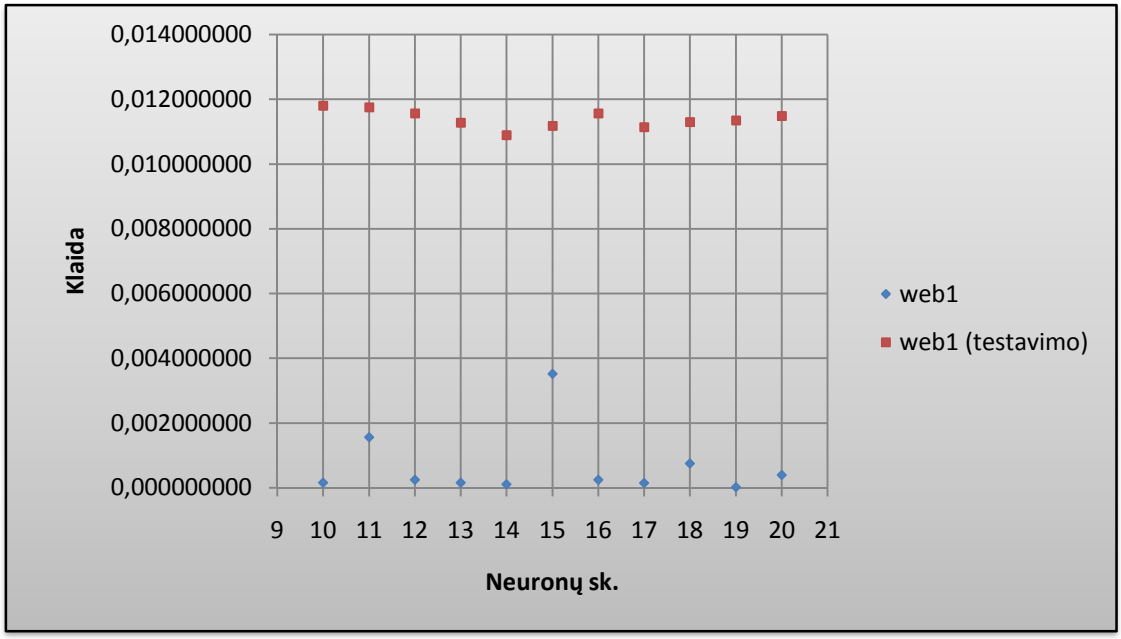
Kai konstanta lygi 0,3 iš 41 paveikslo ir 40 lentelės matyti, kad geriausiai apsimoko tinklas turintis 12 ir 10 neuronų paslėptame sluoksnyje. Tinklas grečiau apsimoko pirmuoju metodu ir daro mažesnę klaidą.

Kai konstanta 0,4.

Konstanta	Metodas	Neuronų skaičius	Mažiausia daroma klaida	Laikas(s)
0,4	**Web1**	19	0,000024362	00:02:55
	Web1 (Testavimo)	14	0,010898450	00:02:06

Lentelė 21. Mažiausia daroma klaida.

Grafiškai pavaizduojami gauti rezultatai:



42 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,4 ir epochų skaičius 400

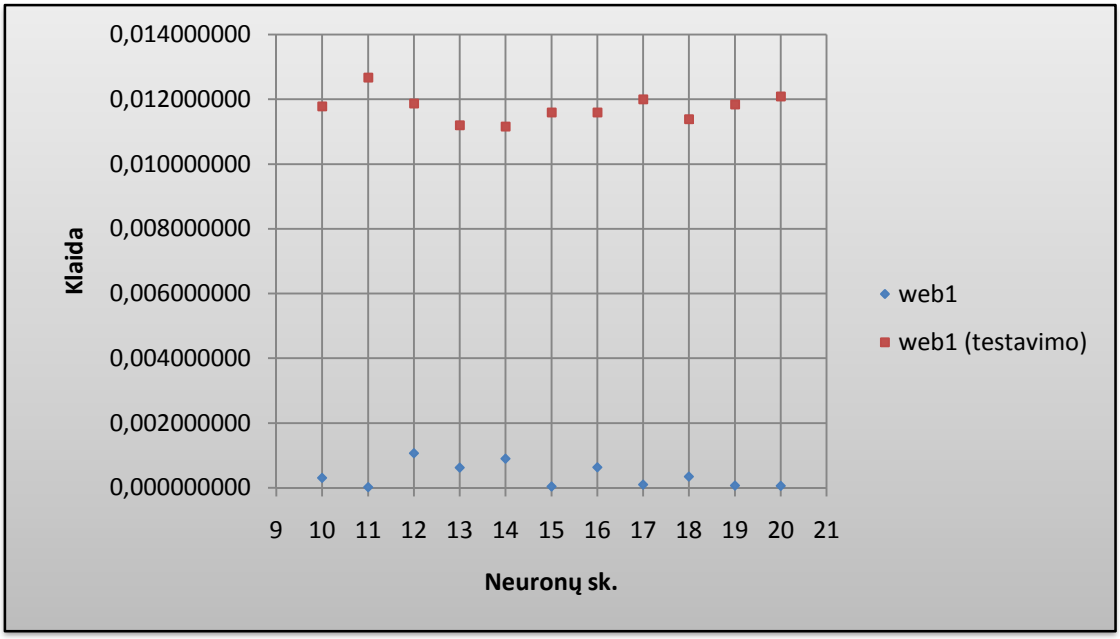
Kaip matyti iš pateiktų duomenų antruoju metodu tinklas apsimoko greičiau, tačiau jo daroma klaida – didesnė. Kai konstanta lygi 0,4 geriausios konstrukcijos tinklas paslėptame sluoksnyje turi 19 ir atitinkamai 14 neuronų.

Kai konstanta 0,5.

Konstanta	Metodas	Neuronų skaičius	Mažiausia daroma klaida	Laikas(s)
0,5	**Web1**	11	0,000023577	00:01:44
	Web1 (Testavimo)	14	0,011164559	00:02:10

Lentelė 22. Mažiausia daroma klaida.

Grafiškai pavaizduojami gauti rezultatai:



43 pav. Neuronų išsidėstymas pagal klaidą kai konstanta 0,5 ir epochų skaičius 400

Atlikus tyrimą galime teigti, kad klaida antruoju metodu labai neišsiskyrė nuo atitinkamų parametrų keitimo. Tačiau galime teigti, kad tinklas, su jau apmokyto tinklo fiksuotais svoriais, apsimoko gerai. Gauti rezultatai pakankamai panašūs, vadinasi ir klaidos gaunamos pakankamai artimos viena kitai. Šiuo paskutiniu atveju esant 400 epochų, kai konstanta lygi 0,5 geriausios konstrukcijos tinklas paslėptame sluoksnyje turi 11 ir atitinkamai 14 neuronų.

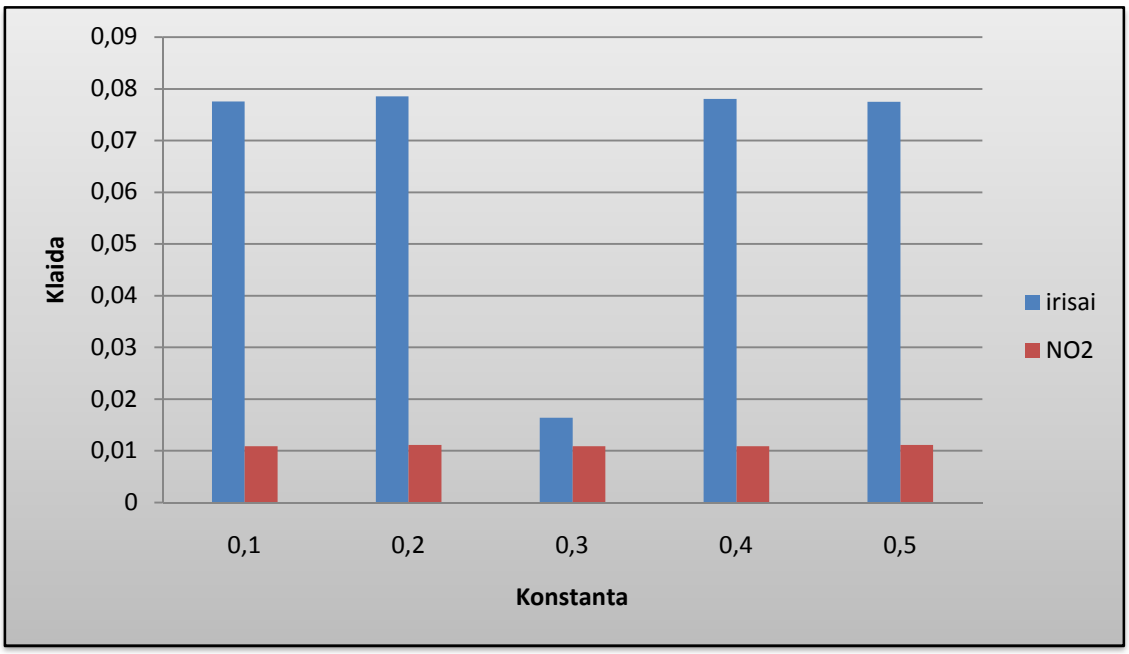
Palyginama klaida tarp irisų ir NO_2 duomenų VS Web Developer 2008 programoje:

Konstanta	VS Web Developer			
	Irisai		***NO_2***	
	Web1	*Web1 (testavimo)*	*Web1*	*Web1 (testavimo)*
0,1	0,0029445222	0,0784517098	0,0004355707	0,0111029885
	0,0000641903	0,0775347137	0,0001638812	0,0120217473
	0,0002350397	0,0781570659	0,0001041019	0,0116419616
	0,0000461343	0,0776735727	0,0006626794	0,0112961762
	0,0002178311	0,0787913791	0,0003961524	0,0113989163
	0,0024971011	0,0775902605	0,0002672378	0,0110728164
	0,0007486593	0,0784101419	0,0005113829	0,0110879741
	0,0000975800	0,0780538207	0,0009832198	0,0111683928
	0,0024798717	0,0794102205	0,0001438541	0,0111375346
	0,0011471844	0,0788733096	0,0001468732	0,0108908010
	0,0037438025	0,0783723412	0,0000571798	0,0112521640
0,2	0,0001410049	0,0953961564	0,0001005608	0,0111535839
	0,0000845609	0,0950853911	0,0023525199	0,0118493358
	0,0000257733	0,0954341188	0,0000434457	0,0112094270
	0,0017911422	0,0785621405	0,0012477200	0,0112866028
	0,0002361592	0,0955128639	0,0003776622	0,0114187177
	0,0004064291	0,0785402925	0,0000043490	0,0114339179
	0,0000958183	0,0954210689	0,0003237261	0,0111681661
	0,0001343199	0,0795073188	0,0004609849	0,0113171346
	0,0002552366	0,0948410433	0,0003108497	0,0114029746
	0,0033660696	0,0956598194	0,0001010882	0,0113690686
	0,0003837633	0,0954210088	0,0019843903	0,0113467492
0,3	0,0002771391	0,0942414304	0,0002427484	0,0108903162
	0,0003669699	0,0791165668	0,0026238455	0,0115790017
	0,0007137130	0,0945796262	0,0000374439	0,0117689105
	0,0040973232	0,0771408892	0,0013111201	0,0111010732
	0,0051668369	0,0934285529	0,0000427154	0,0111444369
	0,0000112972	0,0945616045	0,0004542138	0,0111291440
	0,0001999195	0,0785391949	0,0000380563	0,0112955528
	0,0007678750	0,0787205208	0,0001969193	0,0119821626
	0,0002569466	0,0164267569	0,0000968213	0,0112671456
	0,0000737940	0,0948930451	0,0016126189	0,0111951919
	0,0014473585	0,0787465504	0,0020621446	0,0114212694
0,4	0,0004159412	0,0788155750	0,0001600366	0,0118117527
	0,0011676555	0,0793930182	0,0015653284	0,0117572358
	0,0035566063	0,0782793511	0,0002548161	0,0115744188
	0,0004284525	0,0786369184	0,0001664636	0,0112873288
	0,0000363462	0,0784623320	0,0001156466	0,0108984503
	0,0002552369	0,0780521222	0,0035211635	0,0111907734
	0,0003966251	0,0788491932	0,0002481635	0,0115749554
	0,0001375169	0,0792594328	0,0001488258	0,0111495827
	0,0002201188	0,0947307899	0,0007553796	0,0113032158
	0,0000561830	0,0959280136	0,0000243621	0,0113539549
	0,0003018581	0,0946898148	0,0004014474	0,0114975615

0,5	0,0001375492	0,0779146666	0,0003070331	0,0117891284
	0,0078785981	0,0790386985	0,0000235767	0,0126848191
	0,0007528676	0,0791898863	0,0010731042	0,0118804992
	0,0002744292	0,0781737353	0,0006315871	0,0112086363
	0,0001186356	0,0775142885	0,0009022432	0,0111645592
	0,0002545296	0,0782428521	0,0000468247	0,0116036970
	0,0008123067	0,0781061361	0,0006414160	0,0115992119
	0,0026583294	0,0945331760	0,0001014199	0,0120054476
	0,0000399745	0,0790262299	0,0003469720	0,0113914208
	0,0029703289	0,0949764135	0,0000716327	0,0118514599
	0,0004043260	0,0776273965	0,0000624639	0,0120985081

Lentelė 23. VS Web Developer 2008 programos rezultatai

Mažiausios daromos klaidos palyginimas *VS Web Developer 2008* programoje grafiškai tarp *irisų Web1 (testavimo)* ir *NO_2 Web (testavimo)* duomenų:



44 pav.

Iš 44 paveikslo matome, kad mažesnę klaidą daro NO_2 duomenys kai konstanta kinta nuo 0,1 iki 0,5 ir epochų skaičius lygus 400.

IŠVADOS

- Visual Studio Web Developer 2008 programoje mažesnę klaidą daro tinklas turintis tik vieną paslėptą sluoksnį. Atliekant bandymus su daugiau sluoksnių tinklas apsimoko prastai.
- Atvirkščiai nei VS Web Developer 2008 programa Matlab7.1 programoje po apmokymo mažesnę klaidą daro tinklas, turintis du paslėptus sluoksnius.
- Testuojant tinklą jau apmokyto tinklo fiksuotais svoriais gaunamas klaidos vidurkis išsidėsto pakankamai vienodai. Iš to galime daryti išvadą, kad tinklas apsimoko gerai. Priešingu atveju gautume labai skirtingas klaidas.
- Tiek Matlab7.1, tiek VS Web Developer programose tinklo apmokymo laikas tiesiogiai priklauso nuo jam apmokyti užduotų parametrų.
- Matlab7.1 sistemoje tinklas apsimoko geriau su daugiau duomenų. NO_2 duomenų kiekis gerokai didesnis, todel tinklas apsimoko geriau.
- Apmokant tinklą antruoju metodu VS sistemoje neuroninis tinklas apsimokydavo labai greitai ir klaida buvo artima nuliui. Tinklas apsimokydavo prastai. Priežąstis buvo ta, jog tinklui buvo paduotas tik vienas mokymo vektorius, o tinklas geriau apsimoko kai jam paduodama kuo daugiau duomenų.
- Apmokant tinklą trečiuoju metodu VS sistemoje NT gauti rezultatai buvo tikslesni trokštamiems, tinklas apsimokydavo geriau. Daroma klaida buvo artima pirmąjam metodui.
- Iš tyrime naudotų trijų metodų geriausiai tinklas apsimoko pirmuoju metodu.

LITERATŪRA

1. Dzemyda G. Dirbtiniai neuroniniai tinklai. http://www.estudijos.vpu.lt/course/view.php?id=31
2. Medvedev V. Tiesioginio sklidimo neuroninių tinklų taikymo daugiamačiams duomenims vizualizuoti tyrimai. Daktaro dizertacija, 2007
3. Raudys Š., Dirbtinių neuroninių tinklų tyrimas ir taikymai sprendžiant konkrečius uždavinius: mokslo darbo ataskaita. VGTU, Vilnius, 2005
4. Tyrimams naudoti duomenys: http://en.wikipedia.org/wiki/Iris_flower_data_set
5. Tyrimams naudoti duomenys: http://lib.stat.cmu.edu/datasets/NO2.dat
6. Verbylaitė L., Atgalinio klaidos sklidimo neuroninio tinklo realizavimo problemos ir taikymai. Magistrinis darbas, 2008
7. Verikas A., Gelžinis A. Neuroniniai tinklai ir neuroniniai skaičiavimai. – Kaunas: Technologija, 2003
8. Vyšniauskas V., Vienkrypčių neuroninių tinklų efektyvumo klausimai. – Vilnius: Atviros lietuvos fondas, 1996

SANTRAUKA

Klaidos skleidimo atgal algoritmo tyrimai

Šiame darbe detaliai išanalizuotas klaidos skleidimo atgal algoritmas, atlikti tyrimai. Išsamiai analizuota neuroninių tinklų teorija. Algoritmui taikyti ir analizuoti sistemoje *Visual Studio Web Developer 2008* sukurta programa su įvairiais tyrimo metodais, padedančiais ištirti algoritmo daromą klaidą. Taip pat naudotasi *Matlab 7.1* sistemos įrankiais neuroniniams tinklams apmokyti.

Tyrimo metu analizuotas daugiasluoksnis dirbtinis neuroninis tinklas su vienu paslėptu sluoksniu. Tyrimams naudoti gėlių irisų ir oro taršos duomenys. Atlikti gautų rezultatų palyginimai.

SUMMARY

Investigation of the error back-propagation algorithm

The present work provides an in-depth analysis of the error back-propagation algorithm, as well as information on the investigation carried out. A neural network theory has been analysed in detail. For the application and analysis of the algorithm in the system Visual Studio Web Developer 2008, a program has been developed with various investigation methods, which help to research into the error of the algorithm. For training neural networks, Matlab 7.1 tools have been used.

In the course of the investigation, a multilayer artificial neural network with one hidden layer has been analysed. For the purpose of the investigation, data on irises (plants) and air pollution have been used. Comparisons of the results obtained have been made.

PRIEDAI

Priedas 1. Irisų duomenų bazė.

5,1	3,5	1,4	0,2
4,9	3	1,4	0,2
4,7	3,2	1,3	0,2
4,6	3,1	1,5	0,2
5	3,6	1,4	0,2
5,4	3,9	1,7	0,4
4,6	3,4	1,4	0,3
5	3,4	1,5	0,2
4,4	2,9	1,4	0,2
4,9	3,1	1,5	0,1
5,4	3,7	1,5	0,2
4,8	3,4	1,6	0,2
4,8	3	1,4	0,1
4,3	3	1,1	0,1
5,8	4	1,2	0,2
5,7	4,4	1,5	0,4
5,4	3,9	1,3	0,4
5,1	3,5	1,4	0,3
5,7	3,8	1,7	0,3
5,1	3,8	1,5	0,3
5,4	3,4	1,7	0,2
5,1	3,7	1,5	0,4
4,6	3,6	1	0,2
5,1	3,3	1,7	0,5
4,8	3,4	1,9	0,2
5	3	1,6	0,2
5	3,4	1,6	0,4
5,2	3,5	1,5	0,2
5,2	3,4	1,4	0,2
4,7	3,2	1,6	0,2
4,8	3,1	1,6	0,2
5,4	3,4	1,5	0,4
5,2	4,1	1,5	0,1
5,5	4,2	1,4	0,2
4,9	3,1	1,5	0,2
5	3,2	1,2	0,2
5,5	3,5	1,3	0,2
4,9	3,6	1,4	0,1
4,4	3	1,3	0,2
5,1	3,4	1,5	0,2

5	3,5	1,3	0,3
4,5	2,3	1,3	0,3
4,4	3,2	1,3	0,2
5	3,5	1,6	0,6
5,1	3,8	1,9	0,4
4,8	3	1,4	0,3
5,1	3,8	1,6	0,2
4,6	3,2	1,4	0,2
5,3	3,7	1,5	0,2
5	3,3	1,4	0,2
7	3,2	4,7	1,4
6,4	3,2	4,5	1,5
6,9	3,1	4,9	1,5
5,5	2,3	4	1,3
6,5	2,8	4,6	1,5
5,7	2,8	4,5	1,3
6,3	3,3	4,7	1,6
4,9	2,4	3,3	1
6,6	2,9	4,6	1,3
5,2	2,7	3,9	1,4
5	2	3,5	1
5,9	3	4,2	1,5
6	2,2	4	1
6,1	2,9	4,7	1,4
5,6	2,9	3,6	1,3
6,7	3,1	4,4	1,4
5,6	3	4,5	1,5
5,8	2,7	4,1	1
6,2	2,2	4,5	1,5
5,6	2,5	3,9	1,1
5,9	3,2	4,8	1,8
6,1	2,8	4	1,3
6,3	2,5	4,9	1,5
6,1	2,8	4,7	1,2
6,4	2,9	4,3	1,3
6,6	3	4,4	1,4
6,8	2,8	4,8	1,4
6,7	3	5	1,7
6	2,9	4,5	1,5
5,7	2,6	3,5	1
5,5	2,4	3,8	1,1
5,5	2,4	3,7	1
5,8	2,7	3,9	1,2
6	2,7	5,1	1,6
5,4	3	4,5	1,5

6	3,4	4,5	1,6
6,7	3,1	4,7	1,5
6,3	2,3	4,4	1,3
5,6	3	4,1	1,3
5,5	2,5	4	1,3
5,5	2,6	4,4	1,2
6,1	3	4,6	1,4
5,8	2,6	4	1,2
5	2,3	3,3	1
5,6	2,7	4,2	1,3
5,7	3	4,2	1,2
5,7	2,9	4,2	1,3
6,2	2,9	4,3	1,3
5,1	2,5	3	1,1
5,7	2,8	4,1	1,3
6,3	3,3	6	2,5
5,8	2,7	5,1	1,9
7,1	3	5,9	2,1
6,3	2,9	5,6	1,8
6,5	3	5,8	2,2
7,6	3	6,6	2,1
4,9	2,5	4,5	1,7
7,3	2,9	6,3	1,8
6,7	2,5	5,8	1,8
7,2	3,6	6,1	2,5
6,5	3,2	5,1	2
6,4	2,7	5,3	1,9
6,8	3	5,5	2,1
5,7	2,5	5	2
5,8	2,8	5,1	2,4
6,4	3,2	5,3	2,3
6,5	3	5,5	1,8
7,7	3,8	6,7	2,2
7,7	2,6	6,9	2,3
6	2,2	5	1,5
6,9	3,2	5,7	2,3
5,6	2,8	4,9	2
7,7	2,8	6,7	2
6,3	2,7	4,9	1,8
6,7	3,3	5,7	2,1
7,2	3,2	6	1,8
6,2	2,8	4,8	1,8
6,1	3	4,9	1,8
6,4	2,8	5,6	2,1
7,2	3	5,8	1,6

7,4	2,8	6,1	1,9
7,9	3,8	6,4	2
6,4	2,8	5,6	2,2
6,3	2,8	5,1	1,5
6,1	2,6	5,6	1,4
7,7	3	6,1	2,3
6,3	3,4	5,6	2,4
6,4	3,1	5,5	1,8
6	3	4,8	1,8
6,9	3,1	5,4	2,1
6,7	3,1	5,6	2,4
6,9	3,1	5,1	2,3
5,9	2,7	5,1	1,9
6,8	3,2	5,9	2,3
6,7	3,3	5,7	2,5
6,7	3	5,2	2,3
6,3	2,5	5	1,9
6,5	3	5,2	2
6,2	3,4	5,4	2,3
5,9	3	5,1	1,8

Priedas 2. Oro taršos duomenų bazė (NO_2)

3,71844	7,6912	9,2	4,8	-0,1	74,4	20	600
3,10009	7,69894	6,4	3,5	-0,3	56	14	196
3,31419	4,81218	-3,7	0,9	-0,1	281,3	4	513
4,38826	6,95177	-7,2	1,7	1,2	74	23	143
4,3464	7,51806	-1,3	2,6	-0,1	65	11	115
4,16044	7,67183	2,6	1,6	0,3	224,2	19	527
4,01277	5,52545	-7,9	1,6	0,3	211,9	5	502
2,15176	4,68213	-4,1	3,8	-0,1	63,1	4	453
3,157	7,15618	-12,7	5,2	-0,1	64,5	12	462
2,37955	4,74493	-1,6	3	0,4	58,3	3	554
3,83298	5,81114	-3,1	1,8	0,3	78	2	55
4,48187	8,10892	1	1,2	1,5	215	18	47
4,0483	8,31385	12,2	4	-2,8	230,4	17	572
4,00186	5,22036	-1,5	2,4	0,9	82,7	5	556
3,2308	6,40853	-0,9	3	0,1	235	1	69
4,67189	7,3192	-8,5	0,8	2,9	282,4	20	447
2,73437	6,6174	6,5	4,1	0	88	24	186
3,49651	7,76938	4,2	7,1	-1,1	9,3	10	550
3,67122	6,4677	-1,1	2,5	1	277	6	142
3,67377	7,65064	0,8	3,4	-0,4	70	12	167
3,15274	7,75061	8,2	4,5	0,2	307	14	32
3,42751	5,18178	0,2	0,4	0,2	230	5	112
4,32413	7,63964	-2,1	4,3	-0,2	41,4	11	432
3,65584	8,00703	-2,8	6	-0,2	62,5	15	453
3,93183	7,75319	-5,4	2,5	1,4	83	20	98
2,28238	4,65396	-5,7	4,9	-0,1	58,5	4	409
4,13035	6,67582	6,6	1,6	0,4	104,5	23	579
3,77963	7,79688	5,7	4,8	-1,5	223	11	178
3,97968	8,02224	13,8	0,8	-0,1	201,6	17	602
3,7542	7,48829	3,2	4,2	-0,6	285	12	121
3,45632	7,75148	2,1	4,9	0,2	149	19	60
2,36085	6,12468	5,3	3,2	0	73	1	209
4,23411	7,71244	7,5	1,8	-2,7	273,9	12	530
2,2192	7,56941	10,7	3,1	-0,9	56	13	189
4,53475	7,35947	-9,6	1,3	0,1	64,6	21	437
2,10413	7,99564	3,2	6,5	0	42,8	18	553
4,5819	8,30251	3,5	0,5	-2,2	207,9	8	535
4,03247	5,95584	6,7	1,1	0,6	89,9	5	605
3,21084	6,74052	-6,4	2,8	-0,1	77	22	84
4,04655	6,45205	-3	0,6	1,7	57	6	73

3,94546	6,39359	-13,4	4,4	-0,1	80	6	95
3,89182	7,16317	1,4	0,9	0,2	274	10	76
3,22287	7,643	8,5	4,3	-0,2	322	13	32
3,3569	5,93754	-2,8	2,1	-0,2	65	1	124
3,31782	7,61923	16,6	4,8	-1,1	136,9	14	595
3,81551	8,34854	2,7	7,7	0,1	219	17	159
3,45632	8,1274	19,1	5,1	-1,2	266,6	16	603
2,66723	6,83626	7,8	7,5	0,8	220	21	34
3,76815	5,27811	1,5	3,3	1,6	251	5	156
2,87356	5,68358	0,3	2,1	1,1	220,4	5	538
3,48431	7,76004	-7,3	0,5	-0,3	176	13	84
2,92852	6,30079	-1,6	2	0,1	289,1	7	510
3,32143	5,52146	1,1	1	0,2	9	5	113
4,35543	7,79811	16,1	1,7	-0,2	246,1	19	570
3,7329	6,14419	-7,9	1,2	0,3	56	11	86
2,70805	6,54965	1,1	2,8	0	66	24	59
2,4248	5,64545	0,3	7,1	-0,1	69,5	2	425
3,45632	6,632	4,1	3,9	0,3	212	6	106
4,26127	7,64012	5,9	2,4	2,6	249	16	48
2,57261	6,86589	-6,8	6,9	-0,1	50	10	90
3,79549	6,6896	0,3	0,6	0,7	252	21	104
4,08429	7,36771	1,8	1,7	0,7	91	22	140
5,11259	7,6256	-7,9	1	-0,2	188,9	12	443
3,37417	5,50533	3,4	6,3	0,6	186	2	475
4,13035	8,06778	3,9	2,8	-0,1	43,5	18	576
4,35927	7,76345	9,2	5,7	-0,9	199	14	212
4,27667	8,25713	11	5,5	-1	216	17	193
2,69463	5,57595	-3,5	4,6	-0,1	62,3	5	418
4,04655	7,93487	9,7	1,8	-0,6	255,5	15	529
3,91202	6,32972	1,5	1,6	0,5	275	24	114
3,19458	4,83628	2	1,7	-0,1	63,1	4	549
4,01998	7,78322	3,7	2,2	0	40,5	19	576
3,2068	7,67276	-7,1	1,5	-0,4	53,1	15	517
3,8111	8,22013	2	3,5	-0,2	82	8	64
3,87536	7,53155	-0,7	3,3	-0,2	179,3	20	522
4,43438	7,83992	1	6,9	-0,3	81,7	11	558
2,85647	5,0626	2,4	2,1	0,3	216	4	177
4,2312	8,17245	4,2	5,9	0,3	214	16	108
4,11087	8,24905	0,8	1,3	0,2	215	8	162
3,48124	6,53088	0,5	2,1	0	83	23	58
4,10099	8,18702	-4	0,9	0,9	245,9	16	513
3,32504	7,81682	-3,6	4	0	43,4	18	409
4,75014	7,62462	3,8	2	2,8	283,1	20	543

2,62467	7,23634	5	3,5	-0,1	62	23	196
3,99268	7,98718	-1,6	5,3	0,6	227	18	81
3,95316	7,41156	3,3	6,8	1	197,2	20	472
3,89589	7,91681	-9,1	1,6	0,6	21	14	83
5,115	7,84463	-0,1	1	1,8	174	18	50
2,11626	4,68213	-7,6	4,4	0,1	66,2	3	407
3,71113	8,21311	1,1	9,3	0,1	213,9	17	481
4,3399	6,1506	-0,6	1,3	1,2	337	7	146
2,00148	6,29342	5,6	4,3	0	98	1	211
4,2312	5,8944	-1	1,9	2,7	164	1	397
3,96651	6,57088	5,5	0,8	0,1	89	6	206
4,14155	8,3163	6,3	1,2	1,3	265	17	33
4,21065	7,83003	9,5	0,8	0,1	110,7	19	597
4,54223	8,26178	1	3,4	1,3	248	8	43
3,9279	5,47227	-3,8	3,6	1,9	76,7	6	469
2,93916	6,53088	-4,2	3,8	0,1	64,3	8	503
2,5177	7,0775	4,9	4,9	-0,1	229	11	126
3,39786	4,77912	-3,7	0,9	-0,1	311	2	513
3,19458	5,7462	-6,4	5,1	-0,1	82	6	118
3,82864	4,77912	-1	1,1	2,3	291	3	36
1,335	5,47646	-1	4,4	0,4	5,8	5	553
4,44852	7,87131	5,9	1,3	-0,1	85	9	184
3,71601	7,19519	-2,5	5	0	66,3	22	427
1,79176	4,89035	-4,1	6,3	-0,1	45,1	6	455
4,75617	7,82724	-10,1	1,7	0,6	70,8	7	443
3,88773	5,71043	-2,1	1	0,5	211	6	55
2,95491	4,80402	2,8	5	-0,1	80	4	186
4,31482	7,93164	14,2	8,1	-2,1	209,1	14	577
4,12066	8,17527	-3,9	2,8	0,3	52,4	16	486
3,37759	5,76205	0,1	0,4	0,2	145	2	112
3,99083	7,02909	-16,2	2,6	0,9	73	22	91
4,11087	7,62217	-3,4	2,9	1,7	213	20	81
3,01062	7,17396	-7	7,8	-0,1	49	11	90
4,10759	7,5475	-8	0,3	-0,2	289	14	95
2,54945	5,52943	-1,5	7,5	0,3	3	5	40
1,50408	5,86079	6,5	3,6	0,1	49	2	196
3,57235	7,32778	-4,9	4,6	0,3	359	21	116
4,56331	8,1778	-4,2	1,4	0,2	86	16	101
4,38577	7,73237	-5,2	0,6	0,1	91	7	74
3,44042	6,79571	-4	3,7	-0,2	56,9	11	434
1,72277	6,06146	-4,7	4,8	-0,3	44,8	9	406
4,29729	7,6353	9,5	1,6	-5,4	226,6	11	529
3,35341	7,04925	2,3	3,1	0,7	225	22	120

4,3386	7,67879	4,1	4,4	0,5	297,1	10	473
2,99072	6,93925	4,2	5,5	-0,4	238	9	160
4,237	6,3315	-5,1	1,6	1	251,6	8	468
4,5819	8,26256	3,8	0,7	-0,4	161	8	191
4,17285	7,91498	-3,5	1,8	0,2	222	18	520
3,56105	6,39024	2	0,6	1,1	177	7	139
3,16548	6,34036	-4,1	2,9	-0,1	60,6	8	433
4,39198	7,91206	-5,2	4,5	-0,1	76	9	120
3,01062	7,96728	9,8	2,2	0	44	18	189
4,40672	7,83637	8,9	0,5	-0,6	296	15	184
4,29729	8,21203	2,9	2,2	-0,1	73	17	129
2,64617	5,44674	1,6	2,9	-0,1	220,9	5	545
3,84588	7,66763	0,6	5,8	0	42,1	14	480
4,62399	7,59337	-7,5	3,2	-0,2	59,7	11	435
3,68888	6,52649	-0,3	6,1	-0,2	68,9	6	429
3,79098	8,31899	1,6	2,7	-0,3	209,9	16	523
3,96081	7,32712	9,3	4,3	-0,2	214,1	21	582
4,51196	7,25276	1,1	1,4	1,2	78,9	21	530
3,82647	7,9848	-8,8	2,1	-0,2	48	15	88
4,58802	6,64509	-6,8	1,1	0,1	145,8	24	444
3,02529	7,05531	11,2	3,3	2,3	302,8	23	604
3,66356	7,80344	4,9	2,9	0,4	189	15	133
3,48431	7,61283	20,3	4,2	-3,9	278,9	10	607
2,55723	7,67555	6,8	9,9	-0,2	249	15	160
4,83787	8,19589	-4,2	0,8	0,4	247,2	16	445
4,30271	8,13124	4,1	6,5	0	58,1	9	576
3,25424	6,92756	1,8	4,2	0,3	207	23	130
3,85862	7,52348	3,4	6	0,2	191	14	482
4,21656	7,0076	-3,4	1,5	1,7	76	23	44
4,08261	8,19616	-4,1	0,4	0,4	270,5	17	401
2,5337	4,81218	-6,4	4,7	0,4	75,9	7	451
3,61092	7,61431	0,9	3,9	0,1	107	19	65
3,45947	7,94058	4,3	5,6	-0,9	64	12	174
3,23475	7,4378	2,1	1,6	1	205	20	178
3,84802	8,01334	1,2	2,3	0,1	245	17	168
3,23868	4,54329	6,5	3,7	0	80	4	185
4,23555	7,97488	-1,6	3,5	0,5	34,5	18	480
2,87356	7,23634	1,2	4,3	0	64,3	21	404
4,41764	8,25192	0	3,5	-0,2	77	8	157
4,14155	7,57147	8,4	1,8	-0,9	108	13	191
3,14845	6,74993	-2,8	2	-0,9	231,8	10	504
2,4248	4,56435	-3,9	1,4	-0,1	76,8	4	521
4,02535	6,95559	-7,5	1,3	2,3	236	20	86

4,21656	7,5974	0,3	0,8	1,7	113,3	22	483
3,46574	5,88888	9	1	0,2	151,6	2	601
2,83321	5,64191	4,6	2,5	0,7	229	2	125
2,44235	6,4151	3,6	4	0	68,2	24	573
4,16667	7,85941	1,3	2,2	0,2	227	7	141
4,22244	5,86363	-0,6	0,6	1,7	142	1	79
4,60717	7,82285	3,4	2,3	0,5	8	15	44
1,62924	5,50533	1,5	4,3	0,2	156	4	62
4,3	8,19864	2,4	3,4	0,4	186	16	148
2,73437	6,12249	4,5	2,9	0,6	196	1	125
4,89933	7,98071	-11,6	3,1	0,3	78,8	9	444
4,07414	7,9179	2,7	2,5	0	212	9	131
4,02356	7,71423	-2,1	3,2	-0,1	39,9	7	432
4,25135	7,95893	-1,7	3,7	0	40,8	15	431
3,65325	7,61628	5,1	2,9	1,1	192	20	192
4,26549	5,17615	-2,7	2,2	1,5	219	3	153
4,47506	7,29029	10,8	2,3	0,5	85,5	22	570
3,16969	7,82084	18,1	3,7	-2,1	244	16	568
2,80336	6,87316	5,6	7	0,6	191	21	125
3,69387	7,96207	7,3	2,8	0,7	317	16	35
3,88568	6,92166	6,4	0,5	-2,1	71	9	188
4,35414	7,6834	3,6	2,9	-0,3	74	10	187
4,4128	7,21008	-1,8	1,8	1,1	2	21	135
3,08649	4,84419	7,7	3,5	0,1	81,8	4	589
4,37827	8,19589	-3,2	2	3	211,7	17	400
3,87536	6,86485	0,8	1,5	-0,1	231,5	23	540
4,05352	7,79194	8	4,1	-0,2	207,6	13	585
4,19117	5,77144	-0,4	0,4	1,9	240,4	2	539
3,10459	8,21906	11,6	5,5	-2,5	231,4	8	586
4,19117	7,71244	-9,9	2,7	0,7	78	21	119
4,25561	7,86019	-4,9	2,6	-0,1	243,1	9	481
3,18221	6,8977	-1,4	4,8	-0,1	72	23	57
2,72785	7,18992	7	4,1	0,4	159	22	212
4,28496	7,79975	-0,8	0,8	1,1	80	18	113
3,09558	5,54126	-1,4	2,2	1,7	212	1	142
4,39198	7,94201	-4,2	1,3	-0,2	240	15	74
3,24649	5,20401	6,7	1,5	0	80,5	2	592
4,27528	7,42536	1,1	1,5	2,2	273,1	20	540
2,78501	6,64379	-5,7	1,9	-0,3	74,2	23	517
3,79549	7,83003	6,4	4,3	-0,1	30,4	13	397
4,15261	5,17615	-1,7	0,4	1,1	183,1	5	535
4,07244	6,53669	-0,8	2	0,8	70,6	24	514
4,36055	7,64588	5	1,5	-1,1	243,2	14	527

3,07269	7,61825	8,4	2,8	-1,1	133,1	13	581
2,79728	4,62497	-9,6	2,9	0,8	73	6	86
3,59731	7,46851	-3,5	0,7	0,2	165,1	20	520
4,55703	7,74803	-4,1	2	-0,1	85	15	99
4,70863	7,52726	-6,4	3,3	0	60,5	13	435
4,3108	7,88758	-5	4,7	-0,1	47,2	9	409
3,59182	6,80351	-9,3	3,6	0,3	73	8	83
4,14472	7,80098	8,5	4,5	-0,2	203,2	19	591
3,02042	7,96346	3,6	3,2	1,2	268	18	42
3,17388	6,91473	4,8	4,6	0,2	199	23	109
4,30676	7,80262	-4,6	5,7	-0,2	81	18	122
3,81551	7,51152	0,4	3	-0,8	89,8	11	398
2,83908	7,47022	10,8	2,6	-3,5	262	12	203
3,83945	5,273	-13,8	3,8	0,2	79	5	95
3,85651	8,14989	5,4	3	0,1	234	15	131
4,2485	6,61607	0,1	0,4	2	122,3	22	531
2,96527	6,47851	4,3	3,2	1,1	218	24	138
4,53045	7,5974	2,6	1,3	0,2	233	12	50
3,21487	7,17319	-6	5,6	-0,2	65,5	22	408
3,12676	4,91998	6	3,9	-0,1	72	2	576
3,50255	7,23129	8	3,8	1,7	207,9	22	583
4,51196	7,56631	-5,5	3,5	-0,1	65,1	12	436
3,3673	7,65728	2,5	2,5	-0,2	60,8	10	404
4,42963	6,65028	-3,7	0,5	1,2	325,6	24	445
4,03954	8,08887	15,1	1,9	-5,1	240,7	9	599
4,1463	5,93754	-7,8	0,9	1	111,7	6	447
1,97408	5,43372	7	4,7	0	27,5	2	593
3,76584	7,77317	2,2	3,7	-0,7	211,7	13	523
4,01638	6,39359	-5,8	1,2	1,3	73,7	24	480
3,09104	7,85516	6,5	5,2	-0,2	69	19	196
4,07073	7,33433	-14,7	0,8	0	212	17	457
3,94158	7,64826	4,8	4,4	-0,6	229	13	141
3,43076	7,662	3,4	2,9	0,6	23	18	174
3,627	7,72533	-2,3	4,1	-0,1	61,9	19	417
3,87743	7,46107	-2,8	1,4	-0,1	243	20	102
3,05871	6,2106	5,9	3,3	1,8	248	24	41
3,91202	7,61481	6,9	2,7	0,1	178	19	190
3,96651	5,43808	-0,9	1,6	4,3	96	6	49
4,28496	7,43248	-10,3	4,1	-0,2	61,6	12	459
4,17746	5,20949	-12,9	1,1	1,6	219,5	3	489
4,26549	6,75809	-4,7	1,6	0,7	262,9	9	468
4,01096	6,58203	2,2	1,8	0,1	64,2	6	549
3,15274	5,82305	4,6	4,3	0,3	203	1	127

4,1957	8,12415	5,2	2,8	0,3	209	8	33
3,5293	4,7362	2,9	2,4	0,6	80	3	200
4,38701	7,89283	-3,8	3	0	75,3	15	423
2,93386	7,99564	12,2	5,4	-1,4	234	17	203
3,26576	4,92725	4,4	2,2	0,2	82	2	170
2,83321	5,15329	3,1	1,5	0	179	3	202
4,13357	7,78239	4,8	6,5	-1,1	41,1	10	556
3,80444	7,70841	2,1	3,1	0	79	10	64
4,1239	8,23854	-1,7	5,4	-0,1	58,2	17	417
4,75359	7,82844	3,6	0,8	1,5	87,7	19	529
4,40305	7,68386	0,6	4,7	-1,2	58,2	10	417
2,71469	7,70075	11,5	7,3	-2,1	216	14	210
2,8792	5,2575	4	8,3	0,3	218	5	46
3,3499	7,97728	4,9	2,8	0,3	194	17	126
4,12713	5,42935	-7,7	2,5	0,9	352	2	81
4,49647	7,27586	-7,2	1,3	1	85	21	143
3,84802	7,5761	-5,5	5	-0,1	82	17	118
3,42426	7,82164	16,1	3,1	-1,5	259,3	17	563
4,07244	8,17611	-2	3	1,3	222,5	17	509
4,33336	8,14148	-2,8	4	-0,3	79	8	149
4,26127	7,53636	-2,2	1,7	-0,7	250	13	137
5,23644	8,16934	-1,5	1,9	3,2	83	8	78
4,07414	8,08672	-13,6	3,1	0	77	8	96
4,20469	7,06647	0,2	3	1	90	21	174
3,54385	7,71065	15,8	3,7	-0,3	136,4	16	595
3,81331	5,61313	-4,4	2,9	0,3	74	1	101
3,17388	8,05261	8	6,8	-1,2	280,1	16	546
4,22244	7,46566	6,4	9,6	0,6	264,3	13	471
3,87743	7,78738	7,5	8,1	-0,5	220	15	156
4,00186	7,96555	1,1	3,9	0	78	9	59
3,2308	5,99645	8,6	2,6	0	82	5	600
4,09268	8,27053	-5,1	1,4	-0,7	248,3	16	516
3,13549	4,90527	-8,9	3,9	0,4	78,1	4	450
4,20767	5,97635	-9,7	0,8	0	67,7	1	438
4,237	7,64826	-4,1	2,9	-0,4	85,3	14	519
3,31419	7,02554	-1,6	2,9	0	222,2	22	509
4,18662	7,62071	-0,8	4,7	-0,1	61,4	12	417
3,81331	5,69373	2	1,2	0,4	76	1	141
4,14472	7,74414	-1,3	4,2	-0,2	59	15	149
2,87356	6,40688	-4,1	2,1	-0,1	146,8	24	520
5,02651	7,8808	-6,9	2,2	1	158	7	148
1,70475	5,11199	-4,4	4,2	-0,1	55	4	119
2,8792	7,17855	4,9	5	-0,6	76,2	9	573

3,72086	5,09987	5,8	3	0	78	3	184
3,36038	4,45435	-2,8	5,3	0,4	67	3	508
3,8111	7,6516	7,2	1,5	-0,4	248	16	531
3,91602	5,75257	-7,4	0,9	2,1	293	11	93
3,72569	6,54965	-9,2	4,2	-0,2	78,8	6	408
5,13049	7,26543	-1,7	1,3	1,4	114	21	50
4,90602	8,19699	-10,6	3,4	0,3	81,1	17	438
3,72569	5,07517	-4,8	0,6	0,1	103,7	2	520
3,35341	8,03948	4,9	5,1	-0,7	338	9	172
4,31348	8,1371	6,4	2,2	-2,6	246,6	9	575
2,41591	6,85435	1,2	5	0	65,9	23	404
4,50424	7,51915	3	3,4	0,6	44,1	20	528
2,36085	4,5326	-6,5	2,7	0	77,5	6	451
3,48431	4,85981	-6,3	3,1	0,6	73	2	143
4,08429	7,55538	-4,2	5,3	0	59	15	426
5,37389	6,31355	-13,5	0,9	3	82,5	24	465
4,16044	6,31536	-10,4	2,5	-0,2	62,3	6	460
3,88156	7,52618	-5,5	3	1,6	75	21	98
4,08092	7,74803	6,1	6	0,1	180	15	128
3,23868	6,93049	2,6	2,3	0,6	193	23	163
4,17131	4,76217	1,7	0,3	1,3	112,5	3	563
5,58237	7,70481	-0,2	0,6	1,4	220	12	79
2,4248	5,68698	11	1,2	-4,1	286,6	7	594
4,3108	7,63916	-10,2	4,2	-0,2	58,6	15	459
4,32678	7,67508	3,2	1,7	1,1	249	14	52
2,5096	5,3033	-0,6	2,8	0	76	5	56
4,38203	7,15696	3,3	1,7	0,5	90	22	191
3,71601	7,98582	-2,3	4,2	-0,1	77	18	399
4,2032	7,97384	7,4	1,7	-0,2	165	9	208
4,31749	7,74544	2,2	6,7	0,3	56,6	10	470
3,30689	5,50939	-1,4	5,4	0,5	88	2	167
4,01818	5,70044	2,7	1,2	1,7	185	1	180
3,27714	5,6204	6,2	3,5	0	70,9	5	582
3,27336	7,02909	4,5	2,8	0,2	75,5	21	573
3,92197	7,93057	-8,2	2,6	0,4	50,4	18	407
4,06044	7,59488	4,9	2,9	-0,1	77	11	170
2,99573	7,23346	-2,2	6,1	-0,2	62,9	21	418
4,07923	7,60639	5,5	5,4	-0,6	54,9	11	528
4,17285	7,85671	6,3	3,5	-0,1	77	7	185
4,09101	7,59085	7,5	6,1	-0,5	238	14	162
2,77882	4,86753	2,1	3	0,6	217	2	177
4,2485	8,20303	3,5	3,3	-0,2	51,9	16	576
2,59525	5,90808	-8,1	3,8	-0,1	48,5	1	456

3,13549	5,40268	1,3	0,9	0,7	68,4	5	581
3,06805	6,55393	3,7	2,8	1,1	241	23	132
4,10594	8,00001	4,3	4,6	-0,2	215	17	176
3,2068	7,42476	-3,2	4,9	-0,4	78,9	12	399
4,10099	7,44366	-4,5	5,1	-0,2	47,9	13	409
3,69883	7,58426	10,5	2,1	0,5	214,5	20	598
3,28466	7,7878	0	2,7	-0,1	82	18	70
3,7495	7,99934	4,8	2,6	0,3	194	16	126
3,81551	7,08841	0,7	1,1	0,5	295	22	114
3,97218	7,12125	5,7	1,5	1	65,4	23	593
4,92435	8,05769	2,3	1,3	0,5	152	9	36
2,89037	7,76853	21,1	4,5	-3,6	285,1	11	607
3,79549	7,04141	-0,7	2,7	1,2	73	22	72
2,5416	5,96871	11,1	2	0,2	153,2	3	602
2,4248	6,81892	7,2	3,8	0	58,5	9	578
4,32546	8,18228	4,1	2,1	0,3	177	8	127
3,32504	7,21671	-0,8	3,5	0,3	56,1	23	420
3,97406	7,78489	0,7	2,1	0,6	219,9	7	551
5,32933	4,67283	-14,3	0,5	2,3	257,4	4	466
1,74047	4,31749	1,7	2,3	0	75	3	64
4,50092	8,16138	-2,2	1,7	1,1	69	8	73
4,3412	8,22604	-5,6	2,5	0	87	17	68
3,79098	7,80221	2,5	5	0,1	135	7	128
3,83945	7,17089	-5,3	2,6	0,1	69,4	22	442
4,50314	7,88833	3,8	1,1	2,2	34,9	19	532
4,36691	7,65776	3	1,9	2	265	19	540
3,7013	7,40428	-1,9	3,4	-0,1	43,1	21	431
4,75875	7,6857	1,8	1,8	-1,1	269	10	138
4,56954	8,08364	-7	1,4	-0,1	242,3	15	467
1,22378	6,16331	-3,7	3,9	0,3	30,9	1	406
6,39509	7,57302	-4,6	0,8	1,3	87,2	14	464
4,24133	7,02376	-13,6	2,9	0,1	82,1	22	488
4,57883	7,97281	6,3	4,2	-0,6	81	9	194
3,93769	5,81413	-9,1	2,1	0,6	79	3	83
3,92986	6,608	7,2	0,6	1,3	191,4	24	598
3,33577	7,35628	4,4	2,4	0,3	76,6	20	573
2,95491	7,30586	15,4	1,6	-2,2	279,2	14	565
3,55535	5,11199	0,7	1,1	1,6	48	4	161
2,69463	5,56834	4,8	1,6	0,6	220	2	204
4,32678	6,56386	-1,2	2,1	1,9	192,9	6	446
4,88129	7,84189	0,4	2,9	1,9	279	18	43
4,56539	7,67555	-3,1	1,3	0,3	82,1	14	423
4,20916	8,31066	11,3	3,4	0	200,3	8	603

3,21888	5,52545	-5,8	5	-0,2	70	7	441
2,65324	5,54908	5,5	2	-0,1	75,1	2	582
4,34251	8,06401	3,6	2,3	-0,3	9,4	9	549
3,64545	5,09987	6,7	4,3	0,6	183,8	2	590
1,54756	5,49306	7,1	5,4	1,5	255	6	42
4,18205	7,7411	-5,7	2,7	-0,5	85,4	10	401
4,14946	8,3464	20,8	2,4	-2,4	230	16	605
3,80444	7,91206	1,2	3,3	0,7	219	19	537
4,07923	7,60937	-0,1	3,9	-0,2	74,3	10	430
3,14845	6,76734	-6,8	3,3	0	76,3	24	450
4,24276	7,61727	-4,2	3,7	-0,1	217,8	10	481
2,61007	5,80212	2,2	1	1,3	299	6	140
3,19458	4,55388	2,5	3,9	0,1	195,5	3	548
3,81331	7,57763	-0,2	2,8	-0,1	85	16	111
3,77046	8,06746	19,6	2,6	-2,8	244,4	14	605
2,76632	5,66643	1,3	2,1	1,8	226,8	6	546
3,21084	5,49717	4,2	4,4	1,5	254	5	80
3,37417	8,07527	3,4	5,3	-0,9	10,5	9	550
3,72086	8,02027	-5,2	4,3	-0,1	17	16	116
4,76217	7,6024	-3,9	2,8	0	76	10	68
4,01458	8,25635	8,4	3,1	-0,5	225	17	207
4,30407	7,24708	-10,6	5,5	0	62,5	10	456
3,17805	7,93272	-3,9	4	-0,3	78,4	15	452
1,36098	4,39445	1,3	4,4	0	115	4	59
4,40916	7,66856	-2,9	1,6	0,7	65,3	20	446
4,02892	7,26613	0,9	0,9	0,4	90	21	75
2,89037	7,70616	2,1	6,3	0	44,4	13	574
3,88156	8,11851	0,6	7,1	-0,2	74,6	8	425
4,51852	7,75319	-5,6	1,5	0,1	238,8	10	445
4,01096	7,65207	-4,5	4,2	-0,2	100	19	423
3,74242	7,66011	6,4	3,5	-0,2	90,8	20	571
3,71601	7,59035	11,7	6,4	-0,2	235	13	34
2,85071	6,70564	-5,6	2,4	-0,4	75,5	10	518
4,51305	8,17132	-5,6	3,5	0,6	79	8	101
3,99452	7,76217	12,2	6,2	-2,1	221	14	193
2,40695	7,22621	-6,5	3,8	-0,1	52	21	89
3,91202	7,53423	1,6	1,7	-1,5	336	11	135
4,45899	7,49165	-9,5	2	-0,1	77,3	20	437
3,78872	7,83953	11,5	2,5	-2,7	243,6	15	575
3,92197	7,61874	-12,6	2,1	0,5	81	15	91
3,16969	7,82724	17,8	3,5	-1,7	254,1	17	568
4,46245	7,59589	-2,7	1,3	1,2	159,1	19	400
2,92852	4,81218	2,7	1,9	0,3	204,2	2	479

5,78105	7,87284	1,7	1,2	1,4	97	15	78
1,28093	5,65948	-5,1	3,9	0	46,5	7	406
4,09101	8,04206	6,2	1,2	1,3	261	18	33
3,62966	7,15774	-1,1	1,6	-0,1	65	22	150
3,37759	6,24417	2,7	1,5	0,4	196,9	24	478
5,54713	8,14206	-5,1	1,5	1,5	237	8	152
2,63906	4,49981	3,9	3,6	0,1	68	3	198
3,99268	6,57508	6,2	1,3	1,1	226	24	200
3,66099	4,64439	-3,3	1,9	1,3	88	3	51
2,82731	4,98361	-5,6	4,3	-0,1	78	5	85
4,29592	7,72356	-5,1	0,7	-0,2	216,6	11	445
4,94876	7,76132	0	0,7	0,6	147,7	18	470
2,68785	6,18002	8,9	4,8	1	215	1	34
4,0483	7,21598	8,8	2,1	0,7	90,1	21	579
3,32863	5,50939	10,3	2,2	1,4	37,4	4	608
4,13996	4,99043	1,3	2,3	2,2	277	2	53
3,74242	7,04229	-5,2	2,5	-0,3	73	11	85
4,28082	7,35308	2,4	1,1	0,9	77	21	187
2,79728	4,12713	2,5	5,6	0,3	213,5	3	526
3,65584	7,7698	6,1	2,5	-1	237,2	13	545
4,02535	7,58731	2,1	3,4	-0,7	354	11	40
3,44042	5,63835	-13,8	1	1,8	61	7	91
4,42485	7,64204	10,2	2,6	-2,1	249,5	14	534
3,94352	5,26269	-0,9	1,1	2,1	271	5	48
4,21804	7,25205	7,1	2,4	0,4	63	21	197
3,44999	5,19296	-18,6	2,7	0,8	86	5	92
2,52573	7,31986	16,8	3,4	-0,3	41,5	17	564
4,4705	7,68064	0,3	1,3	1,4	221,8	19	469
4,18205	8,1568	-0,7	5,1	0	57,2	16	425
3,17388	6,94794	-3,4	0,6	-0,5	97	11	119
4,62399	7,14204	-1,6	2,1	2	85	22	78
2,73437	5,0626	-2,1	2,5	-0,1	81,7	4	400
3,2581	6,13123	3,5	8,4	0,6	209,9	7	475
3,22684	6,85857	-4,8	3,1	-0,1	86	23	115
2,98062	5,39816	0,7	1,3	0,8	208	6	180
3,84374	5,14749	-9	1,3	0,4	77	2	88
4,26127	8,16707	6,3	4,8	-0,3	61	15	194
3,39786	6,6174	2,2	3,9	0	67	6	199
3,39451	7,43248	8,9	2	-2,4	201,4	12	581
4,36437	7,76302	12,3	2	-3,7	273,4	11	563
2,26176	5,42495	1,1	3,8	-0,1	79	5	70
3,44042	7,12769	0,1	4,2	1	235	22	141
4,12713	7,41998	-4,6	3	-0,1	82	20	115

2,98062	6,89163	0,1	1	0,5	334	23	66
3,65066	7,97247	-4,8	3,7	0	16	18	116
4,25135	7,75018	9,9	1,8	-1,3	266,3	14	529
3,65842	7,42536	4	5,8	0,2	184,6	20	402
3,99083	7,63723	-5,4	4,1	-0,2	72,5	17	440
4,44147	6,55108	-12	2,8	0,4	82,5	6	444
2,28238	4,98361	1,8	3,5	0,3	121	4	128
3,9279	6,57508	5	0,8	0,3	262,7	6	403
3,3569	7,09257	4,6	3,4	0,2	194	22	109
4,18814	7,83201	-0,2	0,6	-0,3	99,7	7	541
4,2017	7,55329	-0,9	1,1	-0,3	92	13	51
2,41591	7,23778	11,9	4,2	-1,3	233	15	180
4,33336	8,20084	-9	2,4	0	84	8	100
1,58924	5,81711	1,3	3,8	0,3	33	5	174
4,0483	8,26462	7,7	2,6	-0,8	69,1	16	571
2,65324	5,05625	-7	3,2	-0,1	67	20	85
4,06732	7,73193	2,6	4,2	-0,2	77,9	14	514
3,86283	5,99396	1,6	1,7	1,5	209,1	1	545
3,98155	7,64108	6,7	2,3	-0,4	247	10	33
3,81331	7,74587	1,6	2,1	-0,1	146	17	167
4,31615	5,14749	-6,5	1,7	0,8	184	5	154
3,33932	5,66643	6,7	2,4	-0,1	66,1	5	592
3,96651	7,34923	-0,4	1,9	1	77	21	72
3,86073	4,45435	-13,8	4,2	0,2	80	4	95
2,56495	4,58497	1,8	2,3	0,1	59	4	164
4,30946	7,68202	3,5	5	-1	78	11	166
2,94444	6,52942	9,5	6,5	-0,8	210	10	35
4,17439	7,75791	5,2	4,6	-0,8	214	14	176
2,95491	5,78996	8,4	0,5	-2,6	108,5	7	588
4,03247	8,16223	4,7	5,9	0,4	207	17	128

Priedas 3. Gauti rezultatai. 1 palyginimas

Su IRISŲ duomenimis atlikti tyrimai

- „matlab“ – metodas, kai tinklui paduodami vektoriai iš eilės po vieną. Skaičiuojama klaida ir siunčiamas vektorius atgal, atnaujinami svoriai ir paduodamas kitas, naujas vektorius.
- „web1“ – metodas veikia lygiai taip pat kaip ir „matlab“ metodas, tik VS Web Developer programos terpėje.
- „web2“ – metodas, kai paduodamas vienas vektorius, t.y. visų mokymo duomenų vidurkių vektorius.
- „web3“ – metodas, kai paduodamas vienas vektorius, t.y. visų mokymo duomenų sumos vektorius. Prieš sumuojant mokymo vektoriai dauginami iš atsitiktinių koeficientų.

Neuronų sk.	Const	Matlab	laikas	Web1	laikas	Web2	laikas	Web3	laikas
10	0,1	0,0000335695	4,97	0,0029445222	00:00:28	0,000000000042269	00:00:00.4211595	0,0000310994	00:00:00.5307706
11		0,0000234456	5,19	0,0000641903	00:00:30	9,47E-12	00:00:00.4995488	0,0000238284	00:00:00.5307706
12		0,0000180477	5,27	0,0002350397	00:00:28	2,61E-12	00:00:00.4839379	0,0004506819	00:00:00.5619924
13		0,0000151249	5,75	0,0000461343	00:00:33	7,25E-13	00:00:00.5463815	0,0000141500	00:00:00.5932142
14		0,0000190615	5,91	0,0002178311	00:00:36	1,50E-13	00:00:00.5151597	0,0008977205	00:00:00.5932142
15		0,0000092832	7,36	0,0024971011	00:00:33	3,83E-14	00:00:00.5307706	0,0000096331	00:00:00.6088251
16		0,0000205810	6,69	0,0007486593	00:00:43	8,69E-15	00:00:00.5463815	0,0007795115	00:00:00.6400469
17		0,0000056810	10,38	0,0000975800	00:00:44	2,46E-15	00:00:00.5307706	0,0000695092	00:00:00.6400469
18		0,0000101730	10,89	0,0024798717	00:00:46	6,77E-16	00:00:00.5776033	0,0000589352	00:00:00.6556578
19		0,0000036999	11,78	0,0011471844	00:00:50	1,32E-16	00:00:00.6088251	0,0000025405	00:00:00.6712687
20		0,0000169221	11,72	0,0037438025	00:00:53	3,27E-17	00:00:00.6090006	0,0002141062	00:00:00.6870776

Duomenys esant konstantai 0.1 ir epochų skaičiui 400.

Neuronų sk.	Const	Matlab	laikas	Web1	laikas	Web2	laikas	Web3	laikas
10	0,2	0,0000345261	4,64	0,0001410049	00:00:30	4,09E-20	00:00:00.4528466	0,0000436572	00:00:00.5465390
11		0,0000298302	7,38	0,0000845609	00:00:32	3,16E-21	00:00:00.4684620	0,0001185745	00:00:00.5621544
12		0,0000141769	5,16	0,0000257733	00:00:29	1,08E-22	00:00:00.5621544	0,0001277264	00:00:00.5777698
13		0,0000269539	5,77	0,0017911422	00:00:34	7,48E-24	00:00:00.5309236	0,0001198999	00:00:00.5933852
14		0,0000269248	8,56	0,0002361592	00:00:36	2,60E-25	00:00:00.5153082	0,0000092743	00:00:00.6090006
15		0,0000087802	6,69	0,0004064291	00:00:35	1,64E-26	00:00:00.5309236	0,0001688226	00:00:00.6090006
16		0,0000119088	6,67	0,0000958183	00:00:41	8,15E-28	00:00:00.5309236	0,0000785177	00:00:00.6246160
17		0,0000102639	10,30	0,0001343199	00:00:46	3,61E-29	00:00:00.5621544	0,0000682696	00:00:00.6402314
18		0,0000048286	7,34	0,0002552366	00:00:53	1,39E-30	00:00:00.7963854	0,0000434270	00:00:00.6714622
19		0,0000036421	11,52	0,0033660696	00:00:44	6,32E-32	00:00:00.5777698	0,0000937849	00:00:00.6714622
20		0,0000064856	8,05	0,0003837633	00:00:46	6,16E-33	00:00:00.6090006	0,0000512476	00:00:00.7183084

Duomenys esant konstantai 0.2 ir epochų skaičiui 400.

Neuronų sk.	Const	Matlab	laikas	Web1	laikas	Web2	laikas	Web3	laikas
10	0,3	0,0001071110	6,47	0,0002771391	00:00:25	5,30E-29	00:00:00.4528930	0,0001634742	00:00:00.5465950
11		0,0000450043	5,84	0,0003669699	00:00:30	3,77E-31	00:00:00.4685100	0,0000118721	00:00:00.5465950
12		0,0000595714	5,55	0,0007137130	00:00:32	0,000000000000000	00:00:00.4841270	0,0000455110	00:00:00.5778290
13		0,0000205100	7,91	0,0040973232	00:00:31	0,000000000000000	00:00:00.5309780	0,0000742921	00:00:00.5778290
14		0,0000129798	8,53	0,0051668369	00:00:35	0,000000000000000	00:00:00.5153610	0,0000415699	00:00:00.6090630
15		0,0000161723	6,50	0,0000112972	00:00:34	0,000000000000000	00:00:00.5309780	0,0000056273	00:00:00.6090630
16		0,0000097673	9,36	0,0001999195	00:00:37	0,000000000000000	00:00:00.5136648	0,0000068346	00:00:00.5927924
17		0,0000063529	7,06	0,0007678750	00:00:42	0,000000000000000	00:00:00.5303932	0,0000330024	00:00:00.6239920
18		0,0000086118	7,25	0,0002569466	00:00:44	0,000000000000000	00:00:00.5615928	0,0001267748	00:00:00.8118292
19		0,0000038777	7,77	0,0000737940	00:00:44	0,000000000000000	00:00:00.5776477	0,0000234009	00:00:00.6557082
20		0,0000053593	8,03	0,0014473585	00:00:48	0,000000000000000	00:00:00.5932598	0,0000928184	00:00:00.6713203

Duomenys esant konstantai 0.3 ir epochų skaičiui 400.

Neuronų sk.	Const	Matlab	laikas	Web1	laikas	Web2	laikas	Web3	laikas
10	0,4	0,0000464872	4,70	0,0004159412	00:00:26	0,000000000000000	00:00:00.4372620	0,0000120933	00:00:00.5309610
11		0,0000198385	7,38	0,0011676555	00:00:31	0,000000000000000	00:00:00.4997888	0,0000380912	00:00:00.5466440
12		0,0000280697	5,33	0,0035566063	00:00:34	0,000000000000000	00:00:00.4841704	0,0001923146	00:00:00.5778808
13		0,0000205747	5,78	0,0004284525	00:00:36	0,000000000000000	00:00:00.5154204	0,0002738673	00:00:00.7965588
14		0,0000122289	5,92	0,0000363462	00:00:37	0,000000000000000	00:00:00.5154204	0,0001543451	00:00:00.6091332
15		0,0000184434	6,38	0,0002552369	00:00:41	0,000000000000000	00:00:00.5154204	0,0000039970	00:00:00.6247520
16		0,0000109922	6,48	0,0003966251	00:00:42	0,000000000000000	00:00:00.5466615	0,0001039597	00:00:00.6247560
17		0,0000153240	7,27	0,0001375169	00:00:44	0,000000000000000	00:00:00.5466615	0,0000320888	00:00:00.6559938
18		0,0000040061	8,97	0,0002201188	00:00:45	0,000000000000000	00:00:00.5779030	0,0003413672	00:00:00.6559980
19		0,0000038320	8,17	0,0000561830	00:00:47	0,000000000000000	00:00:00.6107205	0,0004365693	00:00:00.6733585
20		0,0000032723	7,95	0,0003018581	00:00:50	0,000000000000000	00:00:00.5950610	0,0000994203	00:00:00.6890180

Duomenys esant konstantai 0.4 ir epochų skaičiui 400

Neuronų sk.	Const	Matlab	laikas	Web1	laikas	Web2	laikas	Web3	laikas
10	0,5	0,0000589003	6,44	0,0001375492	00:00:27	0,000000000000000	00:00:00.4541255	0,0000695752	00:00:00.5480825
11		0,0000305248	5,13	0,0078785981	00:00:27	0,000000000000000	00:00:00.4640490	0,0000639557	00:00:00.5568588
12		0,0000266767	5,31	0,0007528676	00:00:31	0,000000000000000	00:00:00.4949856	0,0000527460	00:00:00.5723271
13		0,0000134427	5,64	0,0002744292	00:00:33	0,000000000000000	00:00:00.4949856	0,0000743005	00:00:00.5723271
14		0,0000211009	8,33	0,0001186356	00:00:35	0,000000000000000	00:00:00.5104539	0,0001557087	00:00:00.5914358
15		0,0000060613	6,72	0,0002545296	00:00:37	0,000000000000000	00:00:00.5291794	0,0000287710	00:00:00.6069999
16		0,0000070683	6,61	0,0008123067	00:00:40	0,000000000000000	00:00:00.5603076	0,0000509671	00:00:00.6069999
17		0,0000076108	6,94	0,0026583294	00:00:42	0,000000000000000	00:00:00.5603076	0,0004039388	00:00:00.6381281
18		0,0000071249	7,05	0,0000399745	00:00:44	0,000000000000000	00:00:00.5771445	0,0002476668	00:00:00.6395385
19		0,0000036247	11,22	0,0029703289	00:00:46	0,000000000000000	00:00:00.5927430	0,0003090852	00:00:00.6551370
20		0,0000039089	7,95	0,0004043260	00:00:47	0,000000000000000	00:00:00.6083415	0,0002832006	00:00:00.7019325

Duomenys esant konstantai 0.5 ir epochų skaičiui 400

Su NO_2 atlikti tyrimai

- „matlab“ – metodas, kai tinklui paduodami vektoriai iš eilės po vieną. Skaičiuojama klaida ir siunčiamas vektorius atgal, atnaujinami svoriai ir paduodamas kitas, naujas vektorius.
- „web1“ – metodas veikia lygiai taip pat kaip ir „matlab“ matodas, tik VS Web Developer programos terpėje.
- „web2“ – metodas, kai paduodamas vienas vektorius, t.y. visų mokymo duomenų vidurkių vektorius.
- „web3“ – metodas, kai paduodamas vienas vektorius, t.y. visų mokymo duomenų sumos vektorius. Prieš sumuojant, mokymo vektoriai dauginami iš atsitiktinių koeficientų.

Neuronų sk.	Const	Matlab	laikas	Web1	laikas	Web2	laikas	Web3	laikas
10	0,1	0,0000119641	12,56	0,000435570695255	00:01:47	0,000000000015762	00:00:01.3276150	0,000005891152529	00:00:01.7649470
11		0,0000004606	14,70	0,000163881237215	00:02:17	0,000000000002598	00:00:01.3432340	0,000003378004925	00:00:01.7649470
12		0,0000009854	16,52	0,000104101904053	00:02:23	0,000000000000669	00:00:01.3744720	0,000011836956957	00:00:01.8118040
13		0,0000009889	17,30	0,000662679421016	00:02:43	0,000000000000158	00:00:01.4057100	0,000061571819813	00:00:01.8118040
14		0,0000006656	18,50	0,000396152350832	00:02:11	0,000000000000031	00:00:01.4057100	0,000001007565894	00:00:01.8430420
15		0,0000002174	20,97	0,000267237822097	00:03:03	0,000000000000006	00:00:01.4681860	0,000007131411986	00:00:01.8430420
16		0,0000004799	22,00	0,000511382889945	00:02:54	0,000000000000001	00:00:01.4681860	0,000034352491470	00:00:01.8742800
17		0,0000003297	24,38	0,000983219758148	00:02:23	0,000000000000000	00:00:01.8911695	0,000040906639330	00:00:01.9224285
18		0,0000003726	24,47	0,000143854124240	00:02:54	0,000000000000000	00:00:01.5004320	0,000086979317646	00:00:01.9067990
19		0,0000001490	27,27	0,000146873174259	00:03:12	0,000000000000000	00:00:01.5629600	0,000156301737324	00:00:02.0005888
20		0,0000020869	41,42	0,000057179769856	00:03:29	0,000000000000000	00:00:01.6723672	0,000005063388463	00:00:02.3913288

Duomenys esant konstantai 0.1 ir epochų skaičiui 400

Neuronų sk.	Const	Matlab	laikas	Web1	laikas	Web2	laikas	Web3	laikas
10	0,2	0,0000061384	18,22	0,000100560789571	00:02:08	6,36E-21	00:00:01.4379232	0,000005979066794	00:00:01.8286632
11		0,0000009369	15,11	0,002352519860059	00:02:45	1,91E-22	00:00:01.7348856	0,000029808662902	00:00:01.9224408
12		0,0000010971	22,28	0,000043445730557	00:02:37	9,01E-24	00:00:01.7192560	0,000051361278677	00:00:01.8755520
13		0,0000002171	26,45	0,001247719966643	00:02:24	3,99E-25	00:00:01.4379232	0,000016921000068	00:00:01.8755520
14		0,0000011175	18,20	0,000377662192955	00:02:34	1,31E-26	00:00:01.4644260	0,000035478872158	00:00:01.9162170
15		0,0000008468	30,06	0,000004348970470	00:03:30	3,99E-28	00:00:02.2000794	0,000100584743572	00:00:01.9972352
16		0,0000006498	31,27	0,000323726063234	00:03:28	1,70E-29	00:00:01.5143737	0,000013417459471	00:00:01.9827367
17		0,0000004426	24,92	0,000460984868299	00:03:26	4,51E-31	00:00:01.7957480	0,000097822803314	00:00:02.2798192
18		0,0000002009	28,44	0,000310849740278	00:03:44	1,39E-32	00:00:01.5460236	0,000094155810190	00:00:02.2799944
19		0,0000004113	40,70	0,000101088172669	00:04:53	0,000000000000000	00:00:01.5773069	0,000199372177872	00:00:01.9833463
20		0,0000047332	41,86	0,001984390349164	00:03:09	0,000000000000000	00:00:01.5156250	0,000001365160461	00:00:01.9687500

Duomenys esant konstantai 0.2 ir epochų skaičiui 400

Neuronų sk.	Const	Matlab	laikas	Web1	laikas	Web2	laikas	Web3	laikas
10	0,3	0,0000017515	17,92	0,000242748447598	00:01:45	1,30E-30	00:00:01.3750000	0,000041907091281	00:00:01.8593750
11		0,0000031432	21,38	0,002623845532458	00:01:50	6,16E-33	00:00:01.3906250	0,000029283030550	00:00:01.7656250
12		0,0000021843	15,61	0,000037443885572	00:01:51	0,000000000000000	00:00:01.3899753	0,000003085190956	00:00:01.9053594
13		0,0000005403	27,55	0,001311120075535	00:02:01	0,000000000000000	00:00:01.4212744	0,000019348956289	00:00:01.8429712
14		0,0000012425	19,22	0,000042715423571	00:02:15	0,000000000000000	00:00:01.3744456	0,000019271577798	00:00:01.7961505
15		0,0000002157	21,97	0,000454213846954	00:02:17	0,000000000000000	00:00:01.4064390	0,000008715699493	00:00:01.8127436
16		0,0000002183	33,69	0,000038056285908	00:02:26	0,000000000000000	00:00:01.4217658	0,000006087633810	00:00:01.8279846
17		0,0000015516	26,17	0,000196919267019	00:02:32	0,000000000000000	00:00:01.4537388	0,000166811448168	00:00:01.8445288
18		0,0000002588	29,64	0,000096821315905	00:02:46	0,000000000000000	00:00:01.4689474	0,000034692216794	00:00:01.8596249
19		0,0000002311	42,91	0,001612618894738	00:02:41	0,000000000000000	00:00:01.5000480	0,000013939981980	00:00:01.8906855
20		0,0000002930	44,14	0,002062144613088	00:02:52	0,000000000000000	00:00:01.4999904	0,000002543742093	00:00:01.9062378

Duomenys esant konstantai 0.3 ir epochų skaičiui 400

Neuronų sk.	Const	Matlab	laikas	Web1	laikas	Web2	laikas	Web3	laikas
10	0,4	0,0000033924	19,27	0,000160036643803	00:01:37	0,000000000000000	00:00:01.3749824	0,000025686929896	00:00:01.7656024
11		0,0000038397	15,34	0,001565328377148	00:01:47	0,000000000000000	00:00:01.3749736	0,000071888614856	00:00:01.8124652
12		0,0000015037	24,03	0,000254816117465	00:01:55	0,000000000000000	00:00:01.4218477	0,000033644428023	00:00:01.8280899
13		0,0000139323	27,08	0,000166463559148	00:02:14	0,000000000000000	00:00:01.4530878	0,000227732618845	00:00:01.8593274
14		0,0000009318	27,48	0,000115646636495	00:02:20	0,000000000000000	00:00:01.4999616	0,000026507642492	00:00:01.8905766
15		0,0000038533	32,16	0,003521163463894	00:02:28	0,000000000000000	00:00:01.4843370	0,000029567861416	00:00:01.9062012
16		0,0000005469	33,27	0,000248163520672	00:02:36	0,000000000000000	00:00:01.4843370	0,000121027494426	00:00:02.0311980
17		0,0000001268	26,44	0,000148825791476	00:02:38	0,000000000000000	00:00:01.6405830	0,000046908633175	00:00:01.9218258
18		0,0000010492	37,14	0,000755379597726	00:02:48	0,000000000000000	00:00:01.4999616	0,000002471051123	00:00:02.1093210
19		0,0000008588	42,05	0,000024362146338	00:02:55	0,000000000000000	00:00:01.4999616	0,000040281281981	00:00:01.9062012
20		0,0000003203	36,72	0,000401447350485	00:03:03	0,000000000000000	00:00:01.5155862	0,000011542985538	00:00:01.9374504

Duomenys esant konstantai 0.4 ir epochų skaičiui 400

Neuronų sk.	Const	Matlab	laikas	Web1	laikas	Web2	laikas	Web3	laikas
10	0,5	0,0000041946	18,45	0,000307033058558	00:01:37	0,000000000000000	00:00:01.3593402	0,000040720548779	00:00:01.7343306
11		0,0000008561	21,97	0,000023576705134	00:01:44	0,000000000000000	00:00:01.9843242	0,000152307491207	00:00:02.1405702
12		0,0000003052	23,77	0,001073104174970	00:01:59	0,000000000000000	00:00:01.4062140	0,000019380919117	00:00:01.8749520
13		0,0000024563	17,83	0,000631587099348	00:01:55	0,000000000000000	00:00:01.3905894	0,000125031715037	00:00:01.8437028
14		0,0000004352	27,50	0,000902243232485	00:02:00	0,000000000000000	00:00:01.4374632	0,000091423557694	00:00:01.8437028
15		0,0000008762	32,09	0,000046824719126	00:02:18	0,000000000000000	00:00:01.4530878	0,000030072374029	00:00:01.8437028
16		0,0000005867	31,70	0,000641415966280	00:02:22	0,000000000000000	00:00:01.4573379	0,000150526707023	00:00:01.8647657
17		0,0000026594	25,03	0,000101419897750	00:02:37	0,000000000000000	00:00:01.4723596	0,000016856576284	00:00:01.8796080
18		0,0000005349	35,81	0,000346971957273	00:02:32	0,000000000000000	00:00:01.4721246	0,000048772996902	00:00:01.8793080
19		0,0000003053	42,92	0,000071632653356	00:03:14	0,000000000000000	00:00:01.5189909	0,000021117205878	00:00:01.9418028
20		0,0000002199	42,73	0,000062463943492	00:03:01	0,000000000000000	00:00:01.5503004	0,000028604549460	00:00:01.9731096

Duomenys esant konstantai 0.5 ir epochų skaičiui 400

Priedas 4. Gauti rezultatai. 2 palyginimas

IRISŲ duomenys. Tyrimai su fiksuotais, apmokyto tinklo, svoriais

Tinklo apmokymo metodai su fiksuotais, apmokyto tinklo, svoriais:

- „web1“ – metodas, kai tinklui paduodami vektoriai iš eilės po vieną. Skaičiuojama klaida ir siunčiamas vektorius atgal, atnaujinami svoriai ir paduodamas kitas, naujas vektorius.
- „web2“ – metodas, kai paduodami keturi vektoriai. Mokymo duomenys dalinami į keturias dalis ir iš kiekvienos dalies skaičiuojamas vidurkis.
- „web3“ – metodas, kai paduodami keturi vektoriai. Mokymo duomenys dalinami į keturias dalis ir iš kiekvienos dalies skaičiuojamas sumos vektorius. Prieš sumuojant, mokymo vektoriai dauginami iš atsitiktinių koeficientų.

Neuronų sk.	Const	Web1		laikas	Web2		laikas	Web3		laikas
		Mokymo	Testavimo		Mokymo	Testavimo		Mokymo	Testavimo	
10	0,1	0,014929716	0,078451710	00:00:23	0,283667549	0,190091044	00:00:00.1523545	0,282606299	0,188424831	00:00:00.6854223
11		0,015238080	0,077534714	00:00:25	0,283769318	0,186350158	00:00:01.7565242	0,285562021	0,191943363	00:00:00.9679440
12		0,015205931	0,078157066	00:00:26	0,286705973	0,190857605	00:00:01.1224720	0,284970923	0,193175235	00:00:01.0147800
13		0,015001648	0,077673573	00:00:28	0,284940249	0,186972522	00:00:01.1709000	0,287140601	0,195366208	00:00:01.0928400
14		0,015090143	0,078791379	00:00:31	0,285803205	0,189544164	00:00:01.2177360	0,285042953	0,190351130	00:00:01.1396760
15		0,015355670	0,077590261	00:00:31	0,282763978	0,189652076	00:00:01.2645720	0,287094579	0,202873288	00:00:01.1865120
16		0,014891235	0,078410142	00:00:33	0,286250468	0,190561417	00:00:01.3114080	0,285575357	0,192040845	00:00:01.2177360
17		0,015020536	0,078053821	00:00:36	0,284692485	0,187409434	00:00:01.3582440	0,286476478	0,192537236	00:00:01.2645720
18		0,014759242	0,079410221	00:00:38	0,285645267	0,185918316	00:00:01.4206920	0,282819173	0,188126264	00:00:01.3114080
19		0,015260039	0,078873310	00:00:39	0,281943707	0,186944994	00:00:01.5143640	0,285742138	0,185469620	00:00:01.3738560
20		0,015153091	0,078372341	00:00:42	0,285635010	0,190544912	00:00:01.5299760	0,285405327	0,188068178	00:00:01.4206920

Duomenys esant konstantai 0.1 ir epochų skaičiui 400.

Neuronų sk.	Const	Web1		laikas	Web2		laikas	Web3		laikas
		Mokymo	Testavimo		Mokymo	Testavimo		Mokymo	Testavimo	
10	0,2	0,016315992	0,095396156	00:00:23	0,285934991	0,189431357	00:00:00.9991680	0,284326480	0,185484533	00:00:00.9211080
11		0,016291988	0,095085391	00:00:24	0,286873037	0,192739412	00:00:01.0616160	0,286640792	0,194026554	00:00:00.9367200
12		0,016204961	0,095434119	00:00:25	0,285853549	0,196060062	00:00:01.1282255	0,286456003	0,191805955	00:00:01.0328825
13		0,016347147	0,078562141	00:00:30	0,284871326	0,190240904	00:00:01.1790600	0,286146281	0,189708377	00:00:01.0805262
14		0,015832398	0,095512864	00:00:31	0,285386425	0,186604847	00:00:01.2214644	0,287417268	0,192629659	00:00:01.1275056
15		0,016329174	0,078540293	00:00:32	0,284747191	0,190062205	00:00:01.2684438	0,283336802	0,186921212	00:00:01.1901448
16		0,016222272	0,095421069	00:00:34	0,286234045	0,193655073	00:00:01.3154232	0,286762962	0,192049254	00:00:01.2214644
17		0,015995926	0,079507319	00:00:34	0,286571947	0,193409703	00:00:01.3761264	0,285635596	0,197316151	00:00:01.2822996
18		0,016015115	0,094841043	00:00:36	0,285451622	0,193474786	00:00:01.4543154	0,284395462	0,187437524	00:00:01.3292130
19		0,015741041	0,095659819	00:00:39	0,286205121	0,194096145	00:00:01.4855910	0,286715912	0,201759156	00:00:01.3917642
20		0,015742375	0,095421009	00:00:42	0,285660528	0,192727177	00:00:01.5317204	0,278793011	0,186593261	00:00:01.5473502

Duomenys esant konstantai 0.2 ir epochų skaičiui 400.

Neuronų sk.	Const	Web1		laikas	Web2		laikas	Web3		laikas
		Mokymo	Testavimo		Mokymo	Testavimo		Mokymo	Testavimo	
10	0,3	0,016309220	0,094241430	00:00:23	0,284809616	0,189376241	00:00:01.0003072	0,287113830	0,205646670	00:00:00.9065284
11		0,016326461	0,079116567	00:00:25	0,286676974	0,190852153	00:00:01.0626360	0,281686559	0,189355171	00:00:00.9532470
12		0,016293233	0,094579626	00:00:26	0,285062134	0,189737303	00:00:01.3126680	0,286321426	0,200535141	00:00:01.0157550
13		0,016355541	0,077140889	00:00:28	0,285752768	0,195434377	00:00:01.1719500	0,286155255	0,190217060	00:00:01.3750880
14		0,016727959	0,093428553	00:00:29	0,287675322	0,201756031	00:00:01.2323921	0,282662142	0,192444180	00:00:01.1075929
15		0,016287140	0,094561605	00:00:31	0,286557454	0,190044183	00:00:01.2635919	0,285996831	0,203589826	00:00:01.1855924
16		0,016283924	0,078539195	00:00:35	0,283742634	0,190219952	00:00:01.3259915	0,286032701	0,188769856	00:00:01.2323921
17		0,015868486	0,078720521	00:00:36	0,286177055	0,190508352	00:00:01.3884725	0,285695138	0,190882840	00:00:01.2638916
18		0,016426757	0,016426757	00:00:37	0,285828322	0,192535730	00:00:01.4511348	0,284269782	0,191397572	00:00:01.326306
19		0,016193164	0,094893045	00:00:36	0,281473752	0,186517218	00:00:01.6383780	0,281468550	0,185374614	00:00:01.3731168
20		0,016226858	0,078746550	00:00:46	0,286563318	0,201203742	00:00:01.8725880	0,285103965	0,189515607	00:00:01.6229096

Duomenys esant konstantai 0.3 ir epochų skaičiui 400.

Neuronų sk.	Const	Web1		laikas	Web2		laikas	Web3		laikas
		Mokymo	Testavimo		Mokymo	Testavimo		Mokymo	Testavimo	
10	0,4	0,016360568	0,078815575	00:00:26	0,284353872	0,189729795	00:00:01.0584200	0,286177872	0,192767805	00:00:00.9650300
11		0,016296334	0,079393018	00:00:27	0,285348200	0,191342388	00:00:01.1206800	0,280192428	0,187254033	00:00:00.9961600
12		0,016250769	0,078279351	00:00:28	0,286383545	0,189332133	00:00:01.1518100	0,286142904	0,190959705	00:00:01.0428550
13		0,016412156	0,078636918	00:00:30	0,285381838	0,192122988	00:00:01.2296350	0,284888474	0,187588435	00:00:01.1219688
14		0,016218631	0,078462332	00:00:32	0,285117298	0,188715044	00:00:01.2622149	0,284839219	0,190255261	00:00:01.1843004
15		0,016695722	0,078052122	00:00:32	0,286830643	0,194279451	00:00:01.3089636	0,285852725	0,190311578	00:00:01.2154662
16		0,016364496	0,078849193	00:00:34	0,282279809	0,189204236	00:00:01.3868781	0,286655745	0,208299500	00:00:01.2777978
17		0,015950679	0,079259433	00:00:39	0,285771245	0,190706561	00:00:01.4186354	0,282029782	0,201451029	00:00:01.3250990
18		0,015959553	0,094730790	00:00:40	0,285687269	0,187636794	00:00:01.4965824	0,286260376	0,192468000	00:00:01.3874566
19		0,015959172	0,095928014	00:00:42	0,282675237	0,185965844	00:00:01.5589400	0,284105447	0,189922571	00:00:01.4498142
20		0,015762376	0,094689815	00:00:43	0,285244279	0,185667386	00:00:01.5903534	0,284235513	0,189577630	00:00:01.5123949

Duomenys esant konstantai 0.4 ir epochų skaičiui 400.

Neuronų sk.	Const	Web1		laikas	Web2		laikas	Web3		laikas
		Mokymo	Testavimo		Mokymo	Testavimo		Mokymo	Testavimo	
10	0,5	0,016662793	0,077914667	00:00:26	0,28489022	0,188922084	00:00:01.0290522	0,286564157	0,195272807	00:00:00.9199103
11		0,016290059	0,079038699	00:00:34	0,28618239	0,194716967	00:00:01.2635799	0,281663276	0,185009372	00:00:01.1567985
12		0,016073552	0,079189886	00:00:34	0,28522785	0,18650673	00:00:01.3525644	0,283823211	0,188368045	00:00:01.2101892
13		0,016639011	0,078173735	00:00:35	0,28606579	0,194371657	00:00:01.3703613	0,286978019	0,193457504	00:00:01.2813768
14		0,016400521	0,077514289	00:00:37	0,2856447	0,191613502	00:00:01.3638394	0,285404475	0,189758972	00:00:01.3474076
15		0,01638493	0,078242852	00:00:37	0,28582173	0,195161633	00:00:01.4295666	0,281229338	0,189839914	00:00:01.3638394
16		0,016417306	0,078106136	00:00:39	0,28561731	0,193398757	00:00:01.4459984	0,28141692	0,184715685	00:00:01.3474076
17		0,015825187	0,094533176	00:00:39	0,28471243	0,190125882	00:00:01.5281574	0,284554654	0,187729822	00:00:01.3389936
18		0,016132287	0,07902623	00:00:40	0,28202037	0,185534556	00:00:01.5143380	0,285330952	0,19307604	00:00:01.4346360
19		0,016182039	0,094976414	00:00:42	0,28623932	0,194590558	00:00:01.5780996	0,286413815	0,198653377	00:00:01.4983976
20		0,016101781	0,077627397	00:00:45	0,28709429	0,191233886	00:00:01.6259208	0,285922259	0,188663802	00:00:01.5462188

Duomenys esant konstantai 0.5 ir epochų skaičiui 400.

NO_2 duomenys. Tyrimai su fiksuotais, apmokyto tinklo, svoriais

Tinklo apmokymo metodai su fiksuotais, apmokyto tinklo, svoriais:

- „web1“ – metodas, kai tinklui paduodami vektoriai iš eilės po vieną. Skaičiuojama klaida ir siunčiamas vektorius atgal, atnaujinami svoriai ir paduodamas kitas, naujas vektorius.
- „web2“ – metodas, kai paduodami keturi vektoriai. Mokymo duomenys dalinami į keturias dalis ir iš kiekvienos dalies skaičiuojamas vidurkis.
- „web3“ – metodas, kai paduodami keturi vektoriai. Mokymo duomenys dalinami į keturias dalis ir iš kiekvienos dalies skaičiuojamas sumos vektorius. Prieš sumuojant, mokymo vektoriai dauginami iš atsitiktinių koeficientų.

Neuronų sk.	Const	Web1		laikas	Web2		laikas	Web3		laikas
		Mokymo	Testavimo		Mokymo	Testavimo		Mokymo	Testavimo	
10	0,1	0,004361084	0,011102989	00:01:34	0,285172880	0,287263149	00:00:02.2783884	0,280691866	0,287376687	00:00:01.6229616
11		0,004133826	0,012021747	00:01:36	0,284325641	0,287929009	00:00:02.3408100	0,282411949	0,287610058	00:00:01.7946210
12		0,004179152	0,011641962	00:01:43	0,282593196	0,287214037	00:00:02.6373464	0,278955077	0,286611655	00:00:01.8882776
13		0,004283386	0,011296176	00:01:58	0,282696981	0,288761143	00:00:02.5437128	0,278008087	0,285511456	00:00:01.9507000
14		0,004265626	0,011398916	00:02:24	0,280488929	0,286763174	00:00:02.4032778	0,280132434	0,286914647	00:00:01.9975296
15		0,004180499	0,011072816	00:02:08	0,279406446	0,285549730	00:00:02.4657006	0,275539790	0,284436279	00:00:01.9038954
16		0,004202641	0,011087974	00:02:13	0,284905300	0,287847821	00:00:02.4983520	0,278769254	0,286282451	00:00:01.9674522
17		0,004261605	0,011168393	00:02:24	0,284059766	0,287489780	00:00:02.6059014	0,275324876	0,285538008	00:00:02.1065670
18		0,004204516	0,011137535	00:02:37	0,279620955	0,285772678	00:00:02.7035748	0,281281866	0,287313796	00:00:02.2034916
19		0,004082739	0,010890801	00:02:40	0,282556557	0,287146399	00:00:02.7660852	0,278319165	0,286021521	00:00:02.3910228
20		0,004080101	0,011252164	00:02:48	0,278563602	0,286111008	00:00:02.8129680	0,279247779	0,286878544	00:00:02.2816296

Duomenys esant konstantai 0.1 ir epochų skaičiui 400.

Neuronų sk.	Const	Web1		laikas	Web2		laikas	Web3		laikas
		Mokymo	Testavimo		Mokymo	Testavimo		Mokymo	Testavimo	
10	0,2	0,004413860	0,011153584	00:01:35	0,284997110	0,287853493	00:00:02.1878640	0,282677735	0,287682374	00:00:01.7346636
11		0,004182585	0,011849336	00:01:37	0,283595955	0,287341515	00:00:02.2437936	0,280438282	0,287492823	00:00:01.7295909
12		0,004249084	0,011209427	00:01:48	0,283324314	0,287689168	00:00:02.3094956	0,283368068	0,288506599	00:00:02.0286110
13		0,004303922	0,011286603	00:02:00	0,280377405	0,286577161	00:00:02.4814971	0,279508253	0,285300254	00:00:01.8104004
14		0,004031896	0,011418718	00:02:31	0,281168221	0,287129118	00:00:02.7001148	0,279700311	0,286090093	00:00:02.0758108
15		0,004187446	0,011433918	00:02:16	0,282458577	0,287367826	00:00:02.5284960	0,279190611	0,286061735	00:00:01.9510000
16		0,004179018	0,011168166	00:02:20	0,283092632	0,287384366	00:00:02.5597120	0,275246238	0,284832944	00:00:02.0446480
17		0,004120019	0,011317135	00:02:34	0,283947744	0,287739795	00:00:02.7157920	0,280963051	0,286575825	00:00:02.1070800
18		0,004112326	0,011402975	00:02:30	0,283073099	0,286869682	00:00:02.9187147	0,280374757	0,286409644	00:00:02.1383097
19		0,004162743	0,011369069	00:02:40	0,282863003	0,286907659	00:00:02.7626337	0,276063101	0,286070156	00:00:02.4504717
20		0,004066013	0,011346749	00:02:45	0,282023951	0,285762692	00:00:02.8406742	0,278907888	0,285925245	00:00:02.3256069

Duomenys esant konstantai 0.2 ir epochų skaičiui 400.

Neuronų sk.	Const	Web1		laikas	Web2		laikas	Web3		laikas
		Mokymo	Testavimo		Mokymo	Testavimo		Mokymo	Testavimo	
10	0,3	0,004216433	0,010890316	00:01:35	0,282487217	0,286741984	00:00:02.2163502	0,280212260	0,287167291	00:00:01.6388505
11		0,004124670	0,011579002	00:01:43	0,280336841	0,286253657	00:00:02.2475664	0,281162435	0,287251270	00:00:01.7324991
12		0,004163711	0,011768910	00:01:53	0,280018499	0,285586818	00:00:02.3412150	0,281951345	0,286988100	00:00:01.8105396
13		0,004133676	0,011101073	00:01:51	0,280745205	0,286374562	00:00:02.3568231	0,276527385	0,285296525	00:00:01.8261477
14		0,004040847	0,011144437	00:02:06	0,283366201	0,287700166	00:00:02.4504717	0,277379208	0,286456435	00:00:01.8729720
15		0,004264793	0,011129144	00:02:13	0,282208603	0,288024326	00:00:02.4972960	0,280832491	0,287591158	00:00:01.9666206
16		0,004068683	0,011295553	00:02:25	0,283048426	0,287515455	00:00:02.5597284	0,280320237	0,285909616	00:00:01.9978368
17		0,004126548	0,011982163	00:02:19	0,278728077	0,285682942	00:00:02.6221608	0,275172322	0,284694790	00:00:02.0914854
18		0,004065435	0,011267146	00:02:36	0,280392763	0,285348853	00:00:02.7002013	0,281679536	0,286625102	00:00:02.1539178
19		0,004035834	0,011195192	00:02:43	0,282168772	0,286907473	00:00:02.7938499	0,278262738	0,286194453	00:00:02.2007421
20		0,004012430	0,011421269	00:02:51	0,283330867	0,287616108	00:00:02.8094580	0,276072158	0,285529559	00:00:02.2787826

Duomenys esant konstantai 0.3 ir epochų skaičiui 400.

Neuronų sk.	Const	Web1		laikas	Web2		laikas	Web3		laikas
		Mokymo	Testavimo		Mokymo	Testavimo		Mokymo	Testavimo	
10	0,4	0,004186845	0,011811753	00:01:33	0,283355084	0,287194339	00:00:02.1851340	0,283480031	0,287910470	00:00:01.6232424
11		0,003990433	0,011757236	00:01:41	0,282284664	0,286829041	00:00:02.2475664	0,278888257	0,286271840	00:00:01.7168910
12		0,004079044	0,011574419	00:01:49	0,283614695	0,288008011	00:00:02.2943907	0,279149738	0,286780266	00:00:01.7949315
13		0,004153666	0,011287329	00:02:02	0,282600374	0,285270226	00:00:02.3880393	0,277812437	0,285508206	00:00:01.9510125
14		0,004237959	0,010898450	00:02:06	0,281594546	0,287256613	00:00:02.4504717	0,278104804	0,285378148	00:00:01.9041882
15		0,004005452	0,011190773	00:02:21	0,279432544	0,285964481	00:00:02.5441203	0,277673820	0,285655492	00:00:02.0290530
16		0,003945365	0,011574955	00:02:15	0,280770877	0,286231066	00:00:02.6533770	0,277189761	0,285238412	00:00:02.0602692
17		0,004006314	0,011149583	00:02:26	0,279705297	0,286099213	00:00:02.6845932	0,276148689	0,285252953	00:00:02.1383097
18		0,004034506	0,011303216	00:02:40	0,277820902	0,284754975	00:00:02.7782418	0,277336349	0,284878852	00:00:02.1851340
19		0,003963629	0,011353955	00:02:47	0,279981815	0,285638384	00:00:02.7938499	0,276857055	0,286502951	00:00:02.2475664
20		0,003916637	0,011497562	00:02:44	0,277565774	0,284982178	00:00:02.9030754	0,275375274	0,284292975	00:00:02.3954338

Duomenys esant konstantai 0.4 ir epochų skaičiui 400.

Neuronų sk.	Const	Web1		laikas	Web2		laikas	Web3		laikas
		Mokymo	Testavimo		Mokymo	Testavimo		Mokymo	Testavimo	
10	0,5	0,003496246	0,011789128	00:01:32	0,283751762	0,287415097	00:00:02.2615488	0,284186276	0,288084495	00:00:01.7275720
11		0,003585953	0,012684819	00:02:07	0,283218116	0,286614910	00:00:02.3002854	0,276603388	0,285625764	00:00:01.7995430
12		0,003808543	0,011880499	00:01:57	0,281026862	0,286942038	00:00:02.3441250	0,279606799	0,286907268	00:00:01.8127900
13		0,003801811	0,011208636	00:02:01	0,281891255	0,286713297	00:00:02.4535175	0,278400570	0,286639969	00:00:01.8596725
14		0,004044214	0,011164559	00:02:10	0,283474035	0,287404148	00:00:02.4836118	0,282604591	0,287766679	00:00:01.9212846
15		0,003805245	0,011603697	00:02:29	0,279962949	0,285679452	00:00:02.5768875	0,279341119	0,286384092	00:00:02.0927450
16		0,003829242	0,011599212	00:02:29	0,282042904	0,286390316	00:00:02.7016545	0,277663300	0,286189944	00:00:02.1706935
17		0,003726556	0,012005448	00:02:28	0,282580280	0,287582851	00:00:02.9827515	0,278317606	0,286140402	00:00:02.1394605
18		0,003831880	0,011391421	00:02:30	0,279559546	0,285606581	00:00:02.6386096	0,278085743	0,287024607	00:00:02.1888466
19		0,003907003	0,011851460	00:02:43	0,279182776	0,285698164	00:00:02.7985594	0,275584783	0,284502437	00:00:02.2911283
20		0,003906707	0,012098508	00:02:48	0,285545094	0,288153545	00:00:02.8547784	0,278798102	0,286106746	00:00:02.3117499

Duomenys esant konstantai 0.5 ir epochų skaičiui 400.

Priedas 5. Sistemos Matlab 7.1 programos kodas

MOKYMO DUOMENŲ NUSKAITYMAS

```
X = load('iris.txt');
```

MOKYMO DUOMENŲ NORMAVIMAS

```
for j=1:4
   min(j)=X(1,j);
   max(j)=X(1,j);
end
for j=1:4
   for ij=1:150
      if min(j) > X(ij,j)
         min(j) = X(ij,j);
      end
   end
end
for j=1:4
   for ij=1:150
      if max(j) < X(ij,j)
         max(j) = X(ij,j);
      end
   end
end
for j=1:4
   for ij=1:150
      X1(ij,j)=(X(ij,j)-min(j))/(max(j)-min(j));
   end
end
p = X1;
p1 = p';
```

TROKŠTAMŲ REIKŠMIŲ NUSKAITYMAS

```
t = load('convert.txt');
```

TROKŠTAMŲ REIKŠMIŲ NORMAVIMAS

```
for j=1:2
   mint(j)=t(1,j);
   maxt(j)=t(1,j);
end
for j=1:2
   for ij=1:150
      if mint(j) > t(ij,j)
         mint(j) = t(ij,j);
      end
   end
end
for j=1:2
   for ij=1:150
      if maxt(j) < t(ij,j)
         maxt(j) = t(ij,j);
      end
   end
end
for j=1:2
   for ij=1:150
```

```
        tt(ij,j)=(t(ij,j)-mint(j))/(maxt(j)-mint(j));
    end
end
t1 = tt';
```

TINKLO SU VIENU PASLĖPTU SLUOKSNIU SUKŪRIMAS

```
net=newff(minmax(p1),[12,2],{'logsig','logsig'});
```

TINKLO PARAMETRŲ NUSTATYMAS

```
net.trainParam.show = 50;
net.trainParam.lr = 0.6;
net.trainParam.epochs = 300;
```

PRADŽIOS LAIKO FIKSAVIMAS

```
tll = cputime;
```

TINKLO MOKYMAS

```
[net,tr]=train(net,p1,t1);
```

TINKLO APSIMOKYMO TIKRINIMAS

```
a = sim(net,p1);
a1 = a';
```

TINKLO APSIMOKYMO TRUKMĖ

```
laik = cputime-tll
pause
```

DUOMENŲ ATNORMAVIMAS

```
for i=1:150
    a2(i,1) = (a1(i,1)-0.1)/(0.9-0.1)*(maxt(1)-mint(1))+mint(1);
    a2(i,2) = (a1(i,2)-0.1)/(0.9-0.1)*(maxt(2)-mint(2))+mint(2);
end
```

DUOMENŲ IŠVEDIMAS EKRANE

```
plot(a2(:,1),a2(:,2),'.'); grid;
```

Priedas 6. Visual Studio Web Developer 2008 programos kodas

AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("BackPropogation")]
[assembly: AssemblyDescription("Back Propagation algorythm implementation")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("BackPropogation")]
[assembly: AssemblyCopyright("Copyright ©  2009")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components.  If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is exposed to
COM
[assembly: Guid("3d5900ae-111a-45be-96b3-d9e4606ca793")]

// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Revision and Build
Numbers
// by using the '*' as shown below:
[assembly: AssemblyVersion("1.6.0.0")]
[assembly: AssemblyFileVersion("1.6.0.0")]
```

DesiredGr.aspx

```
<%@ Page Language="C#" %>
<script language="C#" runat="server">

    void Page_Load(Object sender, EventArgs e)
    {
        try
        {
            BackPropogation.Graph c = new BackPropogation.Graph("Trokštami
Rezultatai");
            c.Clear();
            BackPropogation.Matrix values =
BackPropogation.NeuralNetwork.getNNetwork().DesiredResults;
            for (int i = 0; i < values.nRows; ++i)
                if (values.nCols == 2)
```

```
                    c.AddXY(values[i][0], values[i][1]);

            c.Draw(Page, 320, 240);
        }
        catch (Exception exc)
        {
        }
    }

   </script>
```

ResultsGr.aspx

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Configuration" %>
<%@ Import Namespace="System.Collections"%>
<%@ Import Namespace="System.IO"%>

<script language="C#" runat="server">

void Page_Load(Object sender, EventArgs e)
{
    try
    {
        BackPropogation.Graph c = new BackPropogation.Graph("Gauti Rezultatai");
        c.Clear();
        double[][] values =
BackPropogation.NeuralNetwork.getNNetwork().run(BackPropogation.NeuralNetwork.ge
tNNetwork().Data, BackPropogation.NeuralNetwork.getNNetwork().DesiredResults);
        for (int i = 0; i < values.Length; ++i)
            //if (values[i].Length >= 2)
                c.AddXY(values[i][0], values[i][1]);

        c.Draw(Page, 320, 240);
    }
    catch (Exception exc)
    {
       // throw exc;
    }
}

   </script>
```

TestDesiredGr.aspx

```
<%@ Page Language="C#" %>
<script language="C#" runat="server">

    void Page_Load(Object sender, EventArgs e)
    {
        try
        {
            BackPropogation.Graph c = new BackPropogation.Graph("Trokštami
Rezultatai");
            c.Clear();
            BackPropogation.Matrix values =
BackPropogation.NeuralNetwork.getNNetwork().TestDesiredResults;
```

```
            for (int i = 0; i < values.nRows; ++i)
                if (values.nCols == 2)
                    c.AddXY(values[i][0], values[i][1]);

            c.Draw(Page, 320, 240);
        }
        catch (Exception exc)
        {
        }
    }


   </script>
```

TestResultsGr.aspx

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Configuration" %>
<%@ Import Namespace="System.Collections"%>
<%@ Import Namespace="System.IO"%>

<script language="C#" runat="server">

void Page_Load(Object sender, EventArgs e)
{
    try
    {
        BackPropogation.Graph c = new BackPropogation.Graph("Gauti Rezultatai");
        c.Clear();
        double[][] values =
BackPropogation.NeuralNetwork.getNNetwork().run(BackPropogation.NeuralNetwork.ge
tNNetwork().TestData,
BackPropogation.NeuralNetwork.getNNetwork().TestDesiredResults);
        for (int i = 0; i < values.Length; ++i)
            //if (values[i].Length >= 2)
                c.AddXY(values[i][0], values[i][1]);

        c.Draw(Page, 320, 240);
    }
    catch (Exception exc)
    {
       // throw exc;
    }
}

   </script>
```

Data.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Data.aspx.cs"
Inherits="BackPropogation.Data" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
```

```
<head runat="server">
    <title>Duomenys</title>
    <style type="text/css">
        .style1
        {
            width: 200%;
        }
        .style2
        {
            width: 12%;
            margin-left: 40px;
        }
        .style4
        {
            font-family: Calibri;
        }
    </style>
</head>
<body style="font-size: small; font-family: Calibri; background-color: #00FFFF">
    <form id="form1" runat="server">
    <asp:Panel ID="ErrorPanel" runat="server" BackColor="Red" Visible="False">
        <asp:Label ID="ErrorLabel" runat="server"></asp:Label>
    </asp:Panel>
    <div>
        <asp:HyperLink ID="BackLink" runat="server"
NavigateUrl="Default.aspx">Atgal</asp:HyperLink><br />
        <table>
            <tr>
                <td><asp:Button ID="Loaded" runat="server" Text="Pakrauti
Duomenys"
                        onclick="Loaded_Click" CssClass="style4" /></td>
                <td><asp:Button ID="Normalized" runat="server" Text="Atnormouti
Duomenys"
                        onclick="Normalized_Click" CssClass="style4" /></td>
                <td><asp:Button ID="MinMax" runat="server"
                        Text="Minimalios ir maksimalios reikšmės"
onclick="MinMax_Click"
                        CssClass="style4" /></td>
            </tr>
        </table>
        <asp:Panel ID="DataPanel" runat="server">
            <table style="width:100%;">
                <tr>
                    <td class="style2">
                        <asp:Label ID="Label1" runat="server" Text="Pakrauti
duomenys"></asp:Label>
                    </td>
                    <td>
                        <asp:Label ID="Label2" runat="server" Text="Trokštami
rezultatai"></asp:Label>
                    </td>
                </tr>
                <tr>
                    <td class="style2">
                        <asp:Table ID="DataTable" runat="server"
BorderStyle="Solid" BorderWidth="2px">
                        </asp:Table>
                    </td>
                    <td>
                        <asp:Table ID="DesiredResultsTable" runat="server"
BorderStyle="Solid" BorderWidth="2px">
```

```
                    </asp:Table>
                </td>
            </tr>
        </table>
    </asp:Panel>
    <asp:Panel ID="MinMaxPanel" runat="server" Visible="False">
        <table style="width:100%;">
            <tr>
                <td class="style1" style="width: 100%" colspan="2">
                    Minimalios reikšmės</td>
            </tr>
            <tr>
                <td class="style2">
                    <asp:Label ID="Label3" runat="server" Text="Pakrauti
duomenys"></asp:Label>
                </td>
                <td class="style1" style="width: 50%">Trokštami
rezultatai</td>
            </tr>
            <tr>
                <td class="style2">
                    <asp:Table ID="MinDataTable" runat="server"
BorderStyle="Solid" BorderWidth="2px">
                    </asp:Table>
                </td>
                <td class="style1" style="width: 50%">
                    <asp:Table ID="MinDesiredResultsTable" runat="server"
BorderStyle="Solid" BorderWidth="2px">
                    </asp:Table>
                </td>
            </tr>
            <tr>
                <td class="style1" colspan="2">Maksimalios Reikšmės</td>
            </tr>
            <tr>
                <td class="style2">
                    <asp:Label ID="Label4" runat="server" Text="Pakrauti
duomenys"></asp:Label>
                </td>
                <td class="style1" style="width: 50%">Trokštami
rezultatai</td>
            </tr>
            <tr>
                <td class="style2">
                    <asp:Table ID="MaxDataTable" runat="server"
BorderStyle="Solid" BorderWidth="2px">
                    </asp:Table>
                </td>
                <td class="style1" style="width: 50%">
                    <asp:Table ID="MaxDesiredResultsTable" runat="server"
BorderStyle="Solid" BorderWidth="2px">
                    </asp:Table>
                </td>
            </tr>
        </table>
    </asp:Panel>
    <asp:Panel ID="NormalizedPanel" runat="server" Visible="False">
        <table style="width:100%;">
            <tr>
                <td class="style1" colspan="2">Normalizuoti duomenys</td>
            </tr>
```

```
                <tr>
                    <td class="style2">Pakrauti duomenys</td>
                    <td class="style1" style="width: 50%">Trokštami
rezultatai</td>
                </tr>
                <tr>
                    <td class="style2">
                        <asp:Table ID="NormalizedDataTable" runat="server"
BorderStyle="Solid" BorderWidth="2px">
                        </asp:Table>
                    </td>
                    <td class="style1" style="width: 50%">
                        <asp:Table ID="NormalizedDesiredResultsTable"
runat="server" BorderStyle="Solid"
                                    BorderWidth="2px">
                        </asp:Table>
                    </td>
                </tr>
            </table>
        </asp:Panel>
    </div>
    </form>
</body>
</html>
```

Data.aspx.cs

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

namespace BackPropogation
{
    public partial class Data : System.Web.UI.Page
    {
        /// <summary>
        ///     Initialies page before loading
        /// </summary>
        /// <param name="sender">Object sender</param>
        /// <param name="e">Event occured</param>
        protected void Page_Load(object sender, EventArgs e)
        {
            try
            {
                TableRow tr;

                for (int i = 0; i < NeuralNetwork.getNNetwork().Data.nRows; ++i)
                {
                    tr = new TableRow();
                    for (int j = 0; j < NeuralNetwork.getNNetwork().Data.nCols;
++j)
```

```
                    {
                    TableCell td = new TableCell();
                        td.Text =
NeuralNetwork.getNNetwork().Data[i][j].ToString();
                        td.BorderWidth =
System.Web.UI.WebControls.Unit.Pixel(1);
                        tr.Cells.Add(td);
                    }
                    DataTable.Rows.Add(tr);
                }

                for (int i = 0; i <
NeuralNetwork.getNNetwork().DesiredResults.nRows; ++i)
                {
                    tr = new TableRow();
                    for (int j = 0; j <
NeuralNetwork.getNNetwork().DesiredResults.nCols; ++j)
                    {
                        TableCell td = new TableCell();
                        td.Text =
NeuralNetwork.getNNetwork().DesiredResults[i][j].ToString();
                        td.BorderWidth =
System.Web.UI.WebControls.Unit.Pixel(1);
                        tr.Cells.Add(td);
                    }
                    DesiredResultsTable.Rows.Add(tr);
                }


                tr = new TableRow();
                for (int i = 0; i <
NeuralNetwork.getNNetwork().Data.colMin().Count(); ++i)
                {
                    TableCell td = new TableCell();
                    td.Text =
NeuralNetwork.getNNetwork().Data.colMin()[i].ToString();
                    td.BorderWidth = System.Web.UI.WebControls.Unit.Pixel(1);
                    tr.Cells.Add(td);
                }
                MinDataTable.Rows.Add(tr);

                tr = new TableRow();
                for (int i = 0; i <
NeuralNetwork.getNNetwork().DesiredResults.colMin().Count(); ++i)
                {
                    TableCell td = new TableCell();
                    td.Text =
NeuralNetwork.getNNetwork().DesiredResults.colMin()[i].ToString();
                    td.BorderWidth = System.Web.UI.WebControls.Unit.Pixel(1);
                    tr.Cells.Add(td);
                }
                MinDesiredResultsTable.Rows.Add(tr);


                tr = new TableRow();
                for (int i = 0; i <
NeuralNetwork.getNNetwork().Data.colMax().Count(); ++i)
                {
                    TableCell td = new TableCell();
                    td.Text =
NeuralNetwork.getNNetwork().Data.colMax()[i].ToString();
```

```
                    td.BorderWidth = System.Web.UI.WebControls.Unit.Pixel(1);
                    tr.Cells.Add(td);
                }
                MaxDataTable.Rows.Add(tr);

                tr = new TableRow();
                for (int i = 0; i <
NeuralNetwork.getNNetwork().DesiredResults.colMax().Count(); ++i)
                {
                    TableCell td = new TableCell();
                    td.Text =
NeuralNetwork.getNNetwork().DesiredResults.colMax()[i].ToString();
                    td.BorderWidth = System.Web.UI.WebControls.Unit.Pixel(1);
                    tr.Cells.Add(td);
                }
                MaxDesiredResultsTable.Rows.Add(tr);


                for (int i = 0; i <
NeuralNetwork.getNNetwork().Data.Normalized().nRows; ++i)
                {
                    tr = new TableRow();
                    for (int j = 0; j <
NeuralNetwork.getNNetwork().Data.Normalized().nCols; ++j)
                    {
                        TableCell td = new TableCell();
                        td.Text =
NeuralNetwork.getNNetwork().Data.Normalized()[i][j].ToString();
                        td.BorderWidth =
System.Web.UI.WebControls.Unit.Pixel(1);
                        tr.Cells.Add(td);
                    }
                    NormalizedDataTable.Rows.Add(tr);
                }

                for (int i = 0; i <
NeuralNetwork.getNNetwork().DesiredResults.Normalized().nRows; ++i)
                {
                    tr = new TableRow();
                    for (int j = 0; j <
NeuralNetwork.getNNetwork().DesiredResults.Normalized().nCols; ++j)
                    {
                        TableCell td = new TableCell();
                        td.Text =
NeuralNetwork.getNNetwork().DesiredResults.Normalized()[i][j].ToString();
                        td.BorderWidth =
System.Web.UI.WebControls.Unit.Pixel(1);
                        tr.Cells.Add(td);
                    }
                    NormalizedDesiredResultsTable.Rows.Add(tr);
                }
            }
            catch (Exception exc)
            {
                ErrorLabel.Text = exc.Message;
                ErrorPanel.Visible = true;
            }
        }

        /// <summary>
        ///     Loaded Button Clicked
```

```
        /// </summary>
        /// <param name="sender">Object Sender</param>
        /// <param name="e">Event occured</param>
        protected void Loaded_Click(object sender, EventArgs e)
        {
            DataPanel.Visible = true;
            MinMaxPanel.Visible = false;
            NormalizedPanel.Visible = false;
        }

        /// <summary>
        ///     Normalized Button Clicked
        /// </summary>
        /// <param name="sender">Object Sender</param>
        /// <param name="e">Event occured</param>
        protected void Normalized_Click(object sender, EventArgs e)
        {
            DataPanel.Visible = false;
            MinMaxPanel.Visible = false;
            NormalizedPanel.Visible = true;
        }

        /// <summary>
        ///     MinMax Button Clicked
        /// </summary>
        /// <param name="sender">Object Sender</param>
        /// <param name="e">Event occured</param>
        protected void MinMax_Click(object sender, EventArgs e)
        {
            DataPanel.Visible = false;
            MinMaxPanel.Visible = true;
            NormalizedPanel.Visible = false;
        }
    }
}
```

Data.aspx.designer.cs

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:2.0.50727.3053
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace BackPropogation {

    /// <summary>
    ///    Class for Displaying data
    /// </summary>
    public partial class Data {

        /// <summary>
        /// form1 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
```

```
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.HtmlControls.HtmlForm form1;

        /// <summary>
        /// ErrorPanel control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Panel ErrorPanel;

        /// <summary>
        /// ErrorLabel control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Label ErrorLabel;

        /// <summary>
        /// BackLink control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.HyperLink BackLink;

        /// <summary>
        /// Loaded control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Button Loaded;

        /// <summary>
        /// Normalized control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Button Normalized;

        /// <summary>
        /// MinMax control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
```

```
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Button MinMax;

        /// <summary>
        /// DataPanel control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Panel DataPanel;

        /// <summary>
        /// Label1 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Label Label1;

        /// <summary>
        /// Label2 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Label Label2;

        /// <summary>
        /// DataTable control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Table DataTable;

        /// <summary>
        /// DesiredResultsTable control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Table DesiredResultsTable;

        /// <summary>
        /// MinMaxPanel control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
```

```
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Panel MinMaxPanel;

        /// <summary>
        /// Label3 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Label Label3;

        /// <summary>
        /// MinDataTable control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Table MinDataTable;

        /// <summary>
        /// MinDesiredResultsTable control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Table
MinDesiredResultsTable;

        /// <summary>
        /// Label4 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Label Label4;

        /// <summary>
        /// MaxDataTable control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Table MaxDataTable;

        /// <summary>
        /// MaxDesiredResultsTable control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
```

```
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Table
MaxDesiredResultsTable;

        /// <summary>
        /// NormalizedPanel control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Panel NormalizedPanel;

        /// <summary>
        /// NormalizedDataTable control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Table NormalizedDataTable;

        /// <summary>
        /// NormalizedDesiredResultsTable control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Table
NormalizedDesiredResultsTable;
    }
}
```

Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="BackPropogation._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Back Propagation</title>
    <style type="text/css">
        .style1
        {
            width: 205px;
        }
        .style2
        {
            width: 434px;
        }
        .style3
```

```
        {
            width: 363px;
        }
        .style4
        {
            width: 418px;
        }
        .style5
        {
            width: 460px;
            font-family: Calibri;
            font-size: smaller;
        }
        .style6
        {
            font-family: Calibri;
            font-size: smaller;
        }
        .style7
        {
            font-family: Calibri;
        }
    </style>
</head>
<body style="font-size: medium; font-family: Calibri; background-image: url('');
background-color: #00FFFF;"
    background="Refresh.jpg">
    <form id="form1" runat="server">
    <div>

        <asp:Panel ID="ErrorPanel" runat="server" BackColor="Red"
Visible="False">
            <asp:Label ID="ErrorLabel" runat="server"></asp:Label>
        </asp:Panel>

        <table style="width:100%;">
            <tr>
                <td class="style1">
                    <asp:Label ID="LabelData" runat="server" Text="Mokymo
duomenys"
                        CssClass="style6"></asp:Label>
                    :</td>
                <td>
                    <asp:FileUpload ID="UploadData" runat="server" Font-
Names="Calibri" />
                </td>
            </tr>
            <tr>
                <td class="style1">
                    <asp:Label ID="LabelDesiredResults" runat="server"
Text="Trokštamos reikšmės"
                        CssClass="style6"></asp:Label>
                    :</td>
                <td>
                    <asp:FileUpload ID="UploadDesiredResults" runat="server"
Font-Names="Calibri" />
                </td>
            </tr>
        </table>

    </div>
```

```
    <p>
        <asp:Button ID="Upload" runat="server" onclick="Upload_Click"
            Text="Pakrauti Duomenis" CssClass="style6" />
        <asp:HyperLink ID="ShowDataHyperlink" runat="server"
NavigateUrl="~/Data.aspx"
            Visible="False" style="font-family: Calibri; font-size:
small">Parodyti Duomenis</asp:HyperLink>
        <asp:HyperLink ID="ShowWeightsHyperlink" runat="server"
NavigateUrl="~/Weights.aspx"
            Visible="False" style="font-family: Calibri; font-size:
small">Parodyti Svorius</asp:HyperLink>
    </p>

    <asp:Panel ID="AdvancedPanel" runat="server">
        <table style="width: 100%;">
            <tr>
                <td class="style3" style="vertical-align: top">

                    <asp:Table ID="HiddenNeuronsTable" runat="server"
Width="100%"
                        CssClass="style6">
                        <asp:TableRow runat="server">
                            <asp:TableCell runat="server"><asp:Label
ID="HiddenNeuronsLabel" runat="server"
                                Text="Paslėptų neuronų sluoksniai"></asp:Label>
                            </asp:TableCell>
                            <asp:TableCell runat="server"><asp:TextBox
ID="HiddenNeuronsEdit"
                                runat="server"></asp:TextBox>
                            <asp:ImageButton ID="RefreshButton" runat="server"
Height="20px"
                                ImageUrl="Refresh.jpg" Width="21px"
                            onclick="RefreshButton_Click" />
                            </asp:TableCell>
                        </asp:TableRow>
                    </asp:Table>
                    <asp:Table ID="HiddenLayersTable" runat="server"
style="font-size: smaller">
                    </asp:Table>
                </td>
                <td class="style2" style="vertical-align: top">
                    <table>
                        <tr>
                            <td class="style5">
                                Epochų skaičius:</td>
                            <td class="style4">
                                <asp:TextBox ID="EpochEdit" runat="server"
Width="230px"></asp:TextBox>
                            </td>
                        </tr>
                        <tr>
                            <td class="style5">
                                Konstanta:</td>
                            <td class="style4">
                                <asp:TextBox ID="ConstantEdit" runat="server"
Width="230px"></asp:TextBox>
                            </td>
                        </tr>
                        <tr>
                            <td class="style5">
                                Pradinio svorio dydis:</td>
```

```
                            <td class="style4">
                                <asp:TextBox ID="PrimaryWeightEdit"
runat="server" Width="230px"></asp:TextBox>
                            </td>
                        </tr>
                        <tr>
                            <td class="style5">
                                Mokymo duomenų klaida:</td>
                            <td class="style4">
                                <asp:Label ID="ErrLabel" runat="server"
Text="NaN"></asp:Label></td>
                        </tr>
                                                <tr>
                            <td class="style5">
                                Testavimo duomenų klaida:</td>
                            <td class="style4">
                                <asp:Label ID="TestErrLabel" runat="server"
Text="NaN"></asp:Label></td>
                        </tr>
                        <tr>
                            <td class="style5">
                                Laikas:</td>
                            <td class="style4">
                                <asp:Label ID="TimeLabel" runat="server"
Text="NaN"></asp:Label></td>
                        </tr>
                    </table>
                </td>
                <td  style="vertical-align: top">
                    <table>
                        <tr>
                            <td>
                                <asp:Button ID="TeachButton" runat="server"
onclick="TeachButton_Click"
                                    Text="Apmokyti" CssClass="style7"
Width="195px" />
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <asp:Button ID="TeachButtonConcurrent"
runat="server"
                                    Text="Apmokyti [Modifikuota]"
onclick="TeachButtonConcurrent_Click"
                                    CssClass="style7" Width="195px" />
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <asp:Button ID="TeachButtonModified1"
runat="server"
                                    Text="Apmokyti [Modifikuota 2]"
onclick="TeachButtonModified1_Click"
                                    CssClass="style7" Width="195px" />
                            </td>
                        </tr>
                    </table>
                </td>
            </tr>
        </table>
    </asp:Panel>
```

```
    </form>
    <table>
        <tr>
            <td colspan="2">Mokymo duomenys</td>
        </tr>
        <tr>
            <td><img alt="" src="Graphics/DesiredGr.aspx" /></td>
            <td><a href="Results.aspx?data=teach"><img border="0" alt="" 
src="Graphics/ResultsGr.aspx" /></a></td>
        </tr>
        <tr>
            <td colspan="2">Testavimo duomenys</td>
        </tr>
        <tr>
            <td><img alt="" src="Graphics/TestDesiredGr.aspx" /></td>
            <td><a href="Results.aspx"><img border="0" alt="" 
src="Graphics/TestResultsGr.aspx" /></a></td>
        </tr>
    </table>
</body>
</html>
```

Graph.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;
using System.Collections;


namespace BackPropogation
{
    /// <summary>
    ///     Class for painting Diagram
    /// </summary>
    public class Graph
    {
        private Bitmap b;
        private string Title = "Grafikas";
        private ArrayList chartValues = new ArrayList();
        private float Xorigin = 0, Yorigin = 0;
        private float ScaleX, ScaleY;
        private float Xdivs = 2, Ydivs = 2;
        private int Width, Height;
        private Graphics g;


        struct datapoint
```

```
{

    public float x;

    public float y;

    public bool valid;

}

/// <summary>
///     Constructs new Diagram
/// </summary>
/// <param name="title">Title of the Diagram</param>
public Graph(string title)
{

    Title = title;

}

/// <summary>
///     Clears the Diagaram
/// </summary>
public void Clear()
{
    chartValues.Clear();
}

/// <summary>
///     Adds Point to Diagram
/// </summary>
/// <param name="x">X value of the Point</param>
/// <param name="y">Y value of the Point</param>
public void AddXY(double x, double y)
{

    datapoint myPoint;

    myPoint.x = (float)x;

    myPoint.y = (float)y;

    myPoint.valid = true;

    chartValues.Add(myPoint);

}

/// <summary>
///    Calculates Diagram scale, based on the measures of the drawing
/// </summary>
private void CalculateMeasures()
{
    try
    {
        float minX, minY, maxX, maxY;

        minX = maxX = ((datapoint)chartValues[0]).x;
        minY = maxY = ((datapoint)chartValues[0]).y;
```

```
            foreach (datapoint point in chartValues)
            {
                minX = (point.x < minX ? point.x : minX);
                maxX = (point.x > maxX ? point.x : maxX);
                minY = (point.y < minY ? point.y : minY);
                maxY = (point.y > maxY ? point.y : maxY);
            }
            int x = (int)Math.Floor(minX);
            Xorigin  = (x / 5) * 5 - 5;
            x = ((int)Math.Ceiling(maxX) / 5) *5 + 5;
            ScaleX = x - Xorigin;
            Xdivs = 5;
            Yorigin = ((int)Math.Floor(minY) / 5) * 5 - 5;
            x = ((int)Math.Ceiling(maxY) / 5) * 5 + 5;
            ScaleY = x - Yorigin;
            Ydivs = 5;
        }
        catch (Exception)
        {
            Xorigin = 0; ScaleX = 5; Xdivs = 5;
            Yorigin = 0; ScaleY = 5; Ydivs = 5;
        }
    }

    /// <summary>
    ///     Draws the Diagram
    /// </summary>
    /// <param name="p">Page to draw on</param>
    /// <param name="myWidth">Width of the Drawing</param>
    /// <param name="myHeight">Height of the Drawing</param>
    public void Draw(Page p, int myWidth, int myHeight)
    {

        b = new Bitmap(myWidth, myHeight);

        g = Graphics.FromImage(b);

        Width = myWidth; Height = myHeight;

        CalculateMeasures();

        int i;

        float x, y, x0, y0;

        string myLabel;

        //Pen blackPen = new Pen(Color.Red, 3);

        Brush blackBrush = new SolidBrush(Color.Red);

        Font axesFont = new Font("arial", 10);



        //first establish working area

        p.Response.ContentType = "image/jpeg";

        g.FillRectangle(new
```

```
            SolidBrush(Color.LightYellow), 0, 0, Width, Height);

            int ChartInset = 50;

            int ChartWidth = Width - (2 * ChartInset);

            int ChartHeight = Height - (2 * ChartInset);

            g.DrawRectangle(new Pen(Color.Black, 3), ChartInset, ChartInset,
ChartWidth, ChartHeight);


            //must draw all text items before doing the rotate below

            g.DrawString(Title, new Font("arial", 14), blackBrush, Width / 3,
10);

            //draw X axis labels

            for (i = 0; i <= Xdivs; i++)
            {

                x = ChartInset + (i * ChartWidth) / Xdivs;

                y = ChartHeight + ChartInset;

                myLabel = (Xorigin + (ScaleX * i / Xdivs)).ToString();

                g.DrawString(myLabel, axesFont, blackBrush, x - 4, y + 10);

                //g.DrawLine(blackPen, x, y + 2, x, y - 2);

            }

            //draw Y axis labels

            for (i = 0; i <= Ydivs; i++)
            {

                x = ChartInset;

                y = ChartHeight + ChartInset - (i * ChartHeight / Ydivs);

                myLabel = (Yorigin + (ScaleY * i / Ydivs)).ToString();

                g.DrawString(myLabel, axesFont, blackBrush, 5, y - 6);

                //g.DrawLine(blackPen, x + 2, y, x - 2, y);

            }


            //transform drawing coords to lower-left (0,0)

            g.RotateTransform(180);

            g.TranslateTransform(0, -Height);

            g.TranslateTransform(-ChartInset, ChartInset);
```

```
            g.ScaleTransform(-1, 1);


            //draw chart data

            datapoint prevPoint = new datapoint();

            prevPoint.valid = false;

            foreach (datapoint myPoint in chartValues)
            {

                if (prevPoint.valid == true)
                {
                    x0 = ChartWidth * (prevPoint.x - Xorigin) / ScaleX;
                    y0 = ChartHeight * (prevPoint.y - Yorigin) / ScaleY;

                    x = ChartWidth * (myPoint.x - Xorigin) / ScaleX;
                    y = ChartHeight * (myPoint.y - Yorigin) / ScaleY;


                    g.FillEllipse(blackBrush, x0 - 2, y0 - 2, 4, 4);
                    g.FillEllipse(blackBrush, x - 2, y - 2, 4, 4);

                }

                prevPoint = myPoint;

            }



            //finally send graphics to browser

            b.Save(p.Response.OutputStream, ImageFormat.Jpeg);

            g.Dispose();

            b.Dispose();
        }

    }
}
```

Matrix.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Collections.Generic;
```

```
namespace BackPropogation
{

    /// <summary>
    ///     Class for shuffling Matrices
    /// </summary>
    public class MatrixShuffler
    {
        private static Random random = new Random();
        private int[] shuffler;

        private class MyDescComparer: IComparer<int>
        {
            int IComparer<int>.Compare(int a, int b)
            {
                return b - a;
            }
        }

        /// <summary>
        ///     Creates new Matrix Shuffler
        /// </summary>
        /// <param name="count">Number of values</param>
        public MatrixShuffler(int count)
        {
            List<KeyValuePair<int, int>> list = new List<KeyValuePair<int,
int>>();

            for (int i = 0; i < count; ++i)
                list.Add(new KeyValuePair<int, int>(random.Next(), i));

            var sorted = from item in list
                         orderby item.Key
                         select item;

            shuffler = new int[count];

            int index = 0;
            foreach (KeyValuePair<int, int> pair in sorted)
            {
                shuffler[index] = pair.Value;
                index++;
            }

        }

        /// <summary>
        ///     Number of values
        /// </summary>
        public int Length
        {
            get
            {
                return shuffler.Length;
            }
        }

        /// <summary>
        ///     Sorts the shuffler descending
        /// </summary>
```

```
        public void sortDesc()
        {
            Array.Sort<int>(shuffler, new MyDescComparer());
        }

        /// <summary>
        ///     Finds selected value of the shuffler
        /// </summary>
        /// <param name="index">Number of value</param>
        /// <returns>Selected value</returns>
        public int this[int index] {
            get
            {
                if (shuffler == null || index < 0 || index > shuffler.Count())
                    return 0;
                return shuffler[index];
            }
        }
    }

    /// <summary>
    ///     Class for single Matrix Row
    /// </summary>
    public class MatrixRow
    {
        private double[] row;

        /// <summary>
        ///     Creates new Matrix Row
        /// </summary>
        /// <param name="cols">Number of columns</param>
        public MatrixRow(int cols)
        {
            row = new double[cols];
        }

        /// <summary>
        ///     Copies Matrix row
        /// </summary>
        /// <param name="previousMatrixRow">MatrixRow to copy</param>
        public MatrixRow(MatrixRow previousMatrixRow)
        {
            row = new double[previousMatrixRow.Length()];

            for (int i = 0; i < previousMatrixRow.Length(); ++i)
                row[i] = previousMatrixRow.row[i];
        }

        /// <summary>
        ///     Copies Matrix row from array
        /// </summary>
        /// <param name="previousMatrixRow">Array to copy</param>
        public MatrixRow(double [] previousMatrixRow)
        {
            row = new double[previousMatrixRow.Count()];

            for (int i = 0; i < previousMatrixRow.Count(); ++i)
                row[i] = previousMatrixRow[i];
        }

        /// <summary>
```

```
        ///     Finds selected value
        /// </summary>
        /// <param name="index">Number of value</param>
        /// <returns>Selected value</returns>
        public double this[int index]
        {
            get
            {
                    return row[index];
            }
            set
            {
                row[index] = value;
            }
        }

        /// <summary>
        ///     Counts the lenght of the row
        /// </summary>
        /// <returns>Number of elements in the row</returns>
        public int Length()
        {
            return row.Length;
        }


        /// <summary>
        ///     Finds maximal value in the row
        /// </summary>
        /// <returns>Maximal Value</returns>
        public double Max()
        {
            return row.Max();
        }

        /// <summary>
        ///     Finds minimal value in the row
        /// </summary>
        /// <returns>Minimal value</returns>
        public double Min()
        {
            return row.Min();
        }

        /// <summary>
        ///     Converts matrix row to Double Array
        /// </summary>
        /// <returns>row as array</returns>
        public double[] toDoubleArray()
        {
            double[] output = new double[row.Count()];
            for (int i = 0; i < output.Count(); ++i)
                output[i] = row[i];
            return output;
        }
    }


    /// <summary>
    ///     Basic Matrix class for statistical actions
    /// </summary>
```

```
public class Matrix
{
    private MatrixRow[] m;


    /// <summary>
    ///     Constructs new Matrix
    /// </summary>
    /// <param name="rows">Number of rows</param>
    /// <param name="cols">Number of columns</param>
    public Matrix(int rows, int cols)
    {
        m = new MatrixRow[rows];

        for (int i = 0; i < rows; ++i)
            m[i] = new MatrixRow(cols);
    }

    /// <summary>
    ///     Copies existing Matrix
    /// </summary>
    /// <param name="previousMatrix">Matrix to copy</param>
    public Matrix(Matrix previousMatrix)
    {
        m = new MatrixRow[previousMatrix.nRows];

        for (int i = 0; i < previousMatrix.nRows; ++i)
        {
            m[i] = new MatrixRow(previousMatrix.m[i]);
        }
    }

    /// <summary>
    ///     Copies existing Matrix from array
    /// </summary>
    /// <param name="previousMatrix">Array to copy</param>
    public Matrix(double[][] previousMatrix)
    {
        m = new MatrixRow[previousMatrix.Count()];

        for (int i = 0; i < previousMatrix.Count(); ++i)
        {
            m[i] = new MatrixRow(previousMatrix[i]);
        }
    }

    /// <summary>
    ///     number of rows in the matrix
    /// </summary>
    public int nRows
    {
        get
        {
            return m.Length;
        }
    }

    /// <summary>
    ///     Number of columns in the matrix
    /// </summary>
    public int nCols
```

```
        {
            get
            {
                return m[0].Length();
            }
        }

        /// <summary>
        ///     Finds selected row
        /// </summary>
        /// <param name="index">Row number</param>
        /// <returns>Selected Row</returns>
        public MatrixRow this[int index]
        {
            get
            {
                return m[index];
            }
        }

        /// <summary>
        ///     Counts the average of all rows of the matrix
        /// </summary>
        /// <returns>One row with average values of all rows of the
matrix</returns>
        public MatrixRow Average()
        {
            MatrixRow mr = new MatrixRow(nCols);

            for (int i = 0; i < m.Length; ++i)
                for (int j = 0; j < nCols; ++j)
                    mr[j] += m[i][j];

            for (int j = 0; j < nCols; ++j)
                mr[j] /= m.Length;

            return mr;
        }

        /// <summary>
        ///     Finds part of the matrix (subset of rows)
        /// </summary>
        /// <param name="startRow">Starting row (including)</param>
        /// <param name="endRow">Ending row (excluding)</param>
        /// <returns>Matrix subset of rows</returns>
        /// <exception cref="Exception">Index out of range</exception>
        public Matrix part(int startRow, int endRow)
        {
            try
            {
                Matrix m = new Matrix(endRow - startRow, nCols);

                for (int i = startRow; i < endRow; ++i)
                    m.m[i - startRow] = new MatrixRow(this.m[i]);

                return m;
            }
            catch (IndexOutOfRangeException)
            {
                throw new Exception("Index out of range [" + startRow.ToString()
+ ", " + endRow.ToString());
```

```
    }
}

/// <summary>
///     Finds maximum value in the Matrix
/// </summary>
/// <returns>Maximal value</returns>
public double Max()
{
    double max = m[0][0];

    for (int i = 0; i < m.Count(); ++i)
        if (m[i].Max() > max)
            max = m[i].Max();

    return max;
}

/// <summary>
///     Finds minimum value in the Matrix
/// </summary>
/// <returns>Minimal value</returns>
public double Min()
{
    double min = m[0][0];

    for (int i = 0; i < m.Count(); ++i)
        if (m[i].Min() < min)
            min = m[i].Min();

    return min;

}

/// <summary>
///     Finds minimum value in every column
/// </summary>
/// <returns>Array of minimum values</returns>
public double[] colMin()
{
    double[] min = new double[nCols];

    for (int i = 0; i < nCols; ++i)
    {
        min[i] = m[1][i];
        for (int j = 0; j < nRows; ++j)
            if (m[j][i] < min[i])
                min[i] = m[j][i];
    }
    return min;
}

/// <summary>
///     Finds maximum value in every column
/// </summary>
/// <returns>Array of maximum values</returns>
public double[] colMax()
{
    double[] max = new double[nCols];

    for (int i = 0; i < nCols; ++i)
```

```
    {
        max[i] = m[1][i];
        for (int j = 0; j < nRows; ++j)
            if (m[j][i] > max[i])
                max[i] = m[j][i];
    }
    return max;
}

/// <summary>
///     Adds extra column at the beginning of the matrix
/// </summary>
/// <param name="fill">Digit to fill the Matrix column</param>
/// <returns>Matrix with added column</returns>
public Matrix addColumn(double fill)
{
    Matrix m = new Matrix(nRows, nCols + 1);

    for (int i = 0; i < m.nRows; ++i)
        for (int j = 0; j < m.nCols; ++j)
        {
            if (j == 0)
                m[i][j] = fill;
            else
                m[i][j] = this[i][j - 1];
        }

    return m;
}

/// <summary>
///     Normalizes the matrix
/// </summary>
/// <returns>Normalized Matrix</returns>
public Matrix Normalized()
{
    Matrix normalized = new Matrix(nRows, nCols);
    double[] min = colMin();
    double[] max = colMax();


    for (int i = 0; i < nRows; ++i)
        for (int j = 0; j < nCols; ++j)
            normalized[i][j] = (m[i][j] - min[j]) / (max[j] - min[j]);

    return normalized;
}

/// <summary>
///     Randomly shuffles the matrix
/// </summary>
/// <param name="shuffler">Shuffler to use</param>
public void shuffle(MatrixShuffler shuffler)
{
    MatrixRow[] tmp = new MatrixRow[m.Count()];

    for (int i = 0; i < m.Count(); ++i)
        tmp[shuffler[i]] = m[i];

    m = tmp;
}
```

```
        /// <summary>
        ///     Removes random rows from matrix
        /// </summary>
        /// <param name="ms">Shuffler with values to remove</param>
        /// <returns>Matrix of rows removed</returns>
        public Matrix removeRandomRows(MatrixShuffler ms)
        {
            ms.sortDesc();

            Matrix m = new Matrix(ms.Length, nCols);

            for (int i = 0; i < ms.Length; ++i)
            {
                m.m[i] = this.m[ms[i]];
                removeRow(ms[i]);
            }

            return m;
        }

        /// <summary>
        ///     Removes selected row
        /// </summary>
        /// <param name="number">Row to remove</param>
        private void removeRow(int number)
        {
            MatrixRow[] m = new MatrixRow[nRows - 1];

            for (int i = 0, j = 0; i < nRows; ++i)
            {
                if (i != number)
                    m[j++] = this.m[i];
            }

            this.m = m;
        }

        /// <summary>
        ///     Finds the Weighted sum of matrix row values
        /// </summary>
        /// <param name="weights">Array of weights</param>
        /// <returns>Weighted sum</returns>
        public MatrixRow WeightedSum(double[] weights)
        {
            MatrixRow mr = new MatrixRow(nCols);

            for (int i = 0; i < m.Length; ++i)
                for (int j = 0; j < nCols; ++j)
                    mr[j] += m[i][j] * weights[i];

            return mr;
        }
    }
}
```

NeuralNetwork.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Threading;

namespace BackPropogation
{
    /// <summary>
    ///     Class for Neural Network implementation
    /// </summary>
    public class NeuralNetwork
    {
        private Matrix desiredResults = null,
                       data = null,
                       testDesiredResults = null,
                       testData = null;

        private MatrixShuffler ms = null;
        private int epochs;
        private double constant;
        private double primary;
        private int[] hidden;
        private double error;
        private double testError;
        private String time = "NaN";
        private bool initialized = false;
        private Neuron[][] network;

        /// <summary>
        ///     Array of Neurons
        /// </summary>
        public Neuron[][] NNetwork
        {
            get
            {
                return network;
            }
        }

        /// <summary>
        ///     Number of Hidden Layers
        /// </summary>
        public int HiddenCount
        {
            get
            {
                return (hidden == null ? 0 : hidden.Count() - 1);
            }
            set
```

```
        {
            hidden = new int[value + 1];
            for (int i = 0; i < hidden.Count(); ++i)
                hidden[i] = 1;
        }
    }

    /// <summary>
    ///     Number of Neurons in every layer
    /// </summary>
    public int[] Hidden
    {
        get
        {
            return hidden;
        }
    }

    /// <summary>
    ///     Number of epochs
    /// </summary>
    public int Epochs
    {
        get
        {
            return epochs;
        }
        set
        {
            epochs = value;
        }
    }

    /// <summary>
    ///     Constant to multiply with
    /// </summary>
    public double Constant
    {
        get
        {
            return constant;
        }
        set
        {
            constant = value;
        }
    }

    /// <summary>
    ///     Default value
    /// </summary>
    public double Primary
    {
        get
        {
            return primary;
        }
        set
        {
            primary = value;
        }
```

```
}

/// <summary>
///     Matrix with testing desired results
/// </summary>
public Matrix TestDesiredResults
{
    get
    {
        return testDesiredResults;
    }
}

/// <summary>
///     Matrix with desired results
/// </summary>
public Matrix DesiredResults
{
    get
    {
        return desiredResults;
    }

    set
    {
        desiredResults = value;
        testDesiredResults = desiredResults.removeRandomRows(ms);
    }
}

/// <summary>
///     Matrix with Test Data
/// </summary>
public Matrix TestData
{
    get
    {
        return testData;
    }
}

/// <summary>
///     Matrix with data
/// </summary>
public Matrix Data
{
    get
    {
        return data;
    }
    set
    {
        data = value;
        testData = data.removeRandomRows(ms);
    }
}

private static NeuralNetwork Network = null;

private NeuralNetwork()
{
```

```
            ms = new MatrixShuffler(10);
            ms.sortDesc();
        }

        /// <summary>
        ///     Get the instance of Neural Network
        /// </summary>
        /// <returns>Neural Network</returns>
        public static NeuralNetwork getNNetwork()
        {
            if (Network == null)
            {
                Network = new NeuralNetwork();
                Network.epochs = 300;
                Network.constant = 0.1;
                Network.primary = 100;
                /* test data */
                Network.HiddenCount = 1;
                Network.Hidden[0] = 10;
            }
            return Network;
        }

        /// <summary>
        ///     Randomly generates weights
        /// </summary>
        /// <param name="n">Number of weights to generate</param>
        /// <returns>Array of generated weights</returns>
        private double[] generateWeights(int n)
        {
            double[] tmp = new double[n];
            double sum = 0;
            Random rn = new Random();

            for (int i = 0; i < n; ++i)
                tmp[i] = rn.Next();

            for (int i = 0; i < n; ++i)
                sum += tmp[i];

            for (int i = 0; i < n; ++i)
                tmp[i] /= sum;

            return tmp;
        }

        /// <summary>
        ///     Modified Back Propagation algorhytm, version 2
        /// </summary>
        public void backPropogation2()
        {
            try
            {
                hidden[HiddenCount] = desiredResults.nCols;  //neuronų kiekis
išėjimo sluoksnyje lygus trokštamų duomenų stulpelių kiekiui
                DateTime Start = DateTime.Now;
                if (!initialized)
                {
                    network = new Neuron[hidden.Count()][]; //paslėpti neuronai
+ išėjimo sluoksnis
                    initialized = true;
```

```
                }

                // mokymo duomenų ir trokštamų reikšmių priskyrimas kitam
kintamajam
                Matrix data1 = new Matrix(data.Normalized().addColumn(1));
                Matrix desiredResults1 = new
Matrix(desiredResults.Normalized());

                //-------------------- Pradinių svorių priskyrimas--------------
---------

                for (int i = 0; i < network.Count(); ++i)
                {
                    network[i] = new Neuron[hidden[i]];

                    for (int j = 0; j < network[i].Count(); ++j)
                        network[i][j] = new Neuron((i == 0 ? data1.nCols :
network[i - 1].Count() + 1), primary);
                }

                double[] weights = generateWeights(data1.nRows);

                for (int epoch = 0; epoch < epochs; ++epoch)   // tikrinama ar
nesibaigė epochų skaičius. Viena epocha, kai praeinami visi duomenys
                {
                    for (int currentRow = 0; currentRow < 4; ++currentRow)
                    {
                        //--- Paslėptuose sluoksniuose esančių neuronų
apskaičiavimas-----------

                        int intervalLen = data1.nRows / 4,
                            startRow = currentRow * intervalLen,
                            endRow = (currentRow == 3 ? data1.nRows : startRow +
intervalLen - 1);
                        double[] inputData = data1.part(startRow,
endRow).WeightedSum(weights).toDoubleArray();


                        for (int i = 0; i < network.Count(); ++i)
//perbegam per visus sluoksnius
                        {
                            for (int j = 0; j < network[i].Count(); ++j)
//perbegam per visus neuronus sluoksnyje
                                network[i][j].train(inputData, logicalSygmoid);

                            //suformuojam rezultatų masyvą
                            inputData = new double[network[i].Count() + 1];
                            inputData[0] = 1;

                            for (int j = 1; j < inputData.Count(); ++j)
                                inputData[j] = network[i][j - 1].Output;
                        }

                        //------------------------- Klaidos skaičiavimas--------
---------------
                        //double error = 0;
                        //for (int i = 0; i < desiredResults1.nCols; ++i)
                            //error +=
Math.Pow(desiredResults1.WeightedSum(weights)[i] - inputData[i + 1], 2);
                        //this.error = error / 2;
```

```
                        //Series1->AddXY(ki++, klaidum, "", clRed); // gautos
klaidos išvedimas į ekraną grafiškai

                        //-------------------------- Skleidimas atgal----------
---------------
                        for (int i = network.Count() - 1; i >= 0; --i)
                        {
                            for (int j = 0; j < network[i].Count(); ++j)
                            {
                                double asuma = 0;
                                if (i == network.Count() - 1)
                                    asuma = network[i][j].Output -
desiredResults1.Average()[j];
                                else
                                    for (int ij = 0; ij < network[i +
1].Count(); ++ij)
                                        asuma += network[i + 1][ij].ErrDiff *
network[i + 1][ij].getWeight(j);
                                network[i][j].countEDiff(asuma);
                            }
                        } //end for(i)

                        //------- svorių išvestinių skaičiavimas ir svorių
atnaujinimas--------
                        inputData = data1.WeightedSum(weights).toDoubleArray();

                        for (int i = 0; i < network.Count(); ++i)
//perbegam per visus sluoksnius
                        {
                            for (int j = 0; j < network[i].Count(); ++j)
//perbegam per visus neuronus sluoksnyje
                                network[i][j].updateWeights(inputData,
constant);

                            //suformuojam rezultatų masyvą
                            inputData = new double[network[i].Count() + 1];
                            inputData[0] = 1;

                            for (int j = 1; j < inputData.Count(); ++j)
                                inputData[j] = network[i][j - 1].Output;
                        }
                    }
                }
                DateTime End = DateTime.Now;
                time = (End - Start).ToString();
                error = countError(data, desiredResults);
                testError = countError(testData, testDesiredResults);
            }
            catch (Exception exc)
            {
                initialized = false;
                throw exc;
            }
        }

        /// <summary>
        ///     Modified Back Propagation algorhytm, version 1
        /// </summary>
        public void backPropogation1()
        {
            try
```

```
            {
                hidden[HiddenCount] = desiredResults.nCols;  //neuronų kiekis
išėjimo sluoksnyje lygus trokštamų duomenų stulpelių kiekiui
                DateTime Start = DateTime.Now;
                if (!initialized)
                {
                    network = new Neuron[hidden.Count()][]; //paslėpti neuronai
+ išėjimo sluoksnis
                    initialized = true;
                }

                // mokymo duomenų ir trokštamų reikšmių priskyrimas kitam
kintamajam
                Matrix data1 = new Matrix(data.Normalized().addColumn(1));
                Matrix desiredResults1 = new
Matrix(desiredResults.Normalized());

                //-------------------- Pradinių svorių priskyrimas----------
---------

                for (int i = 0; i < network.Count(); ++i)
                {
                    network[i] = new Neuron[hidden[i]];

                    for (int j = 0; j < network[i].Count(); ++j)
                        network[i][j] = new Neuron((i == 0 ? data1.nCols :
network[i - 1].Count() + 1), primary);
                }

                for (int epoch = 0; epoch < epochs; ++epoch)   // tikrinama ar
nesibaigė epochų skaičius. Viena epocha, kai praeinami visi duomenys
                {
                    for (int currentRow = 0; currentRow < 4; ++currentRow)
                    {
                        //--- Paslėptuose sluoksniuose esančių neuronų
apskaičiavimas-----------

                        int intervalLen = data1.nRows / 4,
                            startRow = currentRow * intervalLen,
                            endRow = (currentRow == 3 ? data1.nRows : startRow +
intervalLen - 1);
                        double[] inputData = data1.part(startRow,
endRow).Average().toDoubleArray();

                        for (int i = 0; i < network.Count(); ++i)
//perbegam per visus sluoksnius
                        {
                            for (int j = 0; j < network[i].Count(); ++j)
//perbegam per visus neuronus sluoksnyje
                                network[i][j].train(inputData, logicalSygmoid);

                            //suformuojam rezultatų masyvą
                            inputData = new double[network[i].Count() + 1];
                            inputData[0] = 1;

                            for (int j = 1; j < inputData.Count(); ++j)
                                inputData[j] = network[i][j - 1].Output;
                        }

                        //------------------------- Klaidos skaičiavimas--------
---------------
```

```
//                          double error = 0;
                        //for (int i = 0; i < desiredResults1.nCols; ++i)
                          //  error += Math.Pow(desiredResults1.Average()[i] -
inputData[i + 1], 2);
                        //this.error = error / 2;
                        //Series1->AddXY(ki++, klaidum, "", clRed); // gautos
klaidos išvedimas į ekraną grafiškai

                        //-------------------------- Skleidimas atgal----------
---------------
                        for (int i = network.Count() - 1; i >= 0; --i)
                        {
                            for (int j = 0; j < network[i].Count(); ++j)
                            {
                                double asuma = 0;
                                if (i == network.Count() - 1)
                                    asuma = network[i][j].Output -
desiredResults1.Average()[j];
                                else
                                    for (int ij = 0; ij < network[i +
1].Count(); ++ij)
                                        asuma += network[i + 1][ij].ErrDiff *
network[i + 1][ij].getWeight(j);
                                network[i][j].countEDiff(asuma);
                            }
                        } //end for(i)

                        //------- svorių išvestinių skaičiavimas ir svorių
atnaujinimas--------
                        inputData = data1.Average().toDoubleArray();

                        for (int i = 0; i < network.Count(); ++i)
//perbegam per visus sluoksnius
                        {
                            for (int j = 0; j < network[i].Count(); ++j)
//perbegam per visus neuronus sluoksnyje
                                network[i][j].updateWeights(inputData,
constant);

                            //suformuojam rezultatų masyvą
                            inputData = new double[network[i].Count() + 1];
                            inputData[0] = 1;

                            for (int j = 1; j < inputData.Count(); ++j)
                                inputData[j] = network[i][j - 1].Output;
                        }
                    }
                }
                DateTime End = DateTime.Now;
                time = (End - Start).ToString();
                error = countError(data, desiredResults);
                testError = countError(testData, testDesiredResults);
            }
            catch (Exception exc)
            {
                initialized = false;
                throw exc;
            }
        }

        /// <summary>
```

```
        ///     Back Propagation algorhytm implementation
        /// </summary>
        public void backPropogation()
        {
            try
            {
                hidden[HiddenCount] = desiredResults.nCols;  //neuronų kiekis
išėjimo sluoksnyje lygus trokštamų duomenų stulpelių kiekiui
                DateTime Start = DateTime.Now;
                if (!initialized)
                {
                    network = new Neuron[hidden.Count()][]; //paslėpti neuronai
+ išėjimo sluoksnis
                    initialized = true;
                }

                // mokymo duomenų ir trokštamų reikšmių priskyrimas kitam
kintamajam
                Matrix data1 = new Matrix(data.Normalized().addColumn(1));
                Matrix desiredResults1 = new
Matrix(desiredResults.Normalized());

                //-------------------- Pradinių svorių priskyrimas--------------
---------

                for (int i = 0; i < network.Count(); ++i)
                {
                    network[i] = new Neuron[hidden[i]];

                    for (int j = 0; j < network[i].Count(); ++j)
                        network[i][j] = new Neuron((i == 0 ? data1.nCols :
network[i - 1].Count() + 1), primary);
                }

                for (int epoch = 0; epoch < epochs; ++epoch)   // tikrinama ar
nesibaigė epochų skaičius. Viena epocha, kai praeinami visi duomenys
                {
                    for (int currentRow = 0; currentRow < data.nRows;
++currentRow)   // tikrina ar panaudoti visi duomenų vektoriai skaičiavimui
                    {
                        //--- Paslėptuose sluoksniuose esančių neuronų
apskaičiavimas-----------
                        double[] inputData = data1[currentRow].toDoubleArray();

                        for (int i = 0; i < network.Count(); ++i)
//perbegam per visus sluoksnius
                        {
                            for (int j = 0; j < network[i].Count(); ++j)
//perbegam per visus neuronus sluoksnyje
                                network[i][j].train(inputData, logicalSygmoid);

                            //suformuojam rezultatų masyvą
                            inputData = new double[network[i].Count() + 1];
                            inputData[0] = 1;

                            for (int j = 1; j < inputData.Count(); ++j)
                                inputData[j] = network[i][j - 1].Output;
                        }

                        //------------------------- Skleidimas atgal----------
--------------
```

```
                        for (int i = network.Count() - 1; i >= 0; --i)
                        {
                            for (int j = 0; j < network[i].Count(); ++j)
                            {
                                double asuma = 0;
                                if (i == network.Count() - 1)
                                    asuma = network[i][j].Output -
desiredResults1[currentRow][j];
                                else
                                    for (int ij = 0; ij < network[i +
1].Count(); ++ij)
                                        asuma += network[i + 1][ij].ErrDiff *
network[i + 1][ij].getWeight(j);
                                network[i][j].countEDiff(asuma);
                            }
                        } //end for(i)

                        //------- svorių išvestinių skaičiavimas ir svorių
atnaujinimas--------
                        inputData = data1[currentRow].toDoubleArray();

                        for (int i = 0; i < network.Count(); ++i)
//perbegam per visus sluoksnius
                        {
                            for (int j = 0; j < network[i].Count(); ++j)
//perbegam per visus neuronus sluoksnyje
                                network[i][j].updateWeights(inputData,
constant);

                            //suformuojam rezultatų masyvą
                            inputData = new double[network[i].Count() + 1];
                            inputData[0] = 1;

                            for (int j = 1; j < inputData.Count(); ++j)
                                inputData[j] = network[i][j - 1].Output;
                        }
                    }

                    //------------------- Duomenų sumaišymas--------------------
---------
                    MatrixShuffler shuffler = new MatrixShuffler(data1.nRows);
                    data1.shuffle(shuffler);
                    desiredResults1.shuffle(shuffler);
                }
                DateTime End = DateTime.Now;
                time = (End - Start).ToString();
                error = countError(data, desiredResults);
                testError = countError(testData, testDesiredResults);
            }
            catch (Exception exc)
            {
                initialized = false;
                throw exc;
            }
        }

        /// <summary>
        ///     Counts Error
        /// </summary>
        /// <param name="inData">Data</param>
        /// <param name="inDesired">Desired Data</param>
```

```
        /// <returns>Error</returns>
        public double countError(Matrix inData, Matrix inDesired)
        {
            if (!initialized)
                throw new Exception("Tinklas nebuvo sukūrtas");
            // Duomenų išvedimas grafiškai
            double[] desiredMax = inDesired.colMax();
            double[] desiredMin = inDesired.colMin();
            double error = 0;
            Matrix data1 = new Matrix(inData.Normalized().addColumn(1));

            Matrix output = new Matrix(inData.nRows, inDesired.nCols);

            for (int currentRow = 0; currentRow < data1.nRows; ++currentRow)
            {
                double[] inputData = data1[currentRow].toDoubleArray();
                //--- Paslėptuose sluoksniuose esančių neuronų apskaičiavimas---
--------
                for (int i = 0; i < network.Count(); ++i)
//perbegam per visus sluoksnius
                {
                    for (int j = 0; j < network[i].Count(); ++j)
//perbegam per visus neuronus sluoksnyje
                        network[i][j].train(inputData, logicalSygmoid);

                    //suformuojam rezultatų masyvą
                    inputData = new double[network[i].Count() + 1];
                    inputData[0] = 1;

                    for (int j = 1; j < inputData.Count(); ++j)
                        inputData[j] = network[i][j - 1].Output;
                }
                for (int i = 0; i < output.nCols; ++i)
                    output[currentRow][i] = ((inputData[i + 1] - 0.2) / (0.8 -
0.1)) * (desiredMax[i] - desiredMin[i]) + desiredMin[i];

                //------------------------- Klaidos skaičiavimas---------------
-------
                for (int i = 0; i < inDesired.nCols; ++i)
                    error += Math.Pow(output.Normalized()[currentRow][i] -
inDesired.Normalized()[currentRow][i], 2);
            }


            return error / 2 / output.nRows;
        }


        /// <summary>
        ///     Tests teached network
        /// </summary>
        /// <param name="inData">Data</param>
        /// <param name="inDesired">Desired Results</param>
        /// <returns>Array of Results</returns>
        public double[][] run(Matrix inData, Matrix inDesired)
        {
            if (!initialized)
                throw new Exception("Tinklas nebuvo sukūrtas");
            // Duomenų išvedimas grafiškai
            double[] desiredMax = inDesired.colMax();
            double[] desiredMin = inDesired.colMin();
```

```
            Matrix data1 = new Matrix(inData.Normalized().addColumn(1));

            double [][] output = new double[inData.nRows][];

            //System.IO.FileInfo f = new System.IO.FileInfo("C:/log1.txt");
            //System.IO.StreamWriter Tex = f.CreateText();

            for (int currentRow = 0; currentRow < data1.nRows; ++currentRow)
            {
                double [] inputData = data1[currentRow].toDoubleArray();
                //--- Paslėptuose sluoksniuose esančių neuronų apskaičiavimas---
--------
                for (int i = 0; i < network.Count(); ++i)
//perbegam per visus sluoksnius
                {
                    for (int j = 0; j < network[i].Count(); ++j)
//perbegam per visus neuronus sluoksnyje
                        network[i][j].train(inputData, logicalSygmoid);

                    //suformuojam rezultatų masyvą
                    inputData = new double[network[i].Count() + 1];
                    inputData[0] = 1;

                    for (int j = 1; j < inputData.Count(); ++j)
                        inputData[j] = network[i][j - 1].Output;
                }
                output[currentRow] = new double[inDesired.nCols];
                //Tex.WriteLine(inputData[0] + " " + inputData[1] + " " +
inputData[2]);
                for (int i = 0; i < output[currentRow].Count(); ++i)
                    output[currentRow][i] = ((inputData[i+1] - 0.2) / (0.8 -
0.1)) * (desiredMax[i] - desiredMin[i]) + desiredMin[i];
            }
            //Tex.Close();
            return output;
        }

        /// <summary>
        ///     Time of teaching network
        /// </summary>
        public string Time
        {
            get
            {
                return time;
            }
        }


        /// <summary>
        ///     Network error value
        /// </summary>
        public string Error
        {
            get
            {
                return error.ToString();
            }
        }

        /// <summary>
```

```
        ///     Testavimo klaida
        /// </summary>
        public string TestError
        {
            get
            {
                return testError.ToString();
            }
        }


        /// <summary>
        ///     Logical Sygmoid function
        /// </summary>
        /// <param name="x">value</param>
        /// <returns>Logical Sygmoid value</returns>
        double logicalSygmoid(double x)
        {
            return 1 / (1 + Math.Exp(-x)); ;
        }

    }
}
```

Neuron.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Collections.Generic;

namespace BackPropogation
{
    /// <summary>
    ///     Class for single Neuron
    /// </summary>
    public class Neuron
    {
        private double[] weights;
        private double output;
        private bool trained = false;
        private static Random rn = new Random();
        private double errDiff;

        /// <summary>
        ///     Activation function of the neuron
        /// </summary>
        /// <param name="n">Activation function parameter</param>
        /// <returns>Activation function value</returns>
        public delegate double activationFunction(double n);

        /// <summary>
```

```
        ///     Creates new instance of the Neuron
        /// </summary>
        /// <param name="inputCount">Number of inputs</param>
        /// <param name="primary">Default weight of an input</param>
        public Neuron(int inputCount, double primary)
        {
            weights = new double[inputCount];

            for (int i = 0; i < weights.Count(); ++i)
                weights[i] = Math.Round((double)rn.Next(5)) / primary + 5 /
primary;
        }

        /// <summary>
        ///     Error differential
        /// </summary>
        public double ErrDiff
        {
            set
            {
                errDiff = value;
            }
            get
            {
                return errDiff;
            }
        }

        /// <summary>
        ///     Finds the selected weight
        /// </summary>
        /// <param name="i">Number of weight</param>
        /// <returns>Selected Weight</returns>
        public double getWeight(int i)
        {
            if (i < 0 || i >= weights.Count())
                throw new IndexOutOfRangeException();

            return weights[i];
        }

        /// <summary>
        ///     Calculates the number of weights
        /// </summary>
        /// <returns>Number of weights</returns>
        public int weightsCount()
        {
            return weights.Count();
        }

        /// <summary>
        ///     Calculates error differential
        /// </summary>
        /// <param name="x">error</param>
        public void countEDiff(double x)
        {
            errDiff = output * (1 - output) * x;
        }

        /// <summary>
        ///     Trains the neuron
```

```
        /// </summary>
        /// <param name="data">Data</param>
        /// <param name="af">Activation Function</param>
        public void train(double[] data, activationFunction af)
        {


            if (data.Count() != weights.Count())
                throw new Exception("Duomenų kiekis neatitinka neurono įėjimo
kiekį (" + data.Count().ToString() + " <> " + weights.Count().ToString()+ ")");

            double result = 0;
            for (int i = 0; i < data.Count(); ++i)
                result += data[i] * weights[i];

            output = af(result);
            trained = true;
        }

        /// <summary>
        ///     Output value of the Neuron
        /// </summary>
        public double Output
        {
            get
            {
                if (!trained)
                    throw new Exception("Šis neuronas nebuvo apmokytas, todėl
neturi išėjimo duomenų");
                return output;
            }
        }

        /// <summary>
        ///     Updates all weights of Neuron
        /// </summary>
        /// <param name="data">Data</param>
        /// <param name="constant">Constant</param>
        public void updateWeights(double[] data, double constant)
        {
            for (int i = 0; i < data.Count(); ++i)
                weights[i] = weights[i] - constant * errDiff * data[i];
        }
    }
}
```

Results.cs

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Results.aspx.cs"
Inherits="BackPropogation.Results" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
```

```
    <title>Testavimo duomenų rezultatai</title>
</head>
<body style="font-size: small; font-family: Calibri; background-color: #00FFFF">
    <form id="form1" runat="server">
    <asp:Panel ID="ErrorPanel" runat="server" BackColor="Red" Visible="False">
        <asp:Label ID="ErrorLabel" runat="server"></asp:Label>
    </asp:Panel>
    <asp:HyperLink ID="BackLink" runat="server"
NavigateUrl="Default.aspx">Atgal</asp:HyperLink><br />
    <div>
        <asp:Panel ID="MainPanel" runat="server">
        </asp:Panel>
    </div>
    </form>
</body>
</html>
```

Results.aspx.cs

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

namespace BackPropogation
{
    /// <summary>
    ///     Class for output results on the page
    /// </summary>
    public partial class Results : System.Web.UI.Page
    {
        /// <summary>
        ///     Initialies page before loading
        /// </summary>
        /// <param name="sender">Object sender</param>
        /// <param name="e">Event occured</param>
        protected void Page_Load(object sender, EventArgs e)
        {

            try
            {
                Matrix desired = NeuralNetwork.getNNetwork().TestDesiredResults,
                       data = NeuralNetwork.getNNetwork().TestData;
                if (Request.QueryString.Get("data") == "teach")
                {
                    desired = NeuralNetwork.getNNetwork().DesiredResults;
                    data = NeuralNetwork.getNNetwork().Data;
                    Title = "Mokymo duomenų rezultatai";
                }

                double [][] result = NeuralNetwork.getNNetwork().run(data,
desired);
```

```
                Table t = new Table();
                Table t1 = new Table();
                for (int i = 0; i < result.Count(); ++i) //kiekvienas sluoksnis
                {
                    TableRow tr = new TableRow();
                    TableRow tr1 = new TableRow();
                    t.BorderStyle = BorderStyle.Double;
                    t.BorderWidth = 2;

                    t1.BorderStyle = BorderStyle.Double;
                    t1.BorderWidth = 2;
                    for (int j = 0; j < result[i].Count(); ++j) //kiekvienas
neuronas
                    {
                            TableCell td = new TableCell();
                            td.Text = result[i][j].ToString();
                            tr.Cells.Add(td);

                            TableCell td1 = new TableCell();
                            td1.Text = desired[i][j].ToString();
                            tr1.Cells.Add(td1);
                    }
                     t.Rows.Add(tr);
                     t1.Rows.Add(tr1);
                }

                Table mainTable = new Table();
                TableRow mainRow = new TableRow();
                TableCell leftCell = new TableCell();
                TableCell rightCell = new TableCell();
                rightCell.Controls.Add(t);
                leftCell.Controls.Add(t1);
                TableRow headerRow = new TableRow();
                TableCell leftHeader = new TableCell();
                TableCell rightHeader = new TableCell();
                leftHeader.Text = "Trokštami rezultatai";
                rightHeader.Text = "Gauti rezultatai";

                headerRow.Cells.Add(leftHeader);
                headerRow.Cells.Add(rightHeader);
                mainRow.Cells.Add(leftCell);
                mainRow.Cells.Add(rightCell);
                mainTable.Rows.Add(headerRow);
                mainTable.Rows.Add(mainRow);
                MainPanel.Controls.Add(mainTable);
            }
            catch (Exception exc)
            {
                ErrorLabel.Text = exc.Message;
                ErrorPanel.Visible = true;
            }
        }
    }
}
```

Results.aspx.designer.cs

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:2.0.50727.3082
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace BackPropogation {


    public partial class Results {

        /// <summary>
        /// form1 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.HtmlControls.HtmlForm form1;

        /// <summary>
        /// ErrorPanel control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Panel ErrorPanel;

        /// <summary>
        /// ErrorLabel control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Label ErrorLabel;

        /// <summary>
        /// BackLink control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.HyperLink BackLink;

        /// <summary>
        /// MainPanel control.
        /// </summary>
        /// <remarks>
```

```
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Panel MainPanel;
    }
}
```

Weights.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Weights.aspx.cs"
Inherits="BackPropogation.WebForm1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Svoriai</title>
</head>
<body style="font-size: small; font-family: Calibri; background-color: #00FFFF">
    <form id="form1" runat="server">
    <asp:Panel ID="ErrorPanel" runat="server" BackColor="Red" Visible="False">
        <asp:Label ID="ErrorLabel" runat="server"></asp:Label>
    </asp:Panel>
    <asp:HyperLink ID="BackLink" runat="server"
NavigateUrl="Default.aspx">Atgal</asp:HyperLink><br />
    <div>
        <asp:Panel ID="MainPanel" runat="server">
        </asp:Panel>
    </div>
    </form>
</body>
</html>
```

Weights.aspx.cs

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

namespace BackPropogation
{
    /// <summary>
    ///     Class for output weights on the page
    /// </summary>
    public partial class WebForm1 : System.Web.UI.Page
    {
        /// <summary>
```

```
        ///     Initialies page before loading
        /// </summary>
        /// <param name="sender">Object sender</param>
        /// <param name="e">Event occured</param>
        protected void Page_Load(object sender, EventArgs e)
        {

            try
            {
                for (int i = 0; i <
NeuralNetwork.getNNetwork().NNetwork.Count(); ++i) //kiekvienas sluoksnis
                {
                    Table t = new Table();
                    t.BorderStyle = BorderStyle.Double;
                    t.BorderWidth = 2;
                    for (int j = 0; j <
NeuralNetwork.getNNetwork().NNetwork[i].Count(); ++j) //kiekvienas neuronas
                    {
                        TableRow tr = new TableRow();
                        for (int ij = 0; ij <
NeuralNetwork.getNNetwork().NNetwork[i][j].weightsCount(); ++ij) //kiekvienas
svoris
                        {
                            TableCell td = new TableCell();
                            td.Text =
NeuralNetwork.getNNetwork().NNetwork[i][j].getWeight(ij).ToString();
                            tr.Cells.Add(td);
                        }
                        t.Rows.Add(tr);
                    }
                    MainPanel.Controls.Add(t);
                }
            }
            catch (Exception exc)
            {
                ErrorLabel.Text = exc.Message;
                ErrorPanel.Visible = true;
            }
        }
    }
}
```

Weughts.aspx.designer.cs

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:2.0.50727.3082
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace BackPropogation {


    public partial class WebForm1 {

        /// <summary>
```

```
        /// form1 control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.HtmlControls.HtmlForm form1;

        /// <summary>
        /// ErrorPanel control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Panel ErrorPanel;

        /// <summary>
        /// ErrorLabel control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Label ErrorLabel;

        /// <summary>
        /// BackLink control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.HyperLink BackLink;

        /// <summary>
        /// MainPanel control.
        /// </summary>
        /// <remarks>
        /// Auto-generated field.
        /// To modify move field declaration from designer file to code-behind
file.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Panel MainPanel;
    }
}
```