*Liet. matem. rink.*, **44**, *spec. nr.*, 2004, 649–654

# Vizualizavimo paklaidos lygiagrečiose SAMANN algoritmo realizacijose

Viktor MEDVEDEV, Gintautas DZEMYDA (MII)
el. paštas: viktor.m@ktl.mii.lt, dzemyda@ktl.mii.lt

## 1. Įvadas

Požymių išskyrimas ir daugiamačių duomenų projektavimas yra svarbios problemos vektorių atpažinimo ir tyrimo analizėje. Tai leidžia vizualiai analizuoti daugiamačius duomenis, jų grupavimosi tendencijas. Šiame straipsnyje nagrinėjamas dirbtinių neuroninių tinklų panaudojimas daugiamačių duomenų vizualizavimui.

Pastaruoju metu, buvo pasiūlytas didelis skaičius dirbtinių neuroninių tinklų ir jų mokymo algoritmų požymių išskyrimui ir daugiamačių duomenų vizualizavimui. Nustatant ryšius tarp klasikinių metodų ir neuroninių tinklų, buvo pastebėta, kad kai kurie neuroniniai tinklai faktiškai realizuoja požymių išskyrimo ir duomenų projektavimo algoritmus. Tokie tinklai dažnai yra pranašesni už tradicinius metodus. Pavyzdžiui, SAMANN neuroninis tinklas, sukurtas Sammono algoritmui, suteikia apibendrinimo galimybę naujų duomenų vizualizavimui. O tai yra savybė, kurios neturi originalus netiesinės projekcijos algoritmas [7].

## 2. Daugiamačių duomenų vizualizavimas

### *2.1. Sammono projekcija*

Vienas iš daugiamačių duomenų vizualizavimo metodų – Sammono algoritmas. Sammono projekcija yra netiesinio daugelio kintamųjų vektorių atvaizdavimo į mažesnio matavimo erdvę metodas.

Tarkime, kad turime $N$ vektorių $n$-matėje erdvėje $X_i \in R^n$, $i = 1, ..., N$ ir atitinkamai apibrėšime $N$ vektorių $d$-matėje erdvėje ($d = 2$ arba 3)$Y_i \in R^d$, $i = 1, ..., N$. Atstumą tarp vektorių $X_i$ ir $X_j$ $n$-matėje erdvėje pažymėsime $d^*_{ij}$ ir atstumą tarp atitinkamų vektorių $Y_i$ ir $Y_j$ $d$-matėje erdvėje pažymėsime $d_{ij}$[6, 7]. Sammono klaida (1), tai matas, kuris parodo, kaip tiksliai atstumai tarp vektorių išlaikomi pereinant iš didesnio matavimo erdvės į mažesnio matavimo erdvę:

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{(d^*_{ij} - d_{ij})^2}{d^*_{ij}} \Big/ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d^*_{ij}. \qquad (1)$$

### 2.2. *SAMANN algoritmas*

SAMANN – tai specifinė neuroninio tinklo mokymo taisyklė (klaidos sklidimo atgal būdu), leidžianti įprastam tiesioginio sklidimo daugiasluoksniam dirbtiniam neuroniniam tinklui realizuoti Sammono projekciją mokymo be mokytojo būdu. 1 pav. pateiktas tokio tinklo modelis.

Tegul $X = \{X_1, X_2, ..., X_N\}$ yra $n$-mačių įėjimo vektorių aibė. Šie vektoriai naudojami tinklui mokyti, siekiant gauti jų $d$-matę projekciją ($d<n$). Vizualizavimo erdvės dimensija yra $d$, $d<n$. Pažymėkime $j$-otojo elemento $l$-tajame sluoksnyje išėjimą $y_j^{(l)}$, $j = 1, 2, ..., n_l$, $l = 0, 1, 2, ..., L$; čia $n_l$ yra sluoksnio $l$ paslėptų neuronų skaičius, $L$ – sluoksnių skaičius, ir $y_j^{(0)} = X_j$, $j = 1, 2, ..., d$. Svorį jungtyje tarp $i$-tojo neurono ($l$–$1$)-ajame sluoksnyje ir $j$-tojo neurono $l$-tajame sluoksnyje žymėsime $\omega_{ij}^{(l)}$. $\omega_{0j}^{(l)}$ – $j$-tojo neurono $l$-tajame sluoksnyje slenksčio reikšmė, $y_0^{(l)} = 1.0$. Kiekvieno neurono išėjimo reikšmei skaičiuoti naudosime sigmoidinę funkciją $g(h)$, kurios išeinamųjų reikšmių intervalas (0,1); čia $h$ – visų elementų svorių suma. Taigi, $j$-tojo elemento $l$-tajame sluoksnyje išėjimas užrašomas taip:

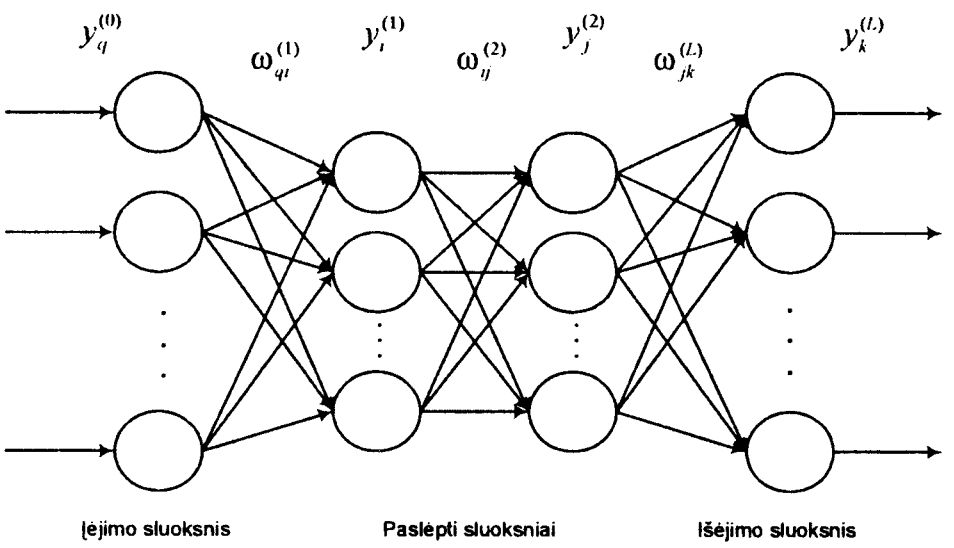
$$y_j^{(l)} = g\left(\sum_{i=0}^{n_l} \omega_{ij}^{(l)} y_i^{(l-1)}\right), \quad l = 1, 2, ..., L. \tag{2}$$

Kiekviename mokymo žingsnyje į neuroninį tinklą paduodami du taškai. Skaičiuojamas atstumas tarp neuroninio tinklo vektorių išėjimų ir nustatomas klaidos matas, naudojant šį atstumą ir atstumą tarp taškų įėjimo erdvėje. Šios klaidos pagrindu gaunama svorių atnaujinimo taisyklė.

Norint atnaujinti svorius, į neuroninį tinklą tuo pat metu turim paduoti vektorių porą, vietoj vieno vektoriaus, kaip sklidimo atgal mokymo algoritme. Norint tai padaryti, galima arba sukonstruoti du identiškus tinklus, arba tiesiog laikyti atmintyje visus pirmojo vektoriaus išėjimo reikšmes, prieš paduodant antrąjį vektorių [5].

SAMANN algoritmas atrodo taip:

1) Atsitiktinai nustatomi SAMANN tinklo svoriai; 2) Atsitiktinai parenkama vektorių pora. Vektoriai po vieną yra paduodami tinklui, tinklas įvertinamas tiesioginio sklidimo būdu; 3) Atnaujinami svoriai, sklidimo atgal būdu, pradedant nuo išėjimo slu-




1 pav. Tiesioginio sklidimo 3-jų sluoksnių neuroninis tinklas Sammono projekcijai.

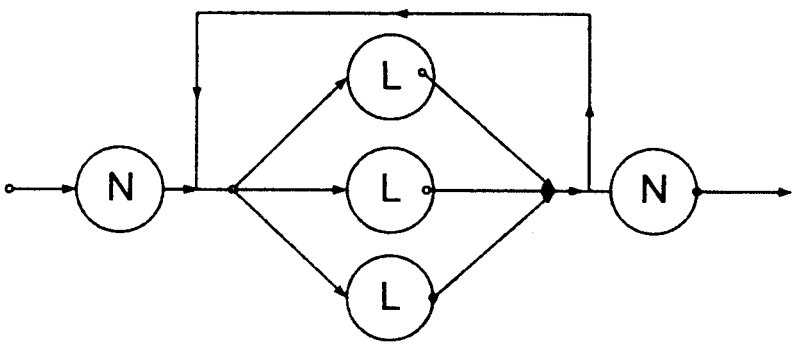
oksnio; 4) Kartojami žingsniai 2–3; 5) Paduodami visi vektoriai ir apskaičiuojami tinklo išėjimai; skaičiuojama Sammono klaida; jeigu Sammono klaidos reikšmė mažesnė už pasirinktą slenkstį arba iteracijų skaičius (žingsniai 2–5) viršija maksimalų skaičių, tuomet sustojam; priešingu atveju vėl pradedam nuo 2 žingsnio.

## 3. Lygiagretus SAMANN algoritmas

Nustatyta, jog SAMANN neuroninio tinklo mokymui reikia daug skaičiuojamųjų sąnaudų. Efektyviausiai ši problema gali būti išspręsta naudojant paskirstytos atminties lygiagrečiąsias sistemas ir taikant metodus nuosekliems algoritmams išlygiagretinti. Tačiau SAMANN algoritmą tiesiogiai išlygiagretinti yra neefektyvu. Todėl šiame darbe pasiūlytas modifikuotas algoritmas, leidžiantis tinklo mokymui vienu metu naudoti keletą procesorių. Šiuo atveju lygiagretusis algoritmas leidžia duotąjį uždavinį išspręsti greičiau, nei tai galėtume padaryti naudodami tik vieną procesorių.

Lygiagretus algoritmas išskaido SAMANN tinklo mokymą į nuoseklias („N“) ir išlygiagretinamas („L“) dalis. Naudosime 3 vienodų parametrų procesorius. Dalį skaičiavimų atlieka vienas procesorius (duomenų įvedimas, vektorių normavimas, pradinis svorių inicializavimas, Sammono klaidos skaičiavimas), o kitus skaičiavimus (svorių atnaujinimas, tinklo išėjimų skaičiavimas) lygiagrečiai vykdo kiti du procesoriai. Skaičiavimų schema pateikta 2 pav. Mokymo procesas yra iteracinis, todėl 2 pav. parodytas ir grįžtamasis ryšys.

**Lygiagretaus SAMANN algoritmo schema.** Neuroninis tinklas apmokomas $n$-mačių vektorių poromis $(X_i, X_j)$, $i, j = \overline{1, N}$, naudojant $v$ iteracijų (viena mokymo iteracija – tai mokymo proceso dalis, kai visas vektorių poras atsitiktine tvarka pateikiame tinklui po vieną kartą). Kiekviename mokymo žingsnyje į neuroninį tinklą paduodami du vektoriai $\mu$ ir $\nu$(kiekvienas procesorius, kuris vykdo tinklo svorių reikšmių skaičiavimus, sukuria du identiškus neuroninius tinklus, ir visus skaičiavimus toliau atlieka savo lokaliojoje dalyje). Kiekvienos iteracijos pradžioje nulinis procesorius ($p_0$) atsitiktiniu būdu permaišo pradinių duomenų masyvą ir paskirsto duomenis tarp likusiųjų 2 procesorių $p_1$ ir $p_2$(pradinio duomenų masyvo elementai dalijami į vienodo arba beveik vienodo dydžio blokus). Kadangi kiekvienos iteracijos pradžioje pradinių duomenų masyvas permaišomas, tai procesoriai kiekvieną kartą gauna tokio pat dydžio skirtingų duomenų blokus. Pateikus vektorius procesoriams, toliau skaičiuojami tinklo išėjimai, t.y. procesoriai $p_1$ ir $p_2$ vykdo Sammono algoritmą. Šie išėjimai yra naudojami atnaujinant atitinkamus tinklo svorius, pradedant nuo išėjimo sluoksnio. Visi procesai atlieka skaičiavimus lygiagrečiai ir tik su lokaliais duomenimis. Po šio




2 pav. Uždavinio išskaidymas į nuoseklias ir išlygiagretinamas dalis.

etapo kiekvienas iš dviejų procesorių jau turi savo lokaliosios dalies svorius $\omega_{ij}^1$ ir $\omega_{ij}^2$, ir nusiunčia juos nuliniam procesoriui. Nulinis procesorius atnaujina svorius ir nusiunčia svorių reikšmes atgal visiems procesoriams. Naudojant ką tik apskaičiuotus svorius, suranda viso tinklo išėjimo vektorius. Po kiekvienos iteracijos tinklo įėjimo ir gauti išėjimo vektoriai naudojami Sammono klaidai $E(1)$ skaičiuoti.

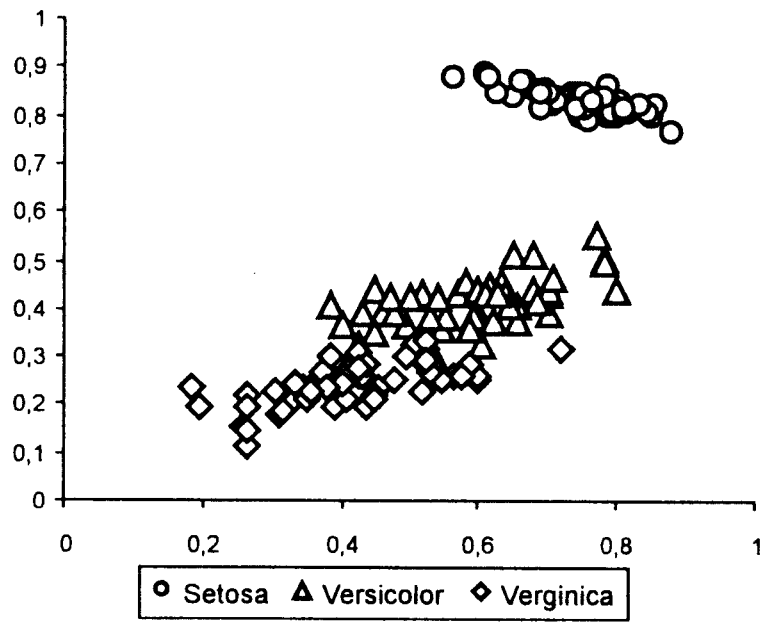
Pastebėta, kad pereinant nuo nuosekliojo algoritmo prie lygiagretaus algoritmo, skaičiuojamieji kaštai sumažėja ne tik dėl lygiagretaus darbo pasidalijimo tarp procesorių, bet ir dėl kitų priežasčių. Skirstant duomenis tarp procesorių, sumažėja vektorių porų, paduodamų į neuroninį tinklą, skaičius. Tai atsitinka dėl to, kad pradinę duomenų aibę turime skaidyti į blokus. Tokių blokų skaičius lygus $k = (p - 1)$, kur $p$ – procesorių skaičius. Jeigu nuosekliajame algoritme tinklo apmokymui naudojama $((N-1)N)/2$ vektorių porų, tai skaičiuojant lygiagrečiuoju algoritmų, vieno duomenų bloko vektorių porų skaičius, kuriais apmokamas tinklas, yra $N(N-k)/(2k^2)$. Naudojant $k$ procesorių, vieno bloko vektorių porų skaičius, lyginant su nuosekliuoju algoritmu, sumažėja maždaug $k^2$ kartų (kai $N$ yra pakankamai didelis).

## 4. Lygiagretaus SAMANN algoritmo tyrimas

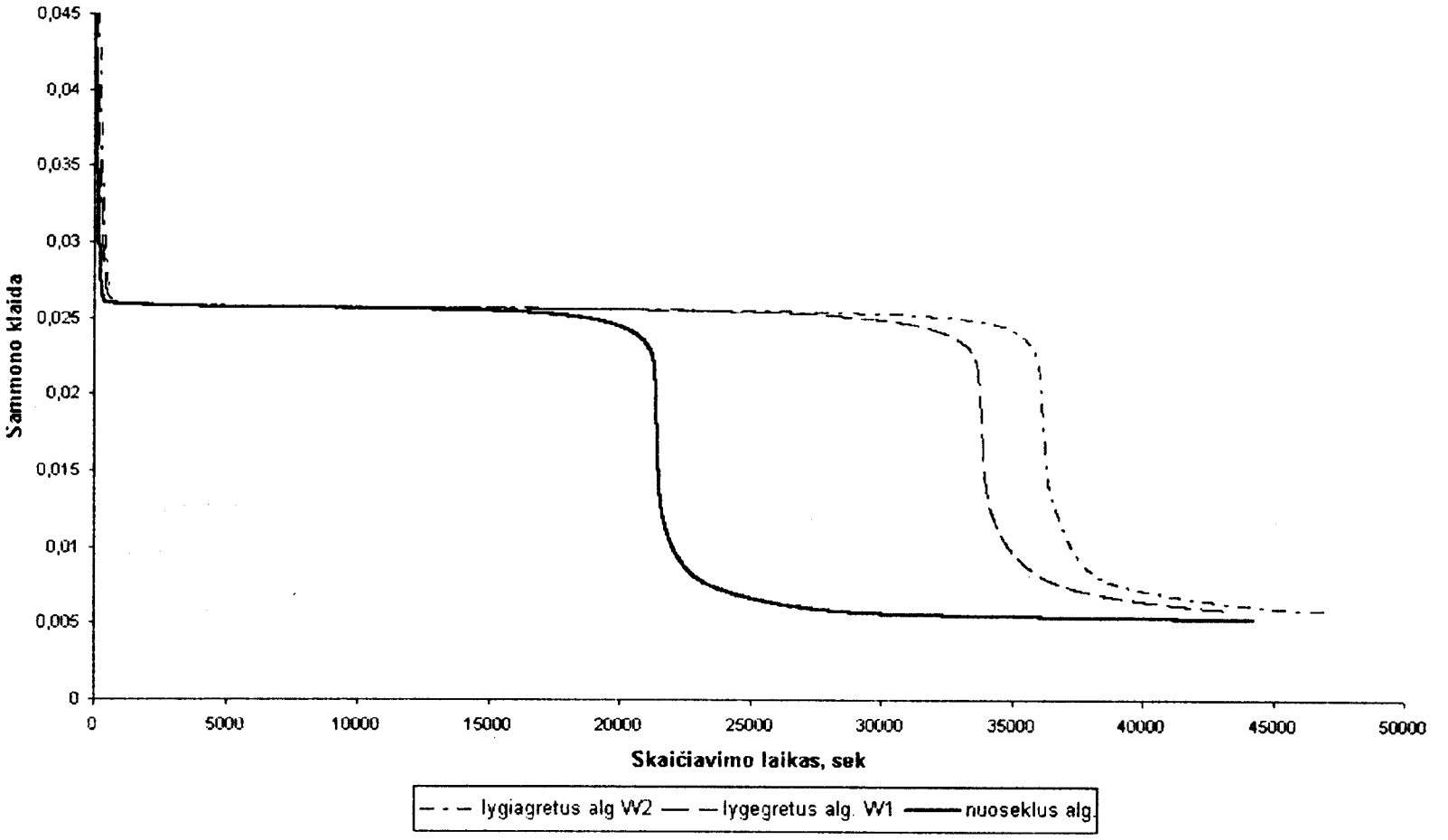
Algoritmo testavimui buvo naudojama irisų duomenų aibė [2]. Aibę (dar žinoma kaip Fišerio irisų aibė) sudaro 150 keturmačių vektorių (3 klasės po 50 vektorių). Analizuojant lygiagretųjį SAMANN algoritmą, buvo nagrinėjamas SAMANN tinklo atskiras atvejis: vienasluoksnis tiesioginio sklidimo dirbtinis neuroninis tinklas, turintis du išėjimus ($d = 2$). Visais atvejais buvo imamas vienodas paslėptojo sluoksnio neuronų skaičius ($N = 10$) ir mokymo greičio parametras ($\eta = 0,5$), o taip pat vienodas pradinių svorių rinkinys. Iteracijų skaičius buvo skirtingas (nuoseklus algoritmas: 200000, lygiagretus algoritmas: 500000).

Tinklo svorių atnaujinimui buvo naudojamos sekančios taisyklės: *W1*: $\omega_{ij} = (\omega_{ij}^1 + \omega_{ij}^2)/2$, t.y. naudojamas paprastas vidurkinimas; *W2*: iš $\omega_{ij}^1$ ir $\omega_{ij}^2$, pasirenkamas tas svorių rinkinys su kuriuo atvaizdavimo paklaida $E$ mažiausia.

Eksperimentiškai įvertinant naujo algoritmo efektyvumą, buvo naudojamas kompiuterių tinklas, valdomas MPI (Message Passing Interface) bibliotekos sukurta




3 pav. Irisų duomenų aibės projekcija plokštumoje.




4 pav. Paklaidos priklausomybė nuo skaičiavimo laiko.

aplinka [1]. Esant toms pačioms pradinėms sąlygoms, skaičiuojant lygiagrečiuoju algoritmu, atlikti tyrimai su 3 procesoriais, apskaičiuotos projekcijos paklaidos, taip pat buvo fiksuojamas skaičiavimo laikas. 3 pav. pavaizduotas irisų duomenų aibės vizualizavimas plokštumoje, skaičiuojant lygiagrečiuoju algoritmu. Rezultatai palyginti su nuosekliojo algoritmo rezultatais (4 pav.).

## 5. Išvados

Nuoseklus algoritmas duoda mažesnes vizualizavimo paklaidas lyginant su lygiagrečiu algoritmu, esant tam pačiam mokymo iteracijų skaičiui, nes atskiras procesorius vienoje iteracijoje disponuoja mažesniu vektorių porų kiekiu nei nuoseklus. Skaičiuojant lygiagrečiu algoritmu sutaupoma skaičiavimo laiko, nes tinklo apmokymo imtis dalinama į atskirus blokus. Tolesnis šio darbo tikslas būtų: a) ieškoti strategijos, kuri leistų sumažinti duomenų persiuntimo kaštus; b) išspręsti kelių neuroninių tinklų kooperavimo problemą, nes, kaip parodė tyrimai, pasiūlyti dviejų procesorių apskaičiuotų svorių jungimo būdai nėra optimalūs.

## Literatūra

1. R. Čiegis, *Lygiagretieji algoritmai*, Technika, Vilnius (2001).
2. A.F. Fisher, The use of multiple measurements in axonomic problems, *Annals of Eugenics*, **7**, 179–188 (1936).
3. B. Lerner, H. Guterman, M. Aladjem, I. Dinstein, On the initialisation of Sammon's nonlinear mapping, *Pattern Analysis and Applications*, **3**(1), 61–68 (2000).
4. J. Mao, K. Jain, Discriminant analysis neural networks, *IEEE Int. Conf. Neural Networks*, **1**, 300–305 (1993).
5. J. Mao, A.K. Jain, Artificial neural networks for feature extraction and multivariate data projection, *IEEE Transactions on Neural Networks*, **6**(2), 296–317 (1969).

6. D. de Ridder, R.P.W. Duin, Sammon's mapping using neural networks: a comparison, *Pattern Recognition Letters*, **18**, 1307–1316 (1997).
7. J.W. Sammon, A nonlinear mapping for data structure analysis, *IEEE Transaction on Computers*, **C-18**, 401–409 (1969).

## SUMMARY

***V. Medvedev, G. Dzemyda. Mapping error in the parallel realizations of SAMANN algorithm***

In this paper we discuss the visualization of multidimensional vectors. A well-known procedure for mapping data from a high-dimensional space onto a lower-dimensional one is Sammon's mapping. This algorithm preserves as well as possible all inter-pattern distances. We investigate an unsupervised back-propagation algorithm to train a multilayer feed-forward neural network (SAMANN) to perform the Sammon's nonlinear projection and propose a parallel algorithm for SAMANN network.

*Keywords:* neutral networks, SAMANN algorithm, Sammon's mapping, visualization.