VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS INSTITUTAS

Viktor MEDVEDEV

TIESIOGINIO SKLIDIMO NEURONINIŲ TINKLŲ TAIKYMO DAUGIAMAČIAMS DUOMENIMS VIZUALIZUOTI TYRIMAI

Daktaro disertacija

Technologijos mokslai, informatikos inžinerija (07T)

VGTU
LEIDYKLA
Vilnius TECHNIKA 2007

Disertacija rengta 2003–2007 metais Matematikos ir informatikos institute.

Darbo mokslinis vadovas

prof. habil. dr. Gintautas Dzemyda (Matematikos ir informatikos institutas, technologijos mokslai, informatikos inžinerija – 07 T).

http://leidykla.vgtu.lt
VGTU leidyklos TECHNIKA 1438 mokslo literatūros knyga

ISBN 978-9955-28-211-2

Viktor MEDVEDEV

TIESIOGINIO SKLIDIMO NEURONINIŲ TINKLŲ TAIKYMO DAUGIAMAČIAMS DUOMENIMS VIZUALIZUOTI TYRIMAI

Daktaro disertacija
Technologijos mokslai, informatikos inžinerija (07T)

2007 12 07. 9,25 sp. l. Tiražas 20 egz.
Vilniaus Gedimino technikos universiteto
leidykla „Technika“, Saulėtekio al. 11, LT-10223 Vilnius
http://leidykla.vgtu.lt
Spausdino UAB „Baltijos kopija“
Kareivių g. 13B, 09109 Vilnius
www.kopija.lt

Reziumė

Disertacijos tyrimų sritis yra daugiamačių duomenų analizė, bei tų duomenų suvokimo gerinimo būdai. Duomenų suvokimas yra sudėtingas uždavinys, ypač kai duomenys nurodo sudėtingą objektą, kuris aprašytas daugeliu parametrų. Disertacijoje nagrinėjami dirbtinių neuroninių tinklų algoritmai daugiamačiams duomenims vizualizuoti. Darbo tyrimų objektas yra dirbtiniai neuroniniai tinklai, skirti daugiamačių duomenų vizualizavimui. Su šiuo objektu yra betarpiškai susiję dalykai: daugiamačių duomenų vizualizavimas; dimensijos mažinimo algoritmai; projekcijos paklaidos; naujų taškų atvaizdavimas; vizualizavimui skirto neuroninio tinklo permokymo strategijos ir parametrų optimizavimas; lygiagretieji skaičiavimai. Pagrindinis disertacijos tikslas yra sukurti ir tobulinti metodus, kuriuos taikant būtų efektyviai minimizuojamos daugiamačių duomenų projekcijos paklaidos naudojantis dirbtiniais neuroniniais tinklais bei projekcijos algoritmais. Darbe atliktų tyrimų rezultatai atskleidė naujas medicininių (fiziologinių) duomenų analizės galimybes.

Disertaciją sudaro devyni skyriai ir literatūros sąrašas. Bendra disertacijos apimtis 144 puslapiai, 86 paveikslai ir 1 lentelė.

Tyrimų rezultatai publikuoti 11 mokslinių leidinių: 2 straipsniai leidiniuose, įtrauktuose į Mokslinės informacijos instituto pagrindinį (ISI Web of Science) sąrašą, 2 straipsniai leidiniuose, įtrauktuose į Mokslinės informacijos instituto konferencijos darbų (ISI Proceedings) sąrašą, 1 skyrius užsienio leidyklos (IOS Press) knygoje, 1 straipsnis užsienio leidyklos (Springer) knygoje, 3 straipsniai Lietuvos periodiniuose leidiniuose, 2 straipsniai konferencijų pranešimų medžiagoje. Tyrimų rezultatai buvo pristatyti ir aptarti 9-iose respublikinėse ir tarptautinėse konferencijose.

Abstract

The research area of this work is the analysis of multidimensional data and the ways of improving apprehension of the data. Data apprehension is rather a complicated problem especially if the data refer to a complex object or phenomenon described by many parameters. The research object of the dissertation is artificial neural networks for multidimensional data projection. General topics that are related with this object: multidimensional data visualization; dimensionality reduction algorithms; errors of projecting data; the projection of the new data; strategies for retraining the neural network that visualizes multidimensional data; optimization of control parameters of the neural network for multidimensional data projection; parallel computing. The key aim of the work is to develop and improve methods how to efficiently minimize visualization errors of multidimensional data by using artificial neural networks. The results of the research are applied in solving some problems in practice. Human physiological data that describe the human functional state have been investigated.

The work is written in Lithuanian. It consists of 9 chapters, the list of references. There are 144 pages of the text, 86 figures, 1 table and 159 bibliographical sources.

The main results of this dissertation were published in 11 scientific papers: 2 articles in periodical scientific publications from the ISI Web of Science list; 2 articles in periodical scientific publications from the ISI Proceedings list; 1 article in the book published by Springer; 1 chapter of the book published by IOS Press; 3 articles in periodical scientific publications from the list approved by the Science Council of Lithuania; 2 articles in the proceedings of scientific conferences. The main results of the work have been presented and discussed at 4 international and 5 national conferences.

Padėka

Nuoširdžiai dėkoju moksliniam vadovui prof. habil. dr. Gintautui Dzemydai už vertingas mokslines konsultacijas, nuoseklų vadovavimą, pagalbą ir kantrybę rengiant šią disertaciją.

Ačiū disertacijos recenzentams doc. dr. Antanui Leonui Lipeikai bei dr. Olgai Kurasovai, atidžiai perskaičiusiems disertaciją ir pateikusiems vertingų patarimų bei kritinių pastabų.

Dėkoju Matematikos ir informatikos instituto Sistemų analizės skyriaus darbuotojams ir kolegoms už naudingus patarimus ir draugišką pagalbą.

Nuoširdžiai dėkoju savo artimiesiems ir draugams už jų paramą, moralinį palaikymą, kantrybę ir supratingumą.

Dėkoju Lietuvos valstybiniam mokslo ir studijų fondui už suteiktą finansinę paramą disertacijos rengimo metu.

Taip pat dėkoju visiems kitiems, kurie tiesiogiai ar netiesiogiai prisidėjo prie šio darbo.

Viktor Medvedev

Turinys

1

Įvadas

1.1. Tyrimų sritis

Kiekvieną dieną mes gauname didžiulius kiekius duomenų, disponuojame informacija ir žiniomis. Duomenys – tai objektyviai egzistuojantys faktai, vaizdai arba garsai, kurie gali būti naudingi tam tikram uždaviniui spręsti. Informacija – tai duomenys, kurių forma ir turinys yra pateikti tinkamiausiu būdu priimti sprendimą. Duomenys virsta informacija, kai jiems suteikiamas kontekstas ir jie priskiriami tam tikrai problemai ar sprendimui. Žinios yra gebėjimas spręsti problemas, atnaujinti arba kitaip sukurti reikšmes remiantis ankstesne patirtimi, įgūdžiais ar išmokimu. Dažniausia problematiška gauti svarbių žinių, suprasti duomenis, atskirti svarbią informaciją nuo menkavertės. Duomenų suvokimas yra gana sudėtingas uždavinys, ypač kai duomenys nurodo sudėtingą objektą, reiškinį, kuris aprašytas daugeliu parametrų. Tokie duomenys vadinami *daugiamačiais duomenimis*. Šio darbo tyrimų sritis yra daugiamačių duomenų analizė, bei tų duomenų suvokimo gerinimo būdai.

1.2. Darbo aktualumas

Technikoje, medicinoje, ekonomikoje, ekologijoje ir daugelyje kitų sričių nuolat susiduriama su daugiamačiais duomenimis. Vystantis technologijoms,

tobulėjant kompiuteriams ir programinei įrangai, kaupiamų duomenų apimtys ypač sparčiai didėja.

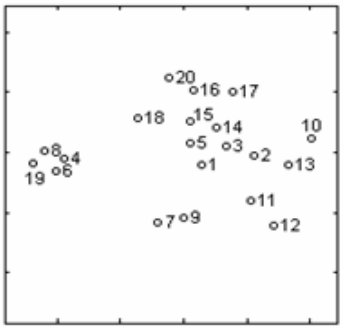
Šiuo metu duomenų analizė yra labai aktuali įvairiose veiklos srityse: moksle, versle, ekonomikoje, medicinoje, sociologijoje ir kt. Duomenų pateikimas žmogui suprantama forma dažnai būna efektyvi priemonė, norint geriau suvokti daugiamačius duomenis, t.y. nustatyti jų struktūrą, tarpusavio ryšius, susidariusias grupes ir pan. Vienas iš galimų būdų yra grafinis informacijos pateikimas arba vizualizavimas. Pagrindinė vizualizavimo idėja – duomenis pateikti tokia forma, kuri leistų vartotojui suprasti duomenis, daryti išvadas ir tiesiogiai įtakoti tolesnį jų srautą. Vizualią informaciją žmogus pajėgus suvokti daug greičiau negu tekstinę.

Per paskutinius 40 metų daugiamačių duomenų vizualizavimo metodai vystomi gana intensyviai siekiant lengvinti duomenų interpretavimą, efektyviai pateikti ir įvertinti duomenų gavybos rezultatus. Taip pat plačiai naudojami projekcijos metodai. Dažnai iškyla būtinybė nustatyti duomenų struktūrą: susidariusias grupes (klasterius), žymiai išsiskiriančius objektus (taškus-atsiskyrėlius), atstumus tarp objektų ir pan. Transformavus daugiamačius duomenis į dvimatę erdvę ir atvaizdavus juos plokštumoje, jau daug paprasčiau suvokti duomenų struktūrą ir sąryšius tarp jų. Pavyzdžiui, 1.1 paveiksle pavaizduota daugiamačių duomenų projekcija plokštumoje. Turint tokį daugiamačių duomenų vizualizavimą, jau galime daryti tam tikras išvadas apie analizuojamą duomenų aibės struktūrą, gauname papildomą informaciją apie duomenis. Paveiksle aiškiai matosi du nagrinėjamų duomenų klasteriai, daugiamačių duomenų tarpusavio išsidėstymas.

analizuojami duomenys

	x_1	x_2	x_3	x_4	x_5	x_6
1	0,57	0,11	0,47	0,54	0,84	0,53
2	0,65	0,14	0,53	0,55	0,89	0,54
3	0,61	0,13	0,49	0,52	0,92	0,57
4	0,44	0,16	0,45	0,42	0,99	0,45
5	0,55	0,11	0,53	0,52	0,92	0,55
6	0,44	0,16	0,43	0,41	0,98	0,43
7	0,49	0,12	0,56	0,46	0,82	0,48
8	0,41	0,17	0,42	0,43	0,98	0,42
9	0,54	0,12	0,59	0,46	0,85	0,48
10	0,69	0,16	0,63	0,56	0,91	0,49
11	0,66	0,16	0,51	0,48	0,87	0,55
12	0,70	0,13	0,52	0,48	0,81	0,52
13	0,65	0,14	0,63	0,52	0,89	0,49
14	0,59	0,11	0,51	0,57	0,89	0,52
15	0,53	0,11	0,52	0,57	0,89	0,50
16	0,52	0,13	0,51	0,61	0,92	0,51
17	0,58	0,12	0,55	0,61	0,95	0,52
18	0,45	0,10	0,52	0,53	0,92	0,51
19	0,42	0,17	0,41	0,42	0,98	0,40
20	0,53	0,11	0,42	0,57	0,91	0,57

duomenų projekcija



1.1 pav. *Duomenų vizualizavimas*

Tačiau duomenis transformuojant į mažesnės dimensijos erdvę, duomenų vizualizavimo iškraipymai, paklaidos yra neišvengiamos. Šių paklaidų minimizavimas išlieka aktualia problema. Ta problema yra pagrindinė šioje disertacijoje sprendžiama **problema**.

Dažnai tenka dirbti su didelės apimties duomenų aibėmis, kurios pastoviai papildomos naujais duomenimis. Todėl naujų taškų atvaizdavimas – dar viena aktuali problema.

Pastaruoju metu stebima tendencija, kad MDS (*angl. Multidimensional Scaling,* daugiamačių skalių metodas) tyrimus vykdantys mokslininkai dažnai atsiriboja nuo kitų metodų tyrimų, ar net ignoruoja tuos metodus. Antra vertus, kitų vizualizavimo metodų tyrimuose nėra lyginimų ar sąsajų su MDS tipo metodais. Šiame darbe siekiama praplėsti MDS tipo metodų realizacijas dirbtinių neuroninių tinklų taikymu, tuo būdu stiprinant ryšį tarp skirtingų vizualios duomenų analizės krypčių.

Darbe nagrinėjami dirbtinių neuroninių tinklų algoritmai daugiamačiams duomenims vizualizuoti. Pavyzdžiui, pasiūlyta specifinė „klaidos sklidimo atgal“ mokymo taisyklė, pavadinta SAMANN, kuri leidžia įprastam tiesioginio sklidimo neuroniniam tinklui realizuoti populiarią Sammono projekciją mokymo be mokytojo būdu.

1.3. Darbo tikslas ir uždaviniai

Pagrindinis disertacijos tikslas yra sukurti ir tobulinti metodus, kuriuos taikant būtų efektyviai minimizuojamos daugiamačių duomenų projekcijos paklaidos naudojantis dirbtiniais neuroniniais tinklais bei projekcijos algoritmais.

Norint pasiekti šį tikslą, reikėjo išspręsti tokius uždavinius:

1) analitiškai apžvelgti daugiamačių duomenų vizualizavimo metodus,
2) ištirti dirbtinių neuroninių tinklų galimybes daugiamačiams duomenims vizualizuoti,
3) išanalizuoti SAMANN algoritmo veikimo principus,
4) sukurti lygiagretųjį SAMANN vizualizavimo algoritmą,
5) pagreitinti (pagerinti) SAMANN tinklo mokymą ir permokymą,
6) surasti optimalias nagrinėjamo algoritmo mokymosi parametro reikšmes,
7) išanalizuoti naujų daugiamačių taškų vizualizavimo galimybes naudojantis dirbtiniais neuroniniais tinklais.

Tyrimų **metodikos** pagrindą sudaro naujų SAMANN neuroninio tinklo mokymo strategijų kūrimas ir jų eksperimentinis tyrimas.

1.4. Tyrimo objektas

Norint atskleisti daugiamačių duomenų struktūras, vien tik klasikinių vizualizavimo metodų nepakanka. Disertacijos tyrimų objektas – dirbtiniai neuroniniai tinklai, skirti daugiamačių duomenų vizualizavimui. Su šiuo objektu betarpiškai susiję dalykai:

1) daugiamačių duomenų vizualizavimas,
2) dimensijos mažinimo (projekcijos) algoritmai,
3) tiesioginio sklidimo dirbtiniai neuroniniai tinklai,
4) daugiamačių duomenų projekcijos į mažesnės dimensijos erdvę paklaidos,
5) naujų taškų (vektorių) atvaizdavimas,
6) daugiamačius duomenis vizualizuojančio neuroninio tinklo (SAMANN) permokymo strategijos,
7) vizualizavimui skirto neuroninio tinklo parametrų optimizavimas,
8) lygiagretieji skaičiavimai.

1.5. Mokslinis naujumas ir ginamieji teiginiai

Sukurtas lygiagretusis SAMANN algoritmas, realizuojantis daugiamačių duomenų vizualizavimą. Lygiagretusis algoritmas leidžia duotąjį uždavinį išspręsti greičiau ir išspręsti sunkesnius uždavinius, nei tai galėtume padaryti naudodami tik vieną procesorių.

SAMANN algoritmas tinka naujų taškų atvaizdavimui. Po SAMANN tinklo mokymo, turint neuroninio tinklo svorių rinkinį, naujas vektorius, pateiktas tinklui, atvaizduojamas į plokštumą labai greitai ir pakankamai tiksliai be jokių papildomų skaičiavimų. Tačiau dirbant su dideliais duomenų kiekiais naujų vektorių gali būti labai daug ir po tam tikro laiko SAMANN tinklą tenka permokyti, kadangi tinklas prisitaiko prie esamų duomenų ir naujus taškus gali atvaizduoti netiksliai. Pasiūlytos ir ištirtos nagrinėjamo algoritmo permokymo strategijos.

Eksperimentiškai nustatyta, kaip parinkti SAMANN tinklo mokymosi parametro reikšmę, kad algoritmas veiktų efektyviai.

1.6. Praktinė vertė

Tyrimų rezultatai atskleidė naujas medicininių (fiziologinių) duomenų analizės galimybes. Tai leido sporto medicinos specialistams įvertinti nesportuojančiųjų sveikatos būklę ir jų galimybę sportuoti.

Tyrimai atlikti pagal Lietuvos valstybinio mokslo ir studijų fondo prioritetinių Lietuvos mokslinių tyrimų ir eksperimentinės plėtros programą „Informacinės technologijos žmogaus sveikatai – klinikinių sprendimų palaikymas (e-sveikata), IT sveikata“; Registracijos Nr.: C-03013; Vykdymo laikas: 2003 m. 09 mėn. – 2006 m. 10 mėn.

1.7. Darbo rezultatų aprobavimas

Tyrimų rezultatai publikuoti 11 moksliniuose leidiniuose: 2 straipsniai leidiniuose, įtrauktuose į Mokslinės informacijos instituto pagrindinį (ISI Web of Science) sąrašą; 2 straipsniai leidiniuose, įtrauktuose į Mokslinės informacijos instituto konferencijos darbų (ISI Proceedings) sąrašą; 1 skyrius užsienio leidyklos (IOS Press) knygoje; 1 straipsnis užsienio leidyklos knygoje; 3 straipsniai Lietuvos periodiniuose leidiniuose; 2 straipsniai konferencijų pranešimų medžiagoje.

Tyrimų rezultatai buvo pristatyti ir aptarti šiose nacionalinėse ir tarptautinėse konferencijose Lietuvoje ir užsienyje:

1. Informacinės technologijos 2004. Kaunas, KTU. 2004 m. sausio 28–29 d.
2. Lietuvos matematikų draugijos XLV konferencija. Kaunas, LŽŪU. 2004 m. birželio 17–18 d.
3. Informacinės technologijos 2005. Kaunas, KTU. 2005 m. sausio 28–29 d.
4. Workshop of the European Chapter on Metaheuristics. Metaheuristics and Large Scale Optimization. Vilnius. 2005 m. gegužės 19–21 d.
5. Lietuvos matematikų draugijos XLVI konferencija. Vilnius, VU. 2005 m. birželio 15–16 d.
6. „Kompiuterininkų dienos–2005“. Klaipėda, KU. 2005 m. rugsėjo 15–17 d.
7. 32nd International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2006. Merin, Čekija. 2006 m. sausio 21–27 d.
8. The 8th International Conference on Artificial Intelligence and Soft Computing, ICAISC 2006. Zakopane, Lenkija. 2006 m. birželio 25–29 d.

   (Pranešimas įvertintas ICAISC’06 Best Presentation Award)

9. The 8th International Conference ICANNGA 2007, Varšuva, Lenkija. 2007 m. balandžio 11–14 d.

   (Pranešimas įvertintas ICANNGA 2007 Best Young Researcher Paper Award in the area of Neural Networks)

1.8. Disertacijos struktūra

Disertaciją sudaro devyni skyriai ir literatūros sąrašas. Disertacijos skyriai: Įvadas, Daugiamačių duomenų vizualizavimo metodai, Dirbtinių neuroninių tinklų koncepcijos, Dirbtinių neuroninių tinklų taikymas daugiamačiams duomenims vizualizuoti, SAMANN neuroninio tinklo mokymo problemos, Lygiagrečios SAMANN algoritmo realizacijos, Naujų taškų atvaizdavimas, SAMANN tinklo taikymai, Išvados. Disertacijos apimtis 145 puslapiai, 86 paveikslai ir 1 lentelė.

2

Daugiamačių duomenų vizualizavimo metodai

Daugiamačių duomenų vizualizavimas – svarbus duomenų analizės įrankis, padedantis geriau suvokti daugiamačių duomenų struktūras, grupavimosi tendencijas, nustatyti atstumus tarp objektų. Tai grafinis duomenų pateikimas, siekiant perduoti informacijos turinį natūraliu (suprantamu) ir tiesioginiu būdais. Daugiamačių duomenų vizualizavimo problemos per pastaruosius 20 metų sulaukė didelio susidomėjimo. Technikoje, medicinoje, ekonomikoje, ekologijoje ir daugelyje kitų sričių nuolat susiduriama su daugiamačiais duomenimis. Vystantis technologijoms, tobulėjant kompiuteriams ir programinei įrangai, kaupiamų duomenų apimtis ypač sparčiai didėja. Didėja ir poreikiai bei nauda, gaunama padarius teisingas išvadas.

Daugiamačiais duomenimis vadinami duomenys, charakterizuojantys vektorių (objektą) $X^j = (x_1^j, x_2^j, ..., x_n^j)$. Parametrai $x_1, x_2, ..., x_n$ vadinami komponentėmis (kintamaisiais), parametrų skaičius n – dimensija (matavimų skaičius). Visų parametrų reikšmių junginys charakterizuoja vieną analizuojamos aibės konkretų objektą $X^j = (x_1^j, x_2^j, ..., x_n^j)$, čia n – parametrų skaičius. Taigi, galima suformuoti analizuojamų duomenų aibę $\mathbb{X}$, sudarytą iš vektorių $X^j \in R^n$, ($X^j = (x_1^j, x_2^j, ..., x_n^j) = \{x_i^j\}$, $i = 1,...,n,\ j = 1,...,m$), čia m – vektorių skaičius. Kiekvienas parametras turi tam tikras skaitines reikšmes, iš kurių

įmanoma sudaryti skaičių lentelę. Kuo didesnės dimensijos duomenys, tuo sunkiau iš lentelės išgauti informacijos apie santykius tarp atskirų objektų.

Daugiamačių duomenų analizės ir vizualizavimo metodai padeda analizuoti ir atvaizduoti žmogui suprantamesne forma sudėtingais ir dažnai nežinomais tarpusavio ryšiais susietus daugiamačius duomenis. Pagrindinis daugiamačių duomenų vizualizavimo tikslas – tai duomenų pavaizdavimas mažesnės dimensijos erdvėje, išsaugant jų tarpusavio panašumo struktūras.

Daugiamačių duomenų (kai dimensija daugiau nei trys) tiesioginis vaizdavimas yra sudėtingas, todėl tokių duomenų pateikimui naudojami įvairūs vizualizavimo metodai. Dažnai neįmanoma pateikti visų duomenų parametrų neprarandant jokios informacijos, todėl būtina ieškoti būdų, leidžiančių tuos praradimus minimizuoti. Metodų įvairovė yra pakankamai plati, todėl būtina įvairių metodų analizė, norint išsirinkti tinkamiausius.

Toliau šiame skyriuje yra nagrinėjamos dvi vizualizavimo metodų grupės: *tiesioginiai vizualizavimo metodai*, kai kiekvienas daugiamačio objekto parametras yra pateikiamas tam tikra vizualia forma; *projekcijos metodai*, kurie leidžia daugiamačius duomenų objektus atitinkančius vektorius pateikti mažesnės dimensijos erdvėje.

Šiame skyriuje daugiamačių duomenų tiesioginiam vizualizavimui ir projekcijos metodams buvo naudojama irisų duomenų aibė (*angl. iris dataset*):

Fišerio irisų duomenys (Fisher, 1936), kurie kartais vadinami tiesiog irisais arba irisų duomenimis, yra klasikiniai testiniai duomenys, naudojami daugiamačių duomenų analizėje. Buvo išmatuota 150-ies gėlių irisų:

- vainiklapių pločiai (*angl. petal weight*),
- vainiklapių ilgiai (*angl. petal height*),
- taurėlapių pločiai (*angl. sepal weight*),
- taurėlapių ilgiai (*angl. sepal height*).

Matuotos trijų rūšių gėlės: Iris Setosa (I klasė), Iris Versicolor (II klasė) ir Iris Virginica (III klasė). Sudaryti 4-mačiai vektoriai. Įvairias metodais nustatyta, kad pirmos klasės irisai „atsiskiria“ nuo kitų dviejų klasių (antros ir trečios). Antra ir trečia klasės dalinai persipina.

2.1. Daugiamačių duomenų tiesioginis vizualizavimas

Daugiamačių duomenų tiesioginio vizualizavimo metoduose nėra apibrėžto formalaus matematinio kriterijaus. Kiekvienas daugiamačio objekto parametras pateikiamas žmogui priimtina vizualia forma.

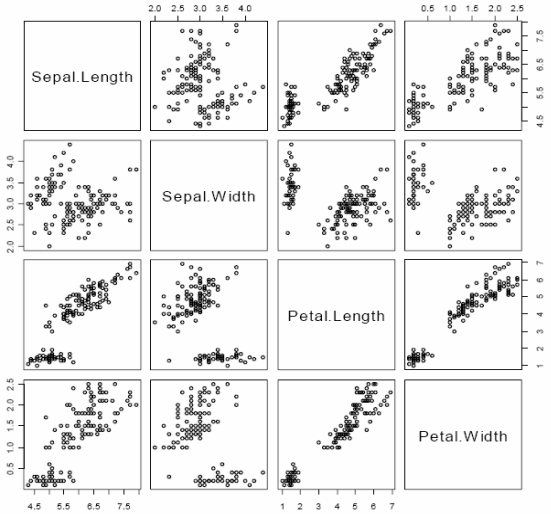
2.1.1. Geometriniai metodai

Geometriniai vizualizavimo metodai – tai didelė grupė vizualizavimo metodų, kuriuose atskiros duomenų vektorių komponenčių (parametrų) reikšmės yra pavaizduojamos naudojant pasirinktos geometrinės figūros ašis. Pagrindinė tokių metodų idėja – vizualizuoti duomenų transformacijas Dekarto arba kitoje koordinačių sistemoje (Sachinopoulou, 2001).

Taškinės diagramos (*angl. scatterplots*) (Andrews, 1972), (Cleveland, 1993), (Hoffman, 2002) yra vienas dažniausiai naudojamų duomenų pateikimo plokštumoje R^2 arba trimatėje erdvėje R^3 būdų (Hoffman, 2002). Įprastai šiuo būdu atvaizduojami dvimačiai arba trimačiai taškai. Turint dvimatį ($n = 2$) arba trimatį ($n = 3$) vektorių, jo pirmos komponentės reikšmė pažymima x ašyje, antros – y ašyje, trečios (jei tokia yra) – z ašyje. Per pažymėtus taškus išvedamos tiesės statmenos ašims. Taškas plokštumoje (erdvėje) atidedamas ten, kur šios tiesės susikerta.

Taškinės diagramos, kuriose naudojamos įvairios spalvos ar formos, gali būti taikomos analizuojant duomenis, kurie turi daugiau nei tris komponentes. Tačiau tada rezultatas nebus toks aiškus, kaip analizuojant dvimačius ar net trimačius duomenis. Viena priežastis yra ta, kad tik x, y ašių naudojimas ir z ašies, spalvos, formos naudojimas neturi vienodo aiškaus vizualizavimo efekto. Šis metodas gali padėti rasti susidariuses grupes, taškus-atsiskyrėlius (*angl. outliers*), trendus ir koreliacijas tarp kelių duomenų parametrų.

Klasikinė taškinė diagrama parodo ryšį tarp dviejų kintamųjų. Daugiamačių duomenų atvaizdavimui naudojamos **taškinių diagramų matricos** *(angl. matrix of scatter plots)* (2.1 pav.).



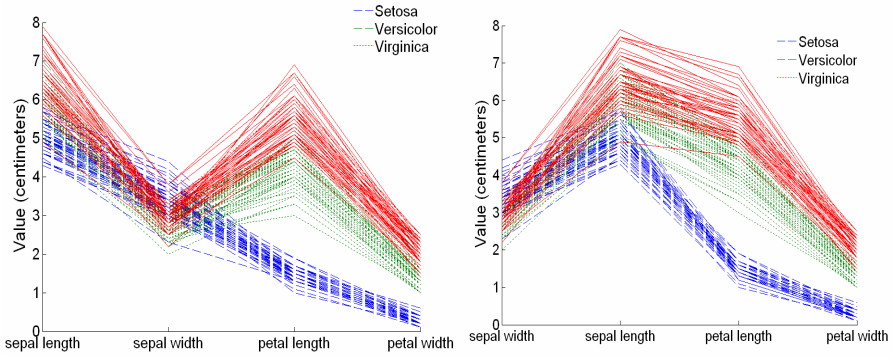
2.1 pav. *Taškinių diagramų matricos pavyzdys (Irisų duomenų aibė)*

Taškinių grafikų matricose n dimensijos yra projektuojamos ant $n \times (n-1)$ taškinių grafikų. Tokioje matricoje atvaizduojamos visos galimos duomenų parametrus atitinkančių komponenčių poros (Keim, 2003). Taškinių grafikų matricos metodas naudingas ieškant koreliacijų tarp dviejų parametrų.

Taškinės diagramos nebūtinai turi būti pateikti stačiakampio masyvo forma. Gali būti naudojamos ir skritulio, šešiakampio ar kitų figūrų formos.

Kitas dažnai naudojamų metodų yra **lygiagrečių koordinačių metodas** *(angl. parallel coordinates)* (Inselberg, 1985), (Inselberg, 1990). Tradicinėje Dekarto koordinačių sistemoje visos ašys yra statmenos viena kitai. Lygiagrečiose koordinatėse visos ašys yra lygiagrečios viena kitai ir išdėstytos vienodu atstumu viena nuo kitos. Lygiagrečiose koordinatėse galima pateikti taškus, linijas, plokštumas, kurių dimensija daugiau nei trys.

Lygiagrečių koordinačių metodo privalumas: metodas gali būti naudojamas didelių matavimų duomenims vizualizuoti, nors labai sutankinus koordinates, yra sunku suvokti duomenų struktūrą. Trūkumas – atvaizduojant didelę duomenų aibę, jų suvokimas tampa labai sudėtingas. Labai svarbi šio metodo savybė yra ta, kad parametrai traktuojami vienodai. Tai leidžia pertvarkyti vaizduojamus parametrus, gauti naują duomenų atvaizdavimą ir geriau suvokti duomenų struktūrą (2.2 pav.).



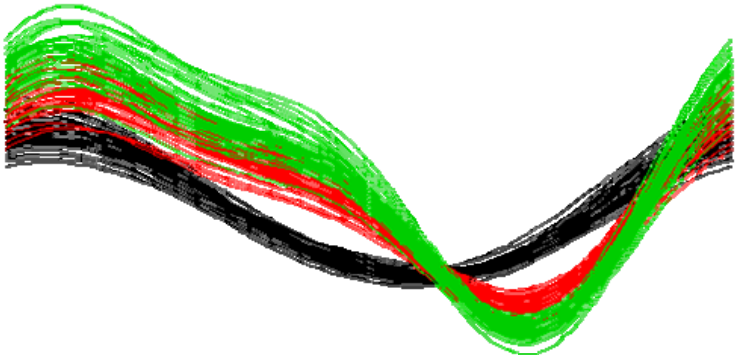
2.2 pav. *Lygiagrečių koordinačių pavyzdys (irisų duomenų aibė)*

Kitas įdomus geometrinis vizualizavimo metodas yra **Andrews kreivės** (Hoffman, 2002). Vieną n-matį vektorių galima pavaizduoti Andrews kreive (sinusoidžių suma), kurios koeficientai yra analizuojamo vektoriaus komponentės (Andrews, 1972) (2.3 pav.):

$$f_x(t) = \frac{x_1}{\sqrt{2}} + x_2 \sin(t) + x_3 \cos(t) + x_4 \sin(2t) + x_5 \cos(2t) + \ldots, \; -\pi<t<\pi.$$

Vienas šio vizualizavimo metodo privalumų – metodą galima taikyti didelės dimensijos duomenų analizei; trūkumas – vizualizuojant didelės aibės duomenis,

atvaizduotus duomenis suvokti gana sunku. Šis metodas gali padėti rasti taškus-atsiskyrėlius (*angl. outliers*) arba vektorius nutolusios nuo pagrindinės vektorių aibės.



2.3 pav. *Andrews kreivių pavyzdys (irisų duomenų aibė)*

Kiti tiesioginio vizualizavimo geometriniai metodai:

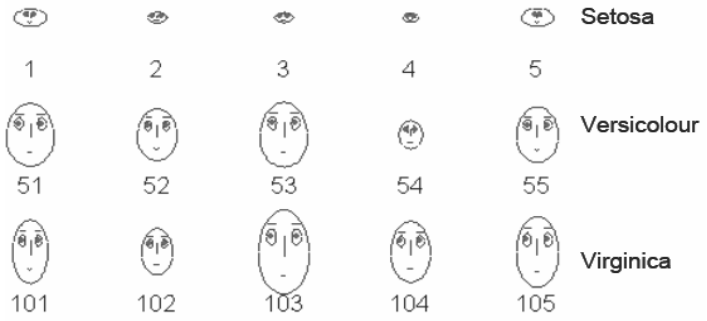
- Projekcijos paieškos metodas (*angl. projection pursuit*) (Huber, 1985),
- Sekcijinis vaizdas (*angl. prosection views*) (Furnas, 1995), (Spence, 1995),
- Hiper pjaustymas (*angl. hyperslice*) (Wijk, 1993),
- Perstatymų matrica (*angl. permutation matrix*) (Bertin, 1983),
- Spindulinis vizualizavimas (*angl. Radviz*), jo modifikacijos (*angl. GridViz*, *PolyViz)* (Hoffman, 2002), ir kt.

2.1.2. Simboliniai metodai

Simboliniai metodai daugiamačių duomenų pateikimui naudoja simbolius (arba ženklus). Daugiamačių duomenų vizualizavimo tikslas yra ne tik atvaizduoti duomenis dvimatėje ar trimatėje erdvėje, bet ir palengvinti daugiamačių duomenų suvokimą. Būtent antrąjį tikslą galima bandyti pasiekti daugiamačius duomenis vizualizuojant simboliniais metodais. Duomenų reikšmės susiejamos su simbolio spalva, forma ar padėtimi.

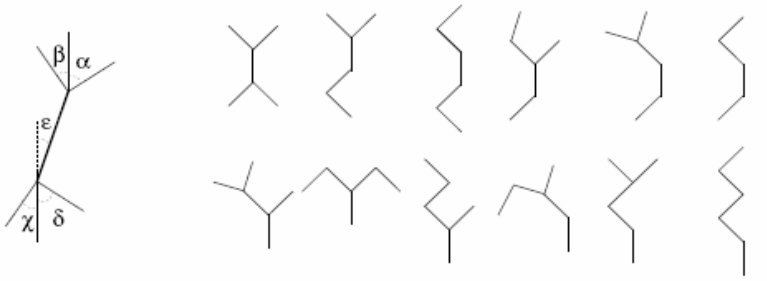
Černovo veidai (*angl. Chernoff Faces*) yra daugiamačių duomenų vizualizavimo metodas, sukurtas statistiko H. Černovo (Chernoff, 1973). Šis metodas yra paprastas, bet labai efektyvūs, nes nustato duomenų ryšius su veido bruožais. Skirtingos duomenų dimensijos projektuojamos į skirtingus veido bruožus. Kiekviena duomenų komponentė nustato kažkurios veido dalies dydį, padėtį ar formą (2.4 pav.). Pvz. viena komponentė susijusi su burnos pločiu, kita su akių forma ir pan. *m d*-mačiai vektoriai gali būti pavaizduoti *m* skirtingais

veidais, kai $d \le 18$ (Jain, 1988). Černovo veidai padeda vizualiai nustatyti taškus-atsiskyrėlius (*angl. outliers*).



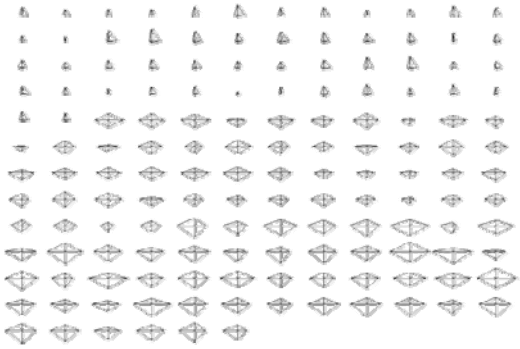
2.4 pav. *Černovo veidų pavyzdys (Irisų duomenų aibė)*

Pickett ir Grinstein (Pickett, 1988) pasiūlė **brūkšnelinės figūros** (*angl. stick figure*) vizualizavimo metodą. Brūkšnelinė figūra susideda iš penkių sujungtų segmentų vadinamų galūnėmis (*angl. limbs*). Viena galūnė pavadinama kūnu (*angl. body*). Kiekviena galūnė turi kelis parametrus, kuriais gali būti išreiškiami analizuojamų duomenų parametrai. Tai galūnės ilgis, jos pasukimo kampas, spalva ir kt. Dažniausiai šiuo metodu atvaizduojami *d*-mačiai duomenys, kai $d \le 5$. Keturi parametrai gali būti išreikšti kiekvienos galūnės orientacija, penktasis – kūno pasvirimu. Kiti parametrai (jeigu jų yra) gali būti išreikšti galūnių ilgiu, storiu ar spalva (Grinstein, 2002). 2.5 paveiksle pavaizduota bazinės konfigūracijos figūra ir brūkšninių figūrų grupė.



2.5 pav. *Brūkšnelinės figūros pavyzdys*

Vienas iš populiariausių simbolinių metodų yra **žvaigždžių metodas** (*angl. star glyphs*) (Chambers, 1983), (Kaski, 1997). Žvaigždės spindulio, išeinančio iš centro, ilgis iliustruoja vieną vektoriaus komponentę (parametrą). Išoriniai spindulių galai yra sujungiami linijomis (2.6 pav.). Taikant šį metodą kartais sunku nustatyti, kaip prasmingai išdėstyti žvaigždes viena šalia kitos.



2.6 pav. *Žvaigždžių pavyzdys (Irisų duomenų aibė)*

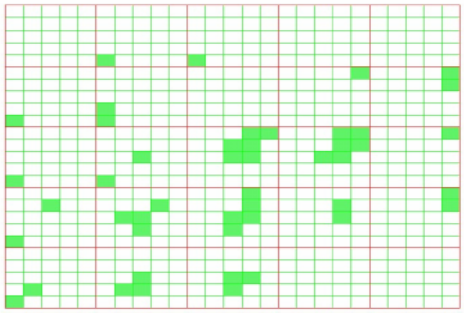
Kiti simboliniai metodai:

- Kontūrinis kodavimas (*angl. shape coding*) (Beddow, 1990),
- Nuspalvintos piktogramos metodas (*angl. color icon*) (Levkowitz, 1991), (Keim, 1994),
- AsbruView (Kosara, 2002),
- SopoView (Messner, 2000),
- Mozaikinė metafora (*angl. mosaic metaphor*) (Nocke, 2005), ir kt.

2.1.3. Hierarchinio vizualizavimo metodai

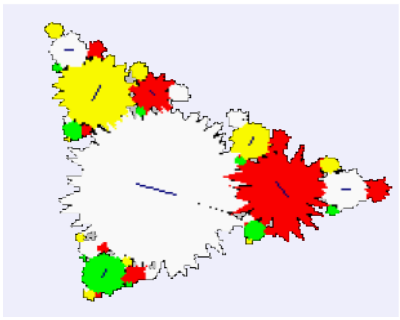
Vienas iš svarbiausių hierarchinio vizualizavimo metodų (*angl. hierarchical display*) yra **dimensijos įterpimo metodas** *(angl. dimensional stacking).* Šis metodas, kaip rekursinis atvaizdavimo metodas iš *n*-matės erdvės į dvimatę, buvo sukurtas LeBlanc ir kt (LeBlanc, 1990). Metodo esmė: dvimatis tinklelis padalijamas į stačiakampius, prieš tai analizuojamų duomenų parametrus suskaidžius į intervalus; du pasirinkti duomenų parametrai pavaizduojami *x* ir *y* ašyse, kiekviena kita parametrų pora yra „įspraudžiama“ į vidinius stačiakampius. Toks įterpimas tęsiamas tol, kol įterpiami visi parametrai. Vidiniai stačiakampiai, kuriuose daugiau nėra vidinių stačiakampių, nuspalvinami, jeigu analizuojamoje duomenų aibėje yra vektorių, kurių parametrų reikšmės atitinka šiuos vidinius stačiakampius (2.7 pav.). Tačiau daugiau nei 9-ių matavimų duomenų atvaizdavimas gana sudėtingas. Kiekvieno parametro reikšmės turi būti išskaidytos į intervalus, kurių būtų ne daugiau kaip 4, 5. „Išoriniai“ parametrai turi skirtingą poveikį lyginant su „vidiniais“. Kyla klausimas, kuriuos parametrus pasirinkti „išoriniais“, kuriuos „vidiniais“, kuriuos žymėti *x* ašyje, kuriuos *y*, kokiais intervalus padalinti parametrus. Šis metodas gali būti naudojamas ieškant klasterių, taškų-atsiskyrėlių ir pan.

2.7 paveiksle (Peng, 2005) pateiktas dimensijų įterpimo pavyzdys irisų duomenims. Duomenų parametrai yra pateikiami tokia tvarka: taurėlapių ilgiai, taurėlapių pločiai, vainiklapių ilgiai, vainiklapių pločiai. Šitie parametrai yra vaizduojami paveiksle, kaip išoriniai horizontalūs, išoriniai vertikalūs, vidiniai horizontalūs ir vidiniai vertikalūs parametrai, atitinkamai. Kiekviena iš keturių dimensijų padalijama į penkis intervalus.



2.7 pav. *Dimensijų įterpimo metodo pavyzdys (irisų duomenų aibė)*

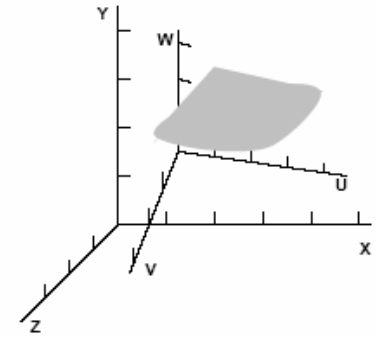
Kitas metodas, kuris rekursyviai vaizduoja koreliacijas tarp atskirų parametrų yra **fraktalų metodas** (*angl. fractal foam*) (Rabenhorst, 1994), (Hoffmann 2002). Pirmiausiai pasirenkama viena vektoriaus komponentė kaip pradinė ir pavaizduojama spalvotu skrituliu. Kitos komponentės yra pateikiamos mažesniais skrituliukais, išdėstytais aplink pagrindinį skritulį. Aplinkinių skritulių dydžiai priklauso nuo koreliacijos tarp pirmos (pradinės) komponentės ir aplinkinius skritulius atitinkančių komponenčių. Procesas kartojamas: maži skrituliukai apie kiekvieną kitą komponentę žymi koreliacijas tarp jų (2.8 pav.). Parametrus, kurie yra labai koreliuoti, atitiks didesni apskritimai. Taip pat gali būti atvaizduojamos kitos statistinės savybės, tokios kaip dispersija, asimetriškumas ir kt. Tai nėra pačių duomenų vizualizavimas, bet jų statistinių įverčių vizualizavimas.



2.8 pav. *Fraktalų metodo pavyzdys (irisų duomenų aibė)*

2.8 paveiksle yra pateiktas fraktalų metodo pavyzdys irisų duomenims: irisų taurėlapių ilgis (centre), vainiklapių ilgis (dešinėje), vainiklapių plotis (viršuje), taurėlapių plotis (apačioje).

Kai tik tapo įmanomas virtualus trimatis vaizdavimas, jis greitai buvo pritaikytas vizualiai duomenų analizei. *N-vision* sistemoje (Beshers, 1993), buvo realizuotas **„pasaulis pasaulyje“ metodas** *(angl. worlds-within-worlds)* (Feiner, 1990). Dimensijų įterpimo paradigma buvo pritaikyta interaktyviai trimatei aplinkai. Ši sistema leidžia tyrinėti *n*-mates erdves, tuo pačiu ir *n*-mačių taškų aibes. Rekursyviai įterpinėjama po tris komponentes iš išorinių į vidinius kubus. Išorinės komponentės fiksuojamos kaip konstantos, o vidinės atvaizduojamos kaip trimatės erdvės (2.9 pav.).



2.9 pav. *„Pasaulis pasaulyje“ metodo pavyzdys*

Plačiai naudojami ir kiti hierarchinio vizualizavimo metodai:

- medžio pavidalo žemėlapiai (*angl. treemaps*) (Shneiderman, 1992),
- kūgio pavidalo medžiai (*angl. cone trees*) (Robertson, 1991),
- informacijos kubas (*angl.infocub*) (Rekimoto, 1993),
- Grotelių metodas (*angl. trellis display*) (Becker, 1996),
- Cheops (Beaudoin, 1996) ir kt.

Kiti tiesioginio vizualizavimo metodai:

Taškus analizuojantys (*angl. pixel-oriented*):

a) Rekursinis vaizdų metodas (*angl. recursive pattern technique*) (Keim, 1995),

b) Skritulio nuopjovos metodas (*angl. circle segment technique*) (Ankerst, 1996),

c) Spiralės ir ašių metodai (*angl. spiral and axes techniques*) (Keim, 1994).

Grafiniai (*angl. graph-based*):

Pagrindinė diagrama (*angl. basic graph*): Tiesė, Politiesė, Kreivė (*angl. straight-line, polyline, curved-line*) (Battista, 1994).

Iškreipimo (deformavimo) (*angl. distortion*):

a) Perspektyvioji siena (*angl. perspective wall*) (Mackinlay, 1991),

b) Sukinys ir lentelės padidinamasis stiklas (*angl. pivot table and table lens*) (Rao, 1994),

c) „Žuvies akis“ (*angl. the fish eye view*) (Furnas, 1986),

d) Hiperboliniai medžiai (*angl. hyperbolic trees*) (Lamping, 1995),

e) Hiper dėžė (*angl. hyperbox*) (Alpern, 1991).

Hibridiniai (*angl. hybrid*): įvairios anksščiau minėtų metodų kombinacijos.

2.2. Projekcijos metodai

Projekcijos metodai (*angl. projection techniques, dimension reduction techniques*) *n*-matės erdvės vektorius transformuoja į *d*-matės erdvės vektorius, kai $d \le n$. Jų tikslas – pateikti daugiamačius duomenis mažesnės dimensijos erdvėje taip, kad būtų kiek galima tiksliau išlaikyta tam tikra duomenų struktūra, bei palengvinti didelės dimensijos duomenų apdorojimą bei interpretavimą.

Duomenų projekcija gali būti suformuluota, kaip atvaizdavimas ψ iš *n*-matės erdvės į *d*-matę erdvę $\psi : R^n \to R^d, d \le n$, toks, kad tam tikras kriterijus $\boldsymbol{C}$ būtų optimizuotas. Šis formulavimas panašus į funkcijos aproksimavimo uždavinį. Tačiau, skirtingai nuo funkcijos aproksimavimo, kur atvaizduojančioji funkcija apskaičiuojama iš mokymo vektorių, kurie yra įėjimo-išėjimo poros (kai išėjimai yra žinomi), projekcijos metoduose norimi išėjimai dažnai nėra žinomi. Duomenų projekcijai dydžio *d* reikšmė paprastai lygi 2 arba 3.

Tarkime, kad yra daugiamačiai duomenys, kurie išreikšti *n*-matės erdvės vektoriais $X^j = (x_1^j, x_2^j, ..., x_n^j)$, ($X^j \in R^n$), $j = 1,...,m$; čia x_i^j yra duomenų *j*-tojo vektoriaus X^j *i*-toji komponentė, atitinkanti *i*-tąjį parametrą; *n* žymi vektoriaus komponenčių skaičių, t.y. erdvės, kuriai priklauso vektorius X^j, dimensiją; *m* – analizuojamų vektorių skaičius. Tikslas – rasti vektoriaus

$X^j = (x_1^j, x_2^j, ..., x_n^j)$ transformaciją $Y^j = (y_1^j, y_2^j, ..., y_d^j)$, kai $d<n$, mažesnės dimensijos erdvėje R^d.

Egzistuoja didelis skaičius projekcijos metodų. Šie metodai skiriasi vienas nuo kito atvaizduojančios funkcijos ψ charakteristikomis, kaip ji gaunama ir koks optimizavimo kriterijus $\boldsymbol{C}$ yra naudojamas. Atvaizduojančioji (projekcijos) funkcija gali būti tiesinė arba netiesinė, ir gali būti gaunama, naudojant mokymą su mokytoju arba be mokytojo. Galimi keturi duomenų projekcijos (vizualizavimo) būdai: tiesinis mokymas be mokytojo; tiesinis mokymas su mokytoju; netiesinis mokymas be mokytojo; netiesinis mokymas su mokytoju. Mokymo su mokytoju būdai turi geresnes charakteristikas, negu mokymo be mokytojo būdai situacijose, kai turimi duomenys tokiam mokymui.

Projekcijos metodai yra skirstomi į dvi dideles grupes: *tiesiniai* ir *netiesiniai*. Tiesiniai metodai yra patrauklesni, kadangi jie reikalauja mažiau skaičiavimų, negu netiesiniai metodai. Tačiau, netiesiniai metodai yra daug veiksmingesni.

Projekcijos metoduose yra naudojamas formalus matematinis kriterijus, pagal kurį minimizuojamas projekcijos iškraipymas. Žinant visų turimų metodų detales ir charakteristikas padaromas teisingas metodo pasirinkimas.

Tiesinės projekcijos metodai skirti analizuojamų duomenų dimensijos mažinimui, atliekant duomenų tiesines transformacijas. Šie metodai yra labai efektyvūs, kai duomenys pasiskirsto po tam tikrą poerdvį. Tikslesnė duomenų struktūra išlaikoma naudojant *netiesinės projekcijos metodus*. Bet ir čia duomenų vizualizavimo iškraipymai yra neišvengiami.

Projekcijos metodai:

1) **Tiesinės projekcijos metodai**:

- Pagrindinių komponenčių analizė (*angl. principal component analysis*),
- Projekcijos paieškos metodas (*angl. projection pursuit*),
- Faktorinė analizė (*angl. factor analysis*),
- Tiesinė diskriminantinė analizė (*angl. linear discriminant analysis*),
- Nepriklausomų komponenčių analizė (*angl. independent component analysis*);

2) **Netiesinės projekcijos metodai**:

- Daugiamatės skalės (*angl. multidimensional scaling*),
- Sammono projekcija (*angl. Sammon mapping*),
- Pagrindinės kreivės (*angl. principal curves*),

- Trianguliacijos metodas (*angl. triangulation*),
- Isomap metodas (*angl. isomap*),
- Lokaliai tiesinis atvaizdavimas (*angl. locally linear embedding*).

Šiame darbe nagrinėjami dažniausiai naudojami projekcijos metodai. Yra daug bandymų apjungti kelis skirtingais principais grindžiamus vizualizavimo metodus. Tai leidžia gauti daugiau naujų žinių apie analizuojamus duomenis, lyginant su atskirų metodų taikymu.

2.2.1. Tiesinės projekcijos metodai

Tarkime, kad yra duomenų matricą $\mathbb{X}$, sudarytą iš vektorių $X^j = (x_1^j, x_2^j, ..., x_n^j) = \{x_i^j\}$, $i = 1,...,n$, $j = 1,...,m$. Tikslas yra surasti tokią tiesinę transformaciją A, kad $Y = A^T X$, $y \in R^d$, $d<n$.

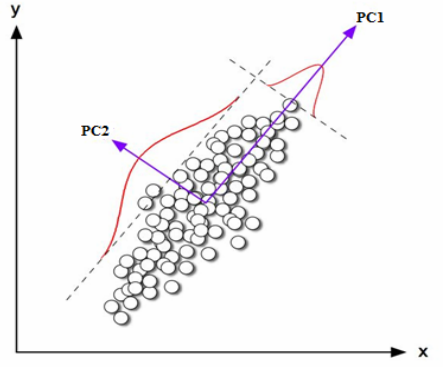
Pagrindinių komponenčių analizė (*angl. principal component analysis, PCA*) yra klasikinis statistikinis tiesinės projekcijos metodas. Šis metodas dar žinomas kaip *Karhunen-Loeve transformacija* arba *Hotelling transformacija*. Tai ortogonali tiesinė duomenų transformacija plačiai naudojama duomenų analizėje (pavyzdžiui, klasifikavimui arba atpažinimui) kaip daugiamačių duomenų dimensijos mažinimo metodas.

Naudojant pagrindinių komponenčių analizės metodą ieškoma geriausio poerdvio daugiamačių duomenų projekcijai, t.y. poerdvio kuriame išlaikyta kiek įmanoma daugiau originalios erdvės duomenų savybių bei informacijos. Metodo tikslas – rasti kryptį, kuria dispersija yra didžiausia (Jollife, 1986), (Haykin, 1999), (Taylor, 2003). Didžiausią dispersiją turinti kryptis vadinama pirmąja pagrindine komponente (PC_1). Ji eina per duomenų centrinį tašką. Taškų visumos kvadratinis vidutinis atstumas iki šios tiesės yra minimalus, t.y. ši tiesė yra kiek galima arčiau visų duomenų taškų (2.10 pav.). Antrosios pagrindinės komponentės (PC_2) ašis taip pat turi eiti per duomenų centrinį tašką ir ji turi būti nekoreliuota (ortogonali) pirmosios pagrindinės komponentės ašiai.

Esminė PCA idėja – sumažinti duomenų dimensiją kiek galima tiksliau išlaikant duomenų dispersijas. Tai padeda geriau vizualizuoti duomenis, o tuo pačiu ir palengvina duomenų suvokimą.

Pagrindinių komponenčių analizės metodą sudaro duomenų kovariacinės matricos nuosavų reikšmių λ (*angl. eigenvalue*) ir nuosavų vektorių e (*angl. eigenvector*) skaičiavimai. Kiekvienas nuosavas vektorius yra vadinamas pagrindine komponente. Pagrindinės komponentės yra lygties $Ce = \lambda e$ sprendinys *e*. Šioje lygtyje e yra vektorius-stulpelis, *C* yra duomenų kovariacinė matrica, ir λ – nuosava reikšmė, randama iš charakteringos lygties $|C - \lambda I| = 0$.

Čia I yra vienetinė matrica, kurios matmenys tokie pat, kaip matricos C. Ženklu $|.|$ apibrėžtas diskriminantas. Komponenčių kryptys yra ortogonalios, ir komponentės su didžiausiomis nuosavomis reikšmėmis turi didžiausią dispersiją. Nuosavų vektorių skaičius yra lygus kintamųjų skaičiui, t.y. n. Nuosavas vektorius, susijęs su didžiausia nuosava reikšme, turi tokią pat kryptį kaip pirmoji pagrindinė komponentė. Antroji pagrindinė komponentė atitinka antrąjį nuosavą vektorių ir t.t. Pagrindinių komponenčių skaičius parenkamas atsižvelgiant į projektinės erdvės dimensiją. Yra sukurti efektyvūs algoritmai pagrindinėms komponentėms rasti. Taip pat, gali būti taikomi ir dirbtiniai neuroniniai tinklai (4 skyrius).



2.10 pav. *Pirmoji (PC1) ir antroji (PC2) pagrindinės komponentės*

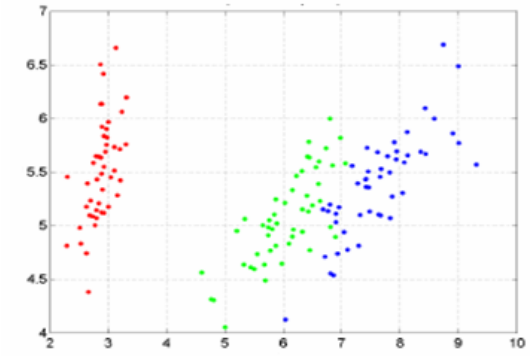
Sudarykime pagrindinių komponenčių matricą $A = \{e_i\}$, $i = 1,...,n$ iš nuosavų vektorių e kaip vektorių-eilučių. Kiekvienas šios matricos vektorius-eilutė yra ortogonalus bet kuriam kitam. Transformuokime bet kurį duomenų vektorių X pagal formulę $Y = A(X - \bar{X})$, čia $X = (x_1^j, x_2^j, ..., x_n^j)^T$, $\bar{X} = (\bar{x}_1, \bar{x}_2, ..., \bar{x}_n)^T$, $\bar{x}_i = \frac{1}{m}\sum_{j=1}^{m} x_i^j, i = 1,...,m$, $A = (e_1, e_2, ..., e_n)^T$.

Daugiamačių duomenų transformacijai galima naudoti ne visus nuosavus vektorius, o tik pirmuosius. Tegu matrica A_d sudaryta iš d pirmųjų nuosavų vektorių. Tai reiškia, kad pradinį duomenų vektorių projektuojame į koordinačių sistemą, turinčią d dimensijų, vizualizavime naudojama $d = 2$ arba $d = 3$.

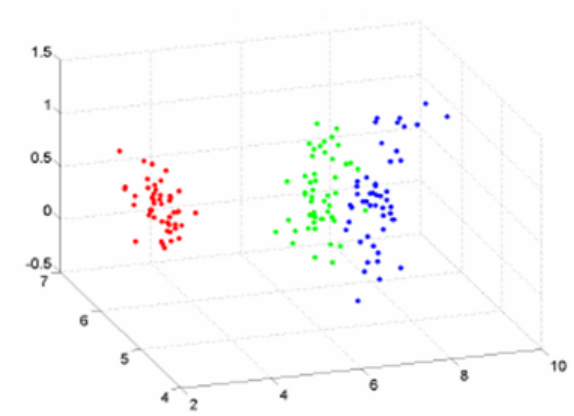
2.11 ir 2.12 paveiksluose pateiktas PCA metodo pavyzdys irisų duomenims, kai duomenys projektuojami į dvimatę ir trimatę erdves (atitinkamai dvi ir trys pagrindinių komponenčių ašys).

Naudojant pagrindinių komponenčių metodą, negalima gauti išsamios duomenų charakteristikos, nes nagrinėjamos tik tiesinės priklausomybės. PCA yra netinkamas metodas daugiamačių duomenų dimensijos mažinimui, jeigu tarp tų duomenų egzistuoja stiprūs netiesiniai sąryšiai. Didelę įtaką PCA metodo

rezultatui daro taškai-atsiskyrėliai, nes jie yra izoliuoti ir dirbtinai padidina dispersijos reikšmę.



2.11 pav. *PCA pavyzdys (irisų duomenų aibės projekcija dvimatėje erdvėje)*



2.12 pav. *PCA pavyzdys (irisų duomenų aibės projekcija trimatėje erdvėje)*

Standartiniai PCA metodai netinka duomenims, sudarytiems iš daug parametrų, nes daugiamatės erdvės duomenims kovariacinės matricos dydis yra labai didelis. *d*-matės erdvės duomenims, tenka skaičiuoti $d \times d$ dydžio matricą, todėl PCA metodas netinka turint kelių šimtų parametrų daugiamačius duomenis.

Projekcijos paieška. Analizuojant *n*-mačius taškus, norima pamatyti jų išsidėstymą *n*-matėje erdvėje. Tačiau, kai $n > 3$, tiesiogiai pamatyti šiuos taškus neįmanoma. Tačiau yra galimybė projektuoti juos į dvimatę ar trimatę erdvę. Skirtingos projekcijos duoda skirtingus rezultatus, be to, ne visada pavyksta išlaikyti duomenų struktūrą. Kaip rasti tinkamą projekciją padeda projekcijos

paieškos metodas (Huber, 1985), (Jones, 1987). Projekcijos paieškos metodas (*angl. projection pursuit, PP*) buvo pasiūlytas ir tyrinėtas eksperimentiškai J. B. Kruskalo (Kruskal, 1972). Šis metodas leidžia surasti „įdomių“ galimų daugiamačių duomenų projekcijų. Projekcija yra laikoma „įdomia“, jeigu ji vaizduoja neatsitiktinius arba ne normaliai pasiskirsčiusių taškų debesis.

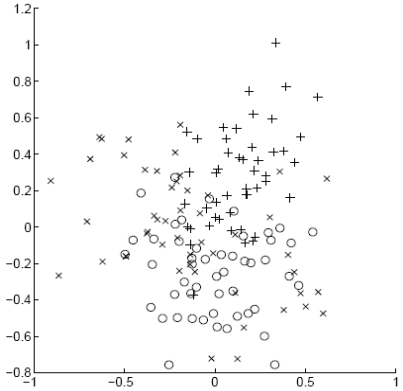
Metodo tikslas – rasti tokias tiesines komponenčių kombinacijas (dažniausiais dvimates ar trimates), kad transformuoti duomenys išlaikytų pradinių duomenų struktūrą. Tarkime, kad yra duomenų matrica $\mathbb{X}$, susidedančią iš *n*-mačių vektorių. Erdvės R^n projekciją į R^2 galima išreikšti taip: $(z_1, z_2) = (\mathbb{X}a^T, \mathbb{X}b^T)$. Čia *a* ir *b* yra *n*-mačiai vektoriai-eilutės; a^T, b^T – transponuoti *a* ir *b* vektoriai, t.y. vektoriai-stulpeliai; z_1, z_2 – gauti po projekcijos *d*-mačiai vektoriai-stulpeliai. Kyla klausimas, kaip parinkti vektorius *a* ir *b*. Visų pirmą reikia nuspręsti, kurią duomenų savybę norime stebėti. Tada reikia nustatyti matą, išreiškiantį šią savybę vektoriais (z_1, z_2). Pavadinkime šį matą $I(z_1, z_2)$. Kartais jis vadinamas *projekcijos indeksu* arba *indekso funkcija* (*angl. projection, index function*). Tarkime, norime nustatyti duomenų klasterius. Dažniausiai naudojamas statistikinis klasterizavimo matas yra vidutinis atstumas iki artimiausio kaimyno (*angl. mean nearest neighbour distance*). Kuo mažesnė šio mato reikšmė viename klasteryje, tuo duomenys yra geriau suklasterizuoti. Taigi, tegu $I(z_1, z_2)$ yra vidutinis atstumas iki artimiausio kaimyno. Užrašas $I(z_1, z_2)$ gali būti pakeistas į $I(\mathbb{X}a', \mathbb{X}b')$. Projekcijos parinkimo uždavinys susiveda į optimizavimo uždavinį: reikia parinkti tokius vektorius *a* ir *b*, kad funkcijos $I(\mathbb{X}a', \mathbb{X}b')$ reikšmė būtų mažiausia. Iš esmės tai ir yra projekcijos paieškos procesas.

Faktorinė analizė. Faktorinėje analizėje (*angl. factor analysis*) (Harman, 1967), (Mardia, 1995) tiriamos tiesinės priklausomybės tarp parametrų (kintamųjų). Ji taikoma kintamųjų skaičiaus mažinimui, daugiamačių duomenų struktūros nustatymui, kintamųjų klasifikavimui ir grupavimui. Faktorinės analizės metodo prielaida yra ta, kad nagrinėjami parametrai priklauso nuo tam tikrų paslėptų faktorių. Metodo tikslas atskleisti tokius ryšius ir daugiamačių duomenų dimensijos mažinimui panaudoti tam tikrą faktoriaus modelį.

Spėjama, kad kiekvienas faktorius įtakoja daug stebimų parametrų. Jis vadinamas *bendru faktoriumi* (*angl. common factor*). Stebimi parametrai yra tiesinė kombinacija bendrų paslėptų faktorių; kiekvieno faktoriaus koeficientas vadinamas *svoriu* (*angl. loading*). Kiekvienas parametras turi dar vieną komponentę, *specifinį faktorių* (*angl. specific factor*), kuri aprašo to parametro nepriklausomą kintamumą (*angl. independent variability*). Specifiniai faktoriai gali būti interpretuojami kaip vektoriaus triukšmas arba paklaida. Faktorių skaičius *k* nustatomas pagal tam tikrus kriterijus.

Vektorius $X = (x_1,...,x_n) \in R^n$ atitinka *k*-faktoriaus modelį, jeigu jis gali būti išreikštas tokia forma (Mardia, 1995): $X = \Delta f + U$. $\Delta = \{\lambda_{ij}, i = 1,...,n, j = 1,...,k\}$ yra konstantų matrica (svoriai), $f = (f_1,...,f_k)$ yra bendrų paslėptų faktorių vektorius ir $U = (u_1,...,u_n)$ yra specifinių faktorių vektorius. Vektorius $X \in R^n$ gali būti užrašytas taip $x_i = \sum_{j=1}^{k} \lambda_{ij} f_i + u_i$, $i = 1,\ldots,n$.

Tiesinė diskriminantinė analizė (*angl. Linear discriminant analysis, LDA*) arba **Fišerio diskriminantinė analizė** (*angl. Fisher discriminant analysis*). Tiesinės diskriminantinės analizės metodas (Fukunaga, 1990), (Duda, 1973), (Duda, 2000) transformuoja daugiamatės erdvės duomenis į mažesnės dimensijos erdvę taip, kad tam tikros klasės atskiriamumo kriterijus būtų optimalus. LDA metodas ieško tokių krypčių, kuriomis klasės yra geriausiai atskiriamos. PCA metodas nekreipia dėmesio į klasės struktūrą. 2.13 paveiksle pateiktas LDA metodu gautas daugiamačių taškų išsidėstymas dvimatėje plokštumoje vizualizuojant irisų duomenis. LDA metodui nepavyko gerai atskirti skirtingų klasių vektorius.



2.13 pav. *LDA pavyzdys (vizualizuoti irisų duomenys)*

Kartais daugiamačių duomenų dimensijos mažinimui naudojamas **Nepriklausomų komponenčių analizės** (*angl. independent component analysis, ICA*) metodas. Nepriklausomų komponenčių analizė, tai metodas, kuris transformuoja pagrindines komponentes į statistiškai nepriklausomas komponentes. ICA metodas gali būti taikomas įvairių problemų sprendimui: vaizdų analizėje, duomenų analizėje, signalų apdorojimui, požymių išskyrimui. Plačiai taikomas statistikoje, neuroniniuose skaičiavimuose. Detalesnė informacija apie ICA yra pateikta literatūroje: (Comon, 1994), (Hyvarinen, 2001), (Roberts, 2000).

2.2.2. Netiesinės projekcijos metodai

Tarkime, kad kiekvieną *n*-matį vektorių $X^i \in R^n$ atitinka mažesnės dimensijos vektorius $Y^i \in R^d$, ($d < n$). Pažymėkime atstumą tarp vektorių X^i ir X^j – d_{ij}^*, o atstumą tarp vektorių Y^i ir Y^j – d_{ij}, $i, j = 1,...,m$. Norint surasti daugiamatės erdvės vektorių X^i projekcijas žemesnio matavimo erdvėje Y^i yra skaičiuojami atstumai tarp visų vektorių atitinkamose erdvėse.

Ieškoma tokia *d*-matės erdvės vektorių $Y^i \in R^d$ projekcija, kad atstumai d_{ij} tarp projektinės erdvės vektorių būtų kiek galima artimesni atstumams d_{ij}^* tarp vektorių pradinėje erdvėje. Dažniausiai atstumams d_{ij} ir d_{ij}^* skaičiuoti naudojami Euklido atstumai. Tačiau d_{ij} nebūtinai turi tenkinti trikampio nelygybės sąlygą $d_{ik} \le d_{ij} + d_{jk}$ ir simetriškumo sąlygą $d_{ij} = d_{ji}$; todėl bendru atveju (kai šios sąlygos nėra tenkinamos) naudojama nepanašumo (*angl. dissimilarity*) sąvoka, o ne atstumas.

Kadangi dažnai neįmanoma surasti tokios projekcijos, kurioje $d_{ij} = d_{ij}^*$ visiems *i* ir *j*, yra apibrėžiamas kokybės kriterijus *J* skirtingų projekcijų įvertinimui. Priklausomai nuo kriterijaus parinkimo, galutinėje projekcijoje tam tikri atstumai tarp taškų (maži ar dideli) bus išlaikyti geriau negu kiti. Yra pasiūlytos trys funkcijos daugiamačių duomenų projekcijos paklaidos įvertinimui (kartu nurodyti ir funkcijų gradientai) (Duda, 2000):

$$S_1 = \frac{1}{\sum\limits_{i<j} d_{ij}^{*2}} \sum_{i<j} (d_{ij} - d_{ij}^*)^2 \ , \ \nabla_{yk} S_1 = \frac{2}{\sum\limits_{i<j} d_{ij}^{*2}} \sum_{j \ne k} \left((d_{kj} - d_{kj}^*) \frac{y_k - y_j}{d_{kj}} \right), \tag{2.1}$$

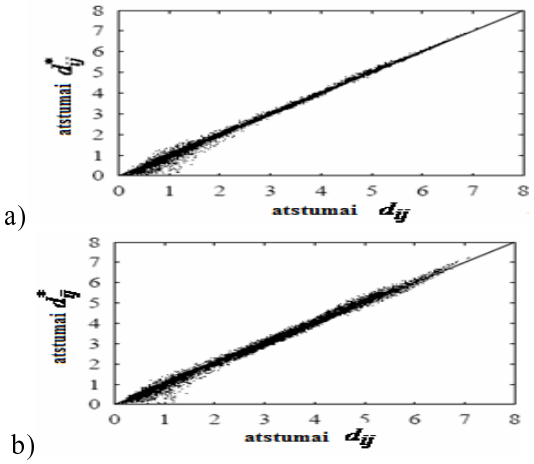
$$S_2 = \sum_{i<j} \left(\frac{d_{ij} - d_{ij}^*}{d_{ij}^*} \right)^2 \ , \ \nabla_{yk} S_2 = 2 \sum_{j \ne k} \left(\frac{d_{kj} - d_{kj}^*}{d_{kj}^{*2}} \cdot \frac{y_k - y_j}{d_{kj}} \right), \tag{2.2}$$

$$S_3 = \frac{1}{\sum\limits_{i<j} d_{ij}^*} \sum_{i<j} \frac{(d_{ij} - d_{ij}^*)^2}{d_{ij}^*} \ , \ \nabla_{yk} S_3 = \frac{2}{\sum\limits_{i<j} d_{ij}^*} \sum_{j \ne k} \left(\frac{d_{kj} - d_{kj}^*}{d_{kj}^*} \cdot \frac{y_k - y_j}{d_{kj}} \right). \tag{2.3}$$

Kriterijus S_1 (2.1) formulėje akcentuoja didžiausias paklaidas, nepriklausomai nuo to ar atstumai d_{ij}^* dideli ar maži. Kriterijus S_2 (2.2) formulėje akcentuoja didžiausias santykines paklaidas, nepriklausomai ar paklaidos reikšmė maža ar didelė. Kriterijus S_3 (2.3) formulėje numato

kompromisą tarp pirmų dviejų kriterijų, pabrėždamas paklaidų sandaugas ir santykines paklaidas.

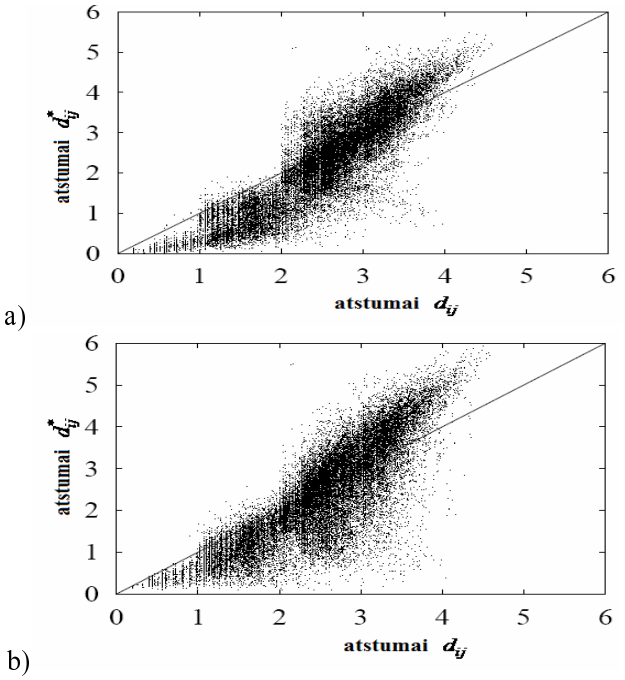
Atstumų d_{ij}^* ir d_{ij} taškinė diagrama, vadinama Šepardo diagrama (*angl. Shepard diagram*), rodo, kokie atstumai geriau atspindi duotus nepanašumus. 2.14 ir 2.15 paveiksluose pavaizduota Šepardo diagrama, gauta minimizuojant kriterijaus funkcijas S_1 ir S_3 irisų ir vėžio duomenims (krūties vėžio duomenų aibę (Blake, 1998), sudaro 699 9-mačiai vektoriai). Kuo arčiau tiesės $y=x$ yra sukoncentruoti taškai, tuo geriau išlaikomi atstumai. Minimizuojant kriterijaus funkciją S_1, geriausiai išlaikomi didesni atstumai, negu mažesni. Minimizuojant funkciją S_3, didesni atstumai išlaikomi blogiau, o mažesni geriau. Tai matosi iš 2.14 ir 2.15 paveikslų. Iš 2.14(a) paveikslo matyti, kad irisų duomenims diagramos taškai labiau sukoncentruoti arti tiesės $y=x$ dešiniajame viršutiniame kampe. 2.14(b) paveiksle toje vietoje taškai labiau išsibarstę aplink tiesės. Vėžio duomenims taškų debesis labiau suspaustas tiesės kryptimi, kai naudojama kriterijaus funkcija S_1 (2.15(a) pav.). Naudojant kriterijų S_3 labiau suspausta tų taškų debesio kairioji pusė (2.15(b) pav.), o dešiniajame viršutiniame kampe taškai labiau išsibarstę negu 2.15(a) paveiksle. Kaip matosi iš 2.14 paveikslo, irisų duomenų aibės atveju, negalima teigti, kad mažesni atstumai išlaikomi geriau naudojant kriterijaus funkciją S_3, negu S_1, nes diagramos taškų išsidėstymas kairiajame apatiniame kampe abiems atvejais yra vienodas. Kriterijaus funkcijos parinkimas gali įtakoti duomenų atvaizdavimo rezultatą. Tai priklauso nuo pradinių duomenų specifikos ir atstumų tarp taškų. 2.16 ir 2.17 paveiksluose pateikti analizuojamų duomenų projekcijos rezultatai, minimizuojant S_1 ir S_3 funkcijas (projekcijos gautos, naudojant daugiamačių skalių metodą).



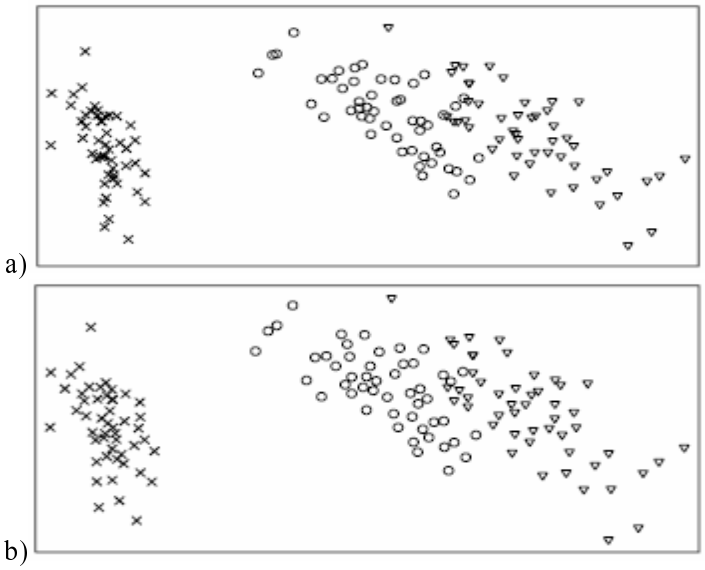
a)

b)

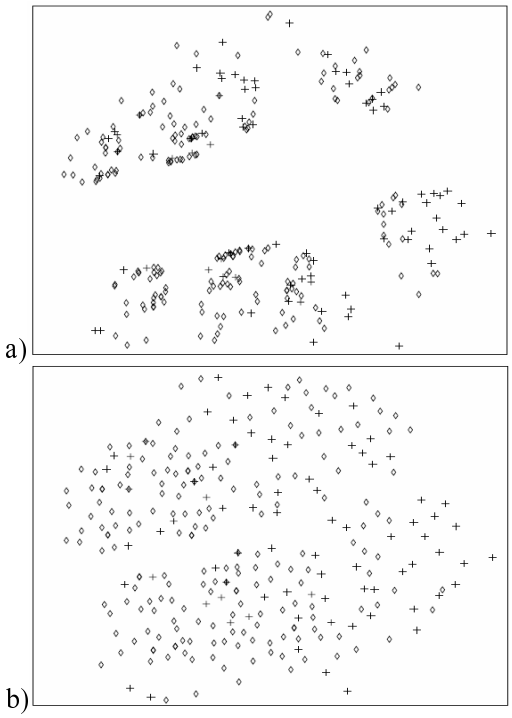
2.14 pav. *Šepardo diagrama irisų duomenims, kai naudojama: a) kriterijaus funkcija S_1, b) kriterijaus funkcija S_3*



2.15 pav. *Šepardo diagrama vėžio duomenims, kai naudojama: a) kriterijaus funkcija S_1, b) kriterijaus funkcija S_3*



2.16 pav. *Irisų duomenų atvaizdavimas, kai naudojama: a) kriterijaus funkcija S_1 (S_1=0,0011); b) kriterijaus funkcija S_3 (S_3=0,0132)*



2.17 pav. *Vėžio duomenų atvaizdavimas, kai naudojama: a) kriterijaus funkcija S_1 (S_1=0,061); b) kriterijaus funkcija S_3 (S_3=0,084)*

Daugiamačių skalių metodas

Daugiamačių skalių metodas (*angl. multidimensional scaling, MDS*) – tai grupė metodų, plačiai naudojamų daugiamačių duomenų analizėje įvairiose šakose, ypač ekonomikoje, socialiniuose moksluose ir kt. *n*-mačiai vektoriai projektuojami į mažesnės dimensijos erdvę (dažniausiai į R^2 arba R^3) siekiant išlaikyti atstumus (nepanašumus) tarp analizuojamos aibės objektų (Kaski, 1997), (Borg, 1997), (Kruskal, 1984). Analizės rezultate gautuose dvimačiuose grafikuose tie objektai, kurie yra panašūs, yra pavaizduojami arčiau vieni kitų, o mažiau panašūs – toliau vieni nuo kitų. Minimizavimo problema yra sprendžiama taip, kad atstumai tarp taškų mažesnės dimensijos erdvėje atitiktų duotus nepanašumus kaip įmanoma geriau. Atvaizdavimas paprastai yra netiesinis, ir padeda atskleisti bendrą analizuojamų duomenų struktūrą.

Pradiniai duomenys, kurie analizuojami šiuo metodu, turi būti kvadratinė, simetrinė matrica, susidedanti iš ryšių tarp analizuojamų duomenų aibės elementų. Tai gali būti atstumų arba nepanašumų matrica. Ryšiais tarp aibės elementų gali būti ir Euklido atstumai. Tačiau, bendru atveju, tai nebūtinai turi būti atstumai griežtai matematine prasme.

Vienas MDS pavyzdys galėtų būti toks: tarkime turime matricą, sudarytą iš atstumų tarp pagrindinių šalies miestų; MDS analizės rezultate gautume miestų išdėstymą žemėlapyje, t.y. dvimatėje plokštumoje (Leeuw, 2003). Kitas MDS matricos pavyzdys yra koreliacijų tarp duomenų parametrų matrica. Jei tie duomenys traktuojami kaip panašumai, MDS algoritmu stipriai koreliuoti parametrai atvaizduojami arti vieni kitų, silpnai koreliuoti – toliau vieni nuo kitų.

Vienas MDS tikslų yra rasti optimalią daugiamačių duomenų konfigūraciją dvimatėje erdvėje. Yra daugybė skirtingų MDS variantų su skirtingomis paklaidų funkcijomis (STRESS) ir jas optimizuojančiais algoritmais (Borg, 1997). Pagal analizuojamus duomenis MDS algoritmai gali būti skirstomi į *metrinius* (*angl. metric*) ir *nemetrinius* (*angl. non-metric*). Pirmasis MDS algoritmas metriniams duomenims buvo pasiūlytas W.S. Torgensono 1952 metais (Torgenson, 1952), vėliau MDS algoritmai buvo taikyti ir nemetriniams duomenis (Shepard, 1962a), (Shepard, 1962b).

Metriniai MDS algoritmai (Taylor, 2003) naudojami tada, kai įmanoma rasti Euklido atstumus tarp analizuojamų duomenų elementų, t.y. analizuojami metriniai duomenys. Pagrindinis šių algoritmų tikslas – pavaizduoti daugiamačius taškus dvimatėje erdvėje taip, kad atstumai tarp dvimačių taškų būtų kiek galima artimesni atstumams tarp atitinkamų daugiamačių taškų. Tam minimizuojama tam tikra paklaidos funkcija.

Tarkime kiekvieną *n*-matį vektorių $X^i \in R^n$ atitinka mažesnės dimensijos vektorius $Y^i \in R^d$, ($d < n$). Pažymėkime atstumą tarp vektorių X^i ir X^j – d_{ij}^*, o atstumą tarp vektorių Y^i ir Y^j – d_{ij}, $i, j = 1,...,m$. Metrinis MDS algoritmas bando priartinti atstumus d_{ij} prie atstumų d_{ij}^*. Jei naudojama kvadratinė paklaidos funkcija, tai minimizuojama tikslo funkcija E_{MDS} gali būti užrašyta taip:

$$E_{MDS} = \sum_{\substack{i,j=1 \\ i<j}}^{m} w_{ij}(d_{ij}^* - d_{ij})^2 . \qquad (2.4)$$

Paklaidos funkcija E_{MDS} dar vadinama STRESS funkcija. Dažnai naudojami tokie svoriai:

$$w_{ij} = \frac{1}{\sum_{\substack{k,l=1 \\ k<l}}^{m} (d_{kl}^*)^2}; \quad w_{ij} = \frac{1}{d_{ij}^* \sum_{\substack{k,l=1 \\ k<l}}^{m} d_{kl}^*}; \quad w_{ij} = \frac{1}{m d_{ij}^*} .$$

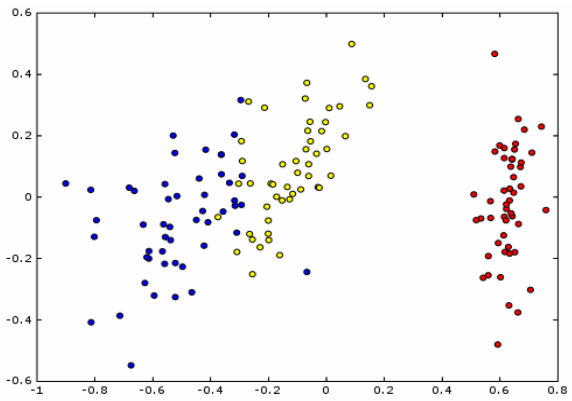
Taip pat gali būti naudojama kiek kitokios išraiškos funkcija, vadinama SSTRESS:

$$E_{MDS} = \sum_{\substack{i,j=1 \\ i<j}}^{m} w_{ij}((d_{ij}^{*})^2 - d_{ij}^2)^2 .$$

Vienas iš paprasčiausių šių funkcijų minimizavimo būdų – gradientinis nusileidimas. Pradėjus nuo atsitiktinės pradinių dvimačių taškų konfigūracijos, iteraciniame procese dvimačių vektorių $Y^i \in R^2$ koordinatės y_k^i, $i = 1,...,m$, $k = 1, 2$, keičiamos pagal formulę $y_k^i(m'+1) = y_k^i(m') - \eta \frac{\partial E_{MDS}(m')}{\partial y_k^i(m')}$. Čia m' yra iteracijos numeris, o η – parametras, įtakojantis optimizavimo žingsnį.

Tačiau galimi ir kiti optimizavimo būdai, tokie kaip SMACOF (*angl. scaling by majorization a complicated function*) algoritmas, pagrįstas tikslo funkcijos mažorizavimu (Borg, 1997), jungtinių gradientų metodas, kvazi-Niutono metodas, deterministinis atkaitinimo modeliavimo algoritmas (*angl. simulated annealing*) (Klock, 1999), genetinio algoritmo ir lokalaus nusileidimo metodų kombinacijos (Mathar, 1993), (Podlipskytė, 2003). Funkcijų STRESS ir SSTRESS minimizavimo uždaviniai yra sudėtingi dėl kriterijų daugiaekstremalumo (Žilinskas, 2003). Darbe (Podlipskytė, 2003) yra pateikta kriterijų STRESS ir SSTRESS galimų optimizavimo strategijų apžvalga bei jų tyrimai.

2.18 paveiksle pateiktas MDS rezultate gautas daugiamačių taškų išsidėstymas dvimatėje plokštumoje vizualizuojant irisų duomenis.



2.18 pav. *MDS pavyzdys (vizualizuoti irisų duomenys)*

Kartais, analizuojant objektus, yra prasmingos ne atstumų skaitinės reikšmės, o atstumų tarp objektų eilės numeriai. Tada tikslinga naudoti *nemetrinius* MDS algoritmus, kuriuose objektų nepanašumai nėra atstumai. Nepanašumą tarp *i*-tojo ir *j*-tojo objektų apibrėžkime realiu skaičiumi δ_{ij}. Kartais šis matas nėra tinkamas Euklido erdvei, todėl jis transformuojamas funkcija *f*:

$f(\delta_{ij})$. Nemetriniuose MDS algoritmuose dažniausiai minimizuojama (2.5) paklaidos (STRESS) funkcija:

$$E_{MDS} = \sum_{\substack{i,j=1 \\ i<j}}^{m} w_{ik}(f(\delta_{kl}) - d_{kl})^2 . \qquad (2.5)$$

MDS algoritmo trūkumai: MDS algoritmo tikslo funkcijos optimizavimas reikalauja palyginimų tarp visų vektorių porų; MDS algoritmai yra neefektyvus, dirbant su didelės apimties duomenų aibėmis; MDS negali vizualizuoti naujus taškus tol, kol nebus perskaičiuoti visi analizuojami vektoriai.

Sammono algoritmas

Sammono projekcija, kartais dar vadinama **Sammono netiesinė projekcija** (Sammon, 1969), yra netiesinis daugelio kintamųjų objektų atvaizdavimo žemesnio matavimo erdvėje metodas. Tai vienas iš gana plačiai naudojamų daugiamačių skalių (MDS) grupės metodų (Bezdek, 1995). Dažniausiai nagrinėjami atvejai, kai projektinės erdvės, į kurią atvaizduojami *n*-mačiai vektoriai, dimensija yra 2 arba 3. Sammono projekcijos tikslas yra minimizuoti skirtumus tarp atstumų tarp taškų pradinėje erdvėje ir atstumų tarp atitinkamų taškų projektinėje erdvėje. Sammono projekcija negali greitai apdoroti naujus taškus be papildomų skaičiavimų. Visi atstumai tarp taškų turi būti skaičiuojami iš naujo, naudojant visą turimą duomenų aibę. Todėl tai sudaro sunkumų, kai duomenys pastoviai atnaujinami arba papildomi naujais taškais, ir kai duomenų aibės yra labai didelės.

Sakykime, kad yra daugiamačiai vektoriai $X^i = (x_1^i, x_2^i, ..., x_n^i)$, $i = 1,...,m$, priklausantys erdvei R^n. Sprendžiamas uždavinys – šiuos *n*-mačius vektorius $X^i = (x_1^i, x_2^i, ..., x_n^i)$, $i = 1,...,m$, atvaizduoti (gauti projekciją) plokštumoje R^2. Juos atitiks dvimačiai vektoriai $Y^1, Y^2, ..., Y^m \in R^2$. Čia $Y^i = (y_1^i, y_2^i)$, $i = 1,...,m$. Pažymėkime d_{ij}^* atstumą tarp daugiamačių vektorių X^i ir X^j, d_{ij} – atstumą tarp vektorius X^i ir X^j atitinkančių dvimačių vektorių Y^i ir Y^j ($i, j = 1,..,m$). d_{ij}^* ir d_{ij} dažniausiai yra Euklido atstumai, bet gali būti naudojami ir kiti. Sammono algoritmas minimizuoja projekcijos paklaidą:

$$E_S = \frac{1}{\sum_{\substack{i,j=1 \\ i<j}}^{m} d_{ij}^*} \sum_{\substack{i,j=1 \\ i<j}}^{m} \frac{(d_{ij}^* - d_{ij})^2}{d_{ij}^*}.$$

Funkcija E_S sutampa su funkcija $E_{MDS} = \sum_{\substack{i,j=1 \\ i<j}}^{m} w_{ij}(d_{ij}^* - d_{ij})^2$, kai $w_{ij} = \frac{1}{d_{ij}^* \sum_{\substack{k,l=1 \\ k<l}}^{m} d_{kl}^*}$.

Sammono paklaida (*angl. Sammon's stress, Sammon's error*) E_S – tai matas, kuris parodo, kaip tiksliai išlaikomi atstumai tarp vektorių pereinant iš didesnio matavimo erdvės į mažesnio matavimo erdvę. Sammono paklaidą labiau įtakoja atstumai tarp artimiausių taškų, negu tarp tolimesnių. Pagrindinis uždavinys minimizuoti šią paklaidos funkciją E_S. Tam gali būti naudojami įvairūs optimizavimo metodai, nurodyti aprašant daugiamačių skalių metodą. J. W. Sammon darbe (Sammon, 1969) pasiūlė vieną funkcijos E_S minimizavimo strategiją, kurią kai kurie tyrėjai vadina pseudo-Niutono, kiti – greičiausio nusileidimo (*angl. steepest-descent*) minimizavimu (Kohonen, 2001). Jis konverguoja greičiau, negu paprastas gradientinis metodas. Bet gali būti naudojamos ir kitos paprastesnės strategijos. Darbe (Press, 1992) nustatyta, kad žymiai efektyvesnis metodas yra jungtinių gradientų metodas (*angl. conjugate-gradient*).

Pseudo-Neutono (greičiausio nusileidimo) metode dvimačių vektorių $Y^i \in R^2$ koordinatės y_k^i, $i = 1,...,m$, $k = 1, 2$, randamos naudojantis (2.6) iteracine formule:

$$y_k^i(m'+1) = y_k^i(m') - \alpha \frac{\frac{\partial E_S(m')}{\partial y_k^i(m')}}{\left|\frac{\partial^2 E_S(m')}{\partial (y_k^i)^2(m')}\right|} \tag{2.6}$$

Dalinės išvestinės randamos pagal (2.7)–(2.9) formules:

$$\frac{\partial E_S}{\partial y_k^i} = -\frac{2}{c} \sum_{\substack{j=1 \\ i \neq j}}^{m} \left(\frac{d_{ij}^* - d_{ij}}{d_{ij}^* d_{ij}} \right) \left(y_k^i - y_k^j \right) \tag{2.7}$$

$$\frac{\partial^2 E_S}{\partial (y_k^i)^2} = -\frac{2}{c} \sum_{\substack{j=1 \\ i \neq j}}^{m} \frac{1}{d_{ij}^* d_{ij}} \left[\left(d_{ij}^* - d_{ij} \right) - \frac{\left(y_k^i - y_k^j \right)^2}{d_{ij}} \left(1 + \frac{d_{ij}^* - d_{ij}}{d_{ij}} \right) \right] \tag{2.8}$$

$$c = \sum_{\substack{i,j=1 \\ i<j}}^{m} d_{ij}^* \tag{2.9}$$

Čia m' yra iteracijos numeris, o α yra parametras, įtakojantis optimizavimo žingsnį. J. W. Sammon jį vadino „magiškuoju faktoriumi“. Šis pavadinimas tradiciškai naudojamas ir kituose darbuose (Kohonen, 2001), (Konig, 2000). Vienoje iteracijoje perskaičiuojami m vektorių $Y^i \in R^2$, $i = 1,...,m$, koordinatės. Gauta projekcijos E_S paklaida priklauso ir nuo parametro α ir nuo vektorių $Y^1, Y^2,..., Y^m \in R^2$ pradinių reikšmių parinkimo. Eksperimentiškai nustatyta, kad mažiausia paklaida gaunama, kai $\alpha \in [0,3;0,4]$ (Sammon, 1969), (Kohonen, 2001). Tačiau negalima teigti, kad tas diapazonas yra optimalus visais atvejais.

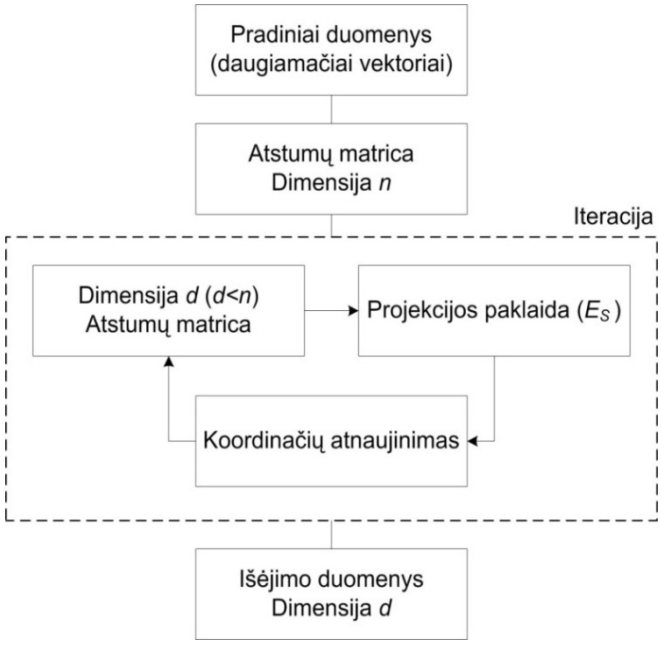
Apibendrinta Sammono algoritmo schema yra tokia:

1. Skaičiuojami atstumai tarp analizuojamų vektorių pradinėje erdvėje;
2. Atsitiktinai parenkami projektinės erdvės vektoriai;
3. Skaičiuojama projekcijos (Sammono) paklaida E_S;
4. Atnaujinamos projektinės erdvės koordinatės pagal (2.6);
5. Jeigu Sammono paklaidos reikšmė mažesnė už pasirinktą slenkstį arba iteracijų skaičius viršija nustatytąjį, tuomet algoritmas sustabdomas, priešingu atveju vėl pradedam nuo 3 žingsnio.

Sammono projekcijos struktūrinė schema pavaizduota 2.19 paveiksle.

Nors šiuo metu yra ir kitų paklaidos E_S optimizavimo metodų, J. W. Sammon pasiūlytas variantas sėkmingai naudojamas darbuose (Dzemyda, 2005), (Kohonen, 2001), (Kaski, 1997), (Konig, 2000).

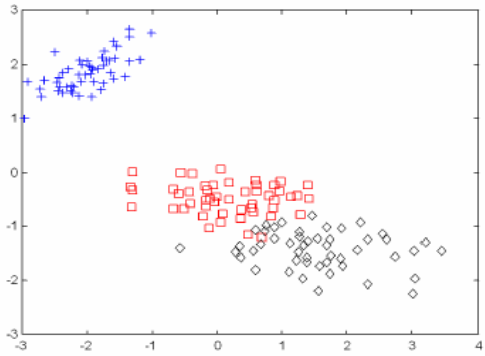
Darbe (Montvilas, 2003a) pasiūlytas nuoseklaus atvaizdavimo metodas daugiamačiams duomenims stebėti realiu laiku. Tokiu atvaizdavimu siekiamas tikslas – išlaikyti vidinę atstumų tarp duomenų vektorių struktūrą po jų atvaizdavimo. Metodas skiriasi nuo Sammono vienalaikio daugiamačių duomenų atvaizdavimo į plokštumą metodo tuo, kad po pradiniame etape vienu metu atvaizduotų keleto duomenų vektorių vėliau galima dirbti nuosekliai realiu laiku. Metodas taip pat tirtas ir darbuose (Montvilas, 2003b), (Montvilas, 2006). Tačiau metodas gali ne visada teisingai atvaizduoti naujus taškus. Pradžioje dalis analizuojamų duomenų aibės taškų yra vaizduojama klasikiniu Sammono metodu, o likę arba atsiradę nauji taškai dėstomi naudojant trianguliacijos metodą. To pasiekoje ganėtinai teisingai gali būti atvaizduojami tik tie taškai, kurie yra panašūs į pradinės aibės taškus. Panašus metodas pasiūlytas darbe (Pekalska, 1999).



2.19 pav. *Sammono projekcijos struktūrinė schema*

2.20 paveiksle pateiktas daugiamačių duomenų projekcijos, taikant Sammono algoritmą, pavyzdys. Analizuoti irisų duomenys.

Vietoj Euklido atstumų, skaičiuojant Sammono paklaidą, gali būti naudojami ir kiti atstumo matai (Yang, 2004).

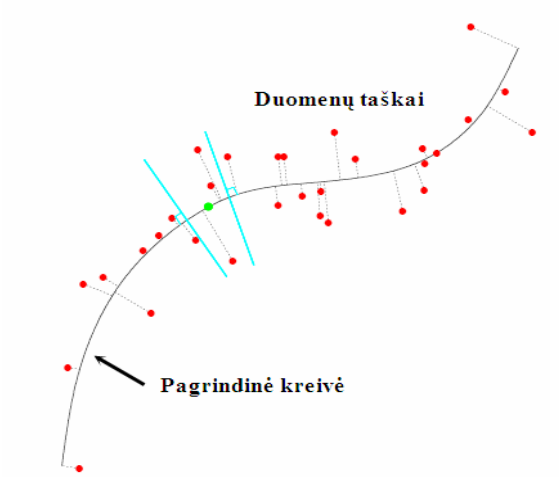


2.20 pav. *Sammono projekcijos pavyzdys (vizualizuoti irisų duomenys)*

Pagrindinės kreivės

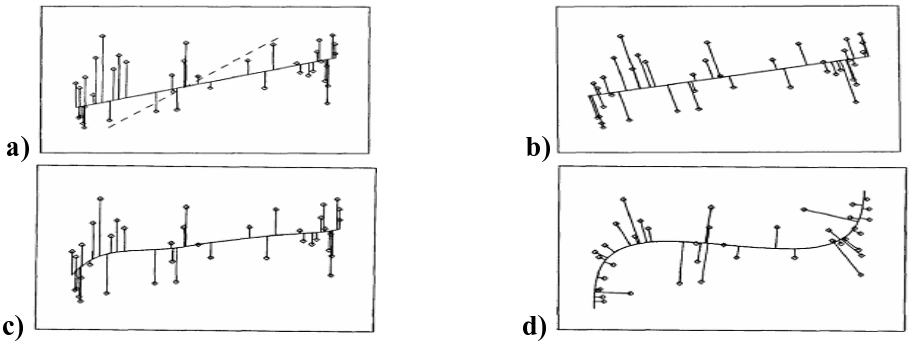
Pagrindinės kreivės *(angl. principal curves)* yra pagrindinių komponenčių netiesinis apibendrinimas, praplečiantis pagrindinių komponenčių analizę.

Pagrindinė komponentė yra tiesinė pagrindinė kreivė. Pagrindinės kreivės kiekvienas taškas yra visų duomenų, kurie projektuojami į jį, vidurkis (Hastie, 1988). Kitaip sakant, pagrindinė kreivė eina minimizuodama atstumus nuo kreivės iki visų duomenų taškų (2.21 pav.).



2.21 pav. *Pagrindinės kreivės pavyzdys*

Darbe (Hastie, 1988) yra pateikti pagrindinių kreivių privalumai lyginant su pagrindinėmis komponentėmis, tiesinės regresijos tiesėmis, glotniomis regresijos kreivėmis (2.22 pav.). Galima daryti išvadą, kad pagrindinė kreivė tiksliausiai aproksimuoja duomenų taškus (2.22(d) pav.). Darbuose (Mulier, 1995), (Ritter, 1992) pateikta pagrindinių kreivių ryšys su saviorganizuojančiais neuroniniais tinklais.



2.22 pav. *Duomenų aproksimavimas įvairiomis linijomis: a) tiesinės regresijos tiese, b) pagrindine komponente, c) glotnia regresijos kreive, d) pagrindine kreive*

Trianguliacijos metodas

Darbe (Lee, 1977) pateiktas nuoseklaus daugiamačių taškų atvaizdavimo į plokštumą metodas naudojant trianguliacijos (*angl. triangulation*) metodą. Atvaizduojant daugiamačius duomenis plokštumoje tiksliai išlaikomi atstumai nuo kiekvieno taško iki kažkurių kitų dviejų taškų; atstumai tarp tų dviejų taškų taip pat yra tiksliai išlaikyti. Tuo būdu gali būti tiksliai išlaikomi $(2m-3)$ atstumų (čia m – analizuojamų taškų skaičius).

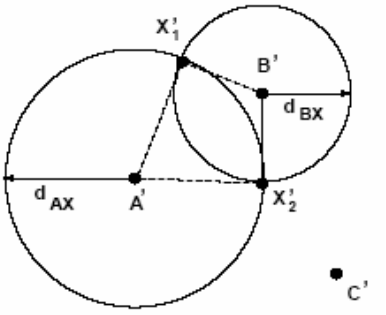
Iš turimų n-mačių taškų sudaromas minimalaus jungimo medis (*angl. minimum spanning tree, MST*) (Graham, 1985), vėliau daugiamačiai taškai atvaizduojami plokštumoje. Minimalaus jungimo medžio idėja paremta grafų teorija. Grafo jungimo medis (*angl. spanning tree*) yra grafas, kuriame visos viršūnės sujungtos į medį. Grafo šakos turi svorius arba ilgius. Medis, kurio svoris yra mažiausias, vadinamas minimalaus jungimo medžiu. Yra daug skirtingų algoritmų minimalaus jungimo medžiui rasti (Kruskal, 1956), (Prim, 1957). Galimi du metodai daugiamačius taškus atvaizduoti plokštumoje naudojantis trianguliacijos metodu: antro arčiausio kaimyno metodas (*angl. the second nearest neighbor approach*) ir atramos taško metodas (*angl. the reference point method*).

Antro arčiausio kaimyno metodas. Tarkime taškas X^j jau atvaizduotas plokštumoje, o taškas X^k tiesiogiai sujungtas su X^j minimalaus jungimo medyje. Reikia atvaizduoti plokštumoje tašką X^k. Taškas X^j turi būti arčiausias taškui X^k iš visų jau atvaizduotų taškų. Sakykime, kad tarp visų atvaizduotų taškų, kurių atstumai iki X^j yra tiksliai išlaikyti, X^i bus taškas, arčiausias iki X^k. Įprastai X^i yra tiesiogiai sujungtas su X^j minimalaus jungimo medyje. Tada naudojant trianguliacijos metodą, X^k plokštumoje atidedamas taip, kad jo atstumai iki X^i ir X^j būtų tiksliai išlaikyti.

Trianguliacijos metodo schema:

- imkime du n-mačius taškus A ir B;
- plokštumoje atidedame jų projekcijas A' ir B' tiksliai išlaikydami atstumus tarp jų $d^*_{AB} = d_{A'B'}$;
- ieškome taško X projekcijos X' plokštumoje, siekiant, kad atstumai d^*_{AX} ir d^*_{BX} būtų tiksliai išlaikyti ir plokštumoje, t.y. $d^*_{AX} = d_{A'X'}$ ir $d^*_{BX} = d_{B'X'}$. Tam brėžiami apskritimai su centrais A' ir B' ir spinduliais d^*_{AX} ir d^*_{BX}. Taško X' vieta bus ten, kur apskritimai susikerta arba liečiasi. Jei apskritimai tik liečiasi, tai bus vienintelis sprendinys X'. Tačiau apskritimai

gali susikirsti dviejuose taškuose X_1', X_2' (2.23 pav.). Artimiausias kaimynas (2.23 pav. tai taškas C') nulemia taško X' vietą.



2.23 pav. *Trianguliacijos metodo schema*

Atramos taško metode parenkamas vienas atramos taškas, iki kurio atstumai nuo visų kitų taškų bus visada išlaikomi. Tuo būdu kiekvienam taškui X^j išlaikomi atstumai iki dviejų taškų: iki taško, kuris tiesiogiai sujungtas su tašku X^j minimalaus jungimo medyje ir iki atramos taško.

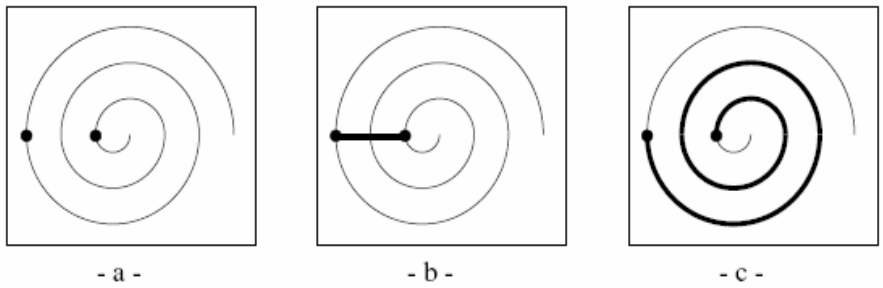
Darbe (Lee, 1977) pasiūlytos taisyklės, pagal kurias taškai surūšiuojami tam tikra tvarka ir tada jie vienas po kito atvaizduojami plokštumoje.

Isomap metodas

Isomap (Tenenbaum, 2000) yra daugiamačių skalių (MDS) grupės metodas, kuris skirtas daugiamačių duomenų dimensijos mažinimui. Šio metodo skirtumas nuo klasikinio MDS metodo yra tas, kad atstumų matas yra apibrėžiamas kitaip. Isomap metode daroma prielaida, kad pradinėje erdvėje analizuojami duomenys išsidėstę ant žemesnio matavimo daugdaros[1]. Prieš tai analizuotuose metoduose, atstumams tarp vektorių skaičiuoti paprastai buvo naudojami Euklido atstumai. Skaičiuojant atstumus neatsižvegiama į daugdaros formą. Naudojant MDS metodą susiduriama su sunkumais, atvaizduojant netiesines duomenų struktūras, tokias kaip, pavyzdžiui, spiralę. Projektuojant spiralę iš dvimatės erdvės į vienmatę, Isomap metodas naudoja geodezinius (kreivinius) atstumus. Geodezinis (arba kreivinis) atstumas, tai trumpiausio kelio ilgis, einant daugdaros kreivu paviršiumi. 2.24 paveiksle pavaizduota, kaip skaičiuojamas Euklido atstumas tarp dviejų taškų (2.24(b) pav.) ir geodezinis atstumas tarp tų pačių taškų (2.24(c) pav.).

[1] Daugdara (*angl. manifold*), tai abstrakti topologinė matematinė erdvė, kurioje kiekvieno taško aplinka yra labai panaši į Euklido erdvę, tačiau jos globali struktūra yra sudėtinga. Pavyzdžiui, Žemės paviršius yra daugdara.

Pagrindinė Isomap metodo idėja yra ta, kad daugiamačių skalių metodas atliekamas ne pradinėje (Euklido) erdvėje, bet netiesinės duomenų daugdaros geodezinėje erdvėje. Taigi taikant Isomap metodą ieškoma tokio poerdvio, kuriame geriausiai išlaikomi geodeziniai atstumai tarp vektorių (taškų).



2.24 pav. *(a) du taškai ant spiralės, (b) Euklido atstumas tarp dviejų taškų, (c) Geodezinis atstumas tarp taškų*

Isomap algoritmą sudaro trys etapai:

1) Daugiamatėje erdvėje surandami kiekvieno taško taškai „kaimynai“;
2) Skaičiuojami geodeziniai atstumai tarp visų įmanomų taškų porų;
3) Surandamos analizuojamų daugiamačių taškų projekcijos mažesnio matavimo erdvėje, naudojant daugiamačių skalių metodą (MDS) ir išlaikant atstumus tarp atitinkamų taškų.

Taškų „kaimynų“ paieškai gali būti naudojamas artimiausių kaimynų metodas (*angl. nearest neighbors algorithm*) arba taškai gali būti grupuojami pasirinkus tam tikro fiksuoto dydžio spindulį. Kaimynystės ryšių vaizdavimui naudojamas grafas, kuriame kiekvienas duomenų taškas (viršūnė) sujungtas su artimiausiu „kaimynu“.

Antrame žingsnyje apskaičiuojami geodeziniai atstumai tarp visų daugdaros taškų porų. Paskutiniame etape, Isomap metode pritaikomas klasikinis MDS metodas atstumų matricai, ir surandamos analizuojamų daugiamačių taškų projekcijas.

Yang (Yang, 2004) pasiūlė vietoj klasikinio MDS metodo naudoti Sammono algoritmą. De Silva ir Tenenbaum (Silva, 2002) sukūrė metodą, kuriame koreguojami grafo atstumai, priklausomai nuo duomenų struktūros ir tankumo.

Kitas metodas, kuris naudoja geodezinius arba kreivinius (*angl. curvilinear*) atstumus yra ***kreivinių atstumų analizė*** (*angl. curvilinear distance analysis*) (4.5 skyrius).

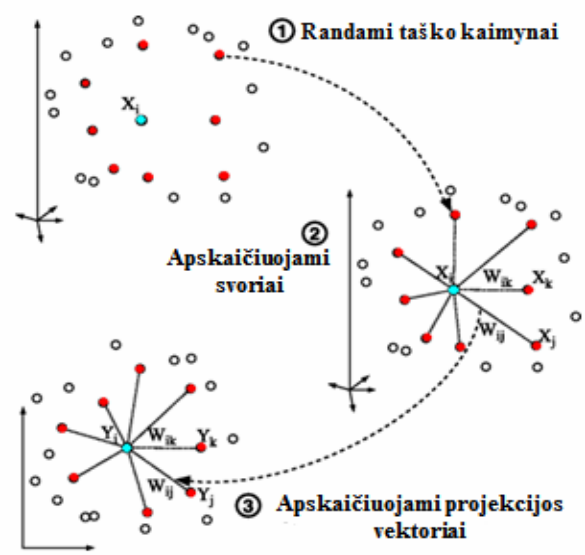
Lokaliai tiesinis atvaizdavimas

2000-ais metais Sam Roweis ir Lawrence Saul pasiūlė (Roweis, 2000) dar vieną daugiamačių duomenų dimensijos mažinimo metodą, **lokaliai tiesinį atvaizdavimą** (*angl. locally linear embedding, LLE*). LLE metodo savybės: surandant analizuojamų daugiamačių duomenų projekcijas mažesnio matavimo erdvėje, išlaikomi kaimynystės ryšiai tik tarp artimiausių taškų; atskleidžiama netiesinės daugdaros globali struktūra; duomenys vienareikšmiškai atvaizduojami į mažesnės dimensijos erdvę.

LLE metodas suranda tokį poerdvį, kuriame geriausiai išlaikoma analizuojamų duomenų lokalioji tiesinė struktūra.

LLE algoritmą sudaro trys etapai:

1) Randami kiekvieno analizuojamojo duomenų taško *k*-artimiausi kaimynai. Tai daroma, konstruojant orientuotąjį grafą, kurio kiekviena viršūnė sujungta briaunomis su taškais „kaimynais“ (2.25(1) pav.).
2) Kiekvienas taškas išreiškiamas, kaip jo kaimynų tiesinė kombinacija. Tam tikslui yra apskaičiuojami svoriai, kurie minimizuoja paklaidas tarp originalių ir transformuotų duomenų (2.25(2) pav.).
3) Fiksavus svorius, apskaičiuojami projekcijos vektoriai (2.25(3) pav.).



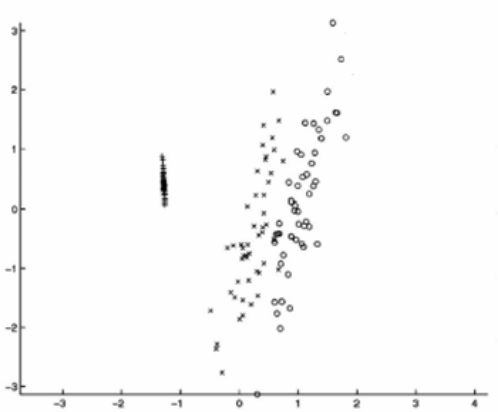
2.25 pav. *LLE algoritmo schema*

Mažinant dimensiją LLE algoritmu, pasiseka identifikuoti daugdaros neaiškią struktūrą. Tuo tarpu, MDS ir PCA metodais tolimi daugiamačiai taškai ant

daugdaros atvaizduojami į artimus taškus plokštumoje, tokiu būdu suardoma daugdaros struktūra. Isomap ir LLE algoritmai duoda gerus vizualizavimo rezultatus, kai analizuojami duomenys sudaro vieną klasterį (grupę). Kai duomenų aibė sudaryta iš kelių, visiškai atskirtų klasterių, projekcijos rezultatai būna žymiai blogesni.

MDS grupės metodai stengiasi išlaikyti santykinius atstumus tarp visų duomenų aibės taškų. LLE metodas nereikalauja išlaikyti atstumų tarp labiausiai nutolusių duomenų taškų, o tik tarp artimiausių kaimynų.

2.26 paveiksle pateiktas LLE metodu gautas daugiamačių taškų išsidėstymas dvimatėje plokštumoje, vizualizuojant irisų duomenis.



2.26 pav. *LLE metodo pavyzdys (vizualizuoti irisų duomenys)*

Kitas metodas, turintis daug bendro su LLE metodu, yra **Laplaso tikriniai atvaizdavimai** (*angl. Laplacian eigenmaps*). Laplacian eigenmaps suranda daugiamačių duomenų projekcijas mažesnio matavimo erdvėje, išlaikant daugdaros pagrindines lokalias savybes (Belkin, 2001), (Belkin, 2003). Lokalios savybės yra pagrįstos atstumais tarp artimiausių vektorių porų.

2.3. Daugiamačių duomenų vizualizavimo sistemos

Pasaulyje yra intensyviai kuriamos ir vystomos duomenų analizės sistemos. Čia pateikiamos kelios iš jų, kuriose yra galimybė vizualizuoti daugiamačius duomenis.

XGobi yra interaktyvi dinaminė vizualizavimo sistema, dirbanti *Windows* ir *Unix* terpėse. Ji skirta daugiamačiams duomenims analizuoti. Galima braižyti taškinius grafikus, taškinių grafikų matricas, lygiagrečias koordinates, atlikti

projekcijos paiešką, visų galimų projekcijų peržiūrą. Naujoje versijoje GGobi (http://www.ggobi.org/) realizuotas daugiamačių skalių metodas.

XGViz (http://public.research.att.com/~stat/xgobi/) yra programa, kurioje realizuotas daugiamačių skalių metodas (Buja, 1998). Sistemoje galima keisti MDS paklaidos funkcijos parametrus, vizualizuoti, t.y. grafiškai pateikti daugiamačius duomenis.

XmdvTool (http://davis.wpi.edu/~xmdv/) sistemoje realizuoti taškinės diagramos, žvaigždžių metodas, lygiagrečių koordinačių metodas, dimensijų įterpimo metodas.

DataDesk (http://www.datadesk.com) yra interaktyvi duomenų analizės ir statistikos sistema, leidžianti geriau suvokti analizuojamus duomenis. Sistema paremta tiriamosios duomenų analizės (*angl. exploratory data analysis*) idėjomis, akcentuojant duomenų struktūros, trendų, klasterių ir taškų-atsiskyrėlių vizualius, interaktyvius paieškos įrankius. Sistemoje taip pat yra daug tradicinių statistinių metodų.

JMP (http://www.jmpdiscovery.com) yra interaktyvi statistikinė sistema, naudojama statistiniam vizualizavimui ir duomenų analizei.

VisDB (http://www.dbs.informatik.uni-muenchen.de/ dbs/projekt/visdb/ visdb.html) – tai vizualios duomenų gavybos ir duomenų bazių tyrinėjimo sistema (Keim, 1994), (Keim, 1996). Joje duomenis galima vizualizuoti lygiagrečiomis koordinatėmis, brūkšnelinių figūrų metodu.

N-vision ir **AutoVisual** (http://www1.cs.columbia.edu/graphics/projects/ AutoVisual/AutoVisual.html) sistemos skirtos analizuoti daugiamačius duomenis „pasaulis pasaulyje“ metodu (Feiner, 1990), (Beshers, 1993).

HiSee (http://hisee.sourceforge.net/), ir **R** (http://cran.r-project.org/) sistemose realizuotas Sammono algoritmas, pagrindinių komponenčių metodas. R sistema turi dar ir papildomas galimybes: klasterinė analizė, tiesinė diskriminantinė analizė, faktorinė analizė, nepriklausomų komponenčių analizė.

S-plus (http://www.insightful.com/products/splus/default.asp) – tai statistikos sistema, kurioje yra realizuoti taškinės diagramos, grotelių metodas ir kiti vizualizavimo metodai.

Visumap (http://www.visumap.net/) sistema skirta darbui su didelių dimensijų duomenimis. Sistemoje realizuota daug tradicinių vizualizavimo, dimensijų mažinimo, klasterizavimo metodų: pagrindinių komponenčių analizė, daugiamačių skalių metodas, saviorganizuojantys neuroniniai tinklai, kreivinių komponenčių analizė. Dar viena sistema, kurioje realizuoti išvardinti metodai yra **Miner3D** (http://www.miner3d.com/).

Yra žinomos ir kitos duomenų analizės sistemos, leidžiančios geriau suvokti analizuojamus duomenis: **Spotfire Enterprise Analytics** (http://www.tibco.com/software/business_intelligence), **ILOG Discovery** (http://www.ilog.com/), **Partek Discovery Suite** (http://www.partek.com/partekds).

2.4. Antrojo skyriaus išvados

Pagrindinis daugiamačių duomenų vizualizavimo tikslas – tai duomenų pavaizdavimas žmogui suprantama ir suvokiama forma. Šiame skyriuje analizuoti tiesioginiai daugiamačių duomenų vizualizavimo metodai ir projekcijos metodai. Tiesioginiuose vizualizavimo metoduose kiekvienas daugiamačio vektoriaus parametras pateikiamas tam tikra vizualia forma. Pagrindiniai tiesioginio vizualizavimo metodai: (a) geometriniai metodai, (b) simboliniai metodai, (c) hierarchinio vizualizavimo metodai. Projekcijos metodų tikslas – pateikti daugiamačius duomenis mažesnės dimensijos erdvėje taip, kad būtų kiek galima tiksliau išlaikyta tam tikra duomenų struktūra, bei palengvinti didelės dimensijos duomenų interpretavimą bei apdorojimą. Yra tiesiniai ir netiesiniai projekcijos metodai. Tiesiniuose metoduose naudojamos tiesinės transformacijos, o netiesiniuose – netiesinės. Populiariausi tiesiniai projekcijos metodai yra: pagrindinių komponenčių analizė, nepriklausomų komponenčių analizė, faktorinė analizė, tiesinė diskriminantinė analizė. Svarbiausi netiesiniai projekcijos metodai yra daugiamačių skalių metodas ir jo atskiras atvejis – Sammono projekcija.

Daugiamačių duomenų analizės ir vizualizavimo metodai padeda atvaizduoti žmogui suprantamesne forma sudėtingais ir dažnai nežinomais tarpusavio ryšiais susietus daugiamačius duomenis. Metodų įvairovė yra pakankamai plati, todėl būtina įvairių metodų analizė, norint išsirinkti tinkamiausius konkrečiam atvejui.

Tiesioginio vizualizavimo metodų analizė parodė, kad naudojant šiuos metodus suprasti duomenų struktūrą yra gana sudėtinga, ypač esant didesnei duomenų dimensijai arba analizuojant didelės apimties duomenų aibę. Daug lengviau suvokti ir interpretuoti rezultatus, gautus projekcijos metodais, kuriuose daugiamatis vektorius transformuojamas į mažesnės dimensijos vektorių.

Lyginant tiesinės ir netiesinės projekcijos metodus, tikslesnė duomenų struktūra išlaikoma naudojant netiesinės projekcijos metodus. Bet ir čia duomenų vizualizavimo iškraipymai yra neišvengiami. Taip pat išlieka problema su naujų taškų atvaizdavimu.

3

Dirbtinių neuroninių tinklų koncepcijos

Norint atskleisti daugiamačių duomenų struktūrą, klasikinių vizualizavimo metodų kartais nepakanka. Šiam tikslui sėkmingai gali būti naudojami dirbtiniai neuroniniai tinklai (DNT). Neuroniniai tinklai yra galingas įrankis, naudojamas tais atvejais, kai formali analizė yra sudėtinga arba neįmanoma, tokiais, kaip atpažinimas, netiesinių sistemų identifikavimas ir valdymas.

3.1. Dirbtinių neuroninių tinklų pagrindai

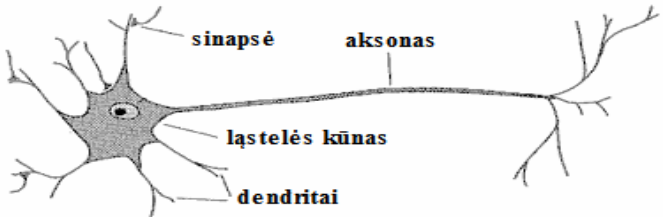
Dirbtiniai neuroniniai tinklai (DNT) (*angl. artificial neural networks*) (Haykin, 1999), (Verikas, 2003) pradėti tyrinėti, kaip biologinių neuroninių sistemų modelis. Pagrindinis dirbtinių neuroninių tinklų teorijos tikslas nėra kuo detaliau modeliuoti biologinius neuronus, o išsiaiškinti ir pritaikyti biologinių neuronų sąveikos mechanizmus efektyvesnėms informacijos apdorojimo sistemoms kurti. Dažnai dirbtiniai neuroniniai tinklai vadinami tiesiog neuroniniais tinklais. Šiuo metu jie vis plačiau naudojami dėl kelių pagrindinių priežasčių. Visų pirma, neuroninis tinklas yra galingas modeliavimo aparatas. Jo pagalba galima modeliuoti ypač sudėtingas funkcijas. Antra, neuroniniai tinklai apmokomi iš pavyzdžių. Turint duomenų pavyzdžius ir naudojant mokymo algoritmus, neuroninis tinklas pritaikomas prie duomenų struktūros. DNT yra

taikomi klasifikavimui, klasterizavimui, funkcijų aproksimavimui, prognozavimui, optimizavimui, medicininei diagnozei, finansinėms prognozėms, intelektinei paieškai, ir kt. Be daugelio kitų sprendžiamų uždavinių, dirbtiniai neuroniniai tinklai sėkmingai taikomi daugiamačiams duomenims vizualizuoti.

Kiekvienas neuroninis tinklas turi įėjimus ir išėjimus. Įėjimuose neuroniniam tinklui perduodamos kintamųjų reikšmes iš išorės. Išėjimuose formuojama prognozė, valdymo signalai ir pan. Be įėjimo ir išėjimo neuronų, dažnai egzistuoja ir tarpiniai (taip vadinami paslėpti) neuronai, atliekantys vidinį vaidmenį tinkle. Įėjimo, paslėpti ir išėjimo neuronai jungiami vieni su kitais. Paprastas neuroninis tinklas turi tiesioginio sklidimo struktūrą: signalai sklinda iš įėjimų pirmyn per visus paslėptus elementus ir pasiekia išėjimo neuronus. Tokia struktūra yra stabili. Tačiau, jei neuroninis tinklas yra rekurentinis (turintis jungtis atgal iš tolimesnių į ankstesniuos neuronus), jis gali būti nestabilus ir dažniausiai turi be galo sudėtingą dinamiką. Dirbama ir su rekurentiniais neuroniniais tinklais, tačiau realių problemų sprendimui dažniausiai naudojami tiesioginio sklidimo.

3.1.1. Biologinis neuronas

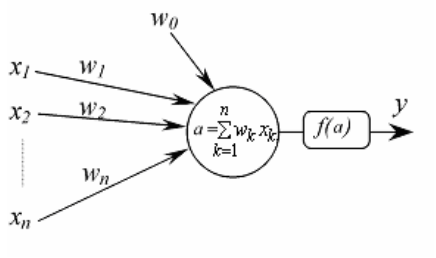
Žmogaus nervų sistema – labai sudėtingas neuronų tinklas. Pagrindinis šios sistemos elementas – smegenys, susideda iš daugelio neuronų, sujungtų vieni su kitais. Kiekvienas *neuronas* yra speciali ląstelė, galinti generuoti elektrocheminį signalą. Neuronas turi išsišakojusią įėjimo struktūrą, vadinamą *dendritais*, ląstelės kūną, vadinamą *soma* ir besišakojančią išėjimo struktūrą, vadinamą *aksonu*. (3.1 pav.). Vienos ląstelės aksonas su kitos ląstelės dendritais jungiasi per *sinapses*. Kai sužadinama pakankamai neuronų, prijungtų prie neurono dendritų, tas neuronas irgi sužadinamas ir jis generuoja elektrocheminį impulsą. Signalas per sinapses perduodamas kitiems neuronams, kurie vėlgi gali būti sužadinami. Neuronas sužadinamas tik tuo atveju, jei bendras dendritais gautas signalas viršija tam tikrą lygį, vadinamą *sužadinimo slenksčiu*. Turint didžiulį skaičių visiškai paprastų elementų, kurių kiekvienas skaičiuoja svorinę įėjimo signalų sumą ir generuoja binarinį signalą, jei suminis signalas viršija tam tikrą lygį, įmanoma atlikti palyginus sudėtingas užduotis (Verikas, 2003).



3.1 pav. *Biologinis neuronas*

3.1.2. Dirbtinio neurono modelis

Dirbtinis neuronas (3.2 pav.) yra svarbiausias neuroninio tinklo elementas. Praėjusio amžiaus ketvirtame dešimtmetyje buvo pasiūlytas dirbtinio neurono modelis. Tai elementarus procesorius, skaičiuojantis įėjimo kintamojo netiesinę funkciją.



3.2 pav. *Dirbtinis neuronas*

Pirmasis formalus dirbtinio neurono apibrėžimas buvo pasiūlytas darbe (McCulloch, 1943):

- Neuronas gauna keletą įėjimo reikšmių. Kiekviena įėjimo jungtis $x_1, x_2, ..., x_n$ (3.2 pav.) turi savo perdavimo koeficientą (svorį) $w_1, w_2, ..., w_n$ ir slenksčio reikšmę w_0.
- Skaičiuojama įėjimo ir svorių sandaugų suma: $a = w_1 x_1 + w_2 x_2 + ... + w_n x_n = \sum_{k=1}^{n} w_k x_k$.

Pagal sužadinimo signalą naudojant aktyvacijos funkciją (neurono perdavimo funkciją) $f(a)$ (3.1) skaičiuojama neurono išėjimo reikšmė y. Ši reikšmė tampa lygi vienam, jeigu suma viršija slenkstinę reikšmę w_0, nuliui – jeigu neviršija.

$$f(a) = \begin{cases} 1, \text{ jeigu } a \geq w_0 \\ 0, \text{ jeigu } a < w_0 \end{cases}. \tag{3.1}$$

Neurono išėjimo reikšmę y galima užrašyti naudojantis (3.2) formule:

$$y = f\left(\sum_{k=1}^{n} w_k x_k \right). \tag{3.2}$$

Funkcijos $f(a)$ reikšmės formulėje (3.1) yra susikoncentravusios aplink tašką w_0, bet ta pati išėjimo reikšmė gali būti gauta, jeigu funkcijos $f(a)$ reikšmės būtų sukoncentruotos aplink tašką 0, o svoris w_0, priskirtas įėjimui x_0 ($x_0 = 1$) ir įtrauktas į (3.2) sumą. Tada (3.2) virsta (3.3) formule:

$$y = f\left(\sum_{k=0}^{n} w_k x_k\right). \tag{3.3}$$

Dirbtinio neurono modelyje gali būti naudojamos ne tik (3.1) funkcija, bet taip pat ir kitokios aktyvacijos funkcijos, pvz. sigmoidinė funkcija $f(a) = \frac{1}{1+e^{-a}}$, hiperbolinis tangentas $f(a) = \tanh(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}}$ ir kt.

Kaip matyti iš 3.1 ir 3.2 paveikslų, biologinio neurono ir dirbtinio neurono modelio sandara yra gana panaši.

Dirbtiniai neuronai gali būti jungiami į *dirbtinius neuroninius tinklus*. Neuroninis tinklas gali būti pavaizduotas kaip grafas, kurio viršūnės yra neuronai, o šakos (su svoriais) jungia neuronų išėjimus su įėjimais (Jain, 1996). Pagal jungimo konstrukciją neuroniniai tinklai gali būti išskiriami į dvi pagrindines grupes:

- tiesioginio sklidimo (*angl. feedforward*) tinklai, kuriuose nėra grafo kilpų;
- atbulinio sklidimo (*angl. feedback*) (arba rekurentiniai) tinklai, kuriuose yra grafo kilpos.

3.1.3. Dirbtinių neuroninių tinklų mokymas

Galimybė mokytis yra esminė intelekto savybė. DNT mokymo procesas gali būti apibrėžtas kaip tinklo struktūros ir jungčių svorių keitimo uždavinys, siekiant, kad tinklas galėtų atlikti jam skirtą užduotį. Neurono mokymo procese iteratyviai keičiamos svorių reikšmės, atsižvelgiant į įėjimo ir išėjimo reikšmes, siekiant gauti reikiamą rezultatą.

Skirtingos tinklų architektūros reikalauja skirtingų jų mokymo algoritmų. Yra trys pagrindinės neuronų mokymo paradigmos:

- Mokymo su mokytoju algoritmai (*angl. supervised learning*);
- Mokymo be mokytojo algoritmai (*angl. unsupervised learning*);
- Hibridinis mokymas (*angl. hybrid learning*).

Jeigu žinomos trokštamos išėjimų reikšmės t, gali būti taikomi taip vadinami *mokymo su mokytoju algoritmai*. Tokiame mokyme tinklo išėjimų reikšmės y kiekvienam įėjimui yra tiesiogiai susijusios su žinomomis trokštamomis to išėjimo reikšmėmis t. Tinklas koreguojamas, keičiant svorių vektorių reikšmes, siekiant gauti kiek galima mažesnę paklaidą, t.y. ieškoma tokių svorių, kad skirtumas tarp trokštamų išėjimo reikšmių t ir reikšmių y, gautų apmokius neuroninį tinklą, būtų kiek galima mažesnis.

Kartais trokštamos tinklo išėjimo reikšmės nėra žinomos. Tada naudojami *mokymo be mokytojo algoritmai*. Juose tinklas mokomas pačiam ieškoti koreliacijų ar panašumų tarp mokymo aibės įėjimų. Čia nėra grįžtamojo ryšio,

pasakančio, kuris atsakymas bus arba yra teisingas. Mokymo be mokytojo algoritmuose nėra mokytojo signalo. Čia turima tik mokymo aibė $\{X^j, j = 1,...,m\}$, susidedanti iš vektorių, priklausančių erdvei R^n. Metodų tikslas yra kategorizuoti mokymo duomenis arba rasti juose kokius nors reguliarumus ar ypatumus. Gali būti sprendžiamas ir toks uždavinys: vektorius X^j, $j = 1,...,m$, reikia atvaizduoti mažesnio matavimo erdvės taškų rinkiniu taip, kad topologiniai ryšiai, esantys tarp X^j, būtų išlaikyti ir tarp naujos aibės taškų (Hassoun, 1995). Mokymo be mokytojo sėkmė glūdi tame, kad parenkamas nuo mokytojo nepriklausomas kriterijus, mokymo metu tas kriterijus optimizuojamas parenkant tinkamas svorių reikšmes.

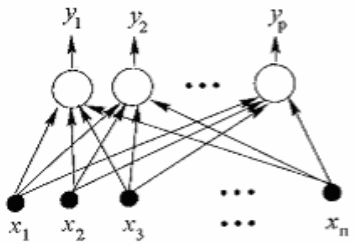
Galimos trys mokymo be mokytojo strategijos, kurios leidžia spręsti atitinkamus trijų tipų duomenų analizės uždavinius:

1. Hebbo tipo mokymas (*angl. Hebbian learning*),
2. varžytinių tipo mokymas (*angl. competitive learning*),
3. saviorganizuojančio žemėlapio (neuroninio tinklo) mokymas (*angl. self-organizing maps*).

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.

3.1.4. Perceptronas, jo mokymas

Paprasčiausios architektūros yra taip vadinami *tiesioginio sklidimo* (*angl. feedforward*) neuroniniai tinklai, kuriuose galimos tik vienkryptės į priekį (*angl. unidirectional forward*) jungtys. Paprasčiausias tokio tipo neuroninis tinklas – *perceptronas*, susidedantis iš vieno sluoksnio *p* neuronų, sujungtų su *n* įėjimais (3.3 pav.) (Silipo, 2003). Neuronų skaičius *p* lygus išėjimų skaičiui. Kai kurie autoriai perceptrone naudoja tik vieną neuroną (Raudys, 2001), tada $p = 1$.



3.3 pav. *Perceptronas*

Perceptrone kiekvienas išėjimas y_i yra įėjimo vektoriaus $X = (x_1, x_2, ..., x_n)$ funkcija, kuri skaičiuojama pagal (3.4) formulę.

$$y_i = f(a_i) = f\left(\sum_{k=0}^{n} w_{ik} x_k\right), \quad i = 1,...,p \tag{3.4}$$

čia w_{ik} jungties iš *k*-tosios įėjimo vektoriaus komponentės į *i*-tąjį išėjimą svoris.

Tarkime, kad yra m mokymo aibės elementų. Įėjimo vektoriai X^j, $j = 1,...,m$, susieti su trokštamomis reikšmėmis $t^j = (t_1^j, t_2^j, ..., t_p^j)$. Perceptrono mokymo procese svoriai w_{ik} keičiami taip, kad tinklo išėjimo reikšmės $Y^j = (y_1^j, y_2^j, ..., y_d^j)$ kiekvienam įėjimui X^j būtų kiek galima artimesnės trokštamoms reikšmėms t^j, t.y. gautųsi kiek galima mažesnė paklaida. Paklaidos matas $E(W)$ apibrėžiamas kaip svorių matricos $W = \{w_{ik}\}$ funkcija. Jeigu paklaidos funkcija $E(W)$ yra diferencijuojama pagal svorius w_{ik}, jos minimumas gali būti randamas gradientiniais optimizavimo metodais. Geriausiai žinomas yra gradientinio nusileidimo algoritmas. Dažniausiai naudojama paklaidos funkcija yra kvadratinių paklaidų (3.5) suma.

$$E(W) = \frac{1}{2}\sum_{j=1}^{m}\sum_{i=1}^{p}(y_i^j - t_i^j)^2 = \sum_{j=1}^{m} E^j(W), \tag{3.5}$$

$$E^j(W) = \frac{1}{2}\sum_{i=1}^{p}(y_i^j - t_i^j)^2 . \tag{3.6}$$

Pradžioje generuojamos atsitiktinės svorių w_{ik} reikšmės. Tada gradientinio nusileidimo algoritmu judama antigradiento kryptimi, svorių reikšmes keičiant pagal (3.7)–(3.9) iteracines formules.

$$w_{ik}(m'+1) = w_{ik}(m') + \Delta w_{ik}(m'), \tag{3.7}$$

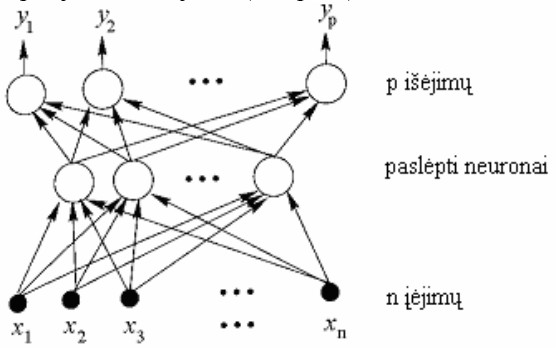
$$\Delta w_{ik}(m') = -\eta \frac{\partial E(m')}{\partial w_{ik}} = -\eta \sum_{j=1}^{m} \frac{\partial E^j(m')}{\partial w_{ik}} = \sum_{j=1}^{m} \Delta w_{ik}^j(m'), \tag{3.8}$$

$$\Delta w_{ik}^j(m') = -\eta \frac{\partial E^j(m')}{\partial w_{ik}} . \tag{3.9}$$

Čia η yra mažas teigiamas skaičius, vadinamas *mokymosi greičiu* (*angl. learning rate*), m' – iteracijos numeris. Galimos dvi svorių keitimo strategijos. Vienu atveju svoriai w_{ik} pakeičiami pagal (3.8) formulę pateikus į tinklą visus mokymo aibės vektorius. Kitoje strategijoje svoriai w_{ik} pakeičiami pagal (3.9) formulę po kiekvieno mokymo aibės vektoriaus pateikimo į tinklą.

3.1.5. Daugiasluoksniai tiesioginio sklidimo neuroniniai tinklai

Tinklai, turintys daugiau nei vieną neuronų sluoksnį, kuriuose galimi tik ryšiai į priekį iš įėjimų į išėjimus, yra vadinami *daugiasluoksniais perceptronais* (*angl. multilayer perceptrons*) arba *daugiasluoksniais tiesioginio sklidimo neuroniniais tinklais* (*angl. multilayer feedforward neural networks*). Kiekvienas toks tinklas sudarytas iš įėjimų aibės, išėjimų neuronų sluoksnio ir paslėptų neuronų sluoksnių tarp įėjimų ir išėjimų (3.4 pav.).



3.4 pav. *Tiesioginio sklidimo neuroninis tinklas*

Tarkime, kad yra daugiasluoksnis neuroninis tinklas, kuriame yra L sluoksnių, $l = 0,1,...,L$, kur sluoksnis $l = 0$ žymi įėjimus, o $l = L$ žymi išėjimus. Kiekviename sluoksnyje l yra n_l neuronų. Kiekvieno i-tojo neurono išėjimų reikšmė y_i l-tajame sluoksnyje gali būti apskaičiuojama pagal (3.4) formulę. Įėjimai x_k į i-tąjį neuroną l-tajame sluoksnyje atitinka išėjimus y_k $(l-1)$-ajame sluoksnyje. Tada (3.4) formulė tampa (3.10) formule.

$$y_i = f_i(a_i) = f_i\left(\sum_{k=0}^{n_{l-1}} w_{ik} y_k\right), \quad i = 1,...,n_l \tag{3.10}$$

Čia a_i atitinka įėjimus į i-tąjį neuroną, $f_i()$ – jų aktyvacijos funkcija, n_{l-1} – neuronų skaičius ($l-1$)-ajame sluoksnyje.

3.1.6. „Klaidos sklidimo atgal“ mokymo algoritmas

Taikant vienasluoksnio perceptrono mokymo idėją daugiasluoksniam neuroniniam tinklui, būtina žinoti paslėptų neuronų išėjimų reikšmes. Jei paklaidos ir aktyvacijos funkcijos yra diferencijuojamos, ieškant minimalios paklaidos gali būti naudojama gradientinio nusileidimo strategija. Algoritmas, kuris realizuoja gradientinio nusileidimo mokymo strategiją daugiasluoksniam

tiesioginio sklidimo neuroniniam tinklui, vadinamas „*klaidos sklidimo atgal" algoritmu* (*angl. back-propagation learning algorithm*). Jį sudaro du žingsniai:

- įėjimų reikšmių „sklidimas pirmyn" iš įėjimų į išėjimų sluoksnį;
- paklaidos „sklidimas atgal" iš išėjimų į įėjimų sluoksnį.

Tiek perceptrono mokyme, tiek „klaidos sklidimo atgal" algoritme naudojama mokymo su mokytoju strategija.

Naudosime (3.6) dalinę kvadratinių sumų paklaidą $E^j(W)$, svoriai bus keičiami pagal (3.9) formulę. Pirmame algoritmo žingsnyje įėjimų reikšmėms X^j apskaičiuojamos išėjimų reikšmės Y^j. Įvertinama paklaidos funkcija $E^j(W)$ išėjimų sluoksnyje L. Tuo baigiama „sklidimo pirmyn" fazė. Jei paklaidos funkcija $E^j(W)$ nelygi nuliui, reikia keisti svorių matricą ΔW. Panašiai, kaip ir vienasluoksniame perceptrone pagal (3.11) formulę keičiami visi svoriai w_{ik} jungčių, kurios jungia k-tąjį neuroną ($l-1$)-ajame sluoksnyje su i-tuoju neuronu (l)-ajame sluoksnyje.

$$\Delta w_{ik}^j = -\eta \frac{\partial E^j}{\partial w_{ik}}. \tag{3.11}$$

Dalinės išvestinės išreiškiamos taip:

$$\frac{\partial E^j}{\partial w_{ik}} = \frac{\partial E^j}{\partial a_i^j} \frac{\partial a_i^j}{\partial w_{ik}}. \tag{3.12}$$

Iš (3.10) formulės gauname:

$$\frac{\partial a_i^j}{\partial w_{ik}} = y_k^j. \tag{3.13}$$

Naudojant pažymėjimą

$$\delta_i^j = \frac{\partial E^j}{\partial a_i^j}, \tag{3.14}$$

(3.13) ir (3.14) išraiškas įstačius į (3.12) ir (3.11) gauname:

$$\frac{\partial E^j}{\partial w_{ik}} = \delta_i^j y_k^j \tag{3.15}$$

$$\Delta w_{ik}^j = -\eta \delta_i^j y_k^j \qquad \begin{array}{l} i \in l - \text{ajam sluoksniui} \\ k \in (l-1) - \text{ajam sluoksniui} \end{array} \tag{3.16}$$

Išėjimų sluoksniui, t.y. kai neuronas $i \in L$-ajam sluoksniui, turime:

$$\delta_i^j = \frac{\partial E^j}{\partial a_i^j} = f'(a_i^j)(y_i^j - t_i^j) \quad i \in \text{išėjimų sluoksniui } L \tag{3.17}$$

Lieka problema, kaip rasti $\frac{\partial E^j}{\partial a_i^j}$ paslėptiems neuronams, t.y. kai $i \in l$-ajam sluoksniui, kai $l < L$. Naudojant dalines išvestines bendru atveju galima parašyti:

$$\delta_i^j = \frac{\partial E^j}{\partial a_i^j} = \sum_{s=1}^{n_{l+1}} \frac{\partial E^j}{\partial a_s^j} \frac{\partial a_s^j}{\partial a_i^j}, \tag{3.18}$$

čia n_{l+1} žymi neuronų $(l+1)$-ajame sluoksnyje skaičių. Išraiška $\frac{\partial E^j}{\partial a_s^j}$ yra lygi dydžiui δ_s^j, apibrėžtam *s*-tajam neuronui $(l+1)$-ajame sluoksnyje. Atsižvelgiant į (3.10) formulę gauname $\frac{\partial a_s^j}{\partial a_i^j} = f'(a_i^j) w_{si}$. Tada paslėptiems *i*-tiesiems neuronams:

$$\delta_i^j = f'(a_i^j) \sum_{s=1}^{n_{l+1}} w_{si} \delta_s^j \quad \begin{matrix} i \in \text{sluoksniui } l < L \\ s \in \text{sluoksniui } l+1 \end{matrix}. \tag{3.19}$$

Iš pradžių reikia apskaičiuoti δ reikšmes išėjimų sluoksnyje *L* pagal (3.17) formulę. Tada palaipsniui skaičiuoti δ reikšmes paslėptiems neuronams tarpiniuose sluoksniuose $l < L$ naudojant $(l+1)$-mo sluoksnio δ reikšmes (3.19).

Kai visi svoriai pakeičiami pagal (3.16) formulę, į tinklą pateikiamas sekantis mokymo vektorius X^j ir procedūra kartojama vėl. Sustojimo kriterijus gali būti arba iš anksto nustatyta paklaidos funkcijos slenksčio reikšmė, arba atitinkamas atliktų iteracijų skaičius.

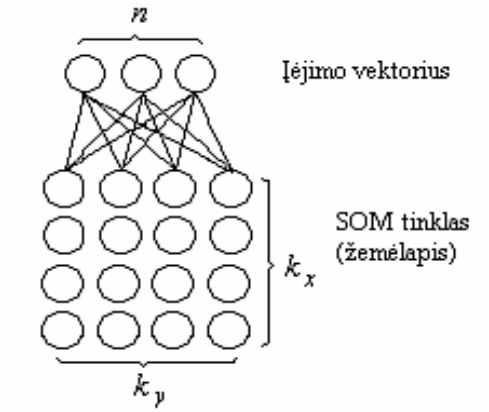
Norint pagreitinti mokymo procesą, yra koreguojama *delta* taisyklė (3.16), pridedant tam tikrą konstantą (angl. *momentum term*) (Haykin, 1999):

$$\Delta w_{ik}^j(m') = -\eta \delta_i^j(m') y_k^j(m') + \alpha \Delta w_{ik}^j(m'-1),$$

čia α dažniausiai yra teigiama konstanta $0 \le \alpha \le 1$, ir yra vadinama *momentum* konstanta (arba reikšmė) (*angl. momentum constant*). Kai $\alpha = 0$, gaunama (3.16) formulė. *Momentum* reikšmė naudojama „klaidos sklidimo atgal“ mokymo algoritme, norint išvengti „užstrigimo“ lokaliame minimume ir pagreitinti tinklo konvergavimo procesą. Mokymo procesas tampa labiau stabilus.

3.2. Saviorganizuojantys neuroniniai tinklai

Saviorganizuojantys neuroniniai tinklai (žemėlapiai) (*angl. self-organizing maps* (SOM)) (Kohonen, 2001) dar vadinami Kohoneno neuroniniais tinklais arba Kohoneno saviorganizuojančiais žemėlapiais. Šio tipo neuroninių tinklų pavadinimas kilo iš to, kad saviorganizuojantis žemėlapis, naudodamas mokymo aibę, pats save sukuria (organizuoja). SOM tinklo esmė – duomenų topografijos išlaikymas. Taškai, esantys arti vieni kitų įėjimo vektorių erdvėje, yra atvaizduojami arti vieni kitų ir SOM žemėlapyje. SOM žemėlapiai gali būti naudojami siekiant vizualiai pateikti duomenų klasterius, bei ieškant daugiamačių duomenų projekcijas į mažesnės dimensijos erdvę, įprastai į plokštumą.



3.5 pav. *Dvimačio SOM tinklo schema*

Saviorganizuojantis neuroninis tinklas (SOM) yra neuronų $M = \{m_{ij}, i = 1,...,k_x, j = 1,...,k_y\}$, išdėstytų dvimačio tinklelio (lentelės) mazguose, masyvas. Dvimačio neuroninio tinklo schema parodyta 3.5 paveiksle. Kiekvienas žemėlapio neuronas sujungtas su kiekvienu įėjimo vektoriumi (3.5 pav., kad jo neperkrauti, pavaizduotos tik pirmos žemėlapio eilutės jungtys su įėjimo vektoriais). Galima stačiakampė arba šešiakampė tinklo struktūra.

Keturkampės tinklo struktūros atveju, k_x yra lentelės eilučių skaičius, k_y – stulpelių skaičius. Kiekvienas *n*-matis mokymo aibės vektorius $X \in \{X^1, X^2, ..., X^m\}$ mokymo metu yra susiejamas su vienu tinklo neuronu, kuris taip pat yra *n*-matis vektorius. Mokymo pradžioje vektorių m_{ij} komponentės generuojamos atsitiktinai. Kiekviename mokymo žingsnyje vienas iš mokymo aibės vektorių $X \in \{X^1, X^2, ..., X^m\}$ pateikiamas į tinklą. Randama, iki kurio neurono m_c vektoriaus X^j Euklido atstumas yra mažiausias. Vektorius

m_c pavadinamas neuronu-nugalėtoju. Neuronų komponentės keičiamos pagal formulę:

$$m_{ij} \leftarrow m_{ij} + h_{ij}^c (X - m_{ij}),$$

čia, $h_{ij}^c = \frac{\alpha}{\alpha \eta_{ij}^c + 1}$, $\alpha = \max\left(\frac{e+1-\hat{e}}{e}; 0{,}01 \right)$ (e – mokymo epochų skaičius, $\hat{e}$ – vykdomos epochos numeris, viena mokymo epocha – tai mokymo proceso dalis, kai visus vektorius pateikiame tinklui po vieną kartą, ją sudaro m mokymo žingsnių) (Dzemyda, 2001). Dydis η_{ij}^c yra kaimynystės tarp neuronų m_c ir m_{ij} eilė. Greta neurono-nugalėtojo esantys neuronai vadinami pirmos eilės kaimynai, greta pirmos eilės kaimynų esantys neuronai, išskyrus jau paminėtus – antros eilės kaimynai ir t.t. Kiekvienos epochos metu perskaičiuojami tie neuronai m_{ij}, kuriems $\eta_{ij}^c \le \max[\alpha \max(k_x, k_y), 1]$.

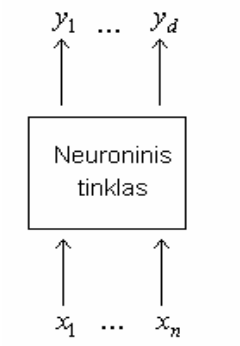
Vienas iš tinklo apsimokymo kokybės įvertinimo kriterijų yra kvantavimo paklaida $(angl.\ quantization\ \ error)\ E_{SOM} = \frac{1}{m} \sum_{j=1}^{m} \| X_j - m_{c(j)} \|$. Tai vidutinis atstumas tarp kiekvieno n-mačio duomenų vektoriaus $X^j = (x_1^j, x_2^j, ..., x_n^j)$ ir jo vektoriaus-nugalėtojo $m_{c(j)}$, m – analizuojamų vektorių skaičius.

3.3. Radialinių bazinių funkcijų neuroniniai tinklai

Radialinių bazinių funkcijų (*angl. radial basis function, RBF*) neuroniniai tinklai (Powell, 1985), (Broomhead, 1988), (Moody, 1989), (Haykin, 1999) sprendžia panašius uždavinius kaip ir daugiasluoksnis perceptronas. Radialinių bazinių funkcijų neuroniniai tinklai paslėptajame sluoksnyje naudoja radialines bazines funkcijas. Bendru atveju abiejų tipų tinklai aprašomi bendra schema, parodyta 3.6 paveiksle.

$X = (x_1, ..., x_n)$ yra įėjimo vektorius, $Y = (y_1, ..., y_d)$ yra išėjimo vektorius. Tikslas – apmokyti tinklą kuo geriau reaguoti į įėjimo vektorius.

Jei $X^j = (x_1^j, ..., x_n^j)$, $j = 1, ..., m$ yra įėjimo vektoriai, $Y^j = (y_1^j, ..., y_d^j)$, $j = 1, ..., m$ yra tinklo išėjimo vektoriai (Y^j yra tinklo reakcija į X^j), $t^j = (t_1^j, ..., t_d^j)$ yra trokštama tinklo reakcija į duomenų vektorius X^j, $j = 1, ..., m$. Nebūtinai $t^j = Y^j$ baigus tinklo mokymą.



3.6 pav. *Bendra neuroninio tinklo schema*

Radialinė bazinė funkcija ϕ, tai funkcija kurios reikšmė priklauso tik nuo atstumų nuo tam tikro pradžios taško, todėl $\phi(x) = \phi(\|x\|)$, arba $\phi(x,c) = \phi(\|x-c\|)$. Naudojant radialines bazines funkcijas paprastai sprendžiamas tam tikros n kintamųjų funkcijos $Y = (y_1(X),...,y_d(X))$, $X \in R^n$, $Y \in R^d$, aproksimavimo pagal jos stebėjimo m argumentų taškuose X^j, $j = 1,...,m$ rezultatus $t^j = (t_1^j,...,t_d^j)$, $j = 1,...,m$ uždavinys. Funkcija aproksimuojama svorine, taip vadinamų radialinių bazinių funkcijų, suma.

Radialinių bazinių funkcijų metodo taikymo pradžia siejama su tikslia duomenų imties interpoliacija daugiamatėje erdvėje. Panaudojant ir išvystant tikslaus interpoliavimo idėjas, gaunamas RBF neuroninis tinklas. Tinklo pagalba sudaroma interpoliuojanti funkcija, kurioje bazinių funkcijų skaičių apsprendžia ne mokymo imties dydis, bet atvaizduojamos funkcijos sudėtingumas.

Realizacijos ypatumai:

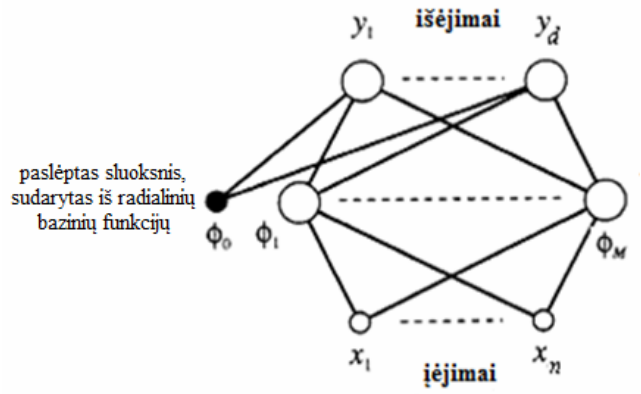
1) Radialinių bazinių funkcijų skaičius M parenkamas daug mažesnis už duomenų taškų skaičių m.
2) Bazinių funkcijų centrų radimas – mokymo proceso dalis.
3) Kiekviena bazinė funkcija gali turėti savus specifinius parametrus (Gausinės bazinės funkcijos atveju, tai plotis σ), kurie taip pat nustatomi mokymo metu.
4) Į svorinę bazinių funkcijų sumą įtraukiami laisvieji nariai. Jie kompensuoja bazinių funkcijų skaičiaus sumažinimą.

Tuo būdu gaunama tokia RBF neuroninio tinklo perdavimo funkcija $y_k(X) = \sum_{j=1}^{M} w_{kj}\phi_j(X) + w_{k0}$.

Laisvasis narys w_{k0} gali būti įtrauktas į sumą pridedant papildomą bazinę funkciją $\phi_0 = 1$:

$$y_k(X) = \sum_{j=0}^{M} w_{kj}\phi_j(X). \qquad (3.20)$$

RBF neuroninis tinklas, atitinkantis (3.20) funkciją parodytas 3.7 paveiksle.



3.7 pav. *RBF neuroninis tinklas*

Gausinės funkcijos atveju:

$$\phi_j(X) = \exp\left(-\frac{\| X - \mu_j \|^2}{2\sigma_j^2}\right),$$

čia μ_j yra radialinės bazinės funkcijos ϕ_j centro vektorius ($\mu_j \in R^n$); $\| X - \mu_j \|$ – atstumas tarp vektorių X ir μ_j.

Jeigu, pavyzdžiui, yra žinomas analizuojamų duomenų klasterių skaičius arba jį galima įvertinti, tai radialinių bazinių funkcijų skaičius gali būti lygus klasterių skaičiui, o μ_j – atitinkamo klasterio svorio centras.

RBF neuroninių tinklų skirtumai nuo daugiasluoksnio perceptrono:

- Visi daugiasluoksnio perceptrono parametrai paprastai nustatomi vienu metu, naudojant globalinį mokymą su mokytoju. Radialinių bazinių funkcijų neuroninis tinklas mokomas dviem etapais, pirmiausia mokymo

be mokytojo metodais nustatant bazinių funkcijų parametrus, tada antrojo sluoksnio svoriai nustatomi naudojant greitas tiesines mokymo su mokytoju procedūras.

- Daugiasluoksnis perceptronas gali turėti daugelį svorių sluoksnių ir norimą sujungimų tarp sluoksnių būdą, nebūtinai naudojant visus įmanomus svorius. Tame pačiame tinkle gali būti naudojamos įvairios perdavimo funkcijos. Radialinių bazinių funkcijų neuroniniai tinklai dažniausiai turi paprastą architektūrą ir susideda paprastai iš dviejų sluoksnių. Pirmajame talpinami bazinių funkcijų parametrai, antrajame formuojamos tiesinės bazinių funkcijų kombinacijos.

3.4. Trečiojo skyriaus išvados

Norint atskleisti daugiamačių duomenų struktūrą, vien tik klasikinių vizualizavimo metodų nepakanka. Šiam tikslui sėkmingai gali būti naudojami dirbtiniai neuroniniai tinklai (Mao, 1995). Neuroniniai tinklai yra galingas įrankis, naudojamas tais atvejais, kai formali analizė yra sudėtinga arba neįmanoma, tokiais, kaip atpažinimas, netiesinių sistemų identifikavimas. Tyrėjus neuroniniai tinklai domina gebėjimu pamėgdžioti žmogaus smegenų veiklą ir galimybe mokytis bei reaguoti. Prisitaikymas arba mokymasis – pagrindinis neuroninių tinklų tyrimų objektas.

Šiame skyriuje susistemintos esminės dirbtinių neuroninių tinklų koncepcijos, kuriomis bus pasinaudota analizuojant neuroninių tinklų galimybes vizualizuoti daugiamačius duomenis: apibūdintas biologinis neuronas, pateiktas dirbtinio neurono modelis, tinklų mokymo būdai, vienasluoksnio perceptrono sandara, jo mokymo algoritmas, daugiasluoksnių tiesioginio sklidimo neuroninių tinklų sandara, jų mokymas „klaidos sklidimo atgal“ algoritmu, saviorganizuojantys neuroniniai tinklai, radialinių bazinių funkcijų neuroniniai tinklai.

4

Dirbtinių neuroninių tinklų taikymas daugiamačiams duomenims vizualizuoti

Tiesioginio sklidimo neuroniniai tinklai plačiai taikomi klasifikavime, klasterizavime, funkcijų aproksimavime, prognozavime, optimizavime. Kita pastaruoju metu plačiai vystoma taikymo sritis – tai daugiamačių duomenų vizualizavimas.

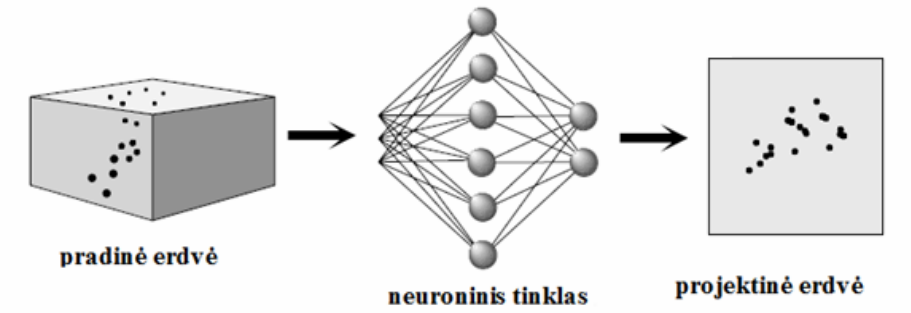
Pastaruoju metu buvo pasiūlytas didelis skaičius dirbtinių neuroninių tinklų ir mokymo algoritmų daugiamačiams duomenims vizualizuoti (Mao, 1995), (Kohonen, 2002), (Kraaijveld, 1995), (Sanger, 1989). Šie tiriamieji darbai gali būti suskirstyti į dvi grupes (nors šis klasifikavimas nėra labai tikslus). Pirmoji grupė, tai nauji arba egzistuojantys neuroninio tinklo modeliai daugiamačių duomenų projekcijai. Šio tipo pavyzdžiai apima saviorganizuojančius neuroninius tinklus (Kohonen, 2001), (Kohonen, 2002), netiesinės projekcijos metodus, netiesinę diskriminantinę analizę, pagrįstą paslėptų neuronų funkcionalumu tiesioginio sklidimo tinklo klasifikatoriuose (Mao, 1995). Šie tinklai atskleidžia įdomias duomenų savybes, kurias „nemato“ klasikiniai metodai. Todėl, jie gali būti naudojami kaip nauji įrankiai ar papildomi būdai daugiamačių duomenų projekcijoms rasti. Antroji grupė tyrinėtojų ištyrė neuroninių tinklų ir mokymo taisyklių savybes ir nustatė ryšius tarp klasikinių metodų ir neuroninių tinklų. Buvo pastebėta, kad kai kurie neuroniniai tinklai

faktiškai realizuoja duomenų projekcijos ir požymių išskyrimo algoritmus. Keletas klasikinių duomenų projekcijos metodų buvo realizuoti naudojant neuroninių tinklų struktūras (Baldi, 1989), (Mao, 1995), (Oja, 1991), (Sanger, 1989). Nors tokių bandymų rezultate nebūtinai gaunami nauji duomenų vizualizavimo metodai, tokie tinklai iš tikrųjų yra pranašesni už tradicinius metodus:

a) dauguma mokymo algoritmų ir neuroninių tinklų adaptuojasi prie besikeičiančių sąlygų, vadinasi jie gali būti pritaikyti realioms aplinkoms, kur reikalingos prisitaikymo sistemos. Tokie algoritmai lengvai realizuojami;

b) naudojant neuroninius tinklus, galima išvengti trūkumų, būdingų klasikiniams algoritmams.

4.1 paveiksle pavaizduota neuroninių tinklų, naudojamų daugiamačių duomenų dimensijos mažinimui, bendra schema.



4.1 pav. *Neuroninis tinklas daugiamačių duomenų vizualizavimui*

Pastaraisiais metais daug dėmesio yra skiriama neuroninių tinklų taikymo galimybėms daugiamačiams duomenims vizualizuoti. 1991 metais Kramer (Kramer, 1991) pasiūlė netiesinį pagrindinių komponenčių analizės metodą (*angl. autoassociator model*) tiesioginio sklidimo neuroninio tinklo mokymui ir pritaikė metodą daugiamačių duomenų projekcijos radimui. Jin ir kt. (Jin, 2000) naudojo neuroninius tinklus kinų kalbos simbolių atpažinimui. Vieni pirmųjų, nagrinėję neuroninių tinklų taikymą daugiamačiams duomenims vizualizuoti, buvo Oja (Oja, 1991), Usui (Usui, 1991). Taip pat dirbtinių neuroninių tinklų plataus spektro taikymams vizualizavime skirtas ir tyrimas (Dzemyda, Kurasova, Medvedev, 2007).

Yra pasiūlyta daug algoritmų tiesioginio sklidimo neuroninio tinklo, skirto dimensijos mažinimui, mokymui. Saund (Saund, 1989) naudojo trijų sluoksnių neuroninį tinklą su vienu paslėptu sluoksniu. Idrissi ir kt. (Idrissi, 2004) nagrinėjo

daugiasluoksnio neuroninio tinklo taikymą daugiamačiams duomenims vizualizuoti, tinklo mokymui naudojant jungtinių gradientų metodą.

Pastaruoju metu stebima tendencija, kad MDS (daugiamačių skalių) tyrimus vykdantys mokslininkai dažnai atsiriboja nuo kitų metodų tyrimų, ar net ignoruoja tuos metodus. Antra vertus, kitų vizualizavimo metodų tyrimuose nėra lyginimų ar sąsajų su MDS tipo metodais. Šiame darbe siekiama praplėsti MDS tipo metodų realizacijas dirbtinių neuroninių tinklų taikymu, tuo būdu stiprinant ryšį tarp skirtingų vizualios duomenų analizės krypčių.

4.1. SAMANN metodas

Sammono projekcija (Sammon, 1969) (2.2.2 skyrius) yra netiesinis daugelio kintamųjų objektų atvaizdavimo žemesnio matavimo erdvėje metodas. Sammono metodas minimizuoja projekcijos (Sammono) paklaidą E_S:

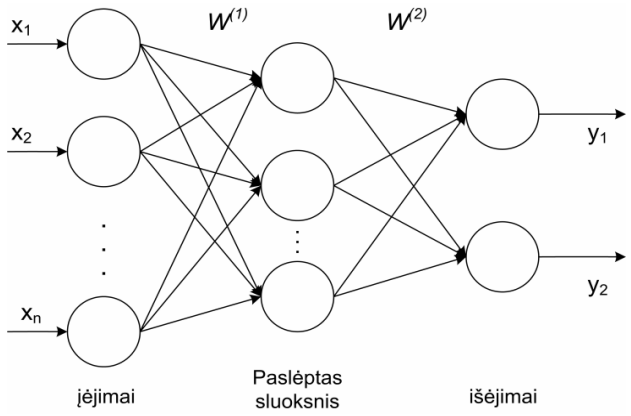
$$E_S = \frac{1}{\sum_{\mu=1}^{m-1}\sum_{\nu=\mu+1}^{m} d^*_{\mu\nu}} \sum_{\mu=1}^{m-1}\sum_{\nu=\mu+1}^{m} \frac{[d^*_{\mu\nu} - d_{\mu\nu}]^2}{d^*_{\mu\nu}}, \qquad (4.1)$$

čia $d^*_{\mu\nu}$ – atstumas tarp *n*-mačių vektorių X^μ ir X^ν, $d_{\mu\nu}$ – atstumas tarp *d*-mačių vektorių Y^μ ir Y^ν, į kuriuos projektuojami vektoriai X^μ ir X^ν ($d < n$). Analizuojamų vektorių skaičius lygus *m*.

Sammono algoritmo vienas iš trūkumų yra tas, kad atsiradus analizuojamų vektorių aibėje naujam taškui, norint ir jį atvaizduoti projektinėje erdvėje, reikia perskaičiuoti visų jau atvaizduotų taškų projekcijas. Mao ir Jain (Mao, 1995) pasiūlė Sammono paklaidą minimizuoti naudojant tiesioginio sklidimo neuroninius tinklus. Pasiūlyta specifinė „klaidos sklidimo atgal“ mokymo taisyklė, pavadinta SAMANN (Sammono algoritmas + dirbtiniai neuroniniai tinklai (*angl. Artificial neural networks, ANN*)), kuri leidžia įprastam tiesioginio sklidimo neuroniniam tinklui (4.2 pav.) realizuoti Sammono projekciją mokymo be mokytojo būdu. 4.2 paveiksle pavaizduotas neuroninio tinklo atskiras atvejis, tirintis du išėjimus, ir skirtas analizuojamų duomenų projekcijos radimui plokštumoje. Detalesnė informacija apie SAMANN algoritmą pateikta5.1 skyriuje.

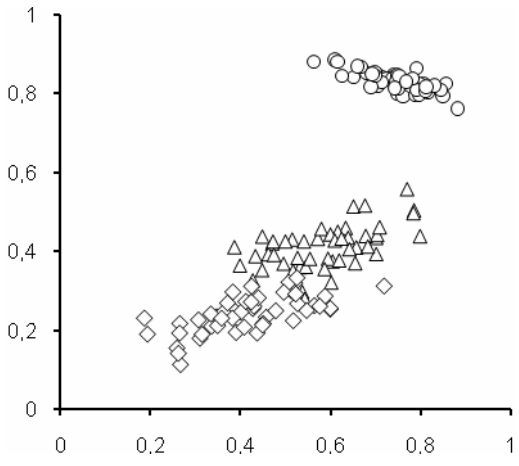
Disertacijos 5–8 skyriuose detaliau analizuojamos SAMANN metodo vizualizavimo galimybės. Atvaizduojant daugiamačius duomenis į mažesnio matavimo erdvę, labiausiai domina keli šio netiesinio atvaizdavimo aspektai: atstumai tarp atitinkamų vektorių turi būti maksimaliai išlaikyti, pereinant į mažesnio matavimo erdvę (t.y. turi būti kuo tikslesnė daugiamačių duomenų projekcija plokštumoje); turi būti išlaikyta pradinės duomenų aibės struktūra (t.y.

visi taškai turi būti atvaizduoti atitinkamai į „savo“ vietas); galimybė vizualizuoti naujus, „nematytus“ taškus.

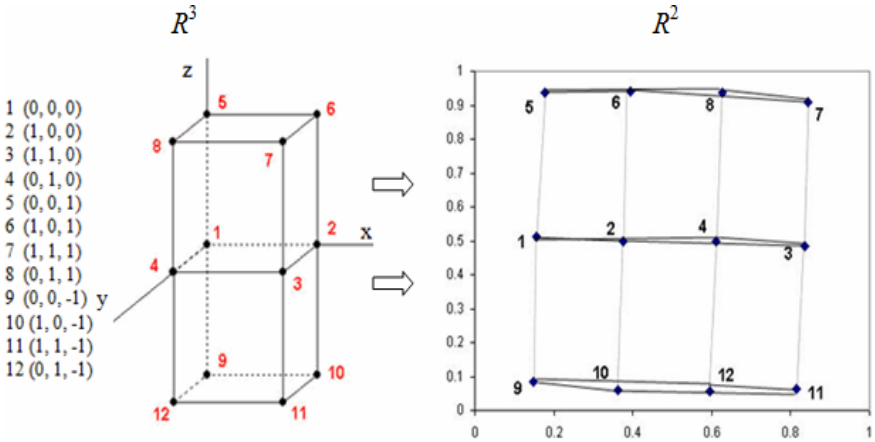


4.2 pav. *Tiesioginio sklidimo dviejų sluoksnių dirbtinis neuroninis tinklas Sammono projekcijai*

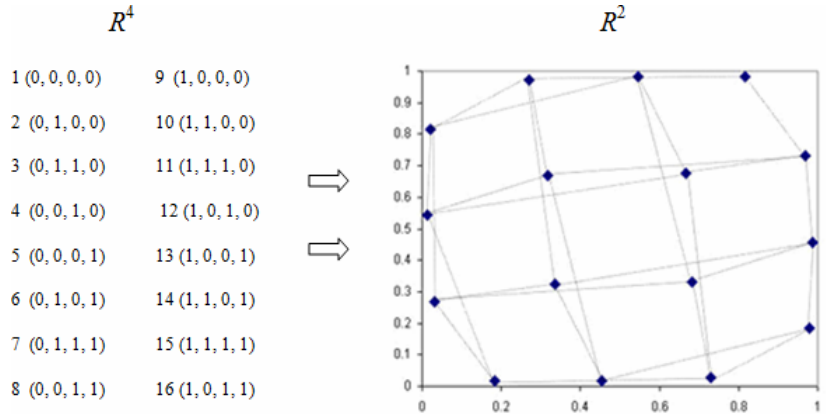
SAMANN algoritmu gautas irisų duomenų išsidėstymas plokštumoje pateiktas 4.3 paveiksle. 4.4 ir 4.5 paveiksluose pavaizduotos atitinkamai dviejų trimačių kubų ir 4-mačio taisyklingo kubo projekcijos plokštumoje. Iš 4.3, 4.4, 4.5 paveikslų matome, kad SAMANN metodas gerai išlaiko atstumus tarp atitinkamų taškų, projektuojant daugiamačius duomenis į plokštumą, ir išlaiko pradinės duomenų aibės struktūrą. Kitos SAMANN tinklo galimybės bei metodo analizė aprašomi 5, 6 ir 7 skyriuose.



4.3 pav. *SAMMAN algoritmu vizualizuoti irisų duomenys (trys grupės)*



4.4 pav. *SAMANN algoritmu gauta dviejų kubų projekcija plokštumoje*



4.5 pav. *SAMANN algoritmu gauta 4-mačio taisyklingo kubo projekcija plokštumoje*

Kaip alternatyva SAMANN mokymo be mokytojo taisyklei, gali būti naudojamas tiesioginio sklidimo neuroninio tinklo mokymas su mokytoju standartiniu „klaidos sklidimo atgal“ algoritmu. Tarkime, kad yra *n*-mačiai taškai X^j, $j=1,...,m$. Šie taškai Sammono algoritmu atvaizduojami plokštumoje, t.y. gaunamos taškų X^j projekcijos Y^j. Vėliau taškai X^j, $j=1,...,m$, pateikiami į tiesioginio sklidimo neuroninį tinklą kaip įėjimai, trokštamos išėjimų reikšmės imamos taškus X^j atitinkančių taškų Y^j komponentės; tinklas apmokomas standartiniu „klaidos sklidimo atgal“ algoritmu. Darbe (Ridder, 1997) SAMANN algoritmas palygintas su dviem daugiamačių duomenų projekcijos metodais paremtais Sammono algoritmu: trianguliacija ir neuroniniu tinklu, apmokomu

standartiniu „klaidos sklidimo atgal“ algoritmu. Darbe (Ridder, 1997) kaip tiesioginio sklidimo neuroninio tinklo mokymo su mokytoju trūkumas atžymėta tai, kad procesas reikalauja didesnių programinių resursų, nes susideda iš dviejų etapų: duomenų mokymui suformavimas Sammono algoritmu ir po to sekantis tinklo mokymas tais duomenimis.

4.2. SOM tinklo taikymas daugiamačiams duomenims vizualizuoti

Saviorganizuojantis neuroninis tinklas yra tinkamas daugiamačių duomenų vizualizavimo įrankis, kuris ne tik gali daugiamačius duomenis atvaizduoti plokštumoje, bet prieš tai juos klasterizuoja. Tokiu būdu tiksliau atskleidžiama duomenų struktūra. Darbe (Flexer, 2001) parodyta, kad SOM tinklas sėkmingai naudojamas ir klasterizavime, ir vizualizavime.

Po SOM tinklo mokymo, analizuojami vektoriai (mokymo aibės ar kitų duomenų) pateikiami į tinklą. Kiekvienam vektoriui randamas neuronas-nugalėtojas. Vektorių pavadinimai (numeriai ar klasės pavadinimai) užrašomi tuose žemėlapio (lentelės) langeliuose, kuriuos atitinka jų neuronai-nugalėtojai, t.y. vektoriai išsidėsto tarp žemėlapio elementų. Tai galima laikyti kaip *n*-mačių taškų išsidėstymą plokštumoje, t.y. SOM tinkle daugiamačiai duomenys transformuojami į tam tikrą diskrečią struktūrą. Taškų vietą plokštumoje nusako tinklelio mazgai, t.y. eilučių ir stulpelių numeriai. Paprasčiausiu atveju gaunama lentelė (stačiakampės tinklo topologijos atveju), kurios langeliuose surašyti analizuojamų vektorių pavadinimai (4.1 lentelė).

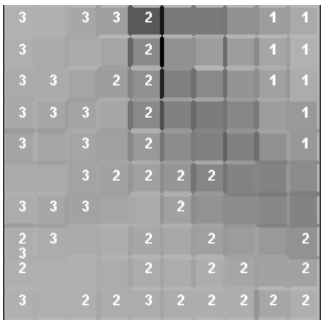
4.1 lentelė. *SOM lentelės ([4x4]) pavyzdys: analizuoti psichologinių testų duomenys (Dzemyda, 2001)*

10, 11, 12,		17, 18	14, 15, 16
13, 21		19	
1			
2, 3, 4	20, 22 ,23		5, 6, 7, 8, 9

Tačiau tokia lentelė nėra labai informatyvi, sunku pasakyti, kaip toli yra vektoriai, esantys gretimuose lentelės langeliuose. Todėl būtina ieškoti metodų, kaip pagerinti tokio vaizdo kokybę.

Unifikuota atstumų matrica (*U-matrica*) (*angl. unified distance matrix*) yra vienas iš populiaresnių SOM tinklo vizualizavimo metodų. U-matricą sudaro atstumai tarp kaimyninių SOM neuronų. Pvz. turint [1x5] tinklą $[m_1, m_2, ..., m_5]$ (1 eilutė, 5 stulpeliai), U-matrica bus vienos eilutės ir devynių stulpelių vektorius

$[u_1, u_{12}, u_2, u_{23}, u_3, u_{34}, u_4, u_{45}, u_5]$. Čia $u_{ij} = \| m_i - m_j \|$ yra atstumas tarp dviejų kaimyninių neuronų, o u_i yra specialiai apibrėžta reikšmė, pvz. vidutinis atstumas tarp kaimyninių neuronų: $u_3 = \frac{u_{23} + u_{34}}{2}$. Radus U-matricą, jos reikšmes reikia pateikti SOM tinkle. Darbuose (Ultsch, 1990), (Kraaijveld, 1995) pasiūlytas metodas, pagal kurį vidutiniai atstumai tarp kaimyninių neuronų yra pateikiami pilkos skalės atspalviais (vėliau imta naudoti ir kitų spalvų skales). Jei vidutiniai atstumai tarp kaimyninių neuronų yra maži, tuos neuronus atitinkantys tinklo langeliai spalvinami šviesia spalva; tamsi spalva reiškia didelius atstumus. Taigi klasteriai yra nustatomi pagal šviesius atspalvius, o ribos – pagal tamsesnius (Kohonen, 2001), (Kohonen, 2002). 4.6 paveiksle, iliustruojančiame U-matricą, matyti, kaip pirmos klasės irisai atsiskiria nuo kitų dviejų; griežtos ribos tarp antros ir trečios klasių nėra.

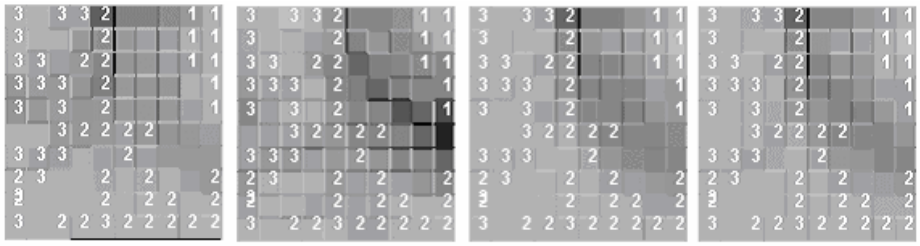


4.6 pav. *U-matrica (analizuoti irisų duomenys)*

Dažniausiai yra naudojama dvimatė U-matrica, tačiau darbe (Takatsuka, 2001) siūloma naudoti trimatį SOM tinklo vizualizavimą, kuris padeda geriau atskirti susidariusius klasterius.

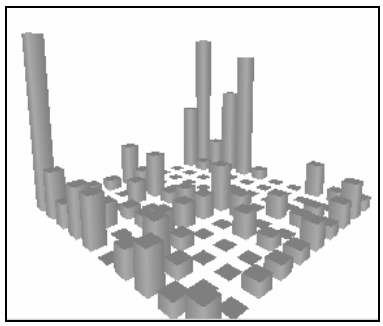
Kiekviena neurono komponentė atitinka skirtingas įėjimo vektoriaus komponentes. Naudojant U-matricos vizualizavimo ideologiją galima atvaizduoti atskiras komponentes. Tokiu būdu gaunamos taip vadinamos *komponenčių plokštumos* (*angl. component planes*) (4.7 pav.) Iš paveikslo galima matyti, kokią įtaką daro viena ar kita komponentė.

SOM tinklo vizualizavimui gali būti naudojamos *histogramos*. Histograma parodo, kiek vektorių priklauso klasteriui, apibrėžtam vienu neuronu. Yra keli histogramų vizualizavimo būdai, vienas iš jų, gautas naudojantis NeNet (http://koti.mbnet.fi/~phodju/nenet/Nenet/General.html), pateiktas 4.8 paveiksle.



a) **b)** **c)** **d)**

4.7 pav. *Komponenčių plokštumos, analizuoti tik irisų: taurėlapių ilgiai (a), taurėlapių pločiai (b), vainiklapio ilgiai (c), vainiklapių pločiai (d)*



4.8 pav. *Neuronų-nugalėtojų histograma (analizuoti irisų duomenys)*

Paminėtųjų SOM tinklo vizualizavimo metodų trūkumas tas, kad ne visada aiškiai atskleidžiama analizuojamų duomenų struktūra. Kartais gautų rezultatų interpretavimas yra sudėtingas uždavinys. Todėl kyla poreikis SOM tinklo vizualizavimui ieškoti geresnio būdo. Kadangi SOM neuronai yra daugiamačiai vektoriai juos galima projektuoti į plokštumą naudojant kurį nors daugiamačių vektorių projekcijos metodą, pvz. Sammono projekciją (Sammon, 1969). Kaip parodyta darbe (Kurasova, 2005), SOM tinklo jungimas su Sammono projekcija yra vienas iš efektyvesnių daugiamačių duomenų vizualizavimo būdų.

4.3. Hebbo mokymas ir jo taikymas pagrindinių komponenčių radimui

Kaip jau buvo parodyta, pagrindinės komponentės gali būti naudojamos daugiamačiams duomenims vizualizuoti. Darbe (Kvedaras, 1974) pateikti pagrindinių komponenčių radimo skaitiniai metodai. Dirbtiniai neuroniniai tinklai, apmokomi Hebbo taisykle, taip pat gali būti naudojami ieškant pagrindinių komponenčių.

Viena iš dirbtinių neuroninių tinklų mokymo be mokytojo taisyklių, paremta Hebbo teorija, yra vadinama *Hebbo mokymo taisykle* (*angl. Hebbian learning*). Kiekvienas įėjimo vektorius siejamas su vienu išėjimo vektoriumi. Šiuo mokymu siekiama, kad kuo dažniau įėjimo aibė pateikiama duotajam neuronui, tuo atitinkamas atsakymas yra stipresnis. Sinapsių koeficientai (svoriai W) kinta proporcingai koreliacijai tarp prieš-sinapsinio X ir po-sinapsinio Y signalų.

Hebbo mokymo idėja: tarkime vektorius $X^j = (x_1, x_2, ..., x_n)$ yra neuroninio tinklo įėjime, tada išėjimo vektorius yra $Y^j = (y_1, y_2, ..., y_d)$; kai į tinklą pateikiamas vektorius $X^j + \varepsilon$ artimas vektoriui X^j, išėjimo vektorius turėtų būti $Y^j + \delta$ artimas vektoriui Y^j. Vektoriai X^j, $j = 1,...,m$, priklauso erdvei R^n, o vektoriai Y^j, $j = 1,...,m$, priklausys erdvei R^d ($d < n$). Būtent dėl šios priežasties Hebbo mokymo taisyklė gali būti taikoma daugiamačių duomenų dimensijai sumažinti, o kai $d = 2$ ir jų atvaizdavimui plokštumoje.

Pradžioje analizuokime vieno neurono atvejį. Neurono išėjimo reikšmė apskaičiuojama taip: $y^j = \sum_{k=1}^{n} x_k^j w_k = (X^j)^T W$, čia X^j – įėjimų vektorius-stulpelis, $(X^j)^T$ – vektorius-eilutė, tai transponuotas įėjimo vektorius-stulpelis X^j, W – svorių vektorius-stulpelis, y^j – išėjimas. Tada mokymo (svorių keitimo) taisyklė:

$$W(m'+1) = W(m') + \eta y^j X^j \text{ arba}$$

$$\Delta W = \eta y^j X^j . \tag{4.2}$$

čia η – mokymo greitis, m' – iteracijos numeris. Apskaičiuojame (4.2) formulės vidurkį pagal visus įėjimų vektorius, gauname:

$$\langle \Delta W \rangle = \eta \langle yX \rangle = \eta \langle XX^T W \rangle . \tag{4.3}$$

(4.3) formulėje ženklu $\langle \cdot \rangle$ žymimas vidurkis, $\langle yX \rangle = \frac{1}{m} \sum_{j=1}^{m} y^j X^j = \frac{1}{m} \sum_{j=1}^{m} (X^j)(X^j)^T (W^j) = \langle XX^T W \rangle$.

Kadangi X ir W nepriklausomi dydžiai, (4.3) formulė gali būti parašyta taip:

$$\langle \Delta W \rangle = \eta \langle yX \rangle = \eta \langle XX^T \rangle \langle W \rangle , \tag{4.4}$$

$C = \langle XX^T \rangle$ yra autokoreliacinė matrica.

$$C = \begin{pmatrix} \langle x_1^2 \rangle & \langle x_1 x_2 \rangle & \dots & \langle x_1 x_n \rangle \\ \langle x_2 x_1 \rangle & \langle x_2^2 \rangle & \dots & \langle x_2 x_n \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle x_n x_1 \rangle & \langle x_n x_2 \rangle & \dots & \langle x_n^2 \rangle \end{pmatrix} \tag{4.5}$$

Matrica C yra simetrinė matrica, pagrindinėje diagonalėje yra įėjimų komponenčių kvadratų vidurkiai. Jos nuosavų ortogonalių vektorių e_k nuosavos reikšmės λ_k, $k = 1,...,n$ yra teigiamos arba lygios nuliui ir yra lygties $Ce_k = \lambda_k e_k$ sprendiniai (Hassoun, 1995).

Normuota Hebbo mokymo taisyklė:

$$\begin{cases} W(1) \quad \text{pasirenkame laisvai} \\ W(m'+1) = \dfrac{W(m') + \eta y^j X^j}{\| W(m') + \eta y^j X^j \|} \end{cases}. \tag{4.6}$$

Naudojant vidutines svorių keitimo reikšmes, (4.6) formulę galima pakeisti taip: $\langle W(m'+1) \rangle = \dfrac{\langle W(m') \rangle + \eta C \langle W(m') \rangle}{\| \langle W(m') \rangle + \eta C \langle W(m') \rangle \|}$. Ši formulė, esant didelei dydžio η reikšmei, gali būti užrašyta: $\langle W(m'+1) \rangle = \dfrac{C \langle W(m') \rangle}{\| C \langle W(m') \rangle \|}$. O tai yra klasikinis metodas (*angl. power method*) simetrinės matricos B maksimalią nuosavą reikšmę atitinkančiam nuosavam vektoriui rasti: $e_1(m'+1) = \dfrac{Be_1(m')}{\| Be_1(m') \|}$. Šiuo klasikiniu metodu randama ir didžiausia nuosava reikšmė λ_1. Norint rasti kitas nuosavas reikšmes naudojamas taip vadinamas „išsiurbimo“ metodas (*angl. deflation method*). Jame išanalizuojamos matricos atimama matrica $e_1 e_1^T$, padauginta iš λ_1: $B' = B - \lambda_1 e_1 e_1^T$. Tuo pačiu metodu randama matricos B' didžiausia nuosava reikšmė, o tai bus antra pagal dydį matricos B nuosava reikšmė. Tokiu pat būdu randamos ir kitos nuosavos reikšmės (Hassoun, 1995).

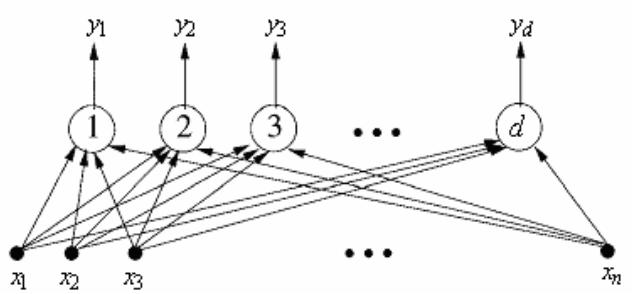
Modifikuota Hebbo taisyklė yra vadinama Oja taisykle:

$$\begin{cases} W(1) \quad \text{pasirenkame laisvai} \\ W(m'+1) = W(m') + \eta y^j X^j - \eta (y^j)^2 W(m'). \\ \qquad\qquad = W(m') + \eta (X^j - y^j W(m')) y^j \end{cases} \tag{4.7}$$

Taigi, naudojant vieną neuroną, apmokomą Hebbo taisykle, galime rasti pirmąją pagrindinę komponentę. Norint rasti *d* pirmųjų pagrindinių komponenčių galimi du būdai, kuriuos ir aptarsime.

Sakykime, kad yra taškai $X^1, X^2, ..., X^m \in R^n$; sprendžiamas uždavinys – gauti jų projekcija erdvėje R^d, t.y. taškus $Y^1, Y^2, ..., Y^m \in R^d$, $d < n$. Pagrindinių komponenčių analizės (PCA) idėja – rasti sistemą, sudarytą iš *d* ortogonalių vektorių ir transformuoti vektorių $X = (x_1, x_2, ..., x_n)$ į vektorių $Y = (y_1, y_2, ..., y_d)$ siekiant išlaikyti duomenų struktūrą (detaliau apie PCA skaitykite 2.2.1 skyriuje).

Norint rasti *d* pirmųjų pagrindinių komponenčių galima naudoti vieno sluoksnio *d* neuronų tinklą (4.9 pav.) ir atitinkamai modifikuoti Hebbo mokymo taisyklę.



4.9 pav. *Įėjimo vektoriaus projekcija į d pirmąsias pagrindines komponentes*

Sakykime, kad yra tinklas, sudarytas iš *d* neuronų. $W_i = (w_{i1}, w_{i2}, ..., w_{in})$ bus jungčių į *i*-tąjį neuroną svorių vektorius. Y. H. Oja išplėtė savo (4.7) mokymo taisyklę neuroniniam tinklui, sudarytam iš *d* neuronų (4.9 pav.). Svorių keitimo taisyklė:

$$\Delta w_{ij} = \eta \left(x_j - \sum_{k=1}^{d} w_{kj} y_k \right) y_i . \tag{4.8}$$

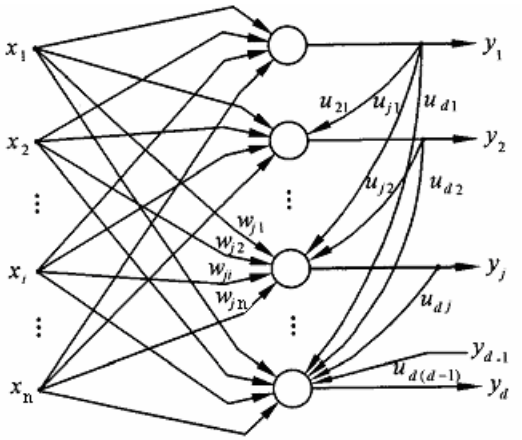
Kitą panašią taisyklę pasiūlė T. D. Sanger (Sanger, 1989):

$$\Delta w_{ij} = \eta \left(x_j - \sum_{k=1}^{i} w_{kj} y_k \right) y_i . \tag{4.9}$$

Abi formulės (4.8) ir (4.9) yra identiškos Oja (4.7) formulei, kai $d = 1$. Kai $d > 1$, jos tarpusavyje skiriasi tik viršutine sumavimo riba. Mokymo pabaigoje gaunami svorių vektoriai W_i yra ortogonalūs. Kartais Oja taisyklė gali nerasti matricos *C* nuosavų vektorių krypčių. Tačiau tuo atveju, *d* svorių vektoriai konverguoja į tokį pat poerdvį kaip ir matricos *C* *d* pirmųjų nuosavų vektorių.

Čia svorių vektoriai priklauso nuo pradinių sąlygų ir nuo duomenų pateikimo į tinklą eilės. Sanger taisyklė yra nejautri pradinėms sąlygoms ir duomenų pateikimo į tinklą tvarkai. Svoriai W_i konverguoja į nuosavus vektorius $\pm e_i$, pirmo neurono $(i=1)$ svorių rinkinys lygus didžiausią nuosavą reikšmę atitinkančiam nuosavam vektoriui, $W_1 = \pm e_1$.

Kitame pagrindinių komponenčių radimo metode naudojamas vienasluoksnis neuroninis tinklas, sudarytas iš d neuronų, su pagalbinėmis jungtimis tarp neuronų (4.10 pav.).



4.10 pav. *PCA neuroninis tinklas su pagalbinėmis jungtimis*

4.4. Kreivinių komponenčių analizė

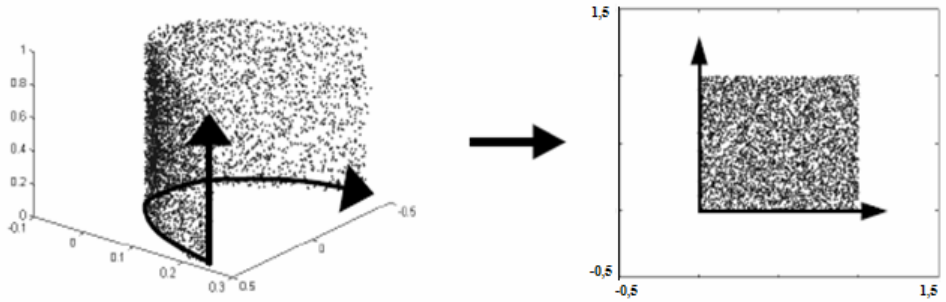
Saviorganizuojantys neuroniniai tinklai yra daugiamačių duomenų vizualizavimo įrankis, kuris daugiamačius duomenis atvaizduoja plokštumoje ir atskleidžia duomenų struktūrą. SOM tinklas transformuoja turimus duomenis į tam tikrą fiksuotą topologinę struktūrą. Tačiau, jeigu ši struktūra nesutampa su vidine turimų duomenų struktūra, tai gautas atvaizdavimas nebus tikslus. Kreivinių komponenčių analizės algoritmas (*angl. curvilinear component analysis*, *CCA*) buvo pasiūlytas tam, kad pradinių duomenų atvaizdavimas būtų patikimesnis (Demartines, 1997). CCA naudoja visiškai naują paklaidos funkciją E (4.10). Algoritmas išskaido duomenų daugdarą ir suranda tų duomenų projekciją mažesnio matavimo erdvėje. Pasagos formos skirstinio projekcija (pereinant iš trimatės erdvės į dvimatę), gauta CCA metodu, pateikta 4.11 paveiksle.

CCA algoritmas yra Kohoneno saviorganizuojančių neuroninių tinklų (SOM) patobulinimas. Galutinė atvaizduojamųjų duomenų struktūra nėra iš

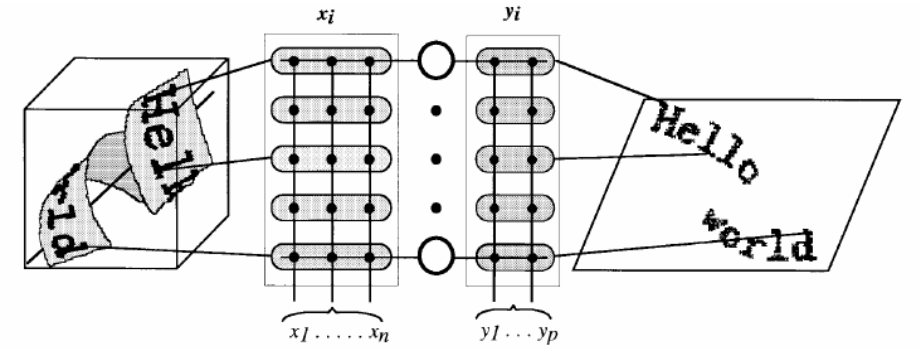
anksto fiksuota, bet yra tolydi erdvė, kurios forma priklauso nuo pradinių duomenų podaugdario. Algoritmą sudaro du atskiri žingsniai:

1) pradinių duomenų vektorių kvantavimas į K kvantuotų pradinės erdvės poerdvių;
2) gautų kvantuotų vektorių netiesinė projekcija.

4.12 paveiksle pavaizduota šį algoritmą realizuojančio tinklo struktūra ir atvaizdavimo iš trimatės erdvės į dvimatę pavyzdys. Iš pradžių analizuojamos atvaizduojamųjų duomenų struktūra ir pradinė projekcija, o vėliau duomenys yra projektuojami, atsižvelgiant į sukonstruotą atvaizdavimą tarp pradinės erdvės ir projektinės erdvės. Po mokymo tinklas turi galimybę projektuoti bet kurį kitą naują vektorių.



4.11 pav. *CCA algoritmo pavyzdys: trimatės erdvės duomenų projekcija į dvimatę erdvę*



4.12 pav. *CCA algoritmo tinklo struktūra*

Tinklo pirmajame sluoksnyje atliekamas pradinių duomenų vektorių kvantavimas, naudojant bet kurį vektorių kvantavimo metodą (Ahalt, 1990). Išėjimo sluoksnis konstruoja netiesinį atvaizdavimą, siekiant minimizuoti

struktūros skirtumus tarp kvantavimo erdvės ir projektinės erdvės. Tam yra naudojama tikslo funkcija (Demartines, 1997):

$$E = \frac{1}{2}\sum_{i}^{K}\sum_{j\neq i}^{K}(d_{ij}^{*} - d_{ij})^2 F(d_{ij}, \lambda_y), \tag{4.10}$$

čia d_{ij}^{*} atstumas tarp daugiamačių vektorių X^i ir X^j, d_{ij} – atstumas tarp vektorius X^i ir X^j atitinkančių dvimačių vektorių Y^i ir Y^j ($i, j = 1,..,m$). $F(d_{ij}, \lambda_y)$ yra svorio funkcija, nuo kurios priklauso lokalios topologijos išlaikymas. Jos reikšmės yra intervale (0; 1). Svorio funkcija yra aprėžta, monotoniškai mažėjanti ir priklauso tik nuo atitinkamų atstumų tarp vektorių projektinėje erdvėje. Būtent tuo CCA metodas skiriasi nuo tradicinių MDS metodų.

CCA metodo idėja yra ta, kad dėmesys koncentruojamas į vektorius, kurie mažiausiai nutolę vienas nuo kito. Paklaidos funkcija, atsižvelgiant į projektinės erdvės vektorius, minimizuojama naudojant tokią procedūrą:

$$\Delta y_j = \alpha(t) F(d_{ij}, \lambda_y)(d_{ij}^{*} - d_{ij})\frac{y_j - y_i}{d_{ij}}, \forall j \neq i,$$

$\alpha(t) = \frac{\alpha_0}{1+t}$ yra mokymosi parametras, priklausantis nuo laiko. Mokymo proceso metu $\alpha(t)$ mažėja.

Lyginant su klasikiniu PCA metodu, CCA metodo privalumas yra tas, kad duomenis, kurie yra pasiskirstę ne pagal normalųjį skirstinį, gali būti atvaizduoti labai tiksliai, naudojant netiesinį atvaizdavimą.

4.5. Kreivinių atstumų analizė

Kreivinių atstumų analizės (*angl. curvilinear distance analysis, CDA*) metodas turi daug bendro su Kreivinių komponenčių analizės metodu ir Sammono netiesine projekcija (Lee, 2000), (Lee, 2004). Šio metodo skirtumas nuo kreivinių komponenčių analizės yra tik tas, kad jis naudoja ne Euklidinius atstumus, o tikrus kreivinius atstumus (kaip ir Isomap metodas, 2.2.2 skyrius). Kaip ir CCA metodas, CDA pagrindinį dėmesį koncentruojama į atstumus tarp tų taškų, kurie yra mažiausiai nutolę vienas nuo kito projektinėje erdvėje. Tikslo funkcija parodo, kaip tiksliai yra išlaikomi atstumai tarp analizuojamų vektorių porų:

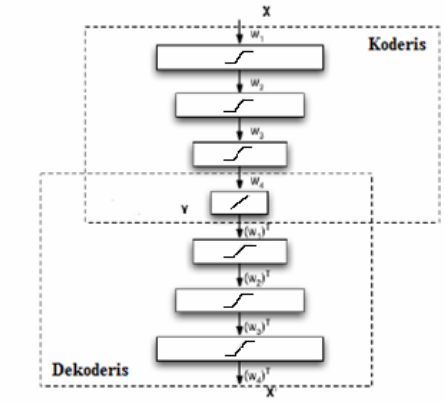
$$E_{CDA} = \frac{1}{2}\sum_{i}^{K}\sum_{j\neq i}^{K}(d_{ij}^{*} - d_{ij})^2 F(d_{ij}, \lambda_y),$$

čia d_{ij}^{*} kreivinis (arba geodezinis) atstumas tarp vektorių pradinėje erdvėje.

4.6. Autokoderiai

Daugiasluoksniai koderiai (*angl. encoders*) yra tiesioginio sklidimo neuroniniai tinklai su nelyginiu skaičiumi paslėptų sluoksnių (DeMers, 1993), (Hinton, 2006b). Vidurinis paslėptas sluoksnis turi *d* neuronų, o įėjimo ir išėjimo sluoksniai turi po *n* neuronų. Autokoderio (*angl. autoencoder*) schemos pavyzdys pavaizduotas 4.13 paveiksle. Autokoderio tinklas (Hinton, 2006b) sudarytas lyg iš dviejų dalių: pirma dalis transformuoja daugiamatės erdvės analizuojamus duomenis į mažesnio matavimo erdvę, o antra dalys rekonstruoja pradinius duomenis iš gautų projekcijų. Tinklo mokymo proceso metu yra minimizuojama vidutinė kvadratinė paklaida, gaunama tarp tinklo įėjimo ir išėjimo reikšmių. Neuroninis tinklas apmokomas pradiniais vektoriais X^i, o tinklo vidurinio paslėptojo sluoksnio neuronų išėjimuose gaunamos analizuojamų duomenų projekcijos Y^i *d*-matėje erdvėje, išlaikant tikslią pradinių duomenų struktūrą.

Jeigu neuroninio tinklo modelyje naudojama tiesinė aktyvacijos funkcija, tai autokoderiai yra labai panašūs į PCA metodą. Netiesiniam atvaizdavimui dažniausiai yra naudojama sigmoidinė aktyvacijos funkcija.

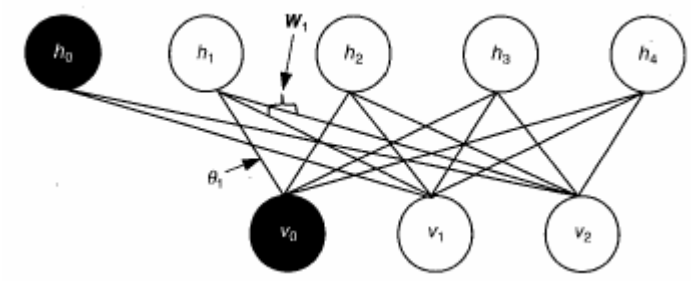


4.13 pav. *Autokoderio struktūrinė schema*

Daugiasluoksniai autokoderiai paprastai turi didelį jungčių skaičių. Todėl klaidos sklidimo atgal algoritmas konverguoja lėtai ir gali patekti į lokalųjį

minimumą. Šis trūkumas yra pašalinamas, prieš tinklo mokymą panaudojant taip vadinamus ribotus Boltzmann mechanizmus (*angl. Boltzmann restricted machines, BRC*) (Hinton, 2006b). BRC, tai neuroninis tinklas, turintis vieną sluoksnį, sudarytą iš paslėptų, nesujungtų tarpusavyje neuronų, ir kitą matomų, nesujungtų tarpusavyje, neuronų sluoksnį (Hinton, 2006a), (Smolensky, 1986). 4.14 paveiksle pavaizduotas tokio tinklo modelis: tinklas iš dviejų matomų neuronų ir keturių paslėptų neuronų. BRC gali būti naudojami neuroninių tinklų, sudarytų iš didelio paslėptų sluoksnių skaičiaus, mokymui, naudojant modeliuojamo atkaitinimo (*angl. simulated annealing*) metodą. Neuroninio tinklo mokymui gali būti naudojami ir genetiniai algoritmai. Autokoderių mokymo procesas konverguoja labai lėtai, jeigu pradinės erdvės ir projektinės erdvės dimensijos yra pakankamai didelės.

Darbe (Hinton, 2006a) yra pateikti pagrindiniai autokoderių privalumai, lyginant su pagrindinių komponenčių metodu ir lokaliai tiesiniu atvaizdavimu. Metodas yra efektyvus dirbant su didelės apimties duomenų aibėmis.



4.14 pav. *Ribotas Boltzmann mechanizmas*

4.7. NeuroScale metodas

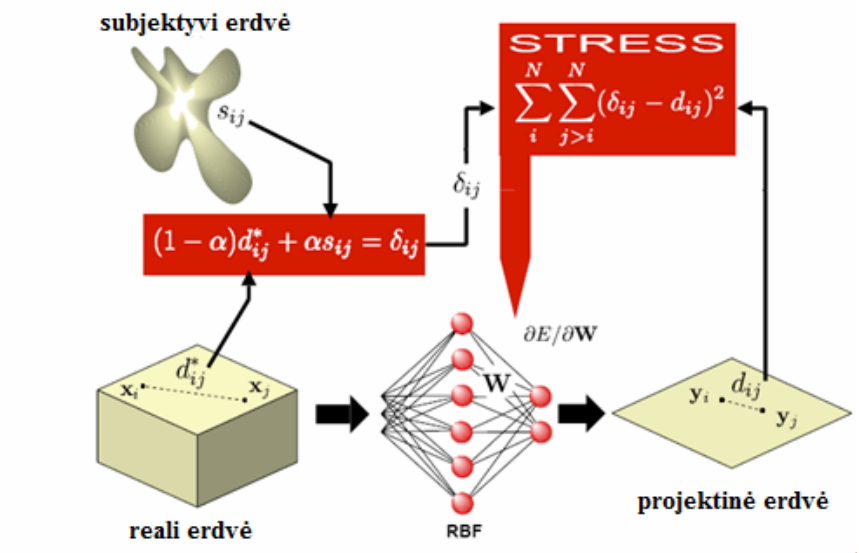
NeuroScale algoritmas yra Sammono projekcijos (2.2.2 skyrius) modifikacija, kurioje panaudojami radialinų bazinių funkcijų (3.3 skyrius) tiesioginio sklidimo neuroniniai tinklai (Lowe, 1996), (Lowe, 1997). NeuroScale naudoja papildomą informaciją apie duomenis ir tos informacijos pagrindu gali koreguoti analizuojamų duomenų projekcijas. Vizualizuojant duomenis, algoritmas bando išlaikyti analizuojamų duomenų geometrinę struktūrą ir atstumus tarp atitinkamų vektorių pradinėje erdvėje ir projektinėje erdvėje.

Sammono algoritmo vienas iš trūkumų tas, kad atsiradus analizuojamų vektorių aibėje naujam taškui, norint ir jį atvaizduoti plokštumoje, reikia perskaičiuoti visų jau atvaizduotų taškų projekcijas. NeuroScale algoritmas, kaip ir SAMANN algoritmas (4.1 skyrius), tokio trūkumo neturi.

NeuroScale metodas transformuoja daugiamatės erdvės vektorius į mažesnio matavimo erdvę, naudojant tiesioginio sklidimo radialinių bazinių funkcijų neuroninį tinklą. Tinklo mokymo algoritmas sutampa su Sammono algoritmu (2.2.2 skyrius), tik yra minimizuojama kita projekcijos paklaida:

$$E = \sum_{i}^{m} \sum_{j<i}^{m} (\delta_{ij} - \|y_i - y_j\|)^2 ,$$

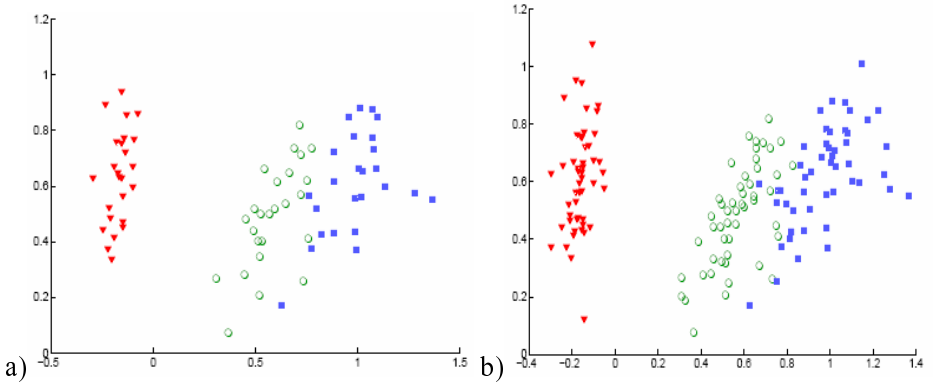
čia $\delta_{ij} = (1-\alpha) d_{ij}^* + \alpha s_{ij}$, $0 \le \alpha \le 1$. s_{ij} – tai papildomos subjektyvios žinios (subjektyvūs nepanašumai) apie kiekvienos vektorių poros nepanašumus. Kiekvienai vektorių porai surandamas ne tik d_{ij}^*, bet ir subjektyvūs nepanašumai s_{ij}. Parametras α leidžia kontroliuoti subjektyvios metrikos įtaką tinklo išėjimams. Kitaip sakant, α padeda surasti kompromisą tarp projekcijų su mokytoju ir be mokytojo. Detalesnis algoritmo aprašymas yra pateiktas literatūroje (Lowe, 1997), (Tipping, 1996). 4.15 paveiksle pavaizduotas NeuroScale metodo schemos pavyzdys (Tipping, 1996).



4.15 pav. *NeuroScale metodo struktūrinė schema*

NeuroScale metodo privalumas yra tas, kad jeigu analizuojamų vektorių aibėje atsiranda naujas *n*-matis taškas *X*, jis pateikiamas jau apmokytam tinklui, tinklo išėjime gaunamos taško *Y*, kuris yra *X* projekcija, komponentės. 4.16(a) paveiksle pateikti rezultatai, vizualizuojant irisų duomenų aibės 75 taškus (po 25 taškus iš kiekvienos klasės), o 4.16(b) paveiksle pateikti rezultatai, vizualizuojant

visus aibės taškus, be papildomo tinklo mokymo. Tinklo „nematyti“ taškai plokštumoje „randa“ vietą tarp savo klasių taškų.



4.16 pav. *NeuroScale metodo pavyzdys (vizualizuoti irisų duomenys): (a) vizualizuoti 75 taškai; (b) vizualizuoti visi 150 taškų (kiti 75 taškai be papildomo tinklo mokymo)*

NeuroScale metodo vizualizavimo rezultatai yra žymiai geresni, lyginant su SOM arba ICA (Noel, 1998). Tačiau šis metodas yra neefektyvus, analizuojant didelės apimties duomenų aibes (kai vektorių skaičius didesnis nei 1000).

Kiti metodai

Kaip alternatyva SOM metodui (3.2 skyrius) buvo sukurtas **generatyvinis topografinis atvaizdavimas** *(angl. generative topographic mapping, GTM*) metodas (Bishop, 1997), (Bishop, 1998). Tai netiesinis paslėptų parametrų modelis. Dar vienas netiesinis daugiamačių duomenų projekcijos metodas yra **Isotop** (Lee, 2002). Tai SOM tinklo ir Sammono algoritmo junginys. Isotop išlaiko taškų kaimynystės ryšius ir geriau negu SOM tinklas išlaiko duomenų pradinę struktūrą.

4.8. Dirbtinių neuroninių tinklų programinės realizacijos

Šiuo metu yra sukurta nemažai programinių sistemų, kuriuose yra realizuoti dirbtiniai neuroniniai tinklai. Vienos sistemos – tai universalūs statistikiniai paketai, kuriuose be neuroninių tinklų yra ir daug kitų duomenų analizės įrankių. Kitos yra specifinės neuroninių tinklų sistemos. Pastarosiose gali būti realizuoti vieno ar kelių tipų neuroniniai tinklai. Toliau pateikta kelių tokių sistemų trumpa analizė.

Matematinėje sistemoje ***Matlab*** (`http://www.mathworks.com`) yra **neuroninių tinklų priemonių paketas** (*angl. neural network toolbox*), kuriame yra įvairios:

- tinklo struktūros sukūrimo funkcijos, kurias naudojant galima sukurti vienasluoksnį perceptroną, tiesioginio sklidimo pasirinkto skaičiaus sluoksnių neuroninį tinklą, Hopfieldo rekurentinį tinklą, radialinių bazinių funkcijų tinklą, saviorganizuojantį neuroninį tinklą ir kt.
- mokymo taisyklės (perceptrono mokymo, „klaidos sklidimo atgal“ algoritmas, Hebbo mokymo, SOM mokymo ir kt.), ir kitos funkcijos.

Netlab Toolbox (`http://www.ncrg.aston.ac.uk/netlab`) yra *Matlab* sistemos neuroninių tinklų priemonių paketas, kuriame yra realizuoti PCA algoritmas, radialinių bazinių funkcijų tinklai, Neuroscale metodas.

Priedai, realizuojantys neuroninius tinklus, yra ir kituose statistikiniuose paketuose, pvz. **STATISTICA: Neural Networks** (`www.statsofttinc.com`), **SPSS Neural Connection** (`http://www.spss.com`) ir kt.

Sistema **PathFinder** (`http://www.zsolutions.com/pathfind.htm`) – tai dirbtinių neuroninių tinklų sistema, kurioje galima sukurti bei apmokyti dviejų sluoksnių (paslėpto ir išėjimų) tiesioginio sklidimo neuroninį tinklą. Tinklo mokymui naudojamas „klaidos sklidimo atgal“ algoritmas. Sukonstruotą tinklą galima apmokyti prognozuoti ir klasifikuoti duomenis.

Sistemoje **SOM-PAK** (`http://www.cis.hut.fi/research/som_lvq_pak.shtml`) (Kohonen, 1996) pateiktos kelios funkcijos: neuroninio tinklo pradinio žemėlapio inicijavimui (*mapinit*, *randinit*, *lininit*), jo mokymui (*vsom*), neuronų-nugalėtojų sužymėjimui juos atitinkančiais duomenų vektorių pavadinimais (*vcal*), žemėlapio vizualizavimui (*visual*, *planes*, *umat*). Taip pat yra funkcija (*sammon*), skirta daugiamačiams duomenims vizualizuoti Sammono algoritmu.

Sistemoje **SOM-TOOLBOX** (`http://www.cis.hut.fi/projects/somtoolbox/`) (Alhoniemi, 2000) pateiktos funkcijos, sukurtos matematine sistema *Matlab* 5; ji arba aukštesnė jos versija yra būtina sistemos naudojimui. SOM-TOOLBOX pakete yra žemėlapio struktūros sukūrimo, duomenų paruošimo, duomenų nuskaitymo iš failo, pradinio žemėlapio inicijavimo, žemėlapio sukūrimo ir mokymo, jo neuronus-nugalėtojus atitinkančių vektorių pavadinimų sužymėjimo ir vizualizavimo bei kitos funkcijos.

Saviorganizuojantys neuroniniai tinklai realizuoti ir sistemose Nenet ir Viscovery SOMine, SOM Option Pack (Davisor Chart) (`http://products.davisor.com/chart/index.html`), Visipoint (`http://www.visipoint.fi/`), Viscovery® SOMine® – Self-Organizing Maps (`www.eudaptics.com/`), Miner3D (`http://www.miner3d.com/`).

RapAnalyst (`www.raptorinternational.com`**)** sistemoje yra galimybė analizuoti bei vizualizuoti duomenis, naudojant trys pagrindines dirbtinio intelekto technologijas: dirbtiniai neuroniniai tinklai, genetiniai algoritmai ir saviorganizuojantys neuroniniai tinklai.

4.9. Ketvirtojo skyriaus išvados

Šiame skyriuje analizuotos dirbtinių neuroninių tinklų galimybės vizualizuoti daugiamačius duomenis, kadangi klasikiniai vizualizavimo metodai kartais yra nepajėgūs susidoroti su savo užduotimis. Tačiau yra ir neuroninių tinklų, kurie realizuoja duomenų projekcijos algoritmus – pagrindinių komponenčių metodą ir Sammono projekciją. Šiame skyriuje taip pat nagrinėti kiti metodai, kuriuose dirbtiniai neuroniniai tinklai naudojami daugiamačiams duomenims vizualizuoti: kreivinių komponenčių analizė, NeuroScale metodas, kreivinių atstumų analizė.

Ne visi iš nagrinėtų neuroninių tinklų leidžia naujų taškų vizualizavimą be tinklo permokymo. Tačiau tokią galimybę turi, pavyzdžiui, NeuroScale ir SAMANN tinklai.

Naudojant specifinį mokymo be mokytojo „klaidos sklidimo atgal“ algoritmą, pavadintą SAMANN, ieškoma tokio daugiamačių taškų išsidėstymo plokštumoje, kad Sammono paklaida būtų minimali. SAMANN tinklo mokymo proceso konvergavimas yra labai lėtas, todėl toliau šiame darbe ieškoma būdų, kaip jį pagreitinti. Taip pat nagrinėjami šio metodo privalumai ir trūkumai, bei galimybė vizualizuoti naujus taškus be papildomo tinklo mokymo. Metodo privalumas yra tas, kad jis gali būti naudojamas naujų analizuojamos aibės vektorių atvaizdavimui plokštumoje neperskaičiavus visų jau atvaizduotų taškų projekcijų.

5

SAMANN neuroninio tinklo mokymo problemos

Nustatant ryšius tarp klasikinių metodų ir neuroninių tinklų, buvo pastebėta, kad kai kurie neuroniniai tinklai faktiškai realizuoja duomenų projekcijos algoritmus. Atliekant tyrimus, keletas klasikinių duomenų projekcijos metodų buvo realizuoti naudojant *neuroninių tinklų struktūras*. Nors tokių bandymų rezultate nebūtinai gaunami nauji projekcijos metodai, tačiau tokie tinklai yra pranašesni už tradicinius metodus. Pavyzdžiui, *SAMANN neuroninis tinklas* (Mao, 1995), sukurtas Sammono netiesinės projekcijos algoritmui, turi naujų duomenų vizualizavimo galimybę, t.y. leidžia surasti naujų taškų projekcijas be papildomų skaičiavimų. O tai yra savybė, kurios neturi originalus Sammono algoritmas.

Pagrindiniai skyriaus rezultatai paskelbti šiuose straipsniuose: (Medvedev, Dzemyda, 2005a), (Medvedev, Dzemyda, 2005b), (Medvedev, Dzemyda, 2006a).

5.1. SAMANN algoritmas

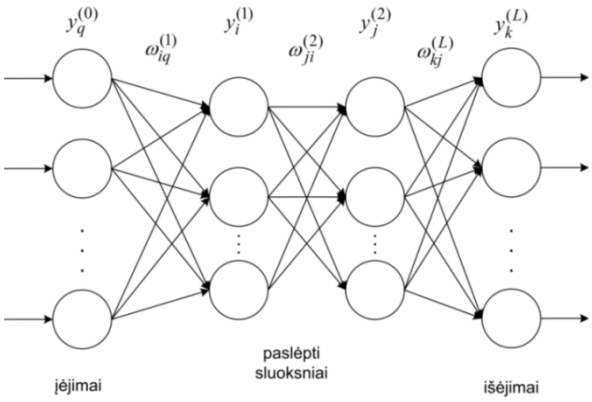
Sammono projekcija (Sammon, 1969) yra netiesinis daugelio kintamųjų objektų atvaizdavimo žemesnio matavimo erdvėje metodas. Jo idėja: atvaizduoti

daugiamačius vektorius mažesnio matavimo erdvėje išlaikant santykinai panašius atstumus tarp vektorių. Sammono metodas minimizuoja projekcijos paklaidą (Sammono paklaidą) E_S:

$$E_S = \frac{1}{\sum_{\mu=1}^{m-1}\sum_{\nu=\mu+1}^{m} d^*_{\mu\nu}} \sum_{\mu=1}^{m-1}\sum_{\nu=\mu+1}^{m} \frac{[d^*_{\mu\nu} - d_{\mu\nu}]^2}{d^*_{\mu\nu}}, \qquad (5.1)$$

čia $d^*_{\mu\nu}$ – atstumas tarp n-mačių vektorių X^μ ir X^ν, $d_{\mu\nu}$ – atstumas tarp d-mačių vektorių Y^μ ir Y^ν, į kuriuos projektuojami vektoriai X^μ ir X^ν ($d < n$). Analizuojamų vektorių skaičius lygus m.

Sammono algoritmo vienas iš trūkumų yra tas, kad atsiradus analizuojamų vektorių aibėje naujam taškui, norint ir jį atvaizduoti mažesnio matavimo erdvėje, reikia perskaičiuoti visų jau atvaizduotų taškų projekcijas. J. Mao ir A. K. Jain darbe (Mao, 1995) pasiūlė Sammono paklaidą minimizuoti naudojant tiesioginio sklidimo neuroninius tinklus. Pasiūlyta specifinė „klaidos sklidimo atgal“ mokymo taisyklė, pavadinta SAMANN, kuri leidžia įprastam tiesioginio sklidimo neuroniniam tinklui (5.1 pav.) realizuoti Sammono projekciją mokymo be mokytojo būdu.



5.1 pav. *Tiesioginio sklidimo trijų sluoksnių DNT Sammono projekcijai*

Tarkime, kad $X = (x_1, x_2, ..., x_n)$ yra n-mačiai vektoriai. Jie bus naudojami tinklo mokymui, kaip įėjimų reikšmės. Kiekviename mokymo žingsnyje neuroniniam tinklui pateikiami du n-mačiai vektoriai, o išėjimuose siekiama gauti jų projekcijas d-matėje erdvėje, t.y. vektorius $Y = (y_1, y_2, ..., y_d)$, ($d < n$). Apskaičiuojami atstumi tarp neuroninio tinklo išėjimų, t.y. d-mačių taškų ir nustatoma Sammono paklaidos reikšmė E_S (5.1). Šios paklaidos pagrindu

keičiami neuronų svoriai. Norint gauti analizuojamų duomenų projekcijas plokštumoje, naudojamas SAMANN neuroninis tinklas turintis du išėjimus.

Pažymėkime j-otojo neurono išėjimą l-tajame sluoksnyje $y_j^{(l)}$, $j=1,...,n_l$, $l=0,...,L$; čia n_l yra l-tojo sluoksnio neuronų skaičius, L – neuronų sluoksnių skaičius. Neuroninio tinklo įėjimai – tai įėjimo vektoriaus komponentės, t.y. $y_j^{(0)}=x_j$, $j=1,...,n$. Jungties tarp i-tojo neurono $(l-1)$-ajame sluoksnyje ir j-tojo neurono l-tajame sluoksnyje svorį žymėkime $w_{ji}^{(l)}$. j-tojo neurono l-tajame sluoksnyje slenksčio reikšmę (*angl. bias*) pažymėkime $w_{j0}^{(l)}$, o $y_0^{(l)}=1$. Kiekvieno neurono išėjimo reikšmei skaičiuoti naudojama sigmoidinė funkcija $f(a)=\dfrac{1}{1+e^{-a}}$, kurios reikšmių intervalas yra (0; 1); čia a – visų neuronų įėjimų ir jų svorių sandaugų suma. Taigi, j-tojo neurono l-tajame sluoksnyje išėjimas išreiškiamas taip $y_j^{(l)}=f\left(\sum\limits_{i=0}^{n_{l-1}} w_{ji}^{(l)} y_i^{(l-1)}\right)$, $l=1,...,L$.

Pakeiskime naudojamus žymėjimus. Anksčiau vektoriaus Y^{μ} komponentės buvo žymimos taip: $Y^{\mu}=(y_1^{\mu}, y_2^{\mu},..., y_d^{\mu})$. Dabar viršutinis indeksas nurodo sluoksnio numerį neuroniniame tinkle, todėl vektoriaus komponentes išreiškime tokiu būdu $Y^{\mu}=(y_1(\mu), y_2(\mu),..., y_d(\mu))$.

Euklido atstumas $d_{\mu\nu}$ tarp dviejų d-mačių vektorių Y^{μ} ir Y^{ν} apskaičiuojamas pagal formulę:

$$d_{\mu\nu}=\left\{\sum_{k=1}^{d}\left[y_k^{(L)}(\mu)-y_k^{(L)}(\nu)\right]^2\right\}^{\frac{1}{2}}.$$

Atkreipkime dėmesį į tai, kad dėl sigmoidinės funkcijos reikšmių intervalo, kiekvieno išėjimo intervalas yra (0; 1). Jei įėjimo reikšmių intervalas yra platus, tai naudojant projekcijos algoritmą, neįmanoma išlaikyti atstumų tarp vektorių. Todėl, visi įėjimo vektoriai normuojami, norint suvienodinti atstumus tarp vektorių pradinėje erdvėje ir projektinėje erdvėje.

Pažymėkime $\lambda=\dfrac{1}{\sum\limits_{\mu=1}^{m-1}\sum\limits_{\nu=\mu+1}^{m} d_{\mu\nu}^{*}}$. Dydžio λ reikšmė nepriklauso nuo tinklo, taigi ji gali būti apskaičiuota iš anksto. Tuomet Sammono paklaida $E_{\mu\nu}$ dviem taškams μ ir ν skaičiuojama pagal (5.2) formulę, o visiems taškams pagal (5.3) formulę.

$$E_{\mu\nu} = \lambda \frac{\left[d^*_{\mu\nu} - d_{\mu\nu} \right]^2}{d^*_{\mu\nu}}, \tag{5.2}$$

$$E_S = \sum_{\mu=1}^{m-1} \sum_{\nu=\mu+1}^{m} E_{\mu\nu} . \tag{5.3}$$

Dydis $E_{\mu\nu}$ yra proporcingas atstumų tarp μ-tojo ir ν-tojo vektorių pokyčiams, todėl jis labiau tinkamas svorių keitimo taisyklėms nustatyti. J. Mao ir A. K. Jain daugiasluoksniam tiesioginio sklidimo tinklui pasiūlė svorių keitimo taisyklę, kuri minimizuoja Sammono paklaidą ir yra pagrįsta gradientinio nusileidimo metodu.

Išėjimo sluoksniui $(l = L)$ gauname:

$$\begin{aligned} \frac{\partial E_{\mu\nu}}{\partial w_{kj}^{(L)}} &= \left(\frac{\partial E_{\mu\nu}}{\partial d_{\mu\nu}} \right) \left(\frac{\partial d_{\mu\nu}}{\partial \left[y_k^{(L)}(\mu) - y_k^{(L)}(\nu) \right]} \right) \left(\frac{\partial \left[y_k^{(L)}(\mu) - y_k^{(L)}(\nu) \right]}{\partial w_{kj}^{(L)}} \right) = \\ &= \left(-2\lambda \frac{d^*_{\mu\nu} - d_{\mu\nu}}{d^*_{\mu\nu}} \right) \left(\frac{y_k^{(L)}(\mu) - y_k^{(L)}(\nu)}{d_{\mu\nu}} \right) \times \\ &\times \left(f'(a_{k,\mu}^{(L)}) y_j^{(L-1)}(\mu) - f'(a_{k,\nu}^{(L)}) y_j^{(L-1)}(\nu) \right) \end{aligned} \tag{5.4}$$

Čia $f'(a_{k,\square}^{(L)})$ yra k-tojo elemento L-tajame sluoksnyje (išėjimo sluoksnyje) sigmoidinės funkcijos išvestinė pagal šio neurono įėjimą $(a_{k,\square})$:

$$f'(a_{k,\square}) = (1 - y_k^{(L)}(\cdot)) y_k^{(L)}(\cdot) . \tag{5.5}$$

Tegu

$$\delta_k^{(L)}(\mu,\nu) = -2\lambda \frac{d^*_{\mu\nu} - d_{\mu\nu}}{d^*_{\mu\nu} d_{\mu\nu}} [y_k^{(L)}(\mu) - y_k^{(L)}(\nu)], \tag{5.6}$$

$$\Delta_{kj}^{(L)}(\mu) = \delta_k^{(L)}(\mu,\nu)[1 - y_k^{(L)}(\mu)] y_k^{(L)}(\mu) , \tag{5.7}$$

$$\Delta_{kj}^{(L)}(\nu) = \delta_k^{(L)}(\mu,\nu) \left[1 - y_k^{(L)}(\nu) \right] y_k^{(L)}(\nu) . \tag{5.8}$$

Įstačius (5.5)–(5.8) formules į (5.4), gauname:

$$\frac{\partial E_{\mu\nu}}{\partial w_{kj}^{(L)}} = \Delta_{kj}^{(L)}(\mu) y_j^{(L-1)}(\mu) - \Delta_{kj}^{(L)}(\nu) y_j^{(L-1)}(\nu) . \tag{5.9}$$

Taigi išėjimo sluoksnio svorių atnaujinimo taisyklė:

$$\Delta w_{kj}^{(L)} = -\eta \frac{\partial E_{\mu\nu}}{\partial w_{kj}^{(L)}} = -\eta(\Delta_{kj}^{(L)}(\mu) y_j^{L-1}(\mu) - \Delta_{kj}^{(L)}(\nu) y_j^{L-1}(\nu)), \qquad (5.10)$$

čia η – mokymosi greičio parametras.

Panašiai gaunamos bendros svorių atnaujinimo taisyklės kiekvienam paslėptam sluoksniui, $l = 1,...,L-1$:

$$\Delta w_{ji}^{(l)} = -\eta \frac{\partial E_{\mu\nu}}{\partial w_{ji}^{(l)}} = -\eta(\Delta_{ji}^{(l)}(\mu) y_i^{l-1}(\mu) - \Delta_{ji}^{(l)}(\nu) y_i^{l-1}(\nu)), \qquad (5.11)$$

čia

$$\Delta_{ji}^{(l)}(\mu) = \delta_j^{(l)}(\mu)\left[1 - y_j^{(l)}(\mu)\right] y_j^{(l)}(\mu), \qquad (5.12)$$

$$\Delta_{ji}^{(l)}(\nu) = \delta_j^{(l)}(\nu)\left[1 - y_j^{(l)}(\nu)\right] y_j^{(l)}(\nu), \qquad (5.13)$$

ir

$$\delta_j^{(l)}(\mu) = \sum_{k=1}^{d} \Delta_{kj}^{(l+1)}(\mu) w_{kj}^{(l+1)}, \qquad (5.14)$$

$$\delta_j^{(l)}(\nu) = \sum_{k=1}^{d} \Delta_{kj}^{(l+1)}(\nu) w_{kj}^{(l+1)}. \qquad (5.15)$$

Kaip ir standartiniame „klaidos sklidimo atgal“ mokymo algoritme, dydžiai $\delta_j^{(l)}(\mu)$ ir $\delta_j^{(l)}(\nu)$ keičiasi grįžtant iš $(l+1)$-ojo sluoksnio į l-tąjį sluoksnį, atitinkamai μ-tajam ir ν-tajam vektoriams.

Iš (5.10) ir (5.11) formulių seka, kad norint atnaujinti neuroninio tinklo svorius, į tinklą tuo pat metu turi būti paduota vektorių pora. Tai galima padaryti sukonstruojant du identiškus tinklus arba tiesiog laikyti atmintyje visus pirmojo vektoriaus išėjimus prieš pateikiant tinklui antrąjį vektorių.

Apibendrinta SAMMAN algoritmo schema yra tokia:

1) Generuojamos atsitiktinės pradinės SAMANN tinklo svorių reikšmės;
2) Atsitiktinai parenkama *n*-mačių vektorių pora, kuri yra pateikiama tinklui; apskaičiuojami tinklo parametrai;
3) Atnaujinami svoriai pagal (5.10) ir (5.11) formules „sklidimo atgal“ būdu, pradedant nuo išėjimo sluoksnio;
4) Kelis kartus kartojami 2–3 žingsniai; tokiu būdu tinklas apmokomas;
5) Tinklui pateikiami visi vektoriai ir apskaičiuojami tinklo išėjimai; skaičiuojama Sammono paklaida (5.1); jeigu jos reikšmė mažesnė už pasirinktą slenkstį arba iteracijų skaičius viršija nustatytąjį, tuomet algoritmas sustabdomas, priešingu atveju, vėl pradedam nuo 2 žingsnio.

Norint pagreitinti tinklo mokymo procesą, kartais svorių atnaujinimo formulės (5.10) ir (5.11) yra keičiamos, pridedant dar *momentum* reikšmę (3.1 skyrius).

SAMANN algoritmo privalumas yra tas, kad jeigu analizuojamų vektorių aibėje atsiranda naujas *n*-matis taškas *X*, jis pateikiamas jau apmokytam SAMANN tinklui, tinklo išėjime gaunamos taško *Y*, kuris yra *X* projekcija, komponentės. Žinoma, jeigu tų naujų taškų yra daug, po kažkurio laiko tinklą reikia permokyti, rasti naujas svorių reikšmes.

Pastebėsime, kad Sammono projekcijos algoritmui reikalinga informacija iš karto apie visus analizuojamos aibės taškus, t. y. algoritmui turime pateikti visą duomenų aibę. SAMANN tinklui pateikiamos vektorių poros. Tinklas įvertina atstumus tarp tų dviejų taškų (vektorių), tačiau saugoja ir koreguoja turimas žinias apie visos aibės taškus. Mokymo metu tinklui padavus naują vektorių porą, koreguojama informacija apie visų mokymo aibės taškų tarpusavio išsidėstymą.

Eksperimentams ir tyrimams buvo sukurta programinė SAMANN tinklo realizacija.

5.2. Optimalios mokymo parametro reikšmės nustatymas

Atvaizduojant daugiamačius duomenis mažesnio matavimo erdvėje, labai svarbu pasiekti gerų vizualizavimo rezultatų per trumpą laiką, tinklo mokymas turi būti efektyvus ir mokymo algoritmas turi greitai konverguoti. SAMANN neuroninio tinklo mokymui reikia daug skaičiuojamųjų sąnaudų, todėl naujus svorius ir tikslią duomenų projekciją siekiama gauti per trumpą laiką. Analizuojant SAMANN tinklą, pastebėta, kad projekcijos paklaida priklauso nuo skirtingų parametrų. Tyrimai parodė, kad norint pasiekti gerų vizualizavimo rezultatų, reikia teisingai parinkti mokymosi parametro η reikšmę. *Momentum* konstanta neturi didelės įtakos šio neuroninio tinklo mokymo procesui, todėl toliau šiame darbe jos reikšmė nagrinėjama nebus.

Iki šiol buvo teigiama (Mao, 1995), kad SAMANN algoritmas duoda geriausius rezultatus, kai tinklo mokymosi parametro reikšmė η imama iš intervalo (0; 1). J. Mao ir K. Jain (Mao, 1995) savo tyrimuose naudojo tokią mokymosi parametro reikšmę: η =0,7. Tačiau tuo atveju tinklo mokymas vyksta lėtai. Viena iš galimų to priežasčių yra tai, kad SAMANN tinklo atveju intervalas (0; 1) nėra pats geriausias. Taigi tikslinga ieškoti optimalios neuroninio tinklo mokymosi parametro reikšmės, kuri nebūtinai turi būti intervale (0; 1). Buvo atlikti tyrimai su skirtingomis duomenų aibėmis, kiekvieną kartą SAMANN tinklo mokymui parenkant naujas mokymosi parametro reikšmes ir fiksuojant gautas analizuojamų duomenų projekcijos paklaidas. Rezultatai buvo palyginti tarpusavyje.

Eksperimentams buvo naudojamos keturios duomenų aibės:

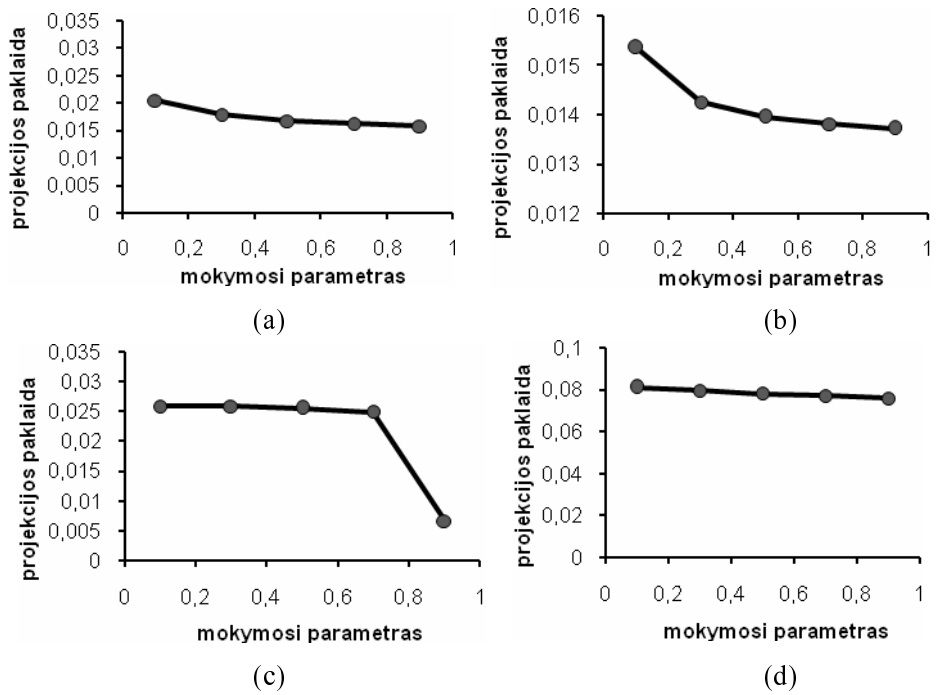
- **Fišerio irisų duomenys (iris dataset)** (Fisher, 1936), kurie kartais vadinami tiesiog irisais arba irisų duomenimis, yra klasikiniai testiniai duomenys, naudojami daugiamačių duomenų analizėje. Buvo išmatuota 150-ies gėlių irisų: vainiklapių pločiai (*angl. petal weight*), vainiklapių ilgiai (*angl. petal height*), taurėlapių pločiai (*angl. sepal weight*), taurėlapių ilgiai (*angl. sepal height*). Matuotos trijų rūšių gėlės: Iris Setosa (I klasė), Iris Versicolor (II klasė) ir Iris Virginica (III klasė). Sudaryti 4-mačiai vektoriai. Įvairias metodais nustatyta, kad pirmos klasės irisai „atsiskiria“ nuo kitų dviejų klasių (antros ir trečios). Antra ir trečia klasės dalinai persipina.
- **Sūrumo duomenų aibė (salinity dataset)** (Ruppert, 1980). Tai vandens sūrumo matavimai, t.y. druskos koncentracija. Aibę sudaro 28 keturmačiai vektoriai.
- **HBK duomenų aibė** (dirbtinė duomenų aibė) (Hawkins, 1984). Tai 75 keturmačiai vektoriai, kurie sudaro 3 atskiras grupes. 1–10 taškai sudaro vieną grupę, 11–14 – antrą grupę ir likę taškai – trečią.
- Atsitiktinai sugeneruotų vektorių aibė. Aibę sudaro 200 keturmačių vektorių $X_i = (x_{i1},...,x_{in}) \in R^4$, $x_{ij} \in (0;1)$.

Šiame skyriuje (o taip pat 6-tame ir 7-tame skyriuose) buvo ieškoma analizuojamų duomenų projekcijų plokštumoje. Atliekant eksperimentus, buvo nagrinėjamas vienasluoksnis tiesioginio sklidimo dirbtinis neuroninis SAMANN tinklas, turintis vieną paslėptą sluoksnį ir du išėjimus (d=2). Visais atvejais buvo imamas vienodas paslėptojo sluoksnio neuronų skaičius (n_2=20), iteracijų skaičius tinklo mokymui M=50000, o taip pat iš anksto fiksuotas pradinių svorių rinkinys. Neuroninio tinklo svorių pradinės reikšmės buvo inicijuojamos parenkant atsitiktines reikšmes iš intervalo $(-0,01;0,01)$. Neuroninio tinklo mokymui naudojamas fiksuotas skaičius mokymo iteracijų. Viena iteracija – tai mokymo proceso dalis, kai visas analizuojamos duomenų aibės vektorių poras panaudojame mokymui po vieną kartą.

Buvo atlikti tyrimai su keturiomis keturmačių vektorių duomenų aibėmis atitinkamai su 28, 75, 150 ir 200 vektoriais (realios duomenų aibės: sūrumo, HBK, irisų; ir atsitiktinai sugeneruotų duomenų aibė). Iš pradžių buvo nagrinėjama daugiamačių duomenų atvaizdavimo tikslumo priklausomybė nuo mokymosi parametro η, kai $\eta \in (0;1)$. SAMANN tinklo mokymui buvo naudojamos tokios η reikšmės: 0,1; 0,3; 0,5; 0,7 ir 0,9. Gauti rezultatai rodo, kad didėjant mokymosi parametro reikšmei, gaunama geresnė (mažesnė) projekcijos paklaida (5.2 pav.). Todėl, buvo atlikti tyrimai su didesnėmis mokymosi parametro reikšmėmis, išeinant už intervalo (0; 1) ribų. Rezultatai iliustruojami 5.3 paveiksle. Pavaizduotos gautos paklaidos, atvaizduojant

analizuojamus duomenis plokštumoje, kai η yra 1, 10, 20, 30, 40, 50, 60, 70, 80, 90 ir 100.

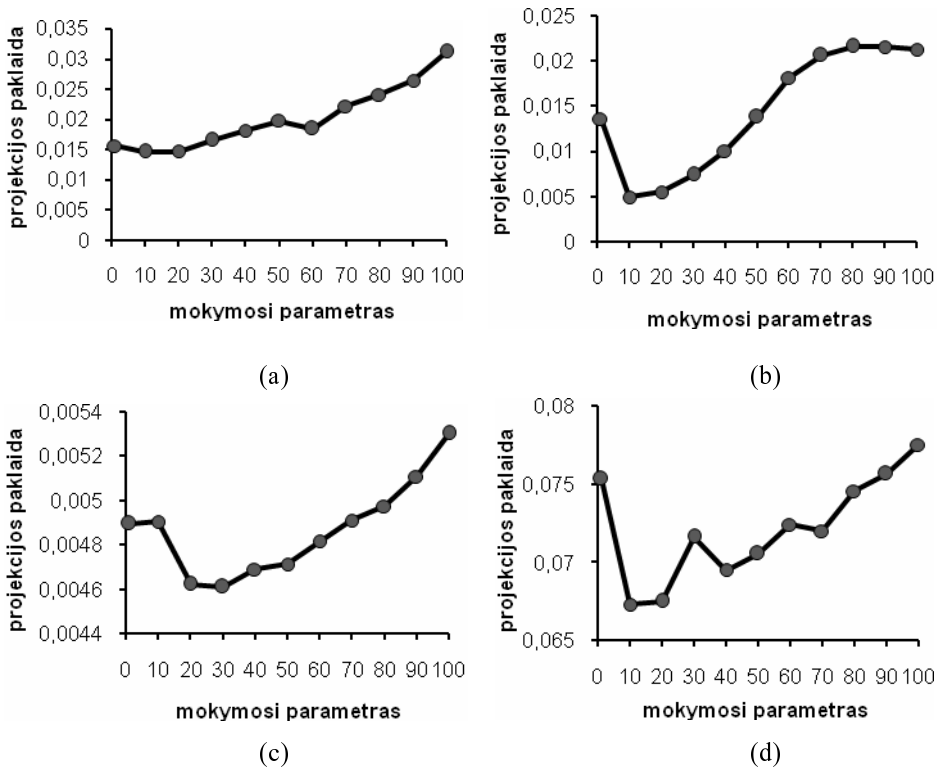
Iš 5.2 ir 5.3 paveikslų matyti, kad geriausi rezultatai gauti, kai $\eta > 1$, bei mokymosi parametro optimali reikšmė nagrinėjamoms duomenų aibėms yra intervale [10; 30]. Sūrumo, HBK duomenų aibėms optimali mokymosi parametro reikšmė yra $\eta = 10$, irisų duomenų aibei – $\eta = 30$. Atsitiktinai sugeneruotų duomenų aibei optimali mokymosi reikšmė gali būti imama iš intervalo [10; 40]. Esant tokioms mokymosi parametro reikšmėms, gaunami geri vizualizavimo rezultatai, duomenys vizualizuojami tiksliau. Taigi, esant fiksuotam iteracijų skaičiui, geresni vizualizavimo rezultatai gali būti pasiekiami per trumpesnį laiką, negu imant mokymosi parametro reikšmes iš intervalo (0; 1).



5.2 pav. *Projekcijos paklaidos priklausomybė nuo mokymosi parametro reikšmės* η, $\eta \in (0;1)$ *(a – sūrumo, b – HBK, c – irisų, d – atsitiktinai sugeneruotų duomenų aibės)*

Atliekant tyrimus buvo fiksuojamas algoritmo vykdymo laikas ir gautos analizuojamų duomenų projekcijos paklaidos kiekvienos iteracijos metu. 5.4 – 5.11 paveiksluose parodytos minėtų keturių duomenų aibių projekcijos paklaidos priklausomybės nuo iteracijų skaičiaus, kai SAMANN tinklo mokymui buvo

naudojamos skirtingos mokymosi parametro η reikšmės. Paveiksluose parodyti tik keli atvejai: $\eta = 0{,}1$, $\eta = 0{,}5$, $\eta = 0{,}9$ (5.4, 5.5, 5.6, 5.7 pav.), ir $\eta = 1$, $\eta = 10$, $\eta = 30$, $\eta = 50$ (5.8, 5.9 , 5.10, 5.11 pav.). 5.8(c) paveikslas patikslina 5.8(b) paveikslą, 5.8(c) paveiksle pavaizduotos tik pirmos 1000 iteracijų. Lyginant 5.8(a) paveikslą ir 5.8(c) paveikslą matome, kad didėjant mokymosi parametro reikšmei, projekcijos paklaidos mažėjimas vyksta žymiai greičiau. O tai rodo, kad didinant η, galima gauti tikslesnę analizuojamų duomenų projekciją per žymiai trumpesnį laiką ir pagreitinti SAMANN tinklo mokymo procesą. 5.12 – 5.15 paveikslai iliustruoja nagrinėjamų duomenų aibių paklaidos priklausomybę nuo skaičiavimų laiko, kai SAMANN tinklo mokymui naudojamos skirtingos mokymosi parametro reikšmės.

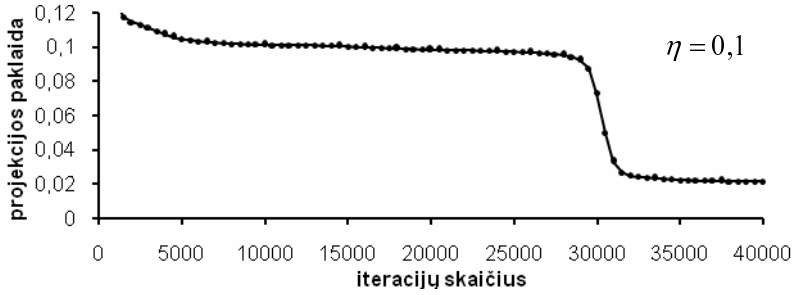


5.3 pav. *Projekcijos paklaidos priklausomybė nuo mokymosi parametro reikšmės* η, $\eta \in [1;100]$ *(a – sūrumo, b – HBK, c – irisų, d – atsitiktinai sugeneruotų duomenų aibės)*

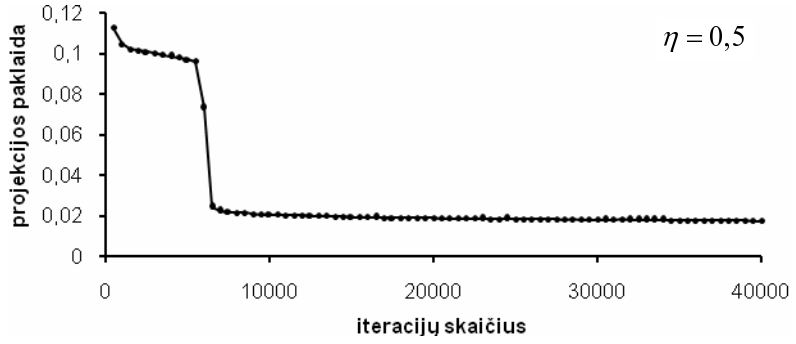
Eksperimentai parodė, kad kuo didesnė mokymosi parametro reikšmė, tuo greičiau pavyksta pasiekti gerus vizualizavimo rezultatus. Tačiau, didėjant

mokymosi parametro reikšmei, didėja paklaidos svyravimai, ir tai gali sukelti tam tikrų SAMANN tinklo mokymo problemų.

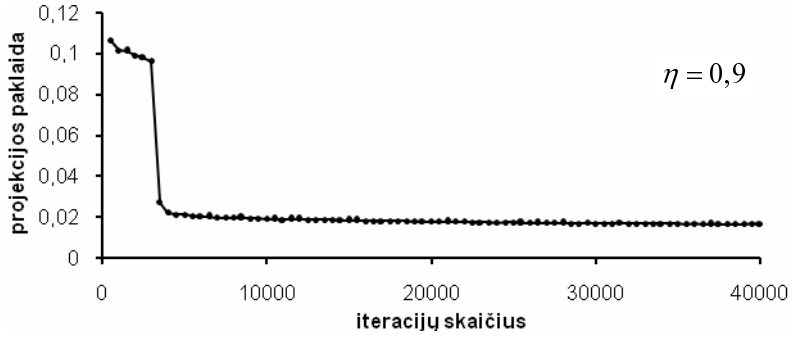
Iš 5.4 – 5.11 paveikslų matyti, kad didinant mokymosi parametro reikšmę, atsiranda paklaidos svyravimai (5.8(d), (e) pav. ir 5.9(c), (d) pav.), kurie gali įtakoti galutinį duomenų projekcijos rezultatą. Tai dar kartą parodo, kad svarbu nustatyti optimalias (tinkamas) mokymosi parametro reikšmes konkrečiai duomenų aibei, su kuriomis projekcijos paklaidos būtų mažiausios.



a)

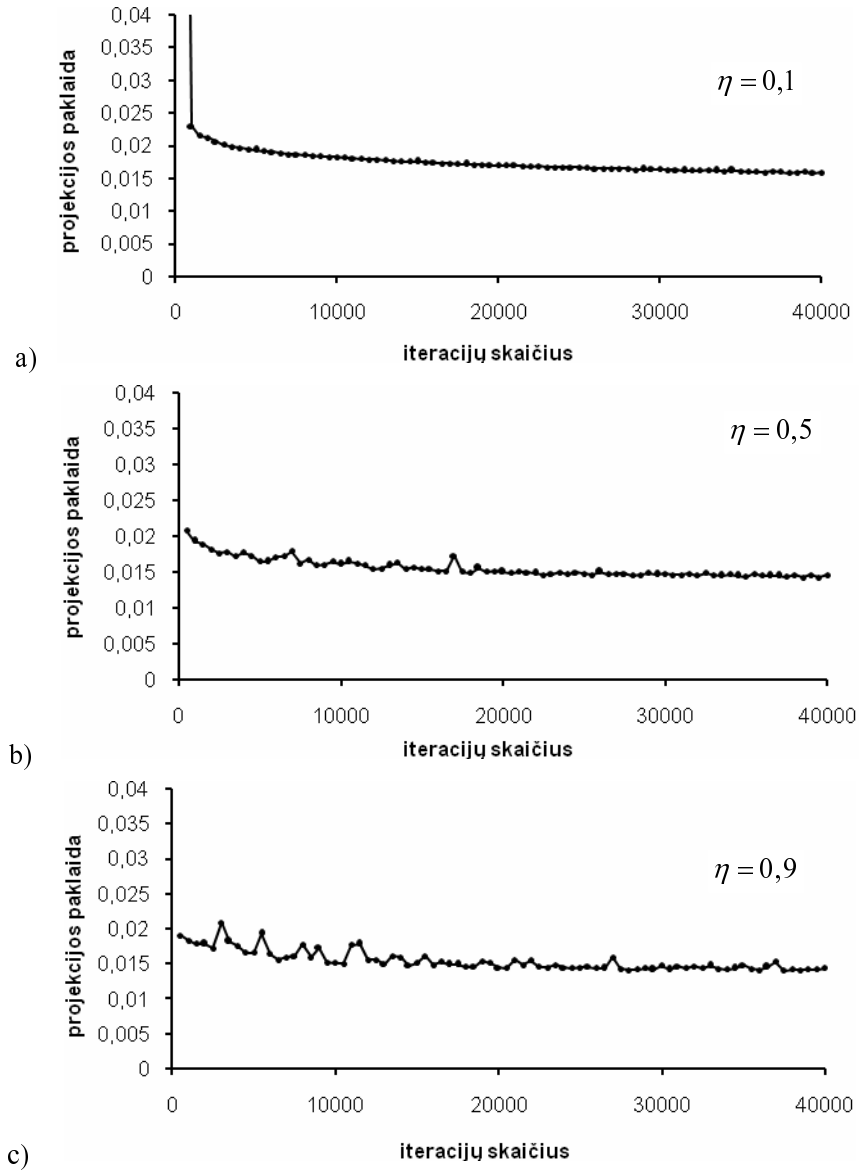


b)

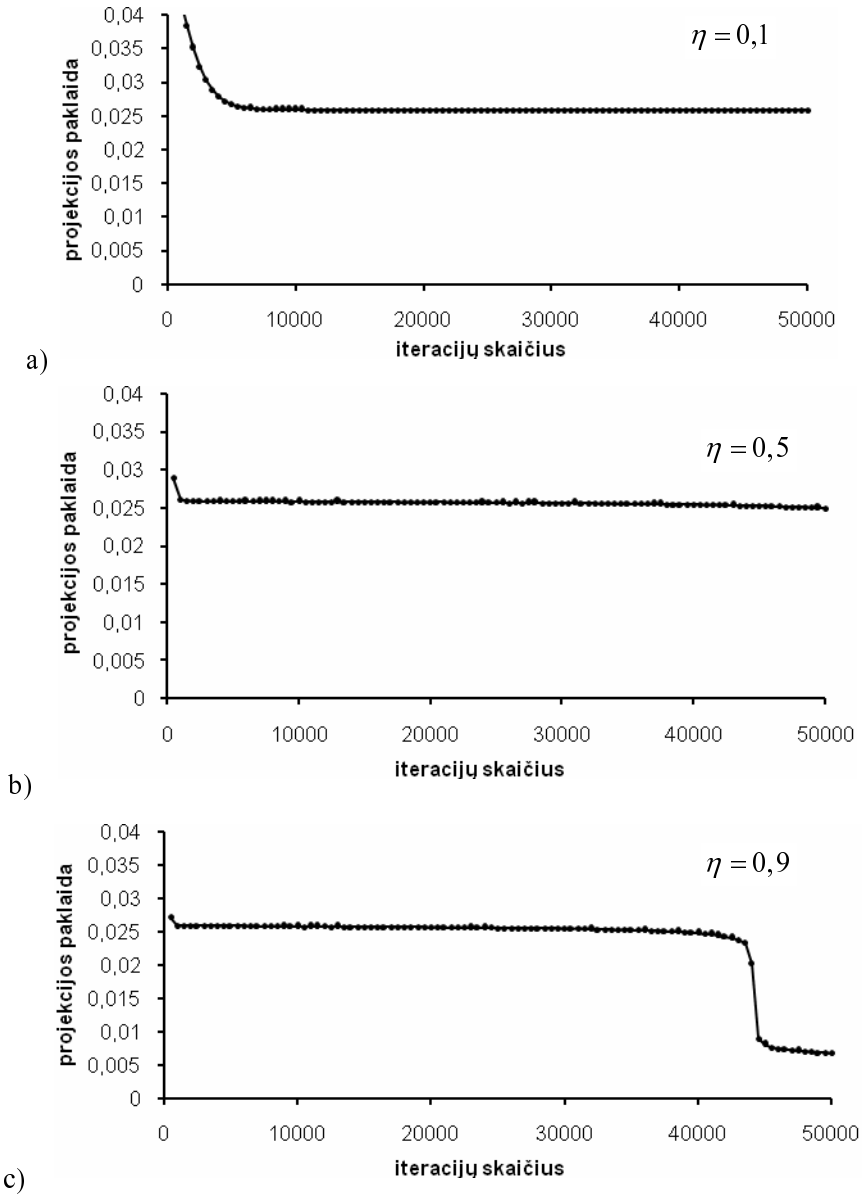


c)

5.4 pav. *Projekcijos paklaidos priklausomybė nuo iteracijų skaičiaus (sūrumo duomenų aibei),* $\eta \in (0,1)$



5.5 pav. *Projekcijos paklaidos priklausomybė nuo iteracijų skaičiaus (HBK duomenų aibei),* $\eta \in (0,1)$

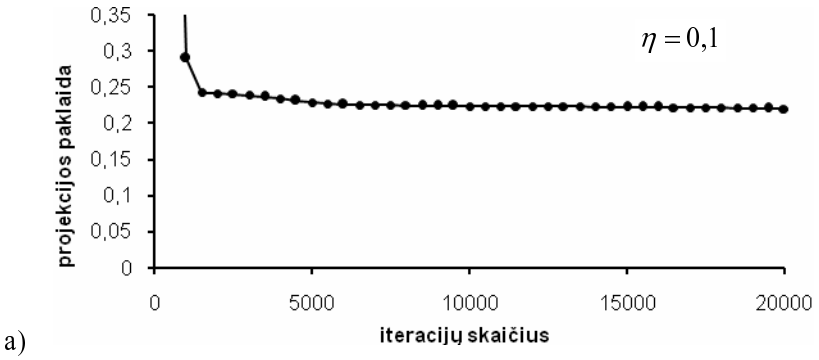


a)

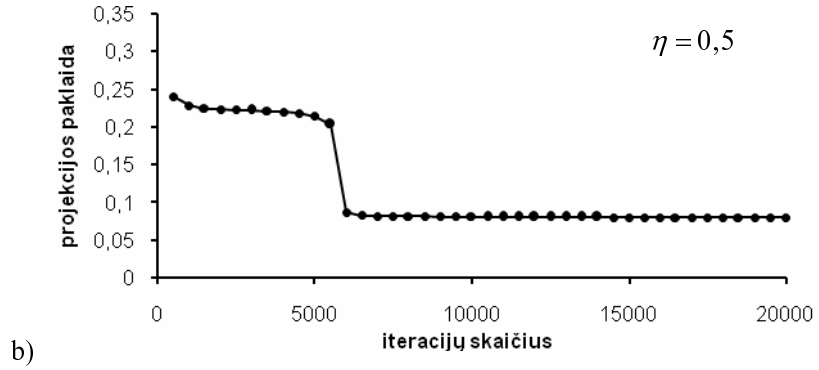
b)

c)

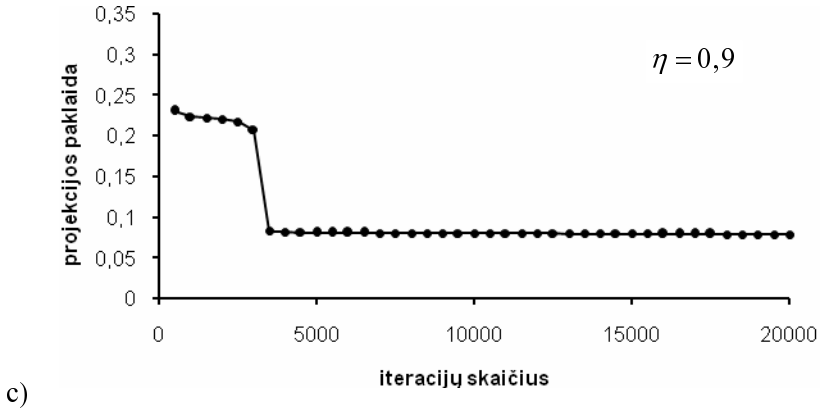
5.6 pav. *Projekcijos paklaidos priklausomybė nuo iteracijų skaičiaus (Irisų duomenų aibei),* $\eta \in (0,1)$



a)

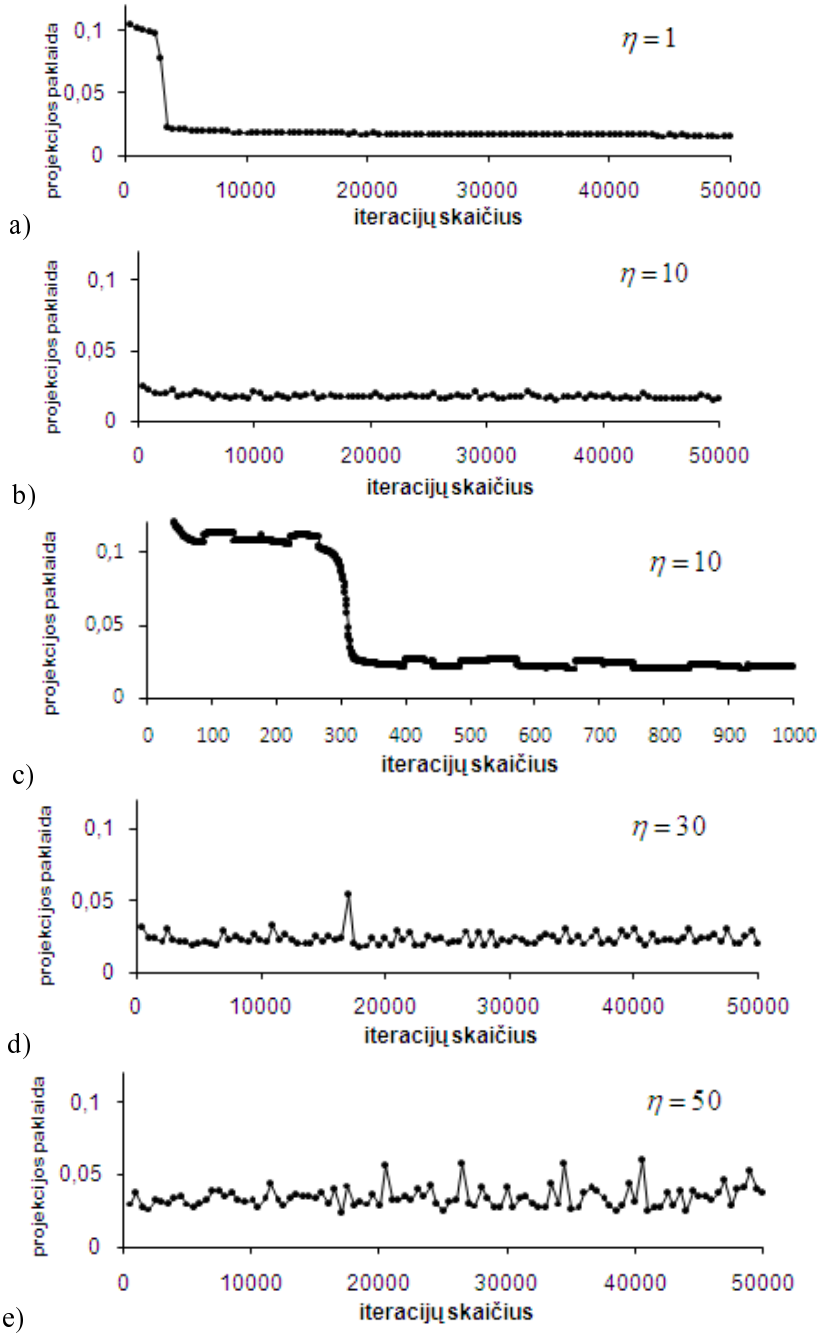


b)

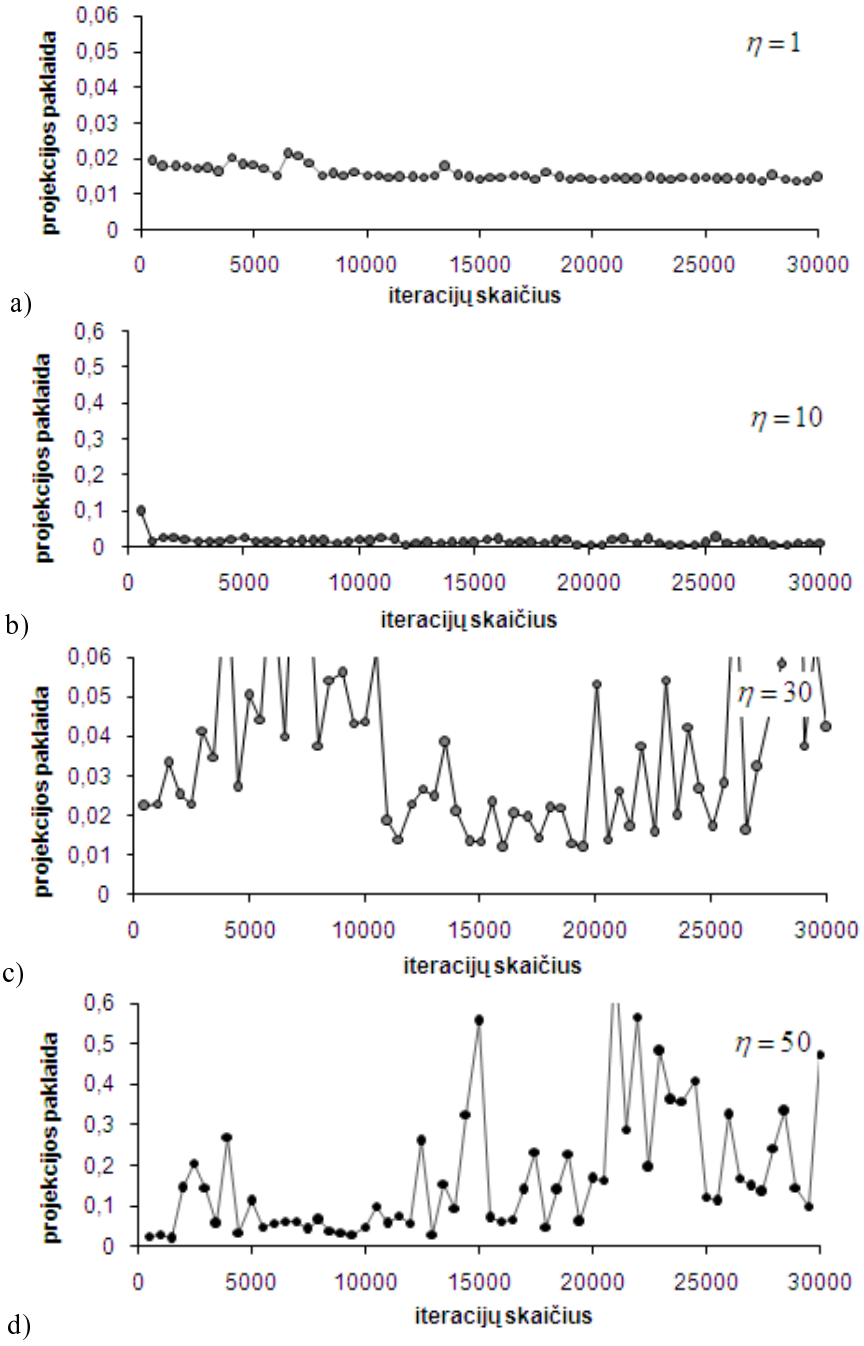


c)

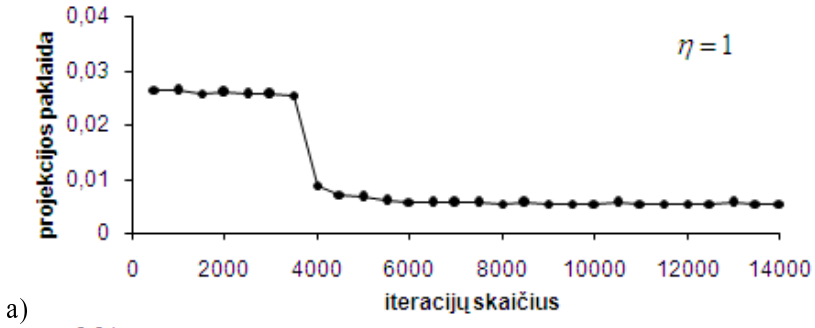
5.7 pav. *Projekcijos paklaidos priklausomybė nuo iteracijų skaičiaus (atsitiktinai sugeneruotų duomenų aibei),* $\eta \in (0,1)$



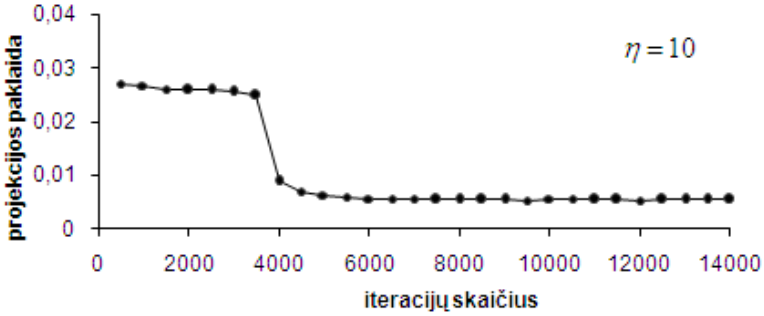
5.8 pav. *Projekcijos paklaidos priklausomybė nuo iteracijų skaičiaus (sūrumo duomenų aibei),* $\eta \in [1,100)$



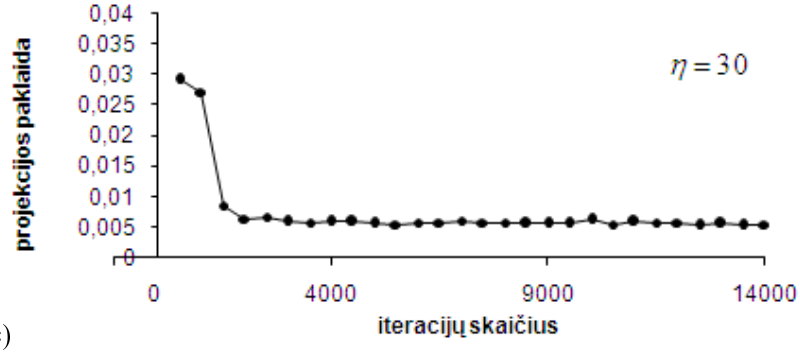
5.9 pav. *Projekcijos paklaidos priklausomybė nuo iteracijų skaičiaus (HBK duomenų aibei),* $\eta \in [1,100)$



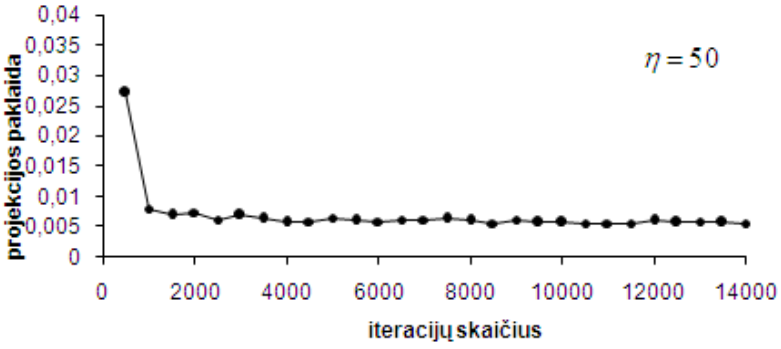
a)



b)

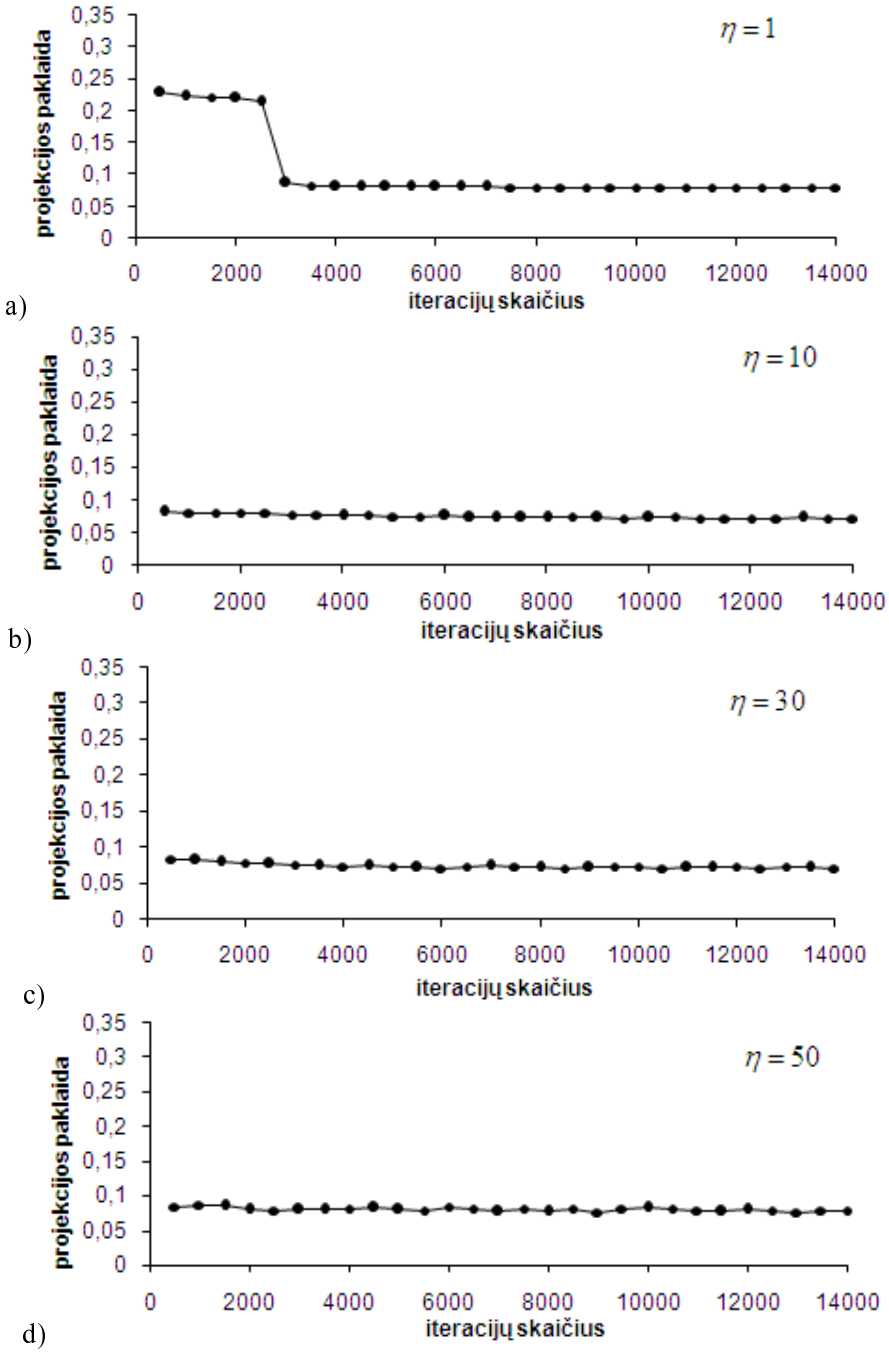


c)

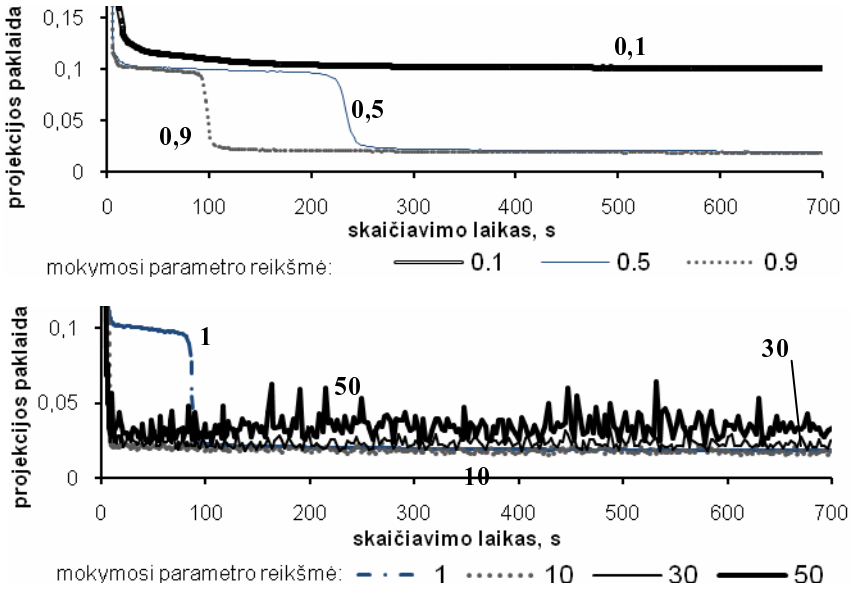


d)

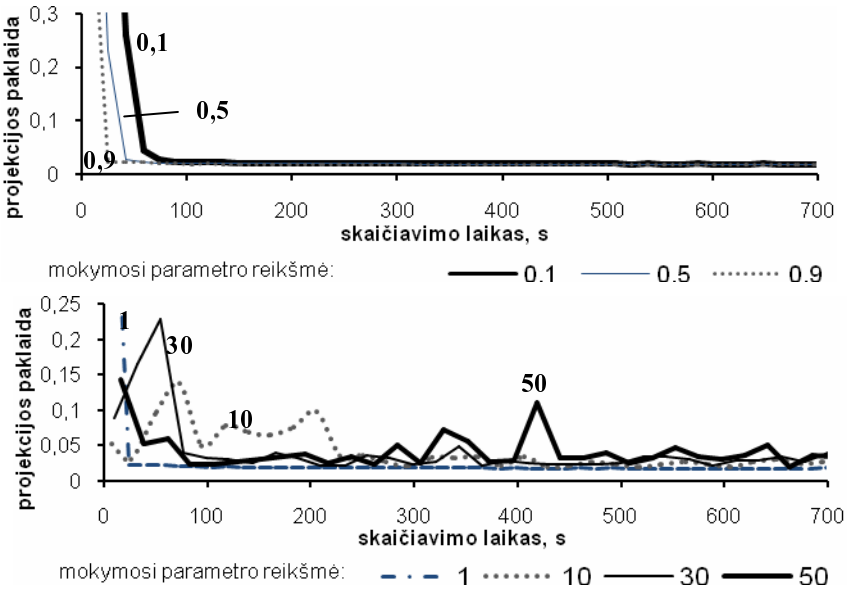
5.10 pav. *Projekcijos paklaidos priklausomybė nuo iteracijų skaičiaus (irisų duomenų aibei),* $\eta \in [1,100)$



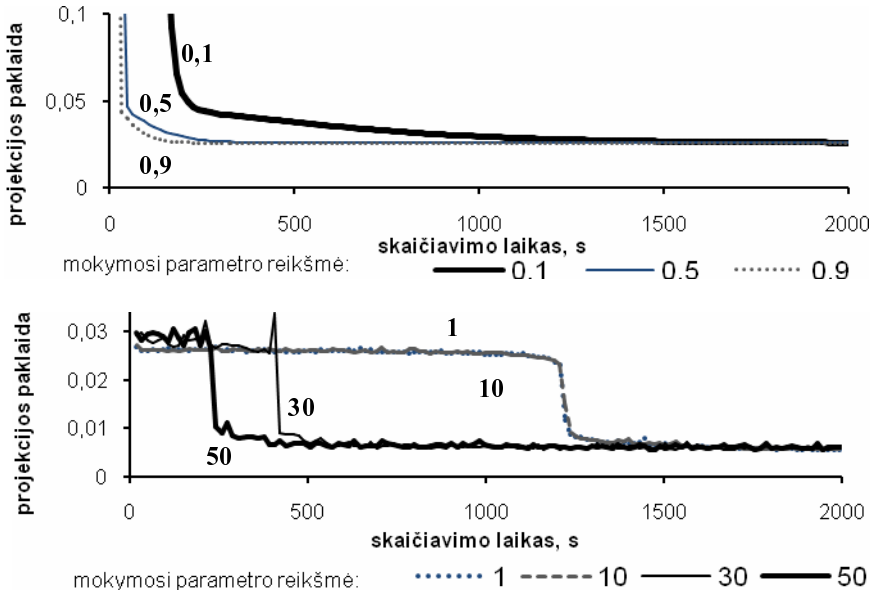
5.11 pav. *Projekcijos paklaidos priklausomybė nuo iteracijų skaičiaus (atsitiktinai sugeneruotų taškų aibei),* $\eta \in [1,100)$



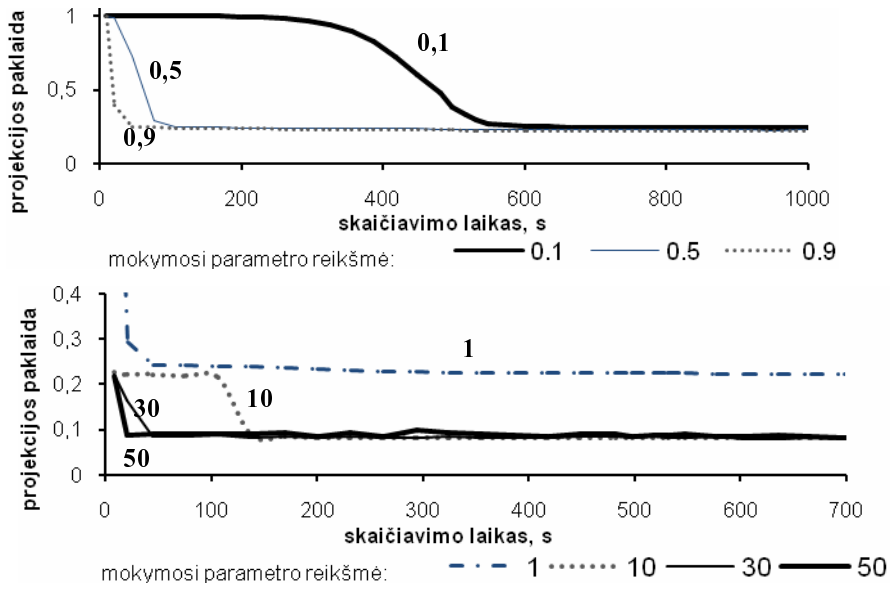
5.12 pav. *Projekcijos paklaidos priklausomybė nuo skaičiavimo laiko (sūrumo duomenų aibė), tinklo mokymui naudojant skirtingas mokymosi parametro reikšmes*



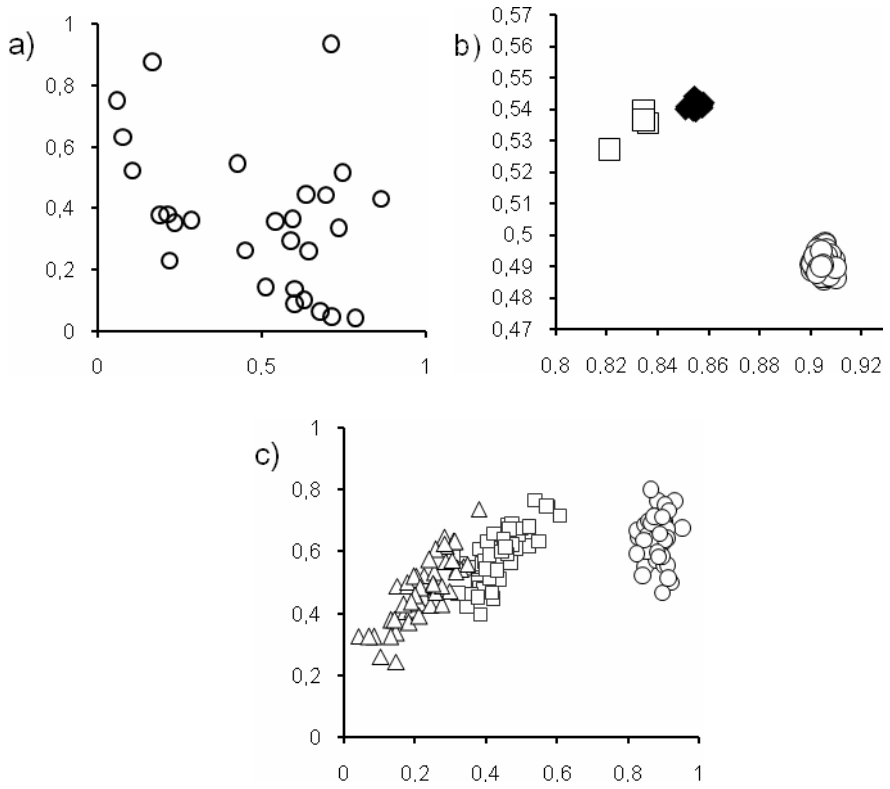
5.13 pav. *Projekcijos paklaidos priklausomybė nuo skaičiavimo laiko (HBK duomenų aibė), tinklo mokymui naudojant skirtingas mokymosi parametro reikšmes*



5.14 pav. *Projekcijos paklaidos priklausomybė nuo skaičiavimo laiko (irisų duomenų aibė), tinklo mokymui naudojant skirtingas η reikšmes*



5.15 pav. *Projekcijos paklaidos priklausomybė nuo skaičiavimo laiko (atsitiktinai sugeneruotų taškų aibė), tinklo mokymui naudojant skirtingas η reikšmes*

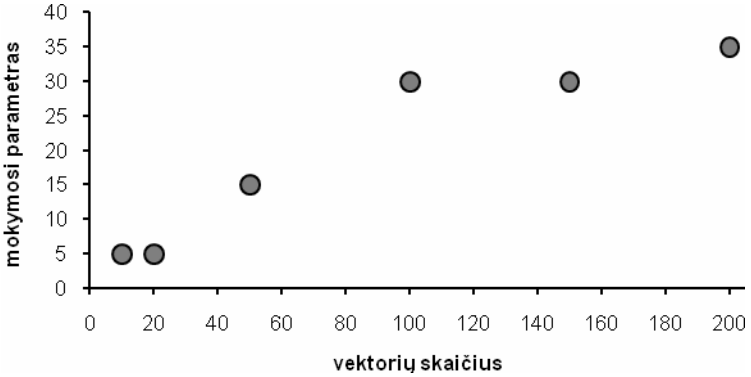


5.16 pav. *Realių duomenų aibių projekcijos plokštumoje, apmokant SAMANN algoritmą (a – sūrumo, b – HBK, c – irisų duomenų aibės)*

5.16 (a, b, c) paveikslai iliustruoja sūrumo, HBK ir irisų duomenų aibių projekcijas plokštumoje, naudojant SAMANN algoritmą. Neuroninio tinklo mokymui buvo naudojamos optimalios nagrinėjamų duomenų aibių mokymosi parametrų reikšmės. Iš paveikslų matyti, kad skaičiuojant analizuojamų daugiamačių duomenų projekcijas SAMANN algoritmų pavyko gerai atvaizduoti duomenis plokštumoje. Vizualiai matosi atskiros HBK ir irisų duomenų aibių grupės.

Papildomai buvo atlikti eksperimentai, kurių tikslas nustatyti optimalią mokymosi parametro reikšmę ir pažiūrėti, kaip ji keičiasi didėjant duomenų aibės vektorių skaičiui. Eksperimentai buvo atlikti su duomenų aibėmis, sudarytomis iš atsitiktinai sugeneruotų vektorių: aibės sudarytos atitinkamai iš 10, 20, 50, 100, 150 ir 200 keturmačių vektorių. Visų nagrinėjamų duomenų aibės vizualizavimui buvo naudojami vienodi SAMANN tinklo parametrai: iteracijų skaičius M=10000; paslėpto sluoksnio neuronų skaičius n_2=20; *momentum* reikšmė 0,05;

fiksuotas pradinių tinklo svorių rinkinys. SAMANN tinklo svorių pradinės reikšmės buvo inicijuojamos parenkant atsitiktines reikšmes iš intervalo $(-0,01;0,01)$. Su kiekviena duomenų aibe buvo atlikta 12 eksperimentų su skirtingomis mokymosi parametro reikšmėmis $\eta \in (0,5;50)$. Buvo pasirinktos tokios mokymosi parametro η reikšmės: 0,5; 0,7; 1; 5; 10; 15; 20; 25; 30; 35; 40; 50. Kiekvienai analizuojamai duomenų aibei buvo nustatyta optimali η reikšmė, t.y. reikšmė su kuria gaunama mažiausia projekcijos paklaida. 5.17 paveiksle pavaizduoti gauti skaičiavimų rezultatai. Skirtingoms duomenų aibėms buvo gautos skirtingos mokymosi parametro reikšmės, priklausomai nuo aibės vektorių skaičiaus. Iš 5.17 paveikslo matyti, kad optimalios mokymosi parametro reikšmės analizuojamoms duomenų aibėms yra intervale (5; 30). Tyrimai parodė, kad atsitiktinai sugeneruotų vektorių aibei, didinant parametrų skaičių, didėja optimalios mokymosi parametro reikšmės dydis.



5.17 pav. *Optimalios mokymosi parametro reikšmės priklausomybė nuo duomenų aibės vektorių skaičiaus*

Nagrinėjant SAMANN tinklo mokymo priklausomybę nuo mokymosi parametro η, eksperimentai parodė, kad optimali mokymosi parametro reikšmė yra intervale (5; 30). Pasirenkant tokias mokymosi parametro reikšmes galima žymiai sumažinti skaičiavimų trukmę (t.y. pagreitinti neuroninio tinklo mokymo procesą) ir gauti gerus vizualizavimo rezultatus per trumpesnį laiką, esant fiksuotam iteracijų skaičiui. Mažos mokymosi parametro reikšmės intervale (0; 1) garantuoja stabilų mokymo paklaidos mažėjimą didėjant iteracijų skaičiui. Tuo tarpu, kai mokymosi parametro reikšmė pasirenkama didesnė (kai $\eta \geq 30$), pastebimi tam tikri paklaidos svyravimai (5.12, 5.13 pav.). Tačiau šie svyravimai yra pakankamai maži, kai mokymosi parametras pasirenkamas iš intervalo (5; 30).

5.3. Penktojo skyriaus išvados

Analizuojant SAMANN neuroninį tinklą pastebėta, kad projekcijos paklaida ir tinklo konvergavimas priklauso nuo pasirinktos mokymosi parametro reikšmės. Eksperimentai parodė, kad kuo didesnė mokymosi parametro reikšmė, tuo greičiau pavyksta pasiekti gerus vizualizavimo rezultatus. Tačiau, didėjant mokymosi parametro reikšmei, didėja paklaidos svyravimai, ir tai gali sukelti tam tikrų SAMANN tinklo mokymo problemų.

Nagrinėjant SAMANN tinklo mokymo proceso priklausomybę nuo mokymosi parametro reikšmės (tiriant kelias duomenų aibes), eksperimentiškai nustatyta, kad optimali mokymosi parametro reikšmė yra intervale (5; 30). Pasirenkant tokias mokymosi parametro reikšmes galima žymiai sumažinti skaičiavimų trukmę ir gauti gerus vizualizavimo rezultatus per trumpesnį laiką, esant fiksuotam iteracijų skaičiui. Mažos mokymosi parametro reikšmės intervale (0; 1) garantuoja stabilų mokymo paklaidos mažėjimą didėjant iteracijų skaičiui. Tuo tarpu, kai mokymosi parametro reikšmė pasirenkama didesnė, pastebimi tam tikri paklaidos svyravimai. Tačiau šie svyravimai yra pakankamai maži, kai mokymosi parametras pasirenkamas iš intervalo (5; 30). Jeigu tiriamai duomenų aibei tinkama mokymosi parametro reikšmė yra nežinoma, tai SAMANN tinklo mokymui geriausiai naudoti mokymosi parametro reikšmę, pavyzdžiui, iš intervalo (1; 5). Pasirinkus jį tokiu būdu, galima išvengti didelių paklaidos svyravimų, ir pagreitinti paklaidos konvergavimą, t.y. gauti mažesnę paklaidą esant mažesniam neuroninio tinklo mokymo iteracijų skaičiui.

6

Lygiagrečios SAMANN algoritmo realizacijos

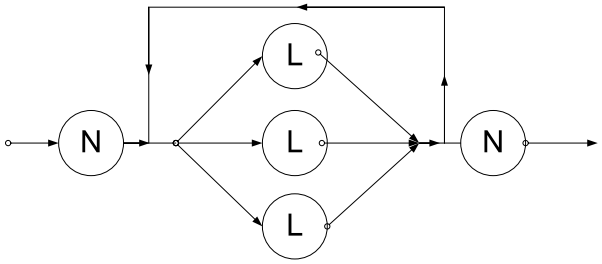
Atliekant praktinius tyrimus, nustatyta, jog SAMANN neuroninio tinklo mokymui reikia daug skaičiuojamųjų sąnaudų. Ši problema gali būti išspręsta naudojant paskirstytos atminties lygiagrečiąsias sistemas ir taikant metodus nuosekliems algoritmams išlygiagretinti. Šiame darbe pasiūlytas lygiagretus algoritmas, leidžiantis tinklo mokymui vienu metu naudoti keletą procesorių. Šiuo atveju lygiagretusis algoritmas leidžia duotąjį uždavinį išspręsti greičiau ir išspręsti sunkesnius uždavinius, nei tai galėtume padaryti naudodami tik vieną procesorių.

Pagrindiniai skyriaus rezultatai paskelbti šiuose straipsniuose: (Medvedev, Dzemyda, 2004a), (Medvedev, Dzemyda, 2004b), (Ivanikovas, Medvedev, Dzemyda, 2007).

6.1. Lygiagretaus SAMANN algoritmo struktūra

Lygiagretus algoritmas išskaido SAMANN tinklo mokymą į nuoseklias („N“) ir išlygiagretinamas („L“) dalis. Kaip pavyzdį, naudosime tris vienodų parametrų procesorius (procesorių skaičius gali būti ir kitas). Dalį skaičiavimų atlieka vienas procesorius, taip vadinamas procesorius-šeimininkas (duomenų įvedimas, vektorių normavimas, pradinis svorių inicializavimas, Sammono

paklaidos skaičiavimas ir t.t.), o kitus pagrindinius skaičiavimus (svorių atnaujinimas, tinklo išėjimų skaičiavimas) lygiagrečiai vykdo keli procesoriai (procesoriai-darbininkai). Skaičiavimų schema pateikta 6.1 paveiksle. Mokymo procesas yra iteracinis, todėl 6.1 paveiksle parodytas ir grįžtamasis ryšys.



6.1 pav. *Uždavinio išskaidymas į nuoseklias ir išlygiagretinamas dalis*

Tarkime, kad yra daugiamačiai duomenys: $X^j = (x_1^j, x_2^j, ..., x_n^j)$, ($X^j \in R^n$), $j = 1,...,m$; m – analizuojamų vektorių skaičius. Tikslas – rasti vektoriaus $X^j = (x_1^j, x_2^j, ..., x_n^j)$ transformaciją $Y^j = (y_1^j, y_2^j, ..., y_d^j)$, $d < n$, mažesnės dimensijos erdvėje R^d (mūsų atveju R^2).

Lygiagretaus SAMANN algoritmo struktūra:

- Neuroninis tinklas apmokomas n-mačių vektorių poromis $(X^i, X^j), i, j = 1,\ldots,m, i \neq j$, naudojant M iteracijų. Šiuo konkrečiu atveju, viena iteracija – tai mokymo proceso dalis, kai visas analizuojamos duomenų aibės vektorių poras panaudojame mokymui po vieną kartą. Atliekamas analizuojamų vektorių normavimas, kad maksimalus atstumas tarp tinklo mokymui naudojamų vektorių būtų lygus vienam. Prieš neuroninio tinklo mokymą pasirenkame kiek procesorių p naudosime ir į kelis blokus skaidysime pradinių duomenų masyvą (jeigu procesorių skaičius yra p, tai duomenų masyvą dalinsime į p-1 blokų).
- Kiekviename mokymo žingsnyje neuroniniam tinklui pateikiami du analizuojamos duomenų aibės vektoriai X^μ ir X^ν. Kiekvienas iš p-1 procesorių, kuris vykdo tinklo svorių reikšmių skaičiavimus, sukuria du identiškus neuroninius tinklus ir visus skaičiavimus toliau atlieka savo lokaliojoje dalyje. Algoritmo realizavimo metu kiekvieno neurono išėjimo reikšmės yra saugomos atmintyje abiem vektoriams.
- Kiekvienos iteracijos pradžioje pradinių duomenų masyvas $\mathbb{X} = \left\{ X^i = (x_1^i, x_2^i, \ldots, x_n^i), i = 1,2,\ldots,m \right\}$ atsitiktiniu būdu permaišomas, taip

kad kiekvieną kartą skaičiavimai būtų atliekami su skirtingomis vektorių poromis. Nulinis procesorius p_0 (procesorius-šeimininkas) paskirsto duomenis tarp likusiųjų p-1 procesorių. Pradinės duomenų aibės elementai dalijami į vienodo (arba beveik vienodo) dydžio blokus. Todėl procesoriai kiekvieną kartą gauna tokio pat dydžio *skirtingų* duomenų blokus. Pateikus vektorius procesoriams, toliau skaičiuojami tinklo išėjimai, t.y. procesoriai vykdo SAMANN algoritmą. Šie išėjimai yra naudojami atnaujinant atitinkamus tinklo svorius (pagal (5.10), (5.11) formules), pradedant nuo išėjimo sluoksnio. Visi procesai atlieka skaičiavimus lygiagrečiai ir tik su lokaliais (jiems paskirtais) duomenimis. Po šio etapo kiekvienas iš p-1 procesorių jau turi savo lokaliosios dalies svorių W rinkinius ir nusiunčia juos nuliniam procesoriui p_0. Nulinis procesorius gauna neuroninio tinklo svorių reikšmes iš procesorių-darbininkų, atnaujina svorius (5.1 skyrius, (5.10) ir (5.11) formulės), skaičiuodamas *vidurkį* tarp atsiustų p-1 svorių komplektų, ir nusiunčia atnaujintas svorių reikšmes atgal visiems procesoriams. Naudojant ką tik apskaičiuotus svorius, randami viso tinklo išėjimo vektoriai $Y^j = (y_1^j, y_2^j), j = 1,...,m$.

- Po kiekvienos iteracijos tinklo įėjimo ir gauti išėjimo vektoriai naudojami projekcijos (Sammono) paklaidai E_s (5.1) skaičiuoti.

Pastebėsime, kad pereinant nuo nuosekliojo algoritmo prie lygiagretaus algoritmo, skaičiuojamieji kaštai sumažėja dėl to, kad skirstant duomenis tarp procesorių, sumažėja vektorių porų, pateikiamų neuroniniam tinklui, skaičius. Tai atsitinka dėl to, kad pradinę duomenų aibę turime skaidyti į blokus. Tokių blokų skaičius lygus $k = p - 1$, kur p – bendras procesorių skaičius. Jeigu nuosekliajame algoritme tinklo mokymui naudojama $\frac{m(m-1)}{2}$ vektorių porų (visos galimos vektorių poros, kai m – tinklo mokymui naudojamų vektorių skaičius), tai skaičiuojant lygiagrečiuoju algoritmu, vieno duomenų bloko vektorių porų skaičius, kuriais apmokomas tinklas, yra $\frac{m(m-k)}{2k^2}$. Gauname, kad naudojant p procesorių, vieno bloko vektorių porų skaičius, lyginant su nuosekliuoju algoritmu, sumažėja maždaug k^2 kartų (kai m yra pakankamai didelis):

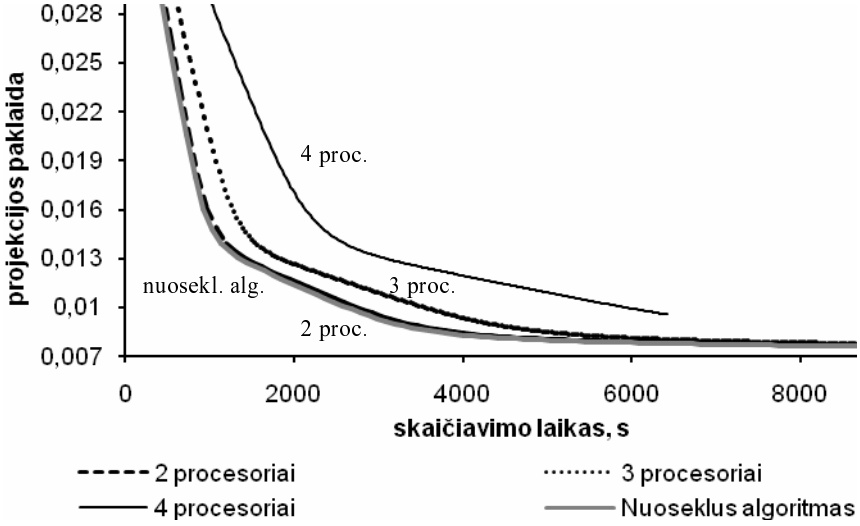
$$\lim_{m\to\infty} \frac{\frac{m(m-1)}{2}}{\frac{m(m-k)}{2k^2}} = k^2 .$$

Kita vertus, negalima tikėtis, kad svorių vidurkinimas, skaičiuojant lygiagrečiuoju algoritmu, duotų tą patį rezultatį kaip ir skaičiuojant nuosekliuoju algoritmu.

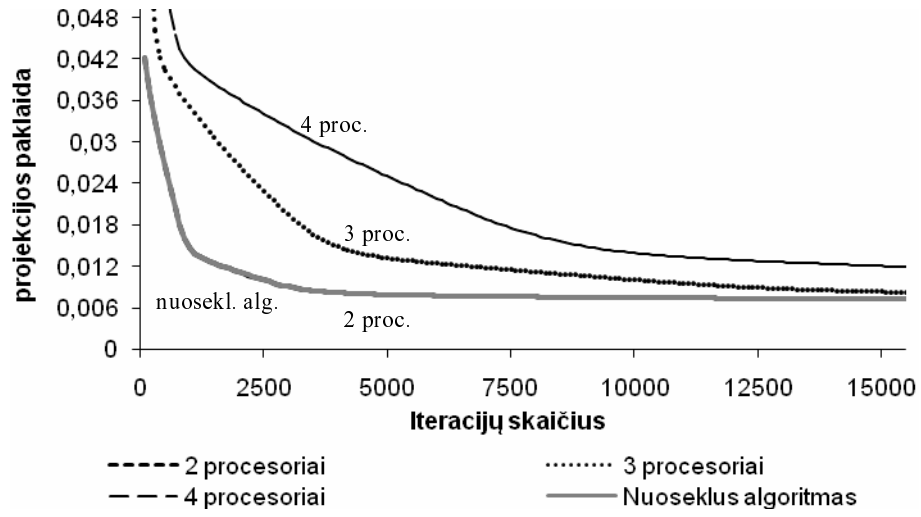
6.2. Lygiagretaus SAMANN algoritmo tyrimas

Algoritmo testavimui buvo naudojama irisų duomenų aibė (Fisher, 1936). Aibę sudaro 150 vektorių (3 klasės po 50 vektorių). Ši aibė yra nedidelė, tačiau gerai tinka lygiagretaus SAMANN algoritmo tyrimams, nes yra žinoma duomenų aibės struktūra.

Analizuojant lygiagretųjį SAMANN algoritmą, buvo nagrinėjamas vienasluoksnis tiesioginio sklidimo dirbtinis neuroninis tinklas, turintis du išėjimus ($d = 2$). Visais atvejais buvo imamas vienodas paslėptojo sluoksnio neuronų skaičius (n_2=10) ir mokymo greičio parametras ($\eta = 0,7$), o taip pat fiksuotas pradinių svorių rinkinys ir fiksuotas iteracijų skaičius M=25000. Lygiagretaus algoritmo efektyvumui eksperimentiškai įvertinti buvo naudojamas kompiuterių tinklas, valdomas MPI (*angl. Message-Passing Interface*) bibliotekos sukurta aplinka (Čiegis, 2005), (Quinn, 2004). Esant toms pačioms pradinėms sąlygoms, skaičiuojant lygiagrečiuoju algoritmu, atlikti tyrimai, kai p=2, 3, 4 (tyrimams buvo naudojami homogeniniai kompiuteriai), ir apskaičiuotos analizuojamų duomenų aibių projekcijos paklaidos (čia p yra bendras procesorių skaičius, t.y. pagrindinė analizuojamų duomenų aibė dalinama į p-1 blokų). Gauti rezultatai palyginti su nuosekliojo algoritmo rezultatais ir pavaizduoti 6.2 ir 6.3 paveiksluose. Lygiagretus algoritmas, kai p=2, iš tikrųjų tinklo mokymui naudoja visus analizuojamos duomenų aibės vektorius, ir buvo pasirinktas tik palyginimui su nuosekliuojo algoritmo rezultatais. Iš 6.2 ir 6.3 paveikslų matyti, kad skaičiuojant nuosekliuoju algoritmu ir lygiagrečiuoju algoritmu (kai p=2) gauti labai panašus projekcijos rezultatai, kadangi neuroninio tinklo mokymui buvo naudojama tiek pat vektorių porų. Skaičiuojant lygiagrečiuoju algoritmu (kai p=3 ir p=4), esant fiksuotam iteracijų skaičiui, gaunamos didesnės projekcijos paklaidos, lyginant su nuosekliojo algoritmo skaičiavimo rezultatais. Didėjant procesorių skaičiui (6.2 pav., kai p=3 ir p=4), esant fiksuotam iteracijų skaičiui, skaičiavimų laikas sumažėja nežymiai, nes didėja duomenų persiuntimo kaštai, reikalingi analizuojamos duomenų aibės projekcijos paklaidai apskaičiuoti. Nuoseklioju algoritmu gaunamos mažesnės projekcijos paklaidos lyginant su lygiagrečiuoju algoritmu, kai $p \geq 3$, nes atskiras procesorius disponuoja viename mokymo žingsnyje mažesnių vektorių porų skaičiumi nei nuoseklus. Skaičiuojant lygiagrečiuoju algoritmu pavyko sutaupyti skaičiavimo laiką, nes tinklo mokymo imtis buvo dalinama į atskirus blokus.

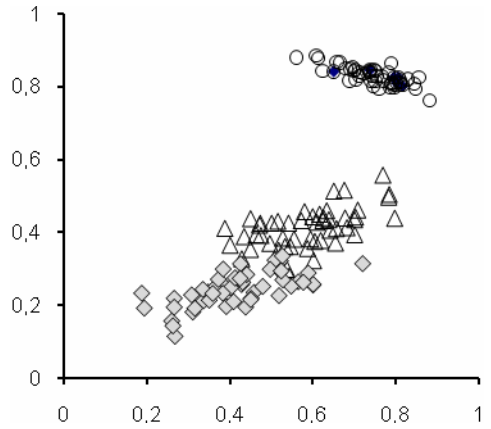


6.2 pav. *Paklaidos priklausomybė nuo skaičiavimo laiko (irisų duomenų aibė)*



6.3 pav. *Paklaidos priklausomybė nuo iteracijų skaičiaus (irisų duomenų aibė)*

6.4 paveiksle pavaizduota irisų duomenų aibės projekcija plokštumoje, skaičiuojant lygiagrečiuoju algoritmu, kai naudojami 3 procesoriai. Lygiagretus algoritmas gerai atvaizdavo pradinę duomenų aibę į plokštumą, vizualiai matosi visos trys klasės.



6.4 pav. *Irisų duomenų aibės projekcija plokštumoje, skaičiuojant lygiagrečiuoju algoritmu, kai naudojami trys procesoriai*

Iš 6.2 paveikslo matyti, kad, pavyzdžiui, esant 2500 sek., nuoseklus algoritmas (skaičiuojant vienu procesoriumi) davė panašias paklaidas, kaip ir skaičiuojant lygiagrečiuoju algoritmu dviem procesoriais. Skaičiuojant lygiagrečiuoju algoritmu, kai naudojami 3 ir 4 proceriai, projekcijos paklaidos gavosi blogesnės. Taigi, iš 6.2 ir 6.3 paveikslų matyti, kad skaičiuojant pateiktu lygiagrečiuoju algoritmu nepavyko pagerinti analizuojamų duomenų projekcijos rezultatų ir (arba) pagreitinti skaičiavimų. Tačiau gauti rezultatai leidžia daryti išvadas, kad kuriant naujas efektyvesnes lygiagrečias SAMANN algoritmo modifikacijas būtina siekti mažinti duomenų persiuntimo kaštus ir racionaliai atlikti atskirais procesoriais apskaičiuotų tinklo svorių apjungimą.

6.3. Lygiagretaus SAMANN tinklo svorių apjungimas

Papildomai buvo atlikti tyrimai, kurių tikslas buvo nustatyti, koks svorių w apjungimo metodas yra geriausias ir kaip jis įtakoja projekcijos rezultatus.

Tinklo svorių atnaujinimui buvo naudojamos tokios taisyklės:

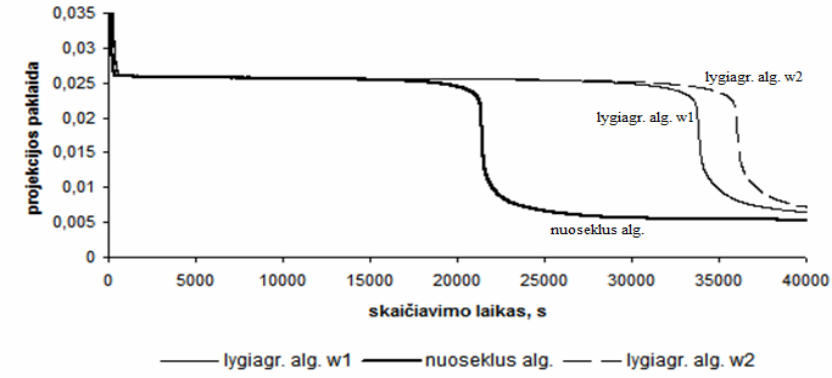
$W1$: $w = \frac{w_1 + w_2}{2}$, t.y. naudojamas svorių vidurkinimas;

$W2$: iš w_1 ir w_2 pasirenkamas tas svorių rinkinys, su kuriuo projekcijos paklaida E_S mažiausia.

Algoritmo testavimui buvo naudojama irisų duomenų aibė (Fisher, 1936). Parinkti tokie SAMANN tinklo parametrai: vienasluoksnis tiesioginio sklidimo

dirbtinis neuroninis tinklas, turintis du išėjimus (d=2), vienodas paslėptojo sluoksnio neuronų skaičius (n_2=10), mokymosi greičio parametras ($\eta = 0{,}5$), o taip pat fiksuotas pradinių svorių rinkinys. Iteracijų skaičius, skaičiuojant nuosekliuoju algoritmu: M=200000, ir skaičiuojant lygiagrečiuoju algoritmu: M=500000.

Esant toms pačioms pradinėms sąlygoms, skaičiuojant lygiagrečiuoju algoritmu, buvo atlikti tyrimai su 3 vienodais procesoriais (procesorius-šeimininkas ir du procesoriai-darbininkai), dalinant analizuojamą aibę į dvi dalis, ir apskaičiuotos duomenų aibės projekcijos paklaidos. Rezultatai palyginti su nuosekliojo algoritmo rezultatais (6.5 pav.). 6.5 paveiksle pavaizduoti rezultatai skaičiuojant nuosekliuoju algoritmu ir lygiagrečiuoju algoritmu (lygiagretus algoritmas $W1$ –tinklo svorių atnaujinimui naudojamas paprastas vidurkinimas; lygiagretus algoritmas $W2$ – svorių atnaujinimui naudojama $W2$ taisyklė, t.y. pasirenkamas tas svorių rinkinys su kuriuo projekcijos paklaida yra mažiausia). Iš 6.5 paveikslo matyti, kad naudojant paprastą vidurkinimą ($W1$) šiek tiek geresni analizuojamos duomenų aibės projekcijos rezultatai gaunami greičiau, nors šiuo atveju tai nelabai įtakoja projekcijos tikslumą.



6.5 pav. *Paklaidos priklausomybė nuo skaičiavimo laiko*

6.4. Nauja lygiagretaus SAMANN algoritmo modifikacija

Atsižvelgiant į prieš tai gautus rezultatus, buvo pasiūlyta kita, modifikuota, lygiagretaus SAMANN algoritmo realizacija. Iš pradžių algoritmas buvo realizuotas naudojant tris homogeninius procesorius (procesorius-šeimininkas ir du procesoriai pagrindiniams skaičiavimams). Skaičiavimai buvo atliekami, naudojant vieną procesorių (pseudo lygiagretinimas), modeliuojant lygiagrečius skaičiavimus.

Lygiagretaus algoritmo mokymo procesą sudaro du etapai:

Pirmas etapas: Kiekvienos iteracijos pradžioje nulinis procesorius (p_0) atsitiktiniu būdu permaišo pradinį duomenų masyvą $A = \mathrm{X} = \left\{ X_i = (x_{i1}, x_{i2} \ldots, x_{in}), i = 1,...,m \right\}$, padalina jį į dvi vienodas arba beveik vienodas dalis (aibės A_1 ir A_2), ir paskirsto tarp likusių procesorių p_1 ir p_2.

Konstruojami du identiški neuroniniai tinklai. Kiekvienas tinklas apmokomas atitinkama duomenų aibės dalimi A_1 arba A_2, naudojant M_1 iteracijų. Šį procesą vykdo lygiagrečiai du procesoriai, pirmasis SAMANN tinklo mokymui naudoja aibės A_1 vektorius, antrasis aibės A_2 vektorius. Po tinklo mokymo (M_1 iteracijų) gaunami kiekvieno atskiro tinklo svorių rinkiniai W_1 ir W_2, išėjimo vektoriai ir atitinkamos visų duomenų aibės A vektorių projekcijų paklaidos E_{S1} ir E_{S2}, kurias daro atitinkami duomenų aibėmis A_1 ir A_2 apmokyti tinklai, kai į tuos tinklus pateikiami visi aibės A vektoriai.

Antras etapas: Toliau skaičiavimus atlieka vienas procesorius p_0. Iš dviejų svorių rinkinių W_1 ir W_2 paliekamas tas, su kuriuo projekcijos paklaida yra mažiausia. Naudojant gautus svorių rinkinius, neuroninis tinklas toliau apmokomas visais duomenų aibės A vektoriais (M_2 iteracijų). Skaičiuojami tinklo išėjimai ir projekcijos paklaida visiems aibės A vektoriams.

Anksčiau aprašytas procesas gali būti kartojamas daug kartų, tol, kol Sammono paklaidos reikšmė taps mažesnė už pasirinktą slenkstį arba iteracijų skaičius viršys nustatytąjį (M_3). Iteracijų skaičius M_3 rodo, kiek kartų kartojami pirmas ir antras lygiagretaus algoritmo mokymo proceso etapai. Po kiekvienos iteracijos M_3 aibės A_1 ir A_2 turėtų būti sudaromos iš naujo. Šiame darbe apsiribota paprasčiausiu atveju, be tokio kartotinumo.

Eksperimentuose buvo naudojamos trys duomenų aibės:

1. Irisų duomenų aibė (Fisher, 1936). Aibę sudaro 150 4-mačių vektorių.
2. Jonosferos duomenų aibė (Sigillito, 1989), (Blake, 1998). Aibę sudaro 351 34-mačių vektorių. Duomenys gauti naudojant radiolokacijos įrenginį, kuris nustato laisvų elektronų buvimą jonosferoje.
3. „Austra“ duomenų aibė (Australian Credit Approval, 2007). Aibę sudaro 690 14-mačių vektorių apie kreditines korteles ir jų panaudojimą. Kiekvienas vektorius priklauso vienai iš dviejų klasių.

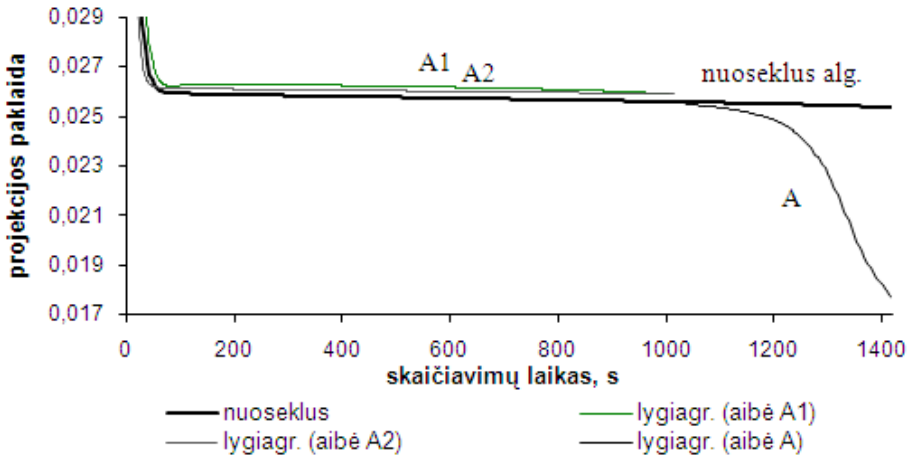
Atliekant eksperimentus buvo nagrinėjamas vienasluoksnis tiesioginio sklidimo dirbtinis neuroninis tinklas, turintis du išėjimus ($d = 2$). Vizualizuojant analizuojamas aibes, buvo naudojami tokie SAMANN tinklo parametrai: paslėptojo sluoksnio neuronų skaičius n_2=20 (irisų aibės duomenims) ir n_2=10 (jonosferos ir „austra“ duomenų aibių duomenims); mokymosi parametras $\eta = 5$;

momentum reikšmė 0,05 ir fiksuotas pradinių svorių rinkinys. Iteracijų skaičius, apmokant SAMANN tinklą aibių A_1 ir A_2 vektoriais, buvo pasirinktas skirtingas (priklausomai nuo duomenų aibės dydžio ir nuo rezultatų, skaičiuojant nuosekliuoju algoritmu): irisų duomenų aibei – 10000, „austra“ ir jonosferos duomenų aibėms – 400.

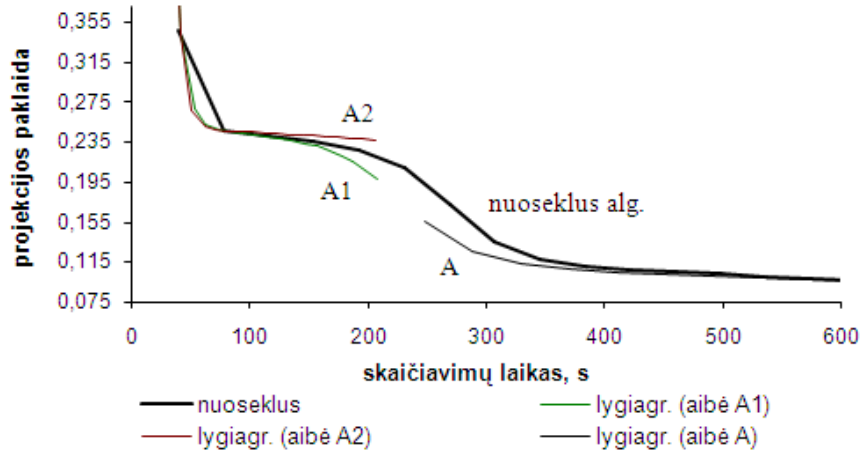
Esant toms pačioms pradinėms sąlygoms, skaičiuojant lygiagrečiuoju algoritmu, buvo atlikti tyrimai su 3 vienodais procesoriais (procesorius šeimininkas p_0, ir procesoriai p_1 ir p_2), dalinant pradinę aibę į dvi dalis, ir apskaičiuotos analizuojamų duomenų projekcijos paklaidos. Gauti rezultatai palyginti su nuosekliojo algoritmo rezultatais ir pavaizduoti 6.6, 6.7, 6.8 paveiksluose. Paveiksluose iš pradžių pavaizduoti lygiagretaus algoritmo rezultatai (projekcijos paklaidos), gauti neuroninį tinklą apmokant aibių A_1 ir A_2 vektoriais, esant fiksuotas iteracijų skaičius (pirmas etapas). Vėliau projekcijos paklaida skaičiuojama tinklą apmokant visais aibės A vektoriais (antras etapas).

Iš 6.6, 6.7, 6.8 paveikslų matyti, kad modifikuotas lygiagretus algoritmas, dalinant pradinę aibę į dvi vienodas dalis, leidžia pasiekti geresnius vizualizavimo rezultatus per trumpesnį laiką (lyginant su nuosekliuoju algoritmu). Analizuojamų duomenų projekcijos paklaidos mažėjimas vyksta greičiau, skaičiuojant lygiagrečiuoju algoritmu.

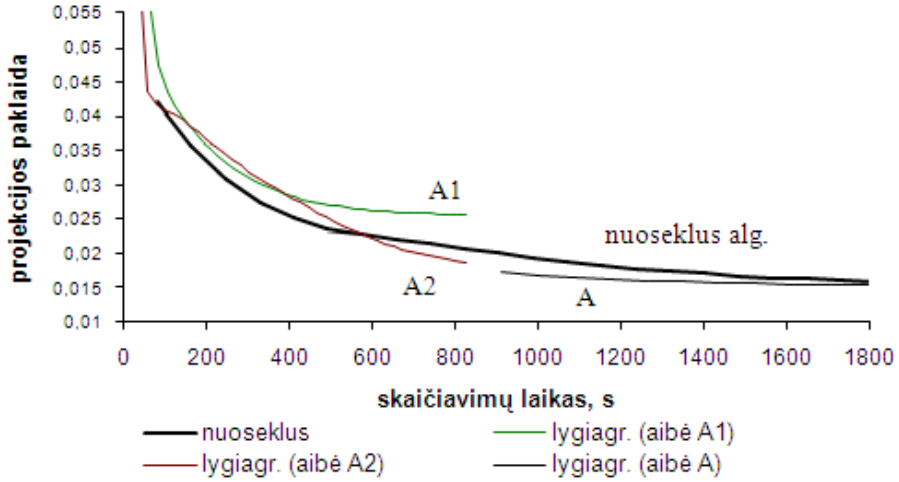
Tyrimai parodė (6.6–6.11 pav.), kad dalinant pradinę duomenų aibę į k dalių, gaunami geresni rezultatai, lyginant su nuosekliuoju algoritmu. Dalinant aibę į vienodas dalis, SAMANN tinklas iš pradžių apmokomas mažesniu vektorių porų skaičiumi. Ir tik po tam tikro iteracijų skaičiaus, kai neuroninio tinklo svorių reikšmės būna jau nusistovėjusios, tinklas toliau apmokomas visais aibės A vektoriais.



6.6 pav. *Projekcijos paklaidos priklausomybė nuo skaičiavimų laiko irisų duomenų aibei (dalinant pradinę aibę į 2 dalis)*



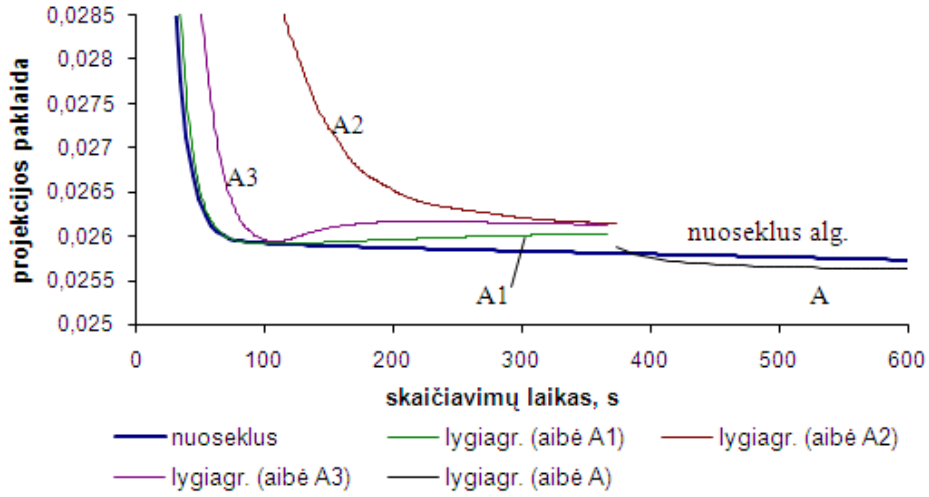
6.7 pav. *Projekcijos paklaidos priklausomybė nuo skaičiavimų laiko jonosferos duomenų aibei (dalinant pradinę aibę į 2 dalis)*



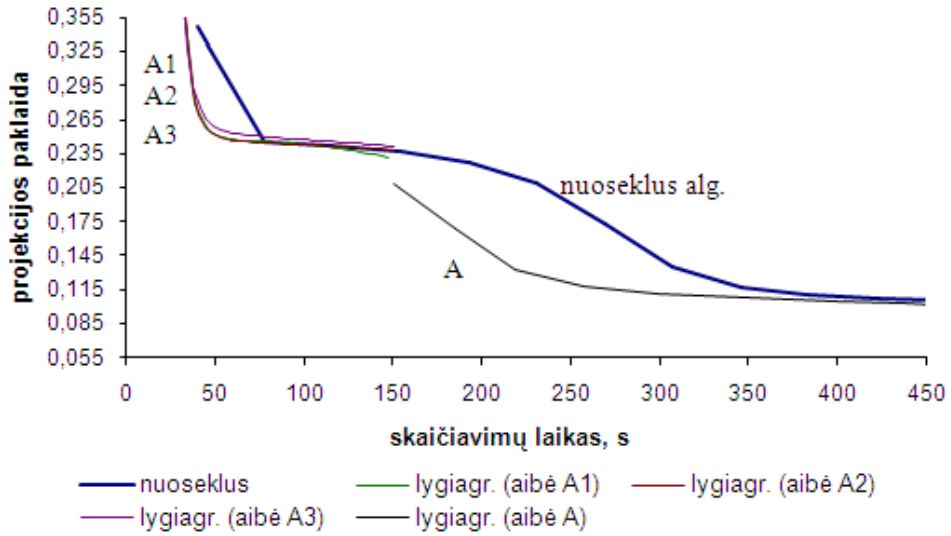
6.8 pav. *Projekcijos paklaidos priklausomybė nuo skaičiavimų laiko „austra“ duomenų aibei (dalinant pradinę aibę į 2 dalis)*

Eksperimentai rodo (6.7, 6.8, 6.10, 6.11 pav.), kad galima rasti tokį analizuojamos duomenų aibės poaibį, kuriuo mokant SAMANN tinklą mažesnės projekcijos paklaidos gaunamos greičiau, negu atveju, kai tinklo mokymui imami visi aibės taškai. Pavyzdžiui, 6.11 paveiksle esant tam tikram laiko momentui (700 sek.) gauta projekcijos paklaida yra beveik 30 % mažesnė naudojant vieną iš atsitiktinai parinktų analizuojamos duomenų aibės poaibių, kuris sudaro 1/3 visos aibės taškų (lyginant su nuosekliuoju algoritmu gauta projekcijos paklaida,

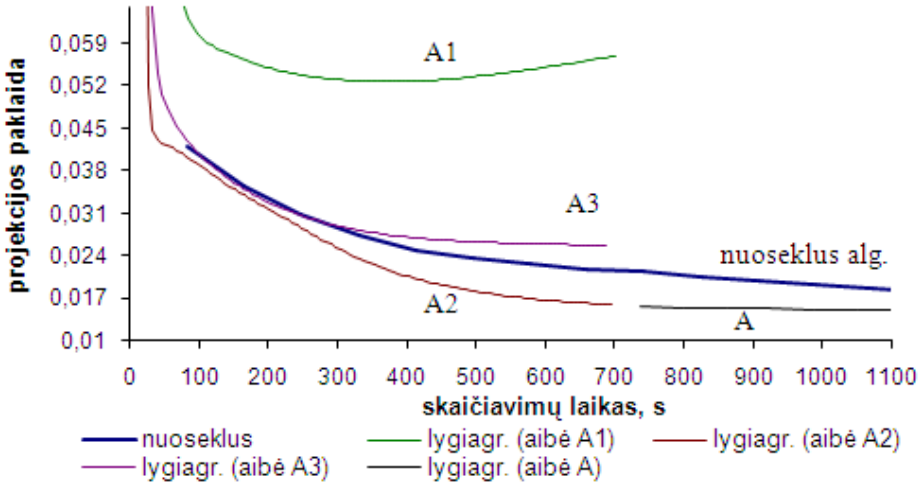
kai tinklo mokymui naudojami visi aibės taškai). Kituose paveiksluose parodytuose rezultatuose išlošimas nėra toks didelis.



6.9 pav. *Projekcijos paklaidos priklausomybė nuo skaičiavimų laiko irisų duomenų aibei (dalinant pradinę aibę į 3 dalis)*



6.10 pav. *Projekcijos paklaidos priklausomybė nuo skaičiavimų laiko jonosferos duomenų aibei (dalinant pradinę aibę į 3 dalis)*



6.11 pav. *Projekcijos paklaidos priklausomybė nuo skaičiavimų laiko „austra“ duomenų aibei (dalinant pradinę aibę į 3 dalis)*

6.5. Šeštojo skyriaus išvados

Ištirtos galimybės tinklo mokymui vienu metu naudoti keletą kompiuterių. Tam buvo pasiūlytos kelios lygiagrečios SAMANN algoritmo realizacijos.

Skaičiuojant lygiagrečiuoju algoritmu, kuris analizuojamą aibę dalina į kelias dalis, vykdo nepriklausomai tinklo mokymą atskiromis dalimis, o po to gautus svorius apjungia juos vidurkinant po kiekvienos mokymo iteracijos (kuomet visos galimos atskiros mokymo aibės dalies vektorių poros pateikiamos tinklui vieną kartą), nepavyko pagerinti atvaizdavimo rezultatų ir (arba) pagreitinti skaičiavimų. Tačiau, analizuojant gautus rezultatus, galime daryti išvadas, kad kuriant naujas lygiagrečias SAMANN algoritmo modifikacijas būtina siekti mažinti duomenų persiuntimo kaštus ir racionaliai atlikti atskirais procesoriais apskaičiuotų tinklo svorių apjungimą. Todėl buvo pasiūlyta lygiagrečiojo algoritmo modifikacija, kuri analizuojamą aibę dalina į kelias dalis, vykdo nepriklausomai tinklo mokymą atskiromis dalimis, o po to tam tikro fiksuoto iteracijų skaičiaus parenka geriausią pasiektą rezultatą visos duomenų aibės požiūriu ir baigia mokyti tinklą pilna duomenų aibe. Tyrimai parodė, kad skaičiuojant taip modifikuotu lygiagrečiuoju SAMANN algoritmu galima pasiekti geresnius vizualizavimo rezultatus per trumpesnį laiką (lyginant su nuosekliuoju algoritmu). Tokio tipo mokymo strategijos gali būti taikomos ir didelės apimties duomenų aibių vizualizavimui. Aukščiau aprašyta mokymo

proceso dalis, kuomet analizuojama aibė dalinama į kelias dalis, vykdomas nepriklausomas tinklo mokymas atskiromis dalimis, o po to tam tikro fiksuoto iteracijų skaičiaus parenkamas geriausias pasiektas rezultatas (visos duomenų aibės požiūriu), gali būti kartojama daug kartų, kol paklaidos reikšmė taps mažesnė už pasirinktą slenkstį arba iteracijų skaičius viršys nustatytąjį. Šiuo atveju prieš kiekvieną tokį pakartojimą būtina analizuojamą duomenų aibę perdalinti į dalis iš naujo (atsitiktinai).

Pastebėta, kad pereinant nuo nuosekliojo algoritmo prie lygiagretaus algoritmo, skaičiuojamieji kaštai sumažėja dėl to, kad skirstant duomenis tarp procesorių, sumažėja vektorių porų, pateikiamų neuroniniam tinklui, skaičius. Tai yra dėl to, kad pradinę duomenų aibę turime skaidyti į blokus.

Eksperimentai parodė, kad galima rasti tokį analizuojamos duomenų aibės poaibį, kuriuo mokant SAMANN tinklą, mažesnės projekcijos paklaidos gaunamos greičiau, negu tinklo mokymui naudojant visus aibės taškus.

7

SAMANN tinklo permokymas

Pagrindiniai skyriaus rezultatai paskelbti šiuose straipsniuose: (Medvedev, Dzemyda, 2005c), (Medvedev, Dzemyda, 2006a), (Medvedev, Dzemyda, 2006b), (Medvedev, Dzemyda, 2006c).

7.1. SAMANN tinklo permokymo strategijos

Po SAMANN tinklo mokymo, turint neuroninio tinklo svorių rinkinį, naujas vektorius, parodytas tinklui, atvaizduojamas į plokštumą labai greitai ir pakankamai tiksliai be jokių papildomų skaičiavimų. Tačiau dirbant su dideliais duomenų kiekiais, naujų vektorių gali būti labai daug ir po tam tikro laiko SAMANN tinklą tenka permokyti, kadangi tinklas prisitaiko prie esamų duomenų ir naujus taškus gali atvaizduoti netiksliai. Todėl buvo pasiūlytos daugiamačius duomenis vizualizuojančio neuroninio tinklo permokymo strategijos ir atlikta jų analizė. Tinklo permokymas turi būti efektyvus ir mokymo algoritmas turi greitai konverguoti. SAMANN neuroninio tinklo mokymui reikia daug skaičiuojamųjų sąnaudų, todėl naujus svorius ir tikslią duomenų projekciją siekiama gauti per trumpą laiką.

Duomenų (vektorių) aibę, kurios vektoriai naudojami SAMANN neuroninio tinklo mokymui, pavadinsime *pradine* duomenų (vektorių) aibe. Nauji vektoriai, kurie dar ne buvo naudojami mokymui, sudaro *naują* duomenų (vektorių) aibę.

Šiame skyriuje pasiūlytos ir ištirtos trys strategijos neuroninio tinklo permokymui. Pirmoji strategija tinklo permokymui naudoja visas įmanomas analizuojamų vektorių poras (vektoriai imami ir iš naujos duomenų aibės ir iš pradinės). Antroji strategija iš pradžių tinklo permokymui naudoja tik naujus vektorius, o paskui visas įmanomas vektorių poras. Trečiojoje strategijoje vienas vektorius imamas iš pradinės duomenų aibės, kitas iš naujos duomenų aibės.

Daugiamačius duomenis vizualizuojančio neuroninio tinklo permokymo strategijos:

1) SAMANN tinklas apmokomas N_1 vektoriais iš pradinės duomenų aibės, gaunamas svorių rinkinys W_1, skaičiuojama projekcijos paklaida $E(N_1)$ ir surandamos vektorių projekcijos plokštumoje. Atėjus N_2 naujų vektorių, neuroninis tinklas permokomas visais N_1+N_2 vektoriais, po kiekvienos iteracijos skaičiuojama projekcijos paklaida $E(N_1+N_2)$ (pradinės ir naujos duomenų aibių požiūriu) ir fiksuojamas skaičiavimo laikas. Surandamas SAMANN tinklo svorių rinkinys W_2.

2) SAMANN tinklas apmokomas N_1 vektoriais iš pradinės duomenų aibės, gaunamas svorių rinkinys W_1, skaičiuojama projekcijos paklaida $E(N_1)$ ir surandamos vektorių projekcijos plokštumoje. Atėjus N_2 naujų vektorių, neuroninis tinklas iš pradžių permokomas N_2 vektoriais iš naujos duomenų aibės, paskui visais N_1+N_2 vektoriais. Po kiekvienos iteracijos skaičiuojama projekcijos paklaida $E(N_1+N_2)$ (pradinės ir naujos duomenų aibių požiūriu) ir fiksuojamas skaičiavimo laikas. Surandami SAMANN tinklo svoriai W_2.

3) SAMANN tinklas apmokomas N_1 vektoriais iš pradinės duomenų aibės, gaunamas svorių rinkinys W_1, skaičiuojama projekcijos paklaida $E(N_1)$. Kadangi norint atnaujinti svorius W, neuroniniam tinklui tuo pat metu pateikiama vektorių porą X^{μ} ir X^{ν}, tai neuroninis tinklas kiekvienoje iteracijoje permokomas $2*N_2$ vektoriais: kiekvieno mokymo žingsnio metu vienas vektorius imamas iš senos duomenų aibės, o kitas iš naujos. Po kiekvienos iteracijos skaičiuojama projekcijos paklaida $E(N_1+N_2)$ (pradinės ir naujos duomenų aibių požiūriu) ir fiksuojamas skaičiavimo laikas. Surandamas SAMANN tinklo svorių rinkinys W_2.

Tyrimo rezultatai

Eksperimentams buvo naudojamos trys duomenų aibės:

a) Irisų duomenų aibė (Fisher, 1936). Aibę (dar žinoma kaip Fišerio irisų aibė) sudaro 150 keturmačių vektorių (3 klasės po 50 vektorių). Kiekvienas vektorius atitinka vienam iš irisų, ir priklauso vienai iš trijų klasių: Iris Setosa, Iris Versicolor ir Iris Virginica;

b) 300 atsitiktinai sugeneruotų vektorių $X_i = (x_{i1},...,x_{in}) \in R^n$ (3 grupės po 100 vektorių, n=5):

$x_{ij} \in [0;0,2]$, i=1,...,100; j=1,...,5, $\sqrt{\sum_{j=1}^{n} (0.1 - x_{ij})^2} \le 0.1$

$x_{ij} \in [0,4;0,6]$, i=101,...,200; j=1,...,5, $\sqrt{\sum_{j=1}^{n} (0.5 - x_{ij})^2} \le 0.1$

$x_{ij} \in [0,8;1]$, i=201,...,300; j=1,...,5, $\sqrt{\sum_{j=1}^{n} (0.9 - x_{ij})^2} \le 0.1$

c) „Austra“ duomenų aibė (Australian Credit Approval, 2007). Aibę sudaro 690 14-mačių vektorių apie kreditines korteles ir jų panaudojimą. Kiekvienas vektorius priklauso vienai iš dviejų klasių.

Kiekviena iš šios testinės duomenų aibės buvo padalinta į dvi dalis: pradinė duomenų aibė ir nauja duomenų aibė. Pirma dalis naudojama pradiniam SAMANN tinklo mokymui, antra dalis – tinklo permokymui. Irisų duomenų aibės atveju, aibė buvo padalinta tokiu būdu: 100 vektorių – pradinė aibė, 50 vektorių – nauja aibė. Atsitiktinai sugeneruotų vektorių aibė: 210 vektorių – pradinė aibė, 90 vektorių – nauja aibė; „austra“ duomenų aibė: 460 vektorių – pradinė aibė, 230 vektorių – nauja vektorių aibė.

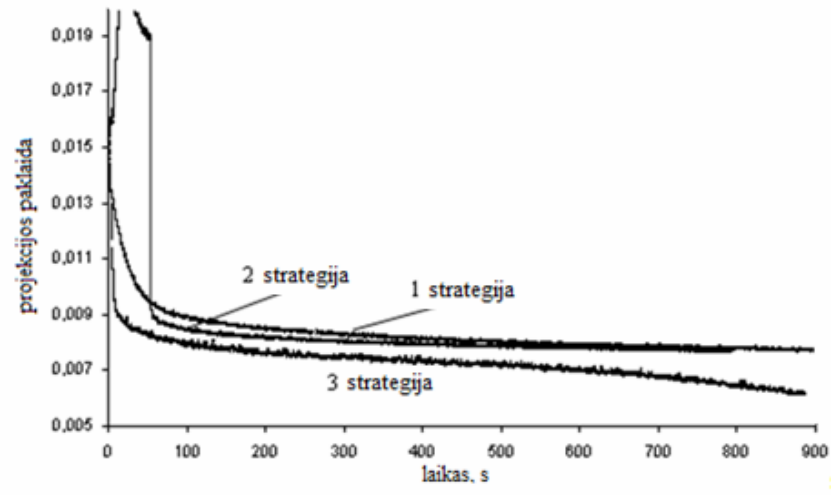
Analizuojant SAMANN tinklo permokymo strategijas, buvo nagrinėjamas SAMANN tinklas su vienu paslėptu sluoksniu, turintis du išėjimus (d=2). Visais atvejais buvo imamas vienodas paslėptojo sluoksnio neuronų skaičius (n_2=20), o taip pat iš anksto fiksuotas pradinių svorių rinkinys. Neuroninio tinklo svorių pradinės reikšmės buvo inicijuojamos parenkant atsitiktines reikšmes iš intervalo $(-0,01;0,01)$. Pradinės duomenų aibės vizualizavimui buvo naudojami tokie parametrai: iteracijų skaičius M=10000, mokymosi parametras $\eta = 10$; naujų vektorių aibei vizualizuoti: mokymosi parametras $\eta = 1$, iteracijų skaičius priklausė nuo pasirinktos strategijos.

Atliekant skaičiavimus buvo fiksuojamas algoritmo vykdymo laikas. 7.1, 7.2 ir 7.3 paveiksluose parodyti skaičiavimų rezultatai. Paveiksluose parodyti SAMANN tinklo permokymo rezultatai tik su naujais vektoriais. Su irisų duomenų aibe ir atsitiktinai sugeneruotų vektorių aibe buvo atlikti tyrimai, kai SAMANN tinklo permokymui buvo naudojamos visos trys strategijos. 7.3 paveikslas vaizduoja projekcijos paklaidos priklausomybę nuo skaičiavimo laiko, permokant tinklą „austra“ duomenų aibės vektoriais. Su „austra“ duomenų aibe buvo atliktas tyrimas tinklo permokymui naudojant tik pirmąją (1) ir trečiąją (3) strategijas. Iš gautų rezultatų (7.1 pav., 7.2 pav. ir 7.3 pav.) matome, kad pirmoji ir antroji strategijos duoda gerus projekcijos rezultatus, tačiau tinklo permokymas vyksta lėtai. Lyginant pirmąją ir antrąją strategijas, antrosios strategijos pradžioje

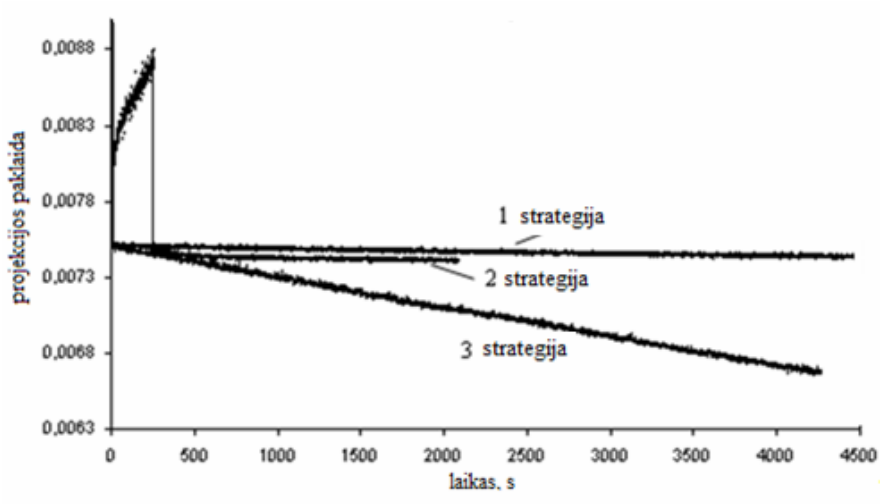
gaunami blogi rezultatai, kai tinklas permokomas tik naujais vektoriais, tačiau po tam tikro laiko (kai tinklas permokomas N_1+N_2 vektoriais) vizualizavimo rezultatai gerėja, gaunamos geresnės projekcijos paklaidos ir mokymo algoritmas konverguoja greičiau, lyginat su pirmąja strategija. Trečioji strategija leidžia per trumpesnį laiką pasiekti gerų vizualizavimo rezultatų, gauti mažesnes projekcijos paklaidas ir pagerinti projekcijos tikslumą lyginant su kitomis strategijomis (geriausiai tai matosi atliekant eksperimentą su atsitiktinai sugeneruotų vektorių aibe, 7.2 pav.). Geriausi projekcijos rezultatai gaunami imant taškus SAMANN tinklo permokymui iš pradinės duomenų aibės ir iš naujos duomenų aibės, t.y. naudojant trečiąją strategiją. Pasiūlyta strategija leidžia sumažinti skaičiavimų trukmę, jeigu naujų vektorių yra žymiai mažiau negu pradinių.

Taip pat, taikant trečiąją strategiją, galima sutaupyti skaičiavimo laiką, norint pasiekti tam tikrą projekcijos paklaidos reikšmę, ir kai iteracijų skaičius nėra iš anksto fiksuotas. Trečioji strategija leidžia sutaupyti skaičiavimų laiką ir todėl, kad tinklo permokymui naudojama mažiau vektorių porų, nes naujos duomenų aibės vektorių skaičius paprastai mažesnis negu pradinės duomenų aibės. Naudojant pirmąją ir antrąją strategijas, tinklo permokymui naudojamos visos įmanomos aibės vektorių poros.

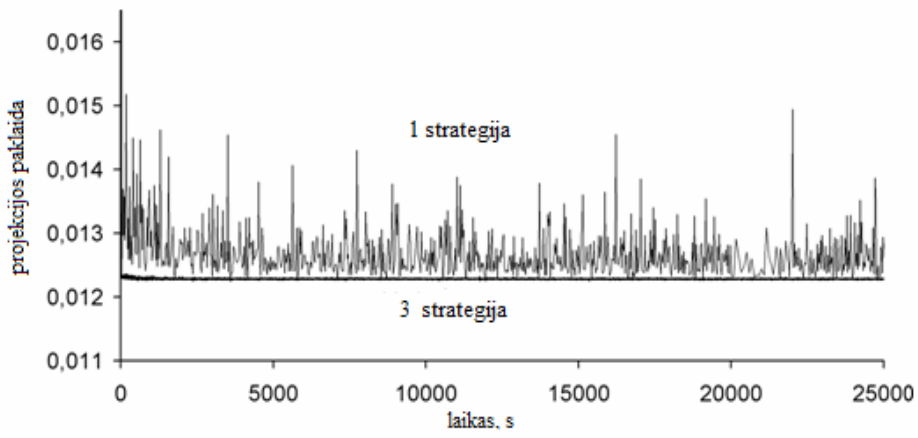
7.4(a), (b), (c) paveiksluose pavaizduotos analizuojamų duomenų aibių (irisų duomenų aibė, atsitiktinai sugeneruotų vektorių aibė ir „austra“ duomenų aibė) gautos projekcijos plokštumoje, kai SAMANN tinklo permokymui buvo naudojama trečioji strategija. Taškai iš pradinės duomenų aibės ir naujos duomenų aibės pavaizduoti skirtingomis spalvomis. Nauji taškai „randa“ vietas tarp savo grupės taškų.



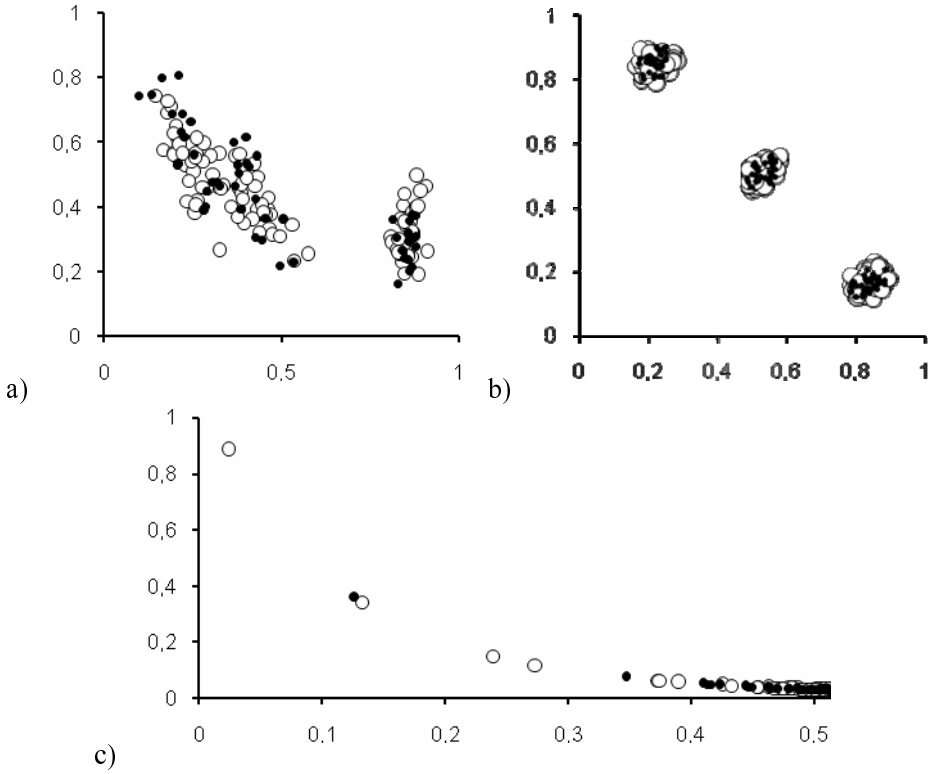
7.1 pav. *Paklaidos priklausomybė nuo skaičiavimo laiko (Irisų duomenų aibė)*



7.2 pav. *Paklaidos priklausomybė nuo skaičiavimo laiko (atsitiktinai sugeneruotų vektorių aibė)*



7.3 pav. *Paklaidos priklausomybė nuo skaičiavimo laiko („austra“ duomenų aibė)*



7.4 pav. *Projekcijos rezultatai plokštumoje: (a) irisų duomenų aibė, (b) atsitiktinai sugeneruotų vektorių aibė, (c) „austra“ duomenų aibė (taškai iš pradinės duomenų aibės ir naujos duomenų aibės pavaizduoti skirtingomis spalvomis; juodos spalvos taškai – nauji taškai)*

7.2. Septintojo skyriaus išvados

Šiame skyriuje analizuojamos tinklo galimybės vizualizuoti naujus duomenis. Dirbant su dideliais duomenų kiekiais naujų vektorių gali būti labai daug ir po tam tikro laiko SAMANN tinklą tenka permokyti. Pasiūlytos ir ištirtos 3 daugiamačius duomenis vizualizuojančio neuroninio tinklo permokymo strategijos. Pirmoji strategija tinklo permokymui naudoja visas įmanomas analizuojamų vektorių poras (vektoriai imami ir iš naujos duomenų aibės ir iš pradinės). Antroji strategija iš pradžių tinklo permokymui naudoja tik naujus vektorius, o paskui visas įmanomas vektorių poras. Trečiojoje strategijoje vienas vektorius imamas iš pradinės duomenų aibės – kitas iš naujos duomenų aibės.

Eksperimentai parodė, kad iš pasiūlytų strategijų geriausia yra ta strategija, kai kiekvieno mokymo žingsnio metu vienas vektorius imamas iš senos duomenų aibės, o kitas iš naujos. Ši strategija duoda mažesnes projekcijos paklaidas lyginant su kitomis strategijomis. Naudojant ją galima sumažinti skaičiavimų trukmę esant fiksuotam iteracijų skaičiui ir pasiekti gerus rezultatus per trumpesnį laiką. Visos pasiūlytos permokymo strategijos gali būti taikomos didelės apimties duomenų aibių vizualizavimui.

Remiantis eksperimentinio tyrimo rezultatais pastebėsime, kad galimas būdas minimizuoti SAMANN tinklo mokymo laiką yra mokymo proceso padalinimas į du subprocesus: (1) tinklo mokymas dalimi analizuojamos duomenų aibės; (2) tinklo permokymas likusia duomenų aibės dalimi arba visa aibe. Šiuo atveju tinklo mokymo proceso pagrindinę dalį užimtų mokymas nepilna duomenų aibe, o tai iš esmės taupytų skaičiavimo laiką.

8

SAMANN tinklo taikymai

Aprašyto ir optimizuoto SAMANN algoritmo realizacija buvo pritaikyta medicininių duomenų analizei. Pagrindiniai skyriaus rezultatai paskelbti šiuose straipsniuose: (Bernatavičienė, Dzemyda, Kurasova, Marcinkevičius, Medvedev, 2007), (Dzemyda, Kurasova, Medvedev, 2007).

8.1. SAMANN tinklo taikymas medicininių duomenų analizei

Dažna problema medicinoje yra būsenos priskyrimas vienai iš jau žinomų klasių (pvz., sveikas ar sergantis kokia nors liga). Tokį priskyrimą paprastai atlieka medikai. Tačiau dabar yra kuriami duomenų gavybos (*angl. data mining*) metodai ir sprendimų paramos (*angl. decission support*) sistemos, kurias galima būtų pritaikyti medicinoje (Lavrac, 1997), (Adlassnig, 2003). Medicininių duomenų klasifikavimo specifika yra tai, kad dažnai perėjimas iš vienos būsenos į kitą nėra labai ryškus. Todėl ypač svarbu įvertinti to perėjimo ribas, kurias bendru atveju nepakanka apibrėžti kokia nors klases skiriančia funkcija. Svarbu rasti sritį, į kurią patenkančius pacientų duomenis būtina atkreipti ypatingą dėmesį. SAMANN metodas šią sritį atvaizduoja dvimatėje erdvėje. Gaunamas vaizdas kur kas lengviau interpretuojamas ir suvokiamas negu lentelėmis pateikiami duomenys.

Kadangi ligą (būseną) charakterizuoja tam tikrų parametrų rinkinys $x_1, x_2, ..., x_n$, todėl iš jų galima sudaryti *n*-matį vektorių $X = (x_1, x_2, ..., x_n)$, nusakantį pacientą. Tegu analizuojamoje aibėje yra *m* objektų (pacientų). Tada galima sudaryti vektorių rinkinį $X^1, X^2, ..., X^m$, kur $X^j = (x_1^j, x_2^j, ..., x_n^j)$. Uždavinys tampa daugiamačių vektorių klasifikavimo uždaviniu. Tarkime, kad yra dvi klasės C_1 ir C_2. Tikslas – priskirti vektorius $X^1, X^2, ..., X^m$ vienai iš klasių C_1 arba C_2. Bendru atveju gali būti ir daugiau klasių (C_1, C_2, …, C_v).

Klasifikavimo paklaidos yra neišvengiamos. Medicinoje tai yra sudėtinga problema, kadangi, jei sergantis žmogus bus priskirtas sveikųjų klasei, pasekmės gali būti lemtingos. Vizualumas (duomenų apie skirtingus pacientus išsidėstymas plokštumoje ir vaizdus klasifikavimo rezultatų pateikimas) leidžia operatyviai priimti sprendimą apie analizuojamus duomenis.

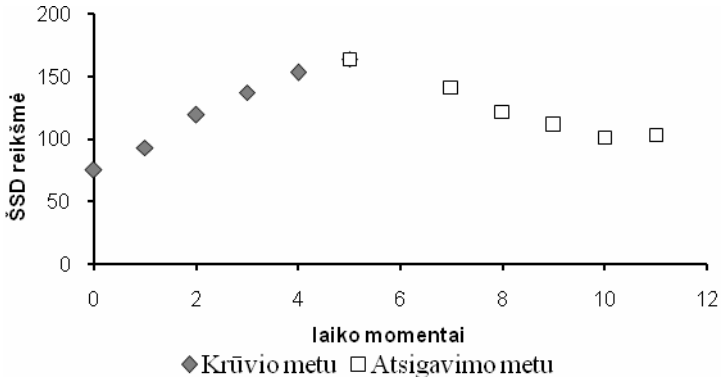
8.1.1. Fiziologiniai duomenys

Fiziologinių duomenų aibė sudaryta iš trijų grupių vyrų duomenų: vyrai, sergantys išemine širdies liga (1 grupė) (61 tiriamasis), ne sportininkai (vyrai, kuriems liga nebuvo diagnozuota) (2 grupė) (110 tiriamųjų), ir profesionalūs sportininkai (vyrai) (3 grupė) (161 tiriamasis). Bendras tiriamųjų skaičius yra 332.

Širdies funkciniams rodikliams vertinti naudota kompiuterizuota 12-os standartinių derivacijų elektrokardiogramų (EKG) analizės sistema „Kaunas-Krūvis“ (Bernatavičienė, 2005).

Šiam tyrimui buvo pasirinkti paprasčiausiai ir lengviausiai nustatomi EKG bei AKS dydžiai: širdies susitraukimų dažnis (ŠSD), sistolinis arterinio kraujo spaudimas (S), diastolinis arterinio kraujo spaudimas (D), intervalas elektrokardiogramoje nuo jungties taško J iki T bangos pabaigos (JT intervalas). Šie keturi skaitiniai parametrai yra fiksuojami tam tikrais laiko momentais, atliekant ergometrinio dviračio tyrimą. Dar papildomai buvo apskaičiuoti du išvestiniai parametrai: (S-D)/S ir JT/RR (RR=60/ŠSD).

Prieš pradedant tyrimą, tiriamiesiems išmatuojami minėti 4 dydžiai: ŠSD, S, D, JT. Pradinis galingumas – 50 W. Tyrimo metu kas minutę galingumas didinamas po 50 W. Prieš galingumo padidinimą matuojami nurodyti keturi dydžiai. Tyrimas baigiamas, kai tiriamasis nepajėgia daugiau jo atlikti arba gydytojas pastebi žymius pakitimus širdies veikloje ir liepia baigti tyrimą. Po to seka organizmo atsigavimo periodas. Kas minutę vėl matuojami tie patys parametrai. Taigi, vieno tyrimo metu gaunamas kelių parametrų reikšmių rinkinys, be to, kiekvienam tiriamajam reikšmių skaičius skiriasi, kadangi jis priklauso nuo maksimalaus galingumo, kuriam esant tiriamasis pajėgė įveikti fizinį krūvį. 8.1 paveiksle pateikiamas vieno tiriamojo ŠSD parametro reikšmės matuotos tam tikrais laiko momentais didinant krūvį ir atsigavimo metu.



8.1 pav. *Širdies susitraukimų dažnio kitimas krūvio ir atsigavimo metu*

Toks reikšmių kitimas atspindi žmogaus širdies veiklą. Visos keturios fiziologinės charakteristikos (ŠSD, JT, S, D) yra svarbios, todėl būtina analizuoti jų visumą. Tokia integrali analizė gydytojui gana sudėtingas uždavinys. Norint supaprastinti šį uždavinį, būtina rasti šios dinamikos integralius įverčius, kurie būtų lengviau suvokiami. Tam gali būti naudojamos įvairios strategijos. Viena iš jų paremta fraktalinių dimensijų skaičiavimais. Ji naudota darbe (Bernatavičienė, 2005).

Fraktalinės dimensijos skaičiuojamos tokiu būdu:

1. Pradiniai duomenys – tai tyrimo metu gautos visų nagrinėjamų parametrų skaitinės reikšmės prie atitinkamų galingumų krūvio metu, o taip pat penkios reikšmės atsigavimo po krūvio metu. Gauti kiekvieno parametro taškiniai įverčiai yra interpoliuojami kubiniu splainu. Tokiu būdu gaunamas kiekvieno parametro funkcijos grafikas.

2. Fraktalinės dimensijos (užimtumo, informacinė, koreliacijos) skaičiuojama remiantis gautais grafikais pagal pateikiamą schemą:

- nagrinėkime kvadratinį tinklelį (langelio dydis ε), uždėtą ant stebimos taškinės struktūros;
- kiekvienoje tinklelio dalyje suskaičiuojamas į ją papuolusių taškų skaičius n_i. Jis dalinamas iš N, bendro taškų skaičiaus $P_i(\varepsilon) = \frac{n_i}{N}$;
- apibrėžkime informacinę funkciją $I \equiv -\sum_{i=1}^{N_\varepsilon} P_i(\varepsilon)\log[P_i(\varepsilon)]$, kur N_ε – užimtų langelių skaičius. Tuomet informacinė dimensija apbrėžiama taip $d_{\inf} \equiv -\lim_{\varepsilon\to 0}\frac{I}{\log(\varepsilon)} = \lim_{\varepsilon\to 0}\sum_{i=1}^{N_\varepsilon}\frac{P_i(\varepsilon)\log\left[P_i(\varepsilon)\right]}{\log(\varepsilon)}$. Pakeitus $I = \log N_\varepsilon$

gausime užimtumo dimensiją, o $I = \log \sum_i (P(\varepsilon))^2$ gausime koreliacijos dimensiją.

Bendras parametrų skaičius yra 18, po 6 parametrus kiekvienai dimensijai.

8.1.2. Fiziologinių duomenų analizė

Iš pradžių buvo atlikta analizuojamų duomenų klasifikacija. Tyrimuose (Bernatavičienė, 2007) buvo naudoti šie klasifikatoriai: Naïve Bayes, klasifikavimo medžiai (*angl. classification trees*), *k* artimiausių kaimynų (*angl. kNN classifier*), ir atraminių vektorių klasifikatorius (*angl. support vector machine*). Nustatyta (Bernatavičienė, 2007), kad klasifikuojant pirmosios (1) ir trečiosios (3) grupės duomenis, be antrosios (2) grupės duomenų, klasifikavimo tikslumas žymiai padidėja. Vadinasi, galima daryti išvadą, kad (2) grupę (nesportuojantys) negalima laikyti klase. Todėl galima stebėti, kaip (2) grupės objektai priskiriami (1) ar (3) klasei, t.y., kurie nesportuojančių asmenų matuoti parametrai yra panašūs į sportuojančiųjų, o kurie į sergančiųjų išemine širdies liga.

Nagrinėjant turimus duomenis, buvo panaudota vizuali daugiamačių duomenų analizė (kai randamos duomenų projekcijos plokštumoje, stebimos susidariusios vizualios jų grupės).

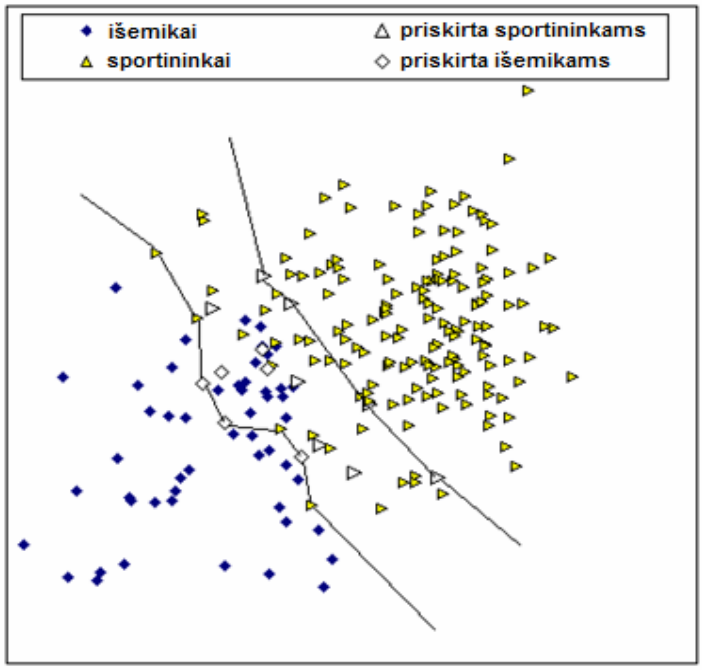
Analizuojamos duomenų aibės projekcijos radimui buvo naudojamas SAMANN tinklas su vienu paslėptu sluoksniu (neuronų skaičius n_2=20), turintis du išėjimus (d=2). SAMANN tinklas buvo apmokytas vektoriais iš dviejų grupių ((1) išemikai ir (3) sportininkai), naudojantis standartiniu sklidimo atgal algoritmu, su tokiais parametrais: mokymosi parametras $\eta = 1$, *momentum* reikšmė – 0,3, iteracijų skaičius - 10000. Viena iteracija, tai visų įmanomų vektorių porų pateikimas SAMANN tinklui vieną kartą. Buvo surasti SAMANN tinklo išėjimai išemikų ir sportininkų duomenų aibės grupėms, ir apskaičiuoti SAMANN tinklo svorių reikšmės *W*. Turint dviejų grupių ((1) išemikai ir (3) sportininkai) projekcijas plokštumoje, galima greitai ir pakankamai tiksliai surasti trečiosios grupės (nesportuojantys) projekcijas plokštumoje, t.y. atvaizduoti trečiąją grupę be papildomų skaičiavimu. (2) grupė (nesportuojantys) pateikiama SAMANN tinklui. Turint jau apskaičiuotas svorių reikšmes, gaunama šių duomenų projekcija plokštumoje.

Duomenų projekcijos rezultatai pateikiami 8.2 paveiksle (parodytos dvi grupės: (1) išemikai ir (3) sportininkai). Taip pat išskirti taškai, kuriuos klasifikatoriai priskyrė ne toms klasėms kaip medikai.

Vizualizuojant duomenis (8.2 pav.), yra nurodomos kelios grupės:

- išemikai (kaip nurodė medikai ir kuriuos dauguma klasifikatorių priskyrė prie išemikų);

- sportininkai, kuriuos daugumą klasifikatorių priskyrė prie išemikų;
- sportininkai (kaip nurodė medikai ir kuriuos dauguma klasifikatorių priskyrė prie sportininkų);
- išemikai, kuriuos dauguma klasifikatorių priskyrė prie sportininkų.



8.2 pav. *Fiziologinių duomenų projekcija (1 ir 3 grupės), naudojant SAMANN algoritmą*

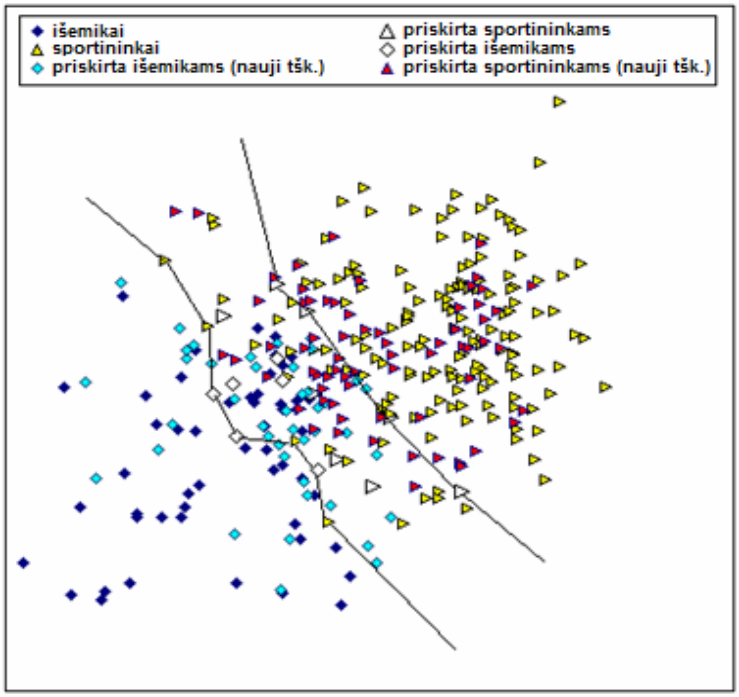
Kaip matyti iš 8.2 paveikslo, didžioji dalis išemikus atitinkančių taškų yra vienoje paveikslo pusėje, priešingoje pusėje dalis taškų, atitinkančių sportininkus. Tačiau šios grupės dalinai persidengia. Todėl tikslinga išskirti tų persimaišiusių grupių taškų sritį. Vienas iš paprasčiausių būdų – sujungti laužtės dalimis priešingai klasei arba neteisingai klasifikuotus artimiausius taškus.

Iš vizualaus vaizdo sunku pasakyti, kokiai klasei priskirti tarp laužčių pakliuvusius taškus, kurie atitinka konkretų tiriamąjį. Šie tyriamieji turi būti medikų detaliai ištirti.

8.3 paveiksle pavaizduotos visų trijų grupių tiriamuosius atitinkantys taškai. Vizualizuojant duomenis, antrosios grupės taškai yra pažymimi pagal tai, kokiai klasei juos priskyrė dauguma klasifikatorių, t.y. nurodomi:

- nesportuojančių asmenų grupės objektai, kuriuos klasifikatorius priskyrė (1) klasei,

- nesportuojančių asmenų grupės objektai, kuriuos klasifikatorius priskyrė (3) klasei.



8.3 pav. *Fiziologinių duomenų projekcija (visos 3 grupės), naudojant SAMANN algoritmą*

Esant būtinybei atvaizduoti naujos duomenų aibės taškus, SAMANN algoritmu nauji taškai randa savo vietą tarp jau atvaizduotų, nereikia iš naujo skaičiuoti visų analizuojamų taškų projekcijų, kaip tą būtina atlikti MDS metodu. Šiuo atveju taip pat nesikeičia pirmų dviejų grupių taškų tarpusavio išsidėstymas po trečios grupės taškų atidėjimo plokštumoje.

Iš rezultatų vizualios analizės galima daryti šias išvadas:

(1) nesportuojantys asmenys, kuriuos atitinkantys taškai vizualiai išsidėstė sportuojančius asmenis atitinkančių taškų srityje (8.3 pav.), yra sveiki ir gali sportuoti, kadangi jų matuoti parametrai nekuo nesiskiria nuo sportuojančiųjų;

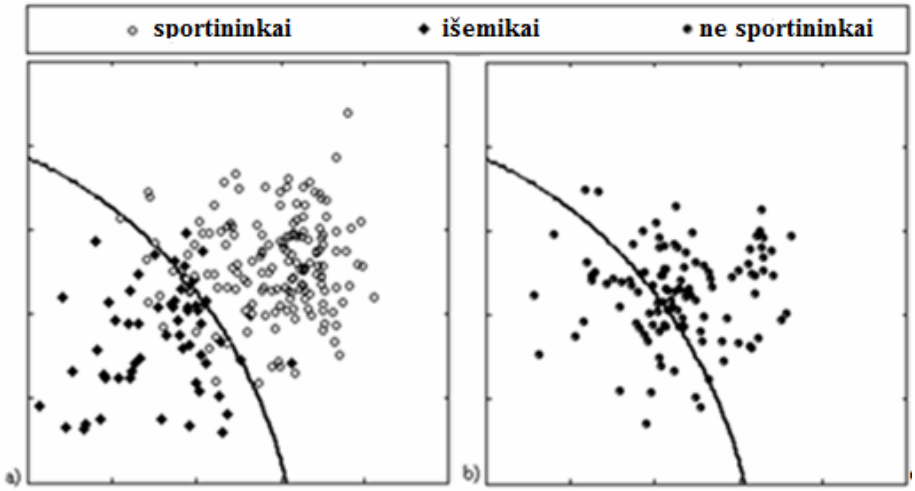
(2) nesportuojantys asmenys, kuriuos atitinkantys taškai vizualiai išsidėstė išemikus atitinkančių taškų srityje (8.3 pav.), gal būt turi sveikatos problemų ir jokiu būdu negalima leisti jiems sportuoti išsamiau jų neištyrus, kadangi jų matuoti parametrai niekuo nesiskiria nuo išemikų;

(3) medikams verta išsamiau ištirti ir stebėti sportininkus, kuriuos atitinkantys taškai išsidėsto arti taškų, atitinkančių išemikus, kadangi yra tikimybė, kad jie turi sveikatos sutrikimų.

Tikslesniam klasių atskyrimui tikslinga naudoti paties klasifikatoriaus gautą skiriamąjį paviršių. Darbuose (Bernataviciene, 2006), (Dzemyda, Kurasova, Medvedev, 2007) pirmosios ir trečiosios grupių taškų atskyrimui buvo panaudotas atraminių vektorių klasifikatorius (*angl. support vector machine, SVM*). Dvimatės fiziologinių duomenų projekcijos gautos naudojant SAMANN algoritmą. 8.4(a) parodytos dvi grupės (išemikai ir sportininkai) ir SVM klasifikatoriumi gautas skiriamasis paviršius, atskiriantis išemikų grupę nuo visų kitų taškų. Taip pat buvo apskaičiuotos neuroninio tinklo svorių reikšmės W apmokant SAMANN tinklą tik (1) ir (3) grupės taškais. Turint tinklo svorius, buvo surastos „naujų“ taškų ((2) grupė – ne sportuojantys) projekcijos, be papildomo tinklo mokymo. 8.4(b) paveiksle pavaizduoti tik antrosios grupės taškai.

8.4 paveiksle klasifikavimo rezultatas (skiriamasis paviršius) gautas analizuojant daugiamačių duomenų dvimates projekcijas. Tyrimai (Bernatavičienė, 2006) parodė, kad klasifikavimas, betarpiškai naudojant daugiamačius duomenis, o ne jų dvimates projekcijas, yra panašios kokybės, kaip ir dvimačių projekcijų analizės atveju, tik daugiamatėje erdvėje prarandamas vaizdumas.

8.4 paveiksle, atsižvelgiant į skiriamąjį paviršių, galima daryti preliminarias išvadas apie tiriamųjų sveikatos būseną.



8.4 pav. *Fiziologinių duomenų projekcija, naudojant SAMANN algoritmą: a) (1) ir (3) grupės; b) tik (2)-ios grupės taškai*

8.2. Aštuntojo skyriaus išvados

SAMANN algoritmas buvo pritaikytas medicininių (fiziologinių) duomenų analizei. Fiziologinių duomenų aibė sudaryta iš trijų grupių vyrų širdies funkcinių rodiklių rinkinių: vyrai, sergantys išemine širdies liga, ne sportininkai (vyrai, kuriems liga nebuvo diagnozuota), ir profesionalūs sportininkai. Analizė leido sporto medicinos specialistams įvertinti nesportuojančių vyrų sveikatos būklę ir jų galimybę sportuoti.

SAMANN algoritmo privalumas vizualizuojant tirtus duomenis yra tai, kad nauji taškai „randa“ savo vietą tarp jau atvaizduotų, nereikia iš naujo skaičiuoti visų analizuojamų taškų projekcijų, kaip tą būtina atlikti MDS grupės metodais, t.y. nesikeičia pirmų dviejų grupių taškų tarpusavio išsidėstymas po trečios grupės taškų atidėjimo plokštumoje. Šiuo atveju pirmas dvi grupes sudaro duomenys su žinomomis charakteristikomis (vyrai, sergantys išemine širdies liga, ir profesionalūs sportininkai).

9

Bendrosios išvados

1. Tiesioginio vizualizavimo metodų analizė parodė, kad naudojant šiuos metodus suprasti duomenų struktūrą yra gana sudėtinga, ypač esant didesnei duomenų dimensijai arba analizuojant didelės apimties duomenų aibę. Daug lengviau suvokti ir interpretuoti rezultatus, gautus projekcijos metodais, kuriuose daugiamatis vektorius transformuojamas į mažesnės dimensijos vektorių. Lyginant tiesinės ir netiesinės projekcijos metodus, tikslesnė duomenų struktūra išlaikoma naudojant netiesinės projekcijos metodus. Bet ir čia duomenų vizualizavimo iškraipymai yra neišvengiami.

2. Analizuojant mokymo be mokytojo „klaidos sklidimo atgal“ SAMANN neuroninį tinklą, nustatyta, kad projekcijos paklaida ir tinklo konvergavimas priklauso nuo pasirinktų parametrų reikšmių. Eksperimentai parodė, kad, kuo didesnė mokymosi parametro reikšmė, tuo greičiau pavyksta pasiekti gerus vizualizavimo rezultatus. Tačiau, didėjant mokymosi parametro reikšmei, didėja paklaidos svyravimai.

3. Tiriant kelias duomenų aibes, nustatyta, kad optimali SAMANN tinklo mokymosi parametro reikšmė yra intervale (5;30). Pasirenkant tokias mokymosi parametro reikšmes, galima žymiai sumažinti skaičiavimų trukmę (iki 3–5 ir net daugiau kartų) ir gauti gerus vizualizavimo rezultatus per trumpesnį laiką, esant fiksuotam iteracijų skaičiui. Mažos mokymosi parametro reikšmės intervale (0;1) garantuoja stabilų (be svyravimų) projekcijos paklaidos mažėjimą didėjant iteracijų skaičiui. Tuo tarpu, kai mokymosi parametro reikšmė pasirenkama didesnė,

pastebimi tam tikri paklaidos svyravimai. Tačiau šie svyravimai yra pakankamai maži, kai mokymosi parametro reikšmė pasirenkama iš intervalo (5;30).

4. Ištirtos galimybės tinklo mokymui vienu metu naudoti keletą kompiuterių. Skaičiuojant lygiagrečiuoju SAMANN algoritmu, kuris analizuojamą aibę dalina į kelias dalis, vykdo nepriklausomai tinklo mokymą atskiromis dalimis, o po to gautus svorius apjungia juos vidurkinant po kiekvienos mokymo iteracijos (kuomet visos galimos atskiros mokymo aibės dalies vektorių poros pateikiamos tinklui vieną kartą), nepavyko pagerinti atvaizdavimo rezultatų ir (arba) pagreitinti skaičiavimų. Tačiau, gauti rezultatai, leidžia daryti išvadas, kad kuriant naujas lygiagrečias SAMANN algoritmo modifikacijas būtina siekti mažinti duomenų persiuntimo kaštus ir racionaliai atlikti atskirais procesoriais apskaičiuotų tinklo svorių apjungimą.

5. Pasiūlyta lygiagrečiojo algoritmo modifikacija, kuri analizuojamą aibę dalina į kelias dalis, vykdo nepriklausomai tinklo mokymą atskiromis dalimis, o po to tam tikro fiksuoto iteracijų skaičiaus parenka geriausią pasiektą rezultatą visos duomenų aibės požiūriu ir baigia mokyti tinklą pilna duomenų aibe. Tyrimai parodė, kad skaičiuojant lygiagrečiuoju SAMANN algoritmu galima pasiekti geresnius vizualizavimo rezultatus per trumpesnį laiką (lyginant su nuosekliuoju algoritmu). Tokio tipo mokymo strategijos gali būti taikomos ir didelės apimties duomenų aibių vizualizavimui.

6. Eksperimentai parodė, kad galima rasti tokį analizuojamos duomenų aibės poaibį, kuriuo mokant SAMANN tinklą, mažesnės projekcijos paklaidos gaunamos greičiau, negu tinklo mokymui naudojant visus aibės taškus.

7. Pereinant nuo nuosekliojo algoritmo prie lygiagretaus algoritmo, skaičiuojamieji kaštai sumažėja ne tik dėl lygiagretaus darbo pasidalijimo tarp procesorių, bet ir dėl to, kad skirstant duomenis tarp procesorių, sumažėja vektorių porų, pateikiamų neuroniniam tinklui, skaičius.

8. Pasiūlytos ir ištirtos trys daugiamačius duomenis vizualizuojančio neuroninio tinklo permokymo strategijos. Eksperimentai parodė, kad iš pasiūlytų strategijų geriausia yra tokia strategija, kai kiekvieno mokymo žingsnio metu vienas vektorius imamas iš senos duomenų aibės, o kitas iš naujos. Ši strategija duoda mažesnes projekcijos paklaidas lyginant su kitomis strategijomis. Ji leidžia sumažinti skaičiavimų trukmę tam pačiam rezultatui pasiekti. Visos trys pasiūlytos permokymo strategijos gali būti taikomos didelės apimties duomenų aibių vizualizavimui. Galimas būdas minimizuoti SAMANN tinklo mokymo laiką yra mokymo proceso padalinimas į du subprocesus: tinklo mokymas analizuojamos duomenų aibės dalimi, vėliau – tinklo permokymas likusia duomenų aibės dalimi arba visa aibe. Šiuo atveju tinklo mokymo proceso pagrindinę dalį užimtų mokymas nepilna duomenų aibe, o tai iš esmės taupytų skaičiavimo laiką.

9. SAMANN algoritmas buvo pritaikytas medicininių (fiziologinių) duomenų analizei. Fiziologinių duomenų aibė sudaryta iš trijų grupių vyrų širdies funkcinių rodiklių rinkinių: vyrai, sergantys išemine širdies liga, ne sportininkai (vyrai, kuriems liga nebuvo diagnozuota), ir profesionalūs sportininkai. Analizė leido sporto medicinos specialistams įvertinti nesportuojančių vyrų sveikatos būklę ir jų galimybę sportuoti. SAMANN algoritmo privalumas vizualizuojant tirtus medicininius duomenis yra tai, kad nauji taškai (duomenys apie nesportuojančius vyrus) „randa“ savo vietas tarp jau atvaizduotų duomenų su žinomomis charakteristikomis.

Literatūros sąrašas

1. Ahalt, A.; Krishnamurthy, A. K.; Chen, P.; Melton, D. E. (1990). Competitive Learning Algorithm for Vector Quantization, Neural Networks, vol. 3: 277–290.
2. Adlassnig, K. P. (2003). *Artificial Intelligence in Medicine.* Elsevier.
3. Alhoniemi, E.; Himberg, J.; Parhankangas, J.; Vesanto, J. (2000). SOM Toolbox for Matlab 5, Helsinky University of Technology, Report A57, Libella Oy Espoo, [žiūrėta 2007-06-07]. Prieiga per internetą <http://www.cis.hut.fi/projects/somtoolbox/>
4. Alpern, B. (1991). "Hyperbox", in *proceeding of IEEE Visulization'91*: 133-139
5. Andrews, D. F. (1972). "Plots of High-Dimensional Data", *Biometrics,* 125–136.
6. Ankerst, M.; Keim, D. A.; Kriegel, H. P. (1996). „Circle Segments: A Technique for Visually Exploring Large Multidimensional Data Sets", in *Proc. Visualization'96*, Hot Topic Session, San Francisco, CA.
7. Australian Credit Approval, [žiūrėta 2007-06-07]. Prieiga per internetą: <http://www.niaad.liacc.up.pt/old/statlog/datasets/australian/australian.doc. html>
8. Baldi, P.; Hornik, K. (1989). "Neural networks and principal component analysis: Learning from examples without local minima". in *IEEE Trans. Neural Networks,* vol. 2: 53–58.

9. Battista, G. D.; Eades, P.; Tamassia, R.; Tollis, I. (1994). "Annotated Bibliography on Graph Drawing Algorithms". *Computational Geometry: Theory and Applications*, Vol. 4: 235–282.

10. Beaudoin, L.; Parent, M. A.; Vroomen, L. C. (1996). "Cheops: A compact explorer for complex hierarchies". in *Proc. Visualization'96*, IEEE Computer Society, 87–92.

11. Becker, R. A.; Cleveland, W. S.; Shyu, M. J. (1996). "The visual design and control of trellis graphics displays". *Journal of Computational and Graphical Statistics*, 5: 123–155

12. Beddow, J. (1990). "Shape Coding of Multidimensional Data on a Mircocomputer Display", in *Visualization'90*, San Francisco, CA, 238–246.

13. Belkin, M.; Niyogi, P. (2001). "Laplacian eigenmaps and spectral techniques for embedding and clustering". *Advances in Neural Informatikon Processing Systems*, Vol. 14. Ed. T. G. Dietterich, S. Becker, and Z. Ghahramani. Cambridge, MA: MIT Press, 585–591.

14. Belkin, M.; Niyogi, P. (2003). "Laplacian eigenmaps for dimensionality reduction and data representation", *Neural Comput.*, vol.15(6): 1373–1396.

15. Bernatavičienė, J.; Berškienė, K.; Ašeriškytė, D.; Dzemyda, G.; Vainoras, A.; Navickas, Z. (2005). "Fraktalinių dimensijų biomedicininio informatyvumo analizė", iš *Biomedicininė inžinerija, Tarptautinės konferencijos pranešimų medžiaga*. Kaunas: Technologija, 27–31.

16. Bernataviciene, J.; Dzemyda, G.; Kurasova, O.; Marcinkevicius, V. (2006). "Decision Support for Preliminary Medical Diagnosis Integrating the Data Mining Methods". in P*roceedings of 5th International Conference on Simulation and Optimisation in Business and Industry*. Ed. H. Pranevičius, O. Vaarmann, E. Zavadskas, 155–160.

17. Bernatavičienė, J.; Dzemyda,G.; Kurasova, O.; Marcinkevičius, V.; Medvedev, V. (2007). "The Problem of Visual Analysis of Multidimensional Medical Data", *Springer optimization and its applications, Models and algorithms for global optimization*. New York: Springer, 277–298.

18. Bertin, J. (1983). *Semiology of Graphics*. W.J.Berg (translator). Semiologie Graphique (Editions Gauthier-Villars). Madison, WI. The University of Wisconsin Press.

19. Beshers, C.; Feiner, S. (1993). "AutoVisual: Rule-based Design of Interactive Multivariate Visualizations", *Computer Graphics & Applications*, Vol. 13(4): 41–49.

20. Bezdek, J. C.; Pal, N. R. (1995). "Index of Topological Preservation for Feature Extraction", *Pattern Recognition* 28(3): 381–391.

21. Bishop, C. M.; Svensen, M.; Wiliams, C. K. I. (1997). "GMT: A Principles Alternative to the Self-Organizing Map", *Advances in Neural Information Processing Systems* 9: 354–363.

22. Bishop, C. M.; Svensen, M.; Williams, C. K. I. (1998). "GTM: The Generative Topographic Mapping", *Neural Computation* 10(1): 215–234.

23. Blake, C. L.; Hettich, S.; Merz, C. J. (1998). *UCI repository of machine learning databases*. Irvine, CA. University of California, Department of Information and Computer Science. [žiūrėta 2007-07-04]. Prieiga per internetą: <http://www.ics.uci.edu/~mlearn/MLRepository.html>

24. Broomhead, D. S.; Lowe, D. (1988). "Multivariable functional interpolation and adaptive networks", *Complex Systems* 2: 321–355.

25. Borg, I.; Groenen, P. (1997). *Modern Multidimensional Scaling: Theory and Applications*. New York: Springer.

26. Buja, A.; Swayne, D. F.; Littman, M.; Dean, N.; Hofmann, H. (1998). "XGvis: Interactive Data Visualization with Multidimensional Scaling", *Journal of Computational and Graphical Statistics*.

27. Chambers, J. M.; Cleveland, W. S.; Kleiner, B.; Tukey, P. A. (1983). *Graphical Methods for Data Analysis*. Belmont, CA: Wadsworth.

28. Chernoff, H. (1973). "The use of faces to represent points in k-dimensional space graphically", *Journal of the American Statistical Association*, Vol. 68: 361–368.

29. Čiegis, R. (2005). *Lygiagretieji algoritmai ir tinklinės technologijos.* Vilnius: Technika.

30. Cleveland, W. S. (1993). *Visualizing Data.* AT&T Bell Laboratorines, Murray Hill, NJ, Hobart Press, Summit NJ.

31. Comon, P. (1994). "Independent component analysis – a new concept?", in *Signal Processing*, Vol. 36: 287–314.

32. Cox, T. F.; Cox, M. A. A. (1994). *Multidimensional scaling.* London: Chapman & Hall.

33. Demartines, P.; Herault, J. (1997). "Curvilinear Component Analysis: A Self-Organizing Neural Network for Nonlinear Mapping of Data Sets", *EEE Trans. Neural Networks,* Vol. 8(1).

34. DeMers, D.; Cottrell, G. (1993). "Non-linear dimensionality reduction", in *Advances in Neural Information Processing Systems*. USA, San Mateo: Morgan Kaufmann. Vol. 5: 580–587.

35. Duda, R. O.; Hart, P. E. (1973). *Pattern Recognition and Scene Analysis*. John Wiley.

36. Duda, R. O.; Hart, P. E.; Stork, D. G. (2000). *Pattern Classification.* 2nd Edition. John-Wiley.

37. Dzemyda, G. (2001). "Visualization of a set of parameters characterized by their correlation matrix", *Computational Statistics and Data Analysis*, 36(1): 15–30.

38. Dzemyda, G. (2005). "Multidimensional data visualization in the statistical analysis of curricula", *Computational Statistics and Data Analysis*, Vol. 49: 265–281.

39. Dzemyda, G.; Kurasova, O.; Medvedev, V. (2007). "Dimension Reduction and Data Visualization Using Neural Networks". *Emerging Artificial Intelligence Applications in Computer Engineering. Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*. Vol. 160, Frontiers in Artificial Intelligence and Applications. Ed. I. Maglogiannis, K. Karpouzis, M. Wallace and J. Soldatos. IOS Press, 25–49.

40. Feiner, S.; Beshers, C. (1990). "World within World: Metaphors for Exploring n-dimensional Virtual Worlds", in *Proc. UIST*, 76–83.

41. Fisher, R. A. (1936). "The Use of Multiple Measurements in Taxonomic Problems", *Annals of Eugenics*, Vol. 7: 179–188.

42. Flexer, A. (2001). "On the use of self-organizing maps for clustering and visualization", Intelligent-Data-Analysis, Vol. 5(5): 373–384.

43. Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition.* 2 ed. Academic Press.

44. Furnas, G. W.; Bederson, B. B. (1995). „Space-Scale Diagrams: Unterstanding Multiscale Interfaces", in *Proc. Human Factors in Computing Systems CHI '95 Conf.*, Denver, CO.

45. Furnas, G. (1986). "Generalized Fisheye Views", in *Proc. Human Factors in Computing Systems CHI'86 Conf.*, Boston, MA, 18–23.

46. Graham, R. L.; Hell, P. (1985). "On the history of the minimum spanning tree problem", *Annals of the History of Computing* 7: 43–57.

47. Grinstein, G. G.; Ward, M. O. (2002). *Introduction to Data Visualization. Information Visualization in data Mining and Knowledge Discovery*. Ed. U.Fayyad, G.G. Grinstein, A. Wierse. Morgan Kaufmann Publishers.

48. Harman, H. H. (1967). *Modern Factor Analysis*. 2nd edition. University of Chicago Press.

49. Hastie, T.; Stuetzle, W. (1988). "Principal curves", *Journal of the American Statistical Association* 84: 502—516.

50. Hassoun, M. H. (1995). *Fundamentals of Artificial Neural Networks.* A Bradford Book. London: the MIT Press.

51. Hinton, G. E.; Osindero, S. (2006a). "A fast learning algorithm for deep belief nets", *Neural Computation*, 18(7): 1527–1554.

52. Hinton, G. E.; Salakhutdinov, R. R. (2006b). "Reducing the dimensionality of data with neural networks", *Science*, 313(5786): 504–507.

53. Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. 2nd ed. Upper Saddle River, New Jersey: Prentice Hall.

54. Hawkins, D. M.; Bradu, D.; Kass, G. V. (1984). „Location of several outliers in multiple regression data using elemental sets", *Technometrics* 26: 197–208.

55. Hyvarinen, A.; Karhunen, J.; Oja, E. (2001). *Independent Component Analysis*. John Wiley & Sons.

56. Hoffman, P. E.; Grinstein, G. G. (2002). „A Survey of Visualizations for High-Dimensional Data Mining", *Information Visualization in Data Mining and Knowledge Discovery*, Ed. by U.Fayyad, G.G. Grinstein, A. Wierse. San Francisco: Morgan Kaufmann Publishers.

57. Huber, P. J. (1985). "Projection Pursuit", *The Annals of Statistics*, Vol. 13(2): 435–474.

58. Hyvärinen, A.; Karhunen, J.; Oja, E. (2001). *Independent Component Analysis*. New York: Wiley.

59. Idrissi, M. J.; Sbihi, A.; Touahni, R. (2004). "An improved neural network technique for data dimensionality reduction in remotely sensed imagery", *International Journal of Remote Sensing,* 25(10): 1981–1986.

60. Inselberg, A. (1985). "The Plane with Parallel Coordinates", *Special Issue on Computational Geometry, The Visual Computer*, Vol. 1: 69–97.

61. Inselberg, A.; Dimsdale, B. (1990). "Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry", in *Visualization*'90, San Francisco, CA, 361–370.

62. Ivanikovas, S.; Medvedev, V.; Dzemyda, G. (2007). "Parallel Realizations of the SAMANN Algorithm", *Lecture Notes in Computer Science, Adaptive and Natural Computing Algorithms*, Vol. 4432: 179–188.

63. Jain, A. K.; Dubes, R. C. (1988). *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice Hall.

64. Jain, A. K.; Mao, J.; Mohiuddin K. (1996). "Artificial Neural networks: a Tutorial", *IEEE Computer*, Vol. 29(3): 31–44.

65. Jin, Y.; Ma M. (2000). "A neural-network dimension reduction method for large-set pattern classification". in *ICMI '00: Proceedings of the Third International Conference on Advances in Multimodal Interfaces*. Springer-Verlag, 426–433.

66. Jollife, I. T. (1986). *Principal Component Analysis*. Springer-Verlag.

67. Jones, M. C.; Sibson, R. (1987). "What is projection pursuit?", *J. of the Royal Statistical Society*, ser. A, 150: 1–36.

68. Chambers, J. M.; Cleveland, W. S.; Kleiner, B.; Tukey P. A. (1983). *Graphical Methods for Data Analysis*. Belmont, CA: Wadsworth Press.

69. Kaski, S., (1997). *Data Exploration Using Self-Organizing Maps PhD thesis*. Helsinki University of Technology, Department of Computer Science and Engineering, [žiūrėta 2007-06-04]. Prieiga per internetą: <http://www.cis.hut.fi/~sami/thesis/>

70. Keim, D. A.; Kriegel, H. P. (1994). „VisDB: Database Exploration using Multidimensional Visualization", *Computer Graphics & Applications*, 40–49.

71. Keim, D. A.; Kriegel, H. P.; Ankerst M. (1995). „Recursive Pattern: A Technique for Visualizing Very Large Amounts of Data", in *Proc. Visualization* '95, Atlanta, GA, 279–286.

72. Keim, D. A.; Kriegel, H.-P. (1996). „Visualization Techniques for Mining Large Databases: A Comparison", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8(6): 923–938.

73. Keim, D. A.; Ward, M. (2003). *Visualization. Intelligent Data Analysis: an Introduction*. Ed. by M. Berthold, D. J. Hand. Springer-Verlag, 403–427.

74. Klock, H.; Buhmann, J. M. (1999). „Data visualization by multidimensional scaling: A deterministic annealing approach", *Pattern Recognition*, Vol. 33(4): 651–669.

75. Kohonen, T.; Hynninen, J.; Kangas, J.; Laaksonen J. (1996). „SOM_PAK: The Self-Organizing Map Program Package", Helsinki University of Technology, Laboratory of Computer and Information Science, Neural Networks Research Centre. Technical Report A31, FIN-02150 Espoo, Finland.

76. Kohonen, T. (2001). *Self-Organizing Maps*. 3nd ed. Springer series in information sciences. Springer-Verlag. Vol. 30.

77. Kohonen, T. (2002). "Self-Organizing Neural networks: Recent Advances and Applications", *Studies in Fuzziness and Soft Computing*, Heidelberg, New York: Physica-Verl. Ed U. Seiffert, L.C. Jain, Vol.78: 1–11.

78. Konig, A. (2000). "Interactive visualization and analysis of hierarchical neural projections for data mining", *IEEE transactions on neural networks*, Vol. 11(3).

79. Kosara, R.; Miksch, S. (2002). *AsbruView: Capturing Complex, Time-Oriented Plans – Beyond Flow-Charts*. Ed. by M. Anderson, B. Meyer, P. Olivier. Diagrammatic Representation and Reasoning, Springer.

80. Kraaijveld, M. A.; Mao, J.; Jain, A. K. (1995). "A Nonlinear Projection Method Based on Kohonen's Topology Preserving Maps", *IEEE Transactions on Neural Networks*, Vol. 6(3): 548–559.

81. Kramer, M. A. (1991). "Nonlinear principal component analysis using autoassociative neural networks", *AIChE Journal*, Vol. 37: 233–243.

82. Kruskal, J. B. (1956). "On the shortest spanning subtree of a graph and the travelling salesman problem", in *Proceedings of the American Mathematical Society*, Vol. 7: 48–50.

83. Kruskal, J. B. (1972). "Linear transformations of multivariate data to reveal clustering", *Multidimensional Scaling: Theory and Application in the Behavioural Sciences, I, Theory*. New York and London: Seminar Press.

84. Kruskal, J. B.; Wish, M. (1984). *Multidimensional Scaling*. Beverly Hills and London: Sage Publications.

85. Kvedaras, B.; Sapagovas, M. (1974). *Skaičiavimo metodai*. Mintis.

86. Kurasova, O., (2005). *Daugiamačių duomenų vizuali analizė taikant savireguliuojančius neuroninius tinklus*. Matematikos ir Informatikos Institutas (daktaro disertacija).

87. Lamping, J.; Rao, R.; Pirolli P. (1995). "A Focus + Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies", in *Proc. Human Factors in Computing Systems CHI '95 Conf.*, Denver, 401–408.

88. Lavrac, N.; Keravnou, E.; Zupan, B. (1997). *Intelligent Data Analysis in Medicine and Pharmacology*. Springer.

89. Lee, R. C. T.; Slagle, J. R.; Blum, H. (1977). "A triangulation method for the sequential mapping of points from N-space to two-space", *IEEE Transactions on Computers*, 288–92.

90. Lee, J. A.; Lendasse, A.; Donckers, N.; Verleysen, M. (2000). „A robust nonlinear projection method". In Eighth *European Symposium on Artificial Neural Networks*, Ed. by M. Verleysen, ESANN'2000, Bruges, Belgium. D-Facto Publications, 13–20.

91. Lee, J. A.; Verleysen, M. (2002). "Nonlinear Projection with the Isotop Method". In *Proceedings of ICANN'2002*, Ed. By J. Dorronsoro, International Conference on Artificial Neural Networks. Madrid: Springer. 933–938.

92. Lee, J. A.; Lendasse, A.; Verleysen, M. (2004). „Nonlinear projection with curvilinear distances: Isomap versus curvilinear distance analysis". *Neurocomputing* 57: 49–76.

93. de Leeuw, W.; van Liere, R. (2003). „Visualization of Multi Dimensional Data using Structure Preserving Projection Methods", *Data Visualization: The State of the Art*, 213–223.

94. LeBlanc, J.; Ward, M. O.; Wittels, N. (1990). "Exploring N-Dimensional Databases", in *Visualization '90*, San Francisco, CA, 230–239.

95. Levkowitz, H. (1991). "Color icons: Merging color and texture perception for integrated visualization of multiple parameters", in *Proceedings of IEEE Visualization* '91, San Diego, California, 164–170.

96. Lowe, D.; Tipping, M. E. (1996). "Feed-forward neural networks and topographic mappings for exploratory data analysis", *Neural Computing and Applications*, Vol. 4: 83–95.

97. Lowe, D.; Tipping, M. E. (1997). "Neuroscale: novel topographic feature extraction using RBF networks", In Mozer,M.C., *Advances in Neural Information Processing Systems,* Ed. by M. Jordan, T. Petsche. London: MIT Press. 543–549.

98. Mackinlay, J. D.; Robertson, G. G.; Card, S. K. (1991). "The Perspective Wall: Detail and Context Smoothly Integrated", in *Proc. Human Factors in Computing Systems CHI* '91 *Conf.*, New Orleans, LA, 173–179.

99. Mao, J.; Jain, A. K. (1995). "Artificial neural networks for feature extraction and multivariate data projection", *IEEE Transactions on Neural Networks*, Vol. 6(2): 296–317.

100. Mardia, K. V.; Kent, J. T.; Bibby J. M. (1995). *Multivariate Analysis. Probability and Mathematical Statistics*. Academic Press.

101. Mathar, R.; Žilinskas, A. (1993). "On Global Optimization in Two-Dimensional Scaling", *Acta Aplicandae Mathematicae*, Vol. 33: 109–118.

102. McCulloch, W. S.; Pitts, W. (1943). "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, Vol. 5: 115–133.

103. Medvedev, V.; Dzemyda, G. (2004a). "Vizualizavimo paklaidos lygiagrečiose SAMANN algoritmo realizacijose", *Lietuvos matematikos rinkinys*, T. 44, Spec. nr., 649–654.

104. Medvedev, V.; Dzemyda, G. (2004b). "Lygiagreti SAMANN vizualizavimo algoritmo realizacija", *Informacinės technologijos 2004*, konferencijos pranešimo medžiaga. Kaunas: Technologija. 344–350.

105. Medvedev, V.; Dzemyda, G. (2005a). "SAMANN neuroninio tinklo mokymo problemos". *Informacinės technologijos 2005*, konferencijos pranešimo medžiaga. Kaunas: Technologija. 394–399, 2005.

106. Medvedev, V.; Dzemyda, G. (2005b). "Vizualizavimui skirto neuroninio tinklo mokymosi greičio optimizavimas". *Lietuvos matematikos rinkinys*, T. 45, Spec. nr., 426–431

107. Medvedev, V.; Dzemyda, G. (2005c). "Daugiamačius duomenis vizualizuojančio neuroninio tinklo permokymo strategijos", *Informacijos mokslai*, Vilnius, Vilniaus universitetas, T. 34: 263–267.

108. Medvedev, V.; Dzemyda G. (2006a). "Optimization of the SAMANN network training", *Journal of Global Optimization,* Springer, Vol. 35(4): 607–623.

109. Medvedev, V.; Dzemyda G. (2006b). “Speed Up of the SAMANN Neural Network Retraining”, *Artificial Intelligence and Soft Computing – ICAISC 2006*, *Lecture Notes in Computer Science*. Springer. Vol. 4029: 94–103.

110. Medvedev, V.; Dzemyda, G. (2006c). “Retraining the Neural Network for Data Visualization”, in *IFIP International Federation for Information Processing*, *Artificial Intelligence Applications and Innovations,* Vol. 204: 27–34.

111. Messner, P. (2000). *Time Shapes – A Visualization for Temporal Uncertainty in Planning*. Master's thesis, Vienna University of Technology, Institute of Software Technology and Interactive Systems, Vienna, Austria.

112. Montvilas, A. M. (2003a). “Features of Sequential Nonlinear Mapping”. Informatica. 14(3): 337–348.

113. Montvilas, A. M. (2003b). “Investigation of Sequential Mapping of Multidimensional Data”. *Electronics and Electrical Engineering*. Kaunas: Technologija. No.6(48): 7–12.

114. Montvilas, A. M. (2006). “Data Sructure Influence on Mapping Error”, *Electronics and Electrical Engineering*, 1(65): 34–37.

115. Moody, J.; Darken C. (1989). “Fast learning in networks of locally-tunned processing units”, *Neural computation*, vil1: 281–294.

116. Mulier, F.; Cherkassky, V. (1995). “Self-organization as an iterative kernel smoothing process”, *Neural Computation*, Vol.7: 1165–1177.

117. Nocke, T.; Schlechtweg, S.; Sschumann, H. (2005). „Icon-based Visualization using Mosaic Metaphors”, in *Proceedings IEEE Information Visualization (IV‘05)*, 103–109.

118. Noel, N. C. (1998). *Dynamical Embedding and Feature Extraction of Electrocephalographic Data.* Ph.D thesis, Aston University, Aston Street, Birmingham B4 7ET, UK.

119. Oja, E. (1991). “Data compression, feature extraction, and autoassociation in feedforward neural networks”, *Artificial Neural Networks*, 737–745.

120. Pekalska, E.; De Ridder, D.; Duin, R. P. W.; Kraaijveld, M. A. (1999). “A new method of generalizing Sammon mapping with application to algorithm speed-up”, in *Proc. of the 5th Annual Conference of the Advanced School for Computing and Imaging ASCI’99Boasson*, Ed. by J. A. Kaandorp, J. F. M. Tonino. Delft, 221–228.

121. Peng, W.; Ward, M. O.; Rundensteiner, E. A. (2005). „Clutter Reduction in Multi-Dimensional Data Visualization Using Dimension Reordering”, in *IEEE Symposium on Information Visualization 2004 (InfoVis 2004)*, 89 – 96.

122. Pickett, R. M.; Grinstein, G. G. (1988). “Iconographic Displays for Visualizing Multidimensional Data”, in *Proc. IEEE Conf. on Systems, Man and Cybernetics*. NJ: IEEE Press. 514–519.

123. Podlipskytė, A. (2003). *Daugiadimensinių duomenų vizualizacija ir jos taikymas biomediciniių duomenų analizei*. Daktaro disertacija. Kauno technologijos universitetas.

124. Powell, M. (1987). "Radial basis functions for multivariable interpolation: A review", *Algorithms for Approximation*, 143–167.

125. Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; Flannery, B. P. (1992). *Numerical Recipes in C.* Second ed. Cambridge: Cambridge University Press.

126. Prim, R. C. (1957). "Shortest connection networks and some generalizations", *Bell System Technical Journal*, Vol. 36: 1389–1401.

127. Quinn, M. J. (2004). *Parallel Programming in C with MPI and OpenMP*. NewYork: McGraw-Hill Inc.

128. Rabenhorst, D. A. (1994). "Interactive exploration of multidimensional data", in *Proceedings of the SPIE Symposium on Electronic Imaging*, Vol. 2179: 277–286.

129. Rao, R.; Card, S. K. (1994). "The Table Lens: Merging Graphical and Symbolic Representation in an Interactive Focus+Context Visualization for Tabular Information", in *Proc. Human Factors in Computing Systems CHI '94 Conf.*, 318–322.

130. Raudys, Š. (2001). *Statistical and neural Classifiers: an Integrated Approach to Design*. Advances in pattern recognition, Springer-Verlag.

131. Rekimoto, J.; Green, M. (1993). "The Information Cube: Using Transparency", in *3d Information Visualization, Proc. 3rd Annual Workshop on Information Technologies & Systems (WITS '93)*, 125–132.

132. de Ridder, D.; Duin, R.P.W. (1997). "Sammon's mapping using neural networks: A comparison", *Pattern Recognition Letters*, Vol. 18: 1307–1316.

133. Ritter, H.; Martinetz, T.; Schulten, K. (1992). *Neural Computation and Self-Organizing Maps: An Introduction*. Reading, MA: Addison-Wesley.

134. Roberts, S.; Everson, R. (2000). *Independent Component Analysis: Principles and Practice*. Cambridge: Cambridge University Press.

135. Robertson, G. G.; Mackinlay, J. D.; Card, S. K. (1991). "Cone Trees: Animated 3D Visualizations of Hierarchical Information", in *Proc. Human Factors in Computing Systems CHI '91 Conf.*, New Orleans, LA, 189–194.

136. Roweis, S. T.; Saul, L. K. (2000). "Nonlinear dimensionality reduction by locally linear embedding", *Science* 290(5500) 2323–2326.

137. Ruppert, D.; Carroll, R. J. (1980). "Trimmed least squares estimation in the linear model", *Journal of the American Statistical Association* 75: 828–838.

138. Sammon, J. W. (1969). "A nonlinear mapping for data structure analysis". *IEEE Transactions on Computers*, C-18(5): 401–409.

139. Sanger, T. D. (1989). "Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network", *Neural Networks,* 2: 459–473.

140. Saund, E. (1989). "Dimensionality-reduction using connectionist networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11: 304–314.

141. Shepard, R. N. (1962a). "The analysis of proximities: Multidimensional scaling with an unknown dstance function", *I. Psychometrika* 27(2): 125–140.

142. Shepard, R. N. (1962b). "The analysis of proximities: Multidimensional scaling with an unknown distance function", *II. Psychometrika* 27(3): 219–246.

143. Shneiderman, B. (1992). "Tree Visualization with Treemaps: A 2D Space-Filling Approach", *ACM Transactions on Graphics*, Vol. 11(1): 92–99.

144. Sigillito, V. G.; Wing, S. P.; Hutton, L. V.; Baker, K. B. (1989). "Classification of radar returns from the ionosphere using neural networks", *Johns Hopkins APL Technical Digest*, 10: 262–266.

145. Silipo, R. (2003). "Neural networks", *Intelligent Data Analysis*, Ed. by M. Berthold and D.J. Hand, 270–320.

146. de Silva, V.; Tenenbaum, J. B. (2002). "Unsupervised learning of curved manifolds". In *Proceedings of the MSRI workshop on nonlinear estimation and classification.* New York: Springer-Verlag. 453–466.

147. Smolensky, P. (1986). "Information processing in dynamical systems: Foundations of harmony theory". in *Parallel Distributed Processing: Volume 1: Foundations*. Cambridge: MIT Press. 194–281.

148. Spence, R.; Tweedie, L.; Dawkes, H.; Su, H (1995). "Visualization for Functional Design", in *Proc. Int. Symp. on Information Visualization (InfoVis '95)*, Atlanta, GA, 4–10.

149. Takatsuka, M. (2001). "An application of the Self-Organizing Map and interactive 3-D visualization to geospatial data", in *Proceedings of the 6th International Conference on GeoComputation*.

150. Taylor, P., (2003). *Statistical Methods. Intelligent Data Analysis: an Introduction.* Ed. by M. Berthold, D. J. Hand. Springer-Verlag. 69–129.

151. Tenenbaum, J. B.; de Silva, V.; Langford, J. C. (2000). "A global geometric framework for nonlinear dimensionality reduction", *Science*, 290(5500): 2319–2323.

152. Tipping, M. E. (1996). *Topographic mappings and feed-forward neural networks*. Ph.D thesis, Aston University, Aston Street, Birmingham B4 7ET, UK.

153. Torgerson, W.S. (1952). "Multidimensional scaling: I. theory and method". *Psychometrika* 17: 401–419.

154. Ultsch, A.; Siemon, H. P. (1990). "Kohonen's Self-Organizing Feature Maps for Exploratory Data Analysis", in *Proc. of INNC'90, International Neural Network Conference*, Dordrecht, Netherlands, 305–308.

155. Usui, S.; Nakauchi, S.; Nakano, M. (1991). "Internal color representation acquired by a five-layer neural network", *Artificial Neural Networks*, 867–872.

156. Verikas, A.; Gelžinis, A. (2003). *Neuroniniai tinklai ir neuroniniai skaičiavimai.* Kaunas: Technologija.

157. van Wijk, J. J.; van Liere, R. D. (1993). „Hyperslice", in *Proc. Visualization '93*, San Jose, CA, 119–125.

158. Yang, L. (2004). "Sammon's nonlinear mapping using geodesic distances", in *ICPR'04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04)*, Vol. 2, Washington, DC, USA, IEEE Computer Society, 303–306.

159. Žilinskas, A.; Podlipskytė, A. (2003). "On multimodality of the SSTRESS criterion for metric multidimensional scaling", *Informatica*, Vol. 14(1): 121–130.

Autoriaus publikacijų sąrašas disertacijos tema

Straipsniai tarptautiniuose periodiniuose leidiniuose, įtrauktuose į Mokslinės informacijos instituto pagrindinį sąrašą (*ISI Web of Science*)

1A. Medvedev V., Dzemyda G., (2006). Optimization of the SAMANN network training. *Journal of Global Optimization*, Springer, Vol. 35(4), p. 607–623. ISSN 0925-5001.

2A. Medvedev V., Dzemyda G., (2006). Speed Up of the SAMANN Neural Network Retraining. Artificial Intelligence and Soft Computing – ICAISC 2006, *Lecture Notes in Computer Science*, Springer, Vol. 4029, p. 94–103. ISSN 0302-9743.

Straipsniai tarptautiniuose periodiniuose leidiniuose, įtrauktuose į Mokslinės informacijos instituto konferencijos darbų (*ISI Proceedings*) sąrašą

3A. Medvedev V., Dzemyda G., (2006). Retraining the Neural Network for Data Visualization. In IFIP International Federation for Information Processing, *Artificial Intelligence Applications and Innovations*, Vol. 204, p. 27–34. ISSN 1571-5736.

4A. Ivanikovas S., Medvedev V., Dzemyda G., (2007). Parallel Realizations of the SAMANN Algorithm. *Lecture Notes in Computer Science,* Adaptive and

Natural Computing Algorithms, Springer, Vol. 4432, p. 179–188. ISSN 0302-9743.

Skyrius užsienio leidyklos (*IOS Press*) knygoje

5A. Dzemyda G., Kurasova O., Medvedev V., (2007). Dimension Reduction and Data Visualization Using Neural Networks. *Emerging Artificial Intelligence Applications in Computer Engineering. Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*. Vol. 160, Frontiers in Artificial Intelligence and Applications. Editors: I. Maglogiannis, K. Karpouzis, M. Wallace and J. Soldatos, p. 25–49, IOS Press. ISBN 978-1-58603-780-2.

Straipsnis užsienio leidyklos (*Springer*) knygoje

6A. Bernatavičienė J., Dzemyda G., Kurasova O., Marcinkevičius V., Medvedev V., (2007). The Problem of Visual Analysis of Multidimensional Medical Data. Models and Algorithms for Global Optimization, *Springer Optimization and Its Applications*, Springer, Vol. 4, p. 277–298. ISBN 978-0-387-36720-0.

Straipsnis Lietuvos periodiniuose leidiniuose

7A. Medvedev V., Dzemyda G., (2004). Vizualizavimo paklaidos lygiagrečiose SAMANN algoritmo realizacijose, *Lietuvos matematikos rinkinys*, T. 44, Spec. nr., p. 649–654. ISSN 0132-2818.

8A. Medvedev V., Dzemyda G., (2005). Daugiamačius duomenis vizualizuojančio neuroninio tinklo permokymo strategijos, *Informacijos mokslai*, Vilnius, Vilniaus universitetas, T. 34, p. 263–267. ISSN 1392-0561.

9A. Medvedev V., Dzemyda G., (2005). Vizualizavimui skirto neuroninio tinklo mokymosi greičio optimizavimas. *Lietuvos matematikos rinkinys,* T. 45, Spec. nr., p. 426–431. ISSN 0132-2818.

Konferencijų pranešimų medžiagoje

10A. Medvedev V., Dzemyda G., (2004). Lygiagreti SAMANN vizualizavimo algoritmo realizacija, *Informacinės technologijos 2004*, konferencijos pranešimo medžiaga, Kaunas, Technologija, p. 344–350. ISBN 9955-09-588-1.

11A. Medvedev V., Dzemyda G., (2005). SAMANN neuroninio tinklo mokymo problemos. *Informacinės technologijos 2005*, konferencijos pranešimo medžiaga, Kaunas, Technologija, p. 394–399. ISBN 9955-09-588-1.