KLAIPĖDOS UNIVERSITETAS
GAMTOS IR MATEMATIKOS MOKSLŲ FAKULTETAS
INFORMATIKOS KATEDRA

ERIKAS SESELSKIS

INF MAG 04 gr. studentas

E. PATARĖJAS GALIMYBĖMS SOCIALINĖS ATSKIRTIES TERPĖJE PASIRINKTI. MAŠINOS APSIMOKYMO ALGORITMŲ PRITAIKYMAS

Baigiamasis magistro darbas

Darbo vadovas prof. A. A. Bielskis

KLAIPĖDA, 2006

ANOTACIJA

Šiuo metu socialinė atskirtis yra didžiulė problema visoje Europoje. Todėl yra skatinami novatoriški sprendimai, padedantys asmenims, priklausantiems socialinei atskirčiai, integruotis į darbo rinką. Kaip viena iš priemonių šiame darbe yra siūlomas dirbtinių neuroninių tinklų pagrindu kurtas e-patarėjas galimybėms socialinės atskirties terpėje pasirinkti. Šis e-patarėjas atsižvelgęs į individualias žmogaus savybes, pasiūlo labiausiai jam tinkančias profesijas. Darbe taip pat yra pateiktas ligos diagnozės nustatymo dirbtinių neuroninių tinklų pagalba modelis.
PAGRINDINIAI ŽODŽIAI: e-patarėjas, mašinos mokymasis, dirbtinis neuroninis tinklas, daugiasluoksnis perceptronas, SOM

ABSTRACT

At the moment social exclusion is a topical problem in a whole Europe. That’s why innovative decisions are prompted for social exclusive group of people in order to facilitate their integration process into the labour market. The stepping-stone of this work is e-advisor for choosing possibilities within social isolation environment. This e-advisor is created in accordance with artificial neural network and considering to individual person’s features give suggestions for the most suitable professions. Also in this work is presented disease diagnostic model, which is defined by artificial neural network.
KEYWORDS: e-advisor, machine learning, artificial neural network, multi-layer perceptron, SOM

TURINYS

ĮVADAS

Šiuo metu Europos Sąjungoje (ES) vis didėjančia problema tampa socialinė atskirtis. ES jau maždaug 15 % gyventojų gyvena rizikuodami nuskursti. Padėtis tam tikrose naujosiose valstybėse narėse ypač kelia nerimą [18]. Laikoma, kad užimtumas yra esminis socialinės integracijos veiksnys ir vienintelis veiksmingas būdas išbristi iš skurdo ne tik todėl, kad jis sukuria pajamas, bet ir dėl to, kad jis skatina socialinį aktyvumą bei asmens raidą.

Siekdamos sušvelninti šią problemą ES valstybės narės inicijavo projektą EQUAL, kurio tikslas „daryti įtaką ir keisti užimtumo politikos įgyvendinimą vietos, nacionaliniu ir net Europos Sąjungos lygmeniu, vykdant novatoriškų sprendimų sklaidą“. Prie šio projekto prisijungė ir Lietuva, kuri kaip vieną iš prioritetų iškėlė atviros visiems darbo rinkos skatinimą, sudarant sąlygas lengviau patekti ar grįžti į darbo rinką asmenims, kurie susiduria su integracijos ar reintegracijos į darbo rinką sunkumais.

Asmenys, priklausantys socialinei atskirčiai, sunkiai integruojasi į darbo rinką ir kaip vienas iš pagalbininkų jiems galėtų būti e-patarėjas, kuris pagal individualias žmogaus savybes pasiūlytų labiausiai jam tinkančias profesijas. Kadangi žmogaus savybių ir jam tinkamų specialybių sąryšiai nėra žinomi, todėl e-patarėjas privalo mokėti surasti dėsningumus turimuose duomenyse, taip pat turėti mokinimosi iš jam duotų naujų duomenų pavyzdžių galimybę. Visomis šiomis savybėmis pasižymi dirbtiniai neuroniniai tinklai, kurie labai sėkmingai pritaikomi kaip besimokantys iš stebimų duomenų .nežinomų funkcijų aproksimavimo mechanizmai.

Taip pat darbe yra atlikti bandymai siekiant nustatyti neuroninių tinklų tinkamumą ligos diagnozės nustatymui. Eksperimentai buvo atlikti matematiniams skaičiavimams skirtame programiniame pakete *MatLab*. Taip pat buvo naudojami neuroninius tinklus realizuojantys įrankiai *SOM Toolbox* bei *NetLab*.

Tikslas: E-patarėjo galimybėms socialinės atskirties terpėje pasirinkti išvystymas.

Uždaviniai:

1. Atlikti darbo teminės literatūros apžvalgą ir analizę, pagrindžiant temos aktualumą ir jos sprendimui reikalingų metodų pasirinkimą.
2. Atlikti ligos diagnozės nustatymo bandymus, panaudojant kelis neuroninių tinklų modelius; palyginti gautus rezultatus.
3. Sukurti e-patarėją padedantį pasirinkti tinkamą profesiją.
4. Suformuluoti magistrinio darbo išvadas ir jas pagrįsti.

1. MAŠINOS APSIMOKYMO ALGORITMŲ APŽVALGA

1.1. Mašinos mokymasis

Mašinos mokymasis (machine learning) yra mokslo sritis nagrinėjanti algoritmus, įgalinančius kompiuterius mokytis pasinaudojant ankstesne patirtimi ir taip pagerinti savo užduočių atlikimo efektyvumą. Ši sritis yra labai artimai susijusi su šablonų aptikimu (pattern recognition) bei statistiniu išvadų darymu (statistical inference). Per paskutiniuosius 20 metų mašinos mokymasis tapo stipriu inžineriniu mokslu, kurio sritys – duomenų klasterizavimas, klasifikavimas neuroniniais tinklais bei netiesinė regresija įgavo stebėtinai platų pritaikymą inžinerijoje, versle bei moksle. Mašinos mokymasis taikomas paieškos, medicininėse diagnostikos bei biržos fondų analizės sistemose, DNR sekų analizatoriuose, kalbos bei rašto atpažinimo sistemose, žaidimuose bei robotikoje. Kur visa tai veda? Laikoma, kad mašinos mokymasis veda prie dalinio kiekvieno mokslinio metodo automatizavimo, pradedant hipotezių generavimu ir baigiant eksperimentinių modelių kūrimu. Manoma, kad mašinos mokymasis, daugiau ar mažiau, suteiks intelektualioms kompiuterių sistemoms analitinio mokslinio mastymo.

Tačiau stengiantis pagerinti mokslo procesą visose stadijose mašinos mokymosi metodai susiduria su įvairiomis neišspręstomis problemomis. Pirma, dauguma mašinos mokymosi metodų skirti darbui tik su vektoriniais duomenimis, todėl esame priversti žymiai turtingesnius nevektorinius duomenis (pvz. tekstas) suspausti ir pertvarkyti į vektorius. Dažniausiai yra naudojamos fiksuotos duomenų struktūros, kai tuo tarpu yra svarbus ir nepastovių struktūrų modeliavimas.

Nepaisant visų kliūčių, dabartiniai mašinos mokymosi tyrimai įspūdingai žengia į priekį tam tikrų problemų sprendime. Tai apima mokymąsi iš nevektorinių duomenų, tokių kaip sužymėti grafai (labeled graphs), tekstas, atvaizdai; daugiapakopius (multiscale) metodus išvadų darymo optimizacijai ir būdingo bruožo radimui dideliuose duomenų masyvuose. Ypač įdomi nauja mašinos mokymosi tyrimų kryptis yra netiesiniai Mercer branduoliai [13] ir taip pat šiuo metu smarkiai populiarėja griežtų ir neraiškiųjų taisyklių apmokymo metodai.

Kai kurios mašinų mokymosi sistemos siekia visiškai panaikinti žmogaus intuicijos poreikį analizuojant duomenis, o kitos bando pasirinkti žmogaus ir kompiuterio bendradarbiavimo poziciją. Tačiau manau, kad žmogaus intuicija visgi negali būti visiškai atmesta, nes apsimokančios sistemos projektuotojas turi apibrėžti kaip duomenys yra pateikti bei kokie mechanizmai bus naudojami dėsningumų duomenyse paieškai.

1.2. Mokymosi algoritmų tipai

Šiuo metu vyrauja trys pagrindinės mokymosi paradigmos, iš kurių kiekviena skirta tam tikram abstrakčiam mokymosi uždaviniui spręsti. Tai yra *mokymasis su mokytoju, mokymasis be mokytojo* ir *mokymasis skatinant.* Trumpai apie kiekvieną iš jų:

- *Mokymasis su mokytoju* (supervised learning) – tai algoritmas, kuris generuoja funkciją verčiančią įvesties duomenis į reikiamą išvestį. Šioje technikoje funkcija kuriama panaudojant apmokymo duomenis. Apmokymo duomenys yra sudaryti iš įėjimo ir išėjimo pavyzdžių porų. Funkcijos rezultatas gali būti tiek diskreti, tiek tolydi reikšmė ar net įėjimo pavyzdžio numanomas klasės pavadinimas, kai sprendžiamas klasifikavimo uždavinys.
- *Mokymasis be mokytojo* (unsupervised learning) – tai algoritmas, kuris bando atrasti dėsningumus jam pateikiamuose įėjimo duomenyse. Nuo mokymosi su mokytoju jis skiriasi tuo, kad su apmokymo duomenimis jis negauna išėjimo reikšmių.
- *Mokymasis skatinant* (reinforcement learning) – yra prieš tai paminėtų dviejų mokymosi algoritmų kombinacija. Jeigu algoritmas gavęs apmokymo duomenis teisingai apskaičiuoja išėjimą, sistema gauna „atlygį“, priešingu atveju ji yra „baudžiama“. Mokymasis skatinant laikomas mokymusi su kritiku.

Kai kurie autoriai dar išskiria ir dalinai prižiūrimą mokymąsi:

- *Dalinai prižiūrimas mokymasis* (semi-supervised learning) – algoritmas, kuris turi galimybę apmokymui naudoti tiek pažymėtus, tiek nepažymėtus apmokymo duomenų pavyzdžius. Dažniausiai naudojamas mažas kiekis pažymėtų ir didelis nepažymėtų duomenų; ir generuojant tinkamą funkciją ar klasifikatorių kombinuojami pažymėti ir nepažymėti pavyzdžiai. Pastebėta, kad mažo kiekio pažymėtų duomenų panaudojimas su nepažymėtais duomenimis gali sukelti žymų mokymosi tikslumo pagerėjimo efektą. Kadangi nepažymėti duomenys yra lengviau gaunami nei pažymėti, todėl ši technika įgauna didelę praktinę reikšmę.

Svarbiausi klausimai, kurios reiktų apsvarstyti prieš organizuojant apmokymo procesą yra:

- Ką sistema objektyviai gali išmokti iš duomenų aibės? Jeigu duomenų yra nepakankamai, tada sistema negali tinkamai apsimokyti remdamasi jais.
- Ko sistema turi mokytis? Kad sistema išspręstų tam tikrą problemą, ji turi mokytis ne visko, o konkrečių savybių, kurios tiesiogiai susijusios su problemos sprendimu.
- Kaip nustatyti ar sistema jau tinkamai apsimokė? Mokymosi proceso testavimas paprastai vykdomas mokymosi paklaidos matavimu. Populiariausi būdai yra:

  Duomenų padalijimas (Partitioning of data.), kai dalis duomenų, tarkim 70 %, yra

naudojama apmokymui, o likusi dalis testavimui.

Vieno atskyrimas (leaving-one-out). Šiame metode mes apmokome sistemą *n* kartų su *n-1* duomenų ir tikriname sistemos reakciją į likusį duomenų pavyzdį. Sistemos apsimokymo tikslumu galime laikyti teisingo sureagavimo į pateiktą likusį duomenų pavyzdį skaičiaus ir visų duomenų skaičiaus *n* santykį.

1.3. Induktyvus mokymasis: mokymasis iš pavyzdžių

Tarkime, kad pavyzdžių aibė yra žinoma. Pavyzdžiai gali būti pateikti keliomis formomis – arba (x_i , y_i), kur $x_i \in D$ yra tiriamos srities erdvės D būsena, o $y_i \in S$ yra sprendimų erdvės S būsena; arba (x_i) kur $i = 1, 2, \ldots, n$, kuri naudojama, kai išėjimo vektorius nėra apibrėžtas. Norima sukurti sistemą, kuri mokytųsi įėjimų–išėjimų asociacijos $\{(x, y)\}$ arba duomenims $\{x\}$ būdingų dėsningumų. Pirmuoju atveju nurodytas mokymasis su mokytoju, kur kiekvienam įėjimo vektoriui x_i yra suteiktas atsakas y_i (klasės žyma). Antruoju atveju nurodytas nekontroliuojamas mokymasis, kurio atveju sistema iš nepažymėtų pavydžių mokosi apmokymo duomenims būdingų charakteristikų.

Induktyvūs sprendimų medžiai ir ID3 algoritmas

Kaip išskirti klases, kokia formulė, taisyklė ar struktūra turi būti naudojama naujų pavyzdžių apibendrinimui? Technika, vadinama induktyviu sprendimų medžiu stebi pavyzdžius ir kuria binarinį medį, kuris gali aiškiai išskirti klases. Medžio kūrimas yra rekursinis požymio ir jo reikšmės išskiriančios visą aibę į dvi grupes pasirinkimo procesas.

Gerai žinomas sprendimo medžio induktyvaus mokymosi iš simbolinių pavyzdžių aibės metodas yra ID3. Šis metodas sukuria optimalų pažymėtų pavyzdžių aibės klasifikavimą atliekantį sprendimų medį. Optimalus reiškia, kad klasifikuojant naują pavyzdį sistema leisis sprendimo medžiu ir patikrins mažiausiai galimą pavyzdžio požymių skaičių, kurio užtenka jo priskyrimui žinomai klasei.

Apmokyti sprendimo medžiai gali būti transformuojami į JEI-TADA taisykles ar formules. Tai yra kompromisas tarp medžio mokymosi tikslumo ir gebėjimo apibendrinti. Sprendimų medis gali būti apgenėjamas iki tam tikro gylio, kas pagreitina sprendimo procesą, tačiau kartu ir padidina klaidos įvykimo tikimybę. Mokymasis iš pavyzdžių gali būti:

- *Inkrementinis*. Kiekvienas naujas pavyzdys prisideda prie sistemos žinių, gautų iš ankstesnių pavyzdžių. Sistema neperskaičiuoja visų ankstesnių pavyzdžių iš naujo.
- *Vienkartinis mokymasis*. Sistema stebi visus pavyzdžius tik kartą ir iš aibės išrenka tik tam tikrus požymius.

Labai galinga mokymosi iš pavyzdžių technika yra neuroniniai tinklai. Jie gali būti

naudojami nestruktūrinių bei struktūrinių žinių įsisavinimui. Po neuroninio tinklo apmokymo struktūrinėmis žiniomis turi būti suformuluojamos taisyklės, kurios gali būti tikslios arba neraiškios (fuzzy). Taip pat gali būti naudojami įvairaus tipo netikslumai bei neaiškumai.

Sistemų mokymasis negali būti aptariamas atskirai nuo jas apibendrinančių savybių. Svarbus samprotavimas remiantis analogija. Jis yra natūralus žmonėms, tačiau jį yra sunku įgyvendinti kompiuterinėje programoje.

Kiti mašinos mokymosi metodai:

- *Mokymasis veikiant, mokymasis stebint ir atrandant*. Sistema pradeda mokytis visai neturėdama žinių arba sukaupusi minimalias žinias: pvz. kilnumo kriterijus. Tačiau kilnumo kriterijus gali nebūti aprioriniu sistemoje, jis gali būti išmokstamas iš vartotojų bei aplinkos reakcijos. Palaipsniui sistema kaupia teisingus sprendimus ir mokosi kaip tinkamai reaguoti. Paprastas pavyzdys yra mokymasis mechaniškai – įsimenant, kai mintinai išmokstami ankstesni sprendimai ir jie naudojami naujiems duomenims gauti. Kitas pavyzdys genetiniai algoritmai. Šie metodai gali būti realizuojami arba kaip dirbtinio intelekto sistemos, kurios apmokomos tikslių žinių, panašiai kaip ir neuroniniai tinklai, kurie be jokių ankstesnių žinių gali būti apmokomi tiek tiksliomis, tiek apytikrėmis žiniomis.
- *Mokymasis iš patarimų*. Tai yra paprastas procesas, kuriame sistema įgyja tam tikros formos žinių ir jas transformuoja į sau priimtiną formą. Tai yra daugiau interpretavimas ir žinių transformavimas į gerai žinomas schemas. Šis metodas yra tipinis dirbtinio intelekto metodas.
- *Mokymasis remiantis analogija*. Tai atvejis, kai sistema mokosi kaip spręsti problemą pasinaudojant ankstesniu analogiškos problemos sprendimu. Mokymasis analogija yra naudojamas dirbtinio intelekto metoduose, tačiau dėl panašumų įvertinimo sunkumo bei apytikslio samprotavimo poreikio šis požiūris mažai pažengęs. Šioje srityje didelį potencialą turi neuroninių tinklų bei neraiškiųjų sistemų naudojimas.
- *Mokymasis iš atvejų* (Case-based learning) yra pagrįstas pavyzdžių aibės naudojimu. Sistema kaupia tik tam tikrą pavyzdžių aibę ir, siekiant priskirti naująjį pavyzdį tam tikrai klasei, naudoja ją geriausio atitikmens tarp naujojo pavyzdžio ir aibėje esančiųjų senųjų pavyzdžių radimui. Šiame metode yra išmokstama pavyzdžių aibė, ne taisyklės. Naujo pavyzdžio klasifikavimas yra pagrįstas artimiausio kaimyno esančio pavyzdžių aibėje radimu [3].

Klaipėdos universiteto Informatikos katedroje vykdomuose tyrimuose jau keli metai yra naudojami įvairūs mašinos mokymo metodai kuriant adaptyviąsias e.mokymosi priemones e.laboratorijoje [20-25].

2. DIRBTINIAI NEURONINIAI TINKLAI

2.1. Įvadas į dirbtinius neuroninius tinklus

Dirbtinis neuroninis tinklas (toliau vadinamas tiesiog *neuroniniu tinklu*) yra lygiagreti bei paskirstyta informacijos apdorojimo struktūra, susidedanti iš apdorojimo elementų (neuronų) (kurie gali turėti lokalią atmintį bei atlikti lokalizuotos informacijos apdorojimą) sujungtų kartu tarpneuroninėmis jungtimis vadinamomis sinapsėmis. Praktiniu požiūriu neuroniniai tinklai yra netiesiniai statistiniai modeliavimo įrankiai, kurie gali būti naudojami sudėtingų ryšių tarp įvesties bei išvesties parametrų modeliavimui arba šablonų duomenyse aptikimui [14].

Neuroniniai tinklai naudojami, kada yra daug duomenų arba jie yra lengvai gaunami, ir kai teorinis modelis ar duomenų sąryšiai yra nežinomi. Neuroniniai tinklai turi universalų aproksimavimo gebėjimą ir sugeba nustatyti didelio sudėtingumo sąryšius.

Pargrindinės neuroninių tinklų charakteristikos yra:

- Mokymasis. Neuroniniai tinklai apmokinimo proceso metu sugeba įgauti žinių, kurias išsaugo sinapsinių svorių pavidalu. Dirbtinio neuroninio tinklo mokymo procedūra vadinama mokymo algoritmu. Tai yra funkcija, kuri modifikuoja tinklo sinapsinius svorius taip, kad tinklas galėtų pasiekti norimą tikslą [16]. Tinklas yra apmokomas turimais duomenimis: įėjimo-išėjimo poromis (mokymasis su mokytoju) arba tik įėjimo duomenimis (mokymasis be mokytojo).
- Apibendrinimas – neuroninis tinklas gavęs naują įeities vektorių, kurio nėra apmokymo duomenyse, pateikia geriausiai apmokymo metu naudotus duomenis atitinkantį atsaką.
- Paralelizmas – apdorojant duomenis neuronai veikia tuo pat metu.
- Atsparumas klaidoms – kelių neuronų netinkamas veikimas neturi didelio poveikio viso tinklo darbui.
- Dalinis tinkamumas – jis reikalingas kai jau žinomi duomenys nevisiškai sutampa su naujaisiais faktais [3].

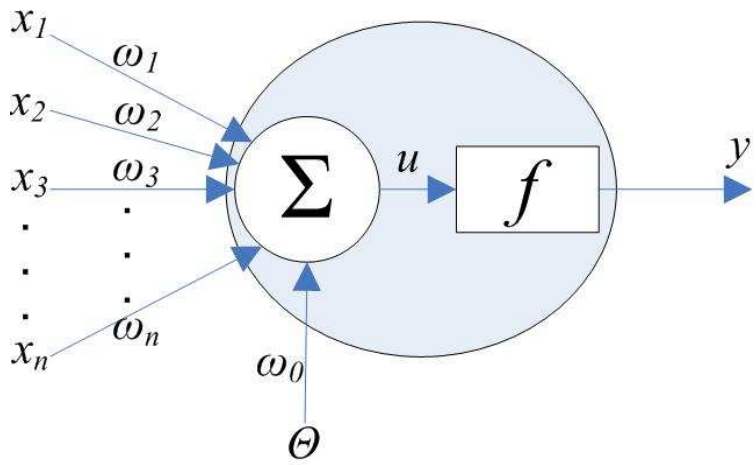
Turbūt didžiausias neuroninių tinklų pranašumas yra tas, kad jie yra labai sėkmingai pritaikomi kaip nežinomų funkcijų aproksimacijos mechanizmas, besimokantis iš stebimų duomenų. Tačiau neuroninių tinklų naudojimas nėra paprastas ir yra būtinos teorinės žinios. Sprendžiant konkretų uždavinį reikia pasirinkti geriausiai jam tinkantį neuroninių tinklų modelį bei mokymosi algoritmą. Modelio pasirinkimas pagrinde priklauso nuo duomenų pateikimo ir pritaikymo. Neverta imti sudėtingesnio modelio nei reikia, nes gali iškilti problemų mokymosi

procese. Daugelis mokymosi algoritmų gali gerai spręsti tą patį uždavinį su teisingais mokymosi parametrais. Algoritmo pasirinkimas ir pritaikymas konkrečiai situacijai reikalauja daug eksperimentuoti.

2.2. Neurono modelis

Neuronas yra pagrindinis neuroninio tinklo informaciją apdorojantis elementas. Jo supaprastintiną modelį galite išvysti 1 pav. Kaip matote neuronas yra sudarytas iš sinapsių aibės, sumatoriaus, slenksčio bei aktyvacijos funkcijos. Kiekviena sinapsė turi savo svorį ω_i, kuris yra dauginamas su įeinančiu signalu x_i. Sumatorius Σ sumuoja visus įėjimo signalus, padaugintus iš atitinkamų sinapsinių svorių. Slenkstis ω_0 nustato vidinio nekintamo dydžio signalo Θ įtaką neuronui. Jis priklausomai nuo ženklo didina arba mažina įėjimą aktyvacijos funkcijai, kuri apibrėžia neurono išėjimo *y* priklausomybę nuo aktyvacijos potencialo *u* reikšmės.

.



1 pav. ***Neurono modelis***

Neurono matematinis modelis yra išreiškiamas 1 formule:

$$y = f(u) = f\left(\sum_{i=1}^{n} x_i \omega_i + \Theta\omega_0\right) \quad (1)$$

čia *y* yra neurono atsako signalas, *f* – neurono aktyvavimo funkcija, ω_i – neurono sinapsės svoris, x_i – įėjimo signalas, ω_0 – papildomas koeficientas įvertinantis vidinio nekintamo dydžio signalo Θ (kuris priklausomai nuo ženklo didina arba mažina įėjimą aktyvacijos funkcijai ir dažniausiai turi -1 vertę) įtaką neuronui ir vadinamas slenksčiu.

Dažniausiai naudojamos neurono aktyvavimo funkcijos:

1. Tiesinė funkcija

$$f(u) = u \tag{2}$$

2. Slenkstinė funkcija (binarinė unipoliarinė funkcija)

$$f(u) = \begin{cases} 1, u > 0 \\ 0, u \le 0 \end{cases} \tag{3}$$

3. Apribota tiesinė funkcija (unipoliarinė rampos funkcija)

$$f(u) = \begin{cases} 1, & u \ge \frac{1}{c} \\ cu, & 0 < u < \frac{1}{c} \\ 0, & u \le 0 \end{cases} \tag{4}$$

čia *c* teigiamoji mastelio konstanta.

4. Logistinė (sigmoidinė unipoliarinė) funkcija

$$f(u) = \frac{1}{1+e^{-u}}, \quad kai \ f(u) \in (0,1) \tag{5}$$

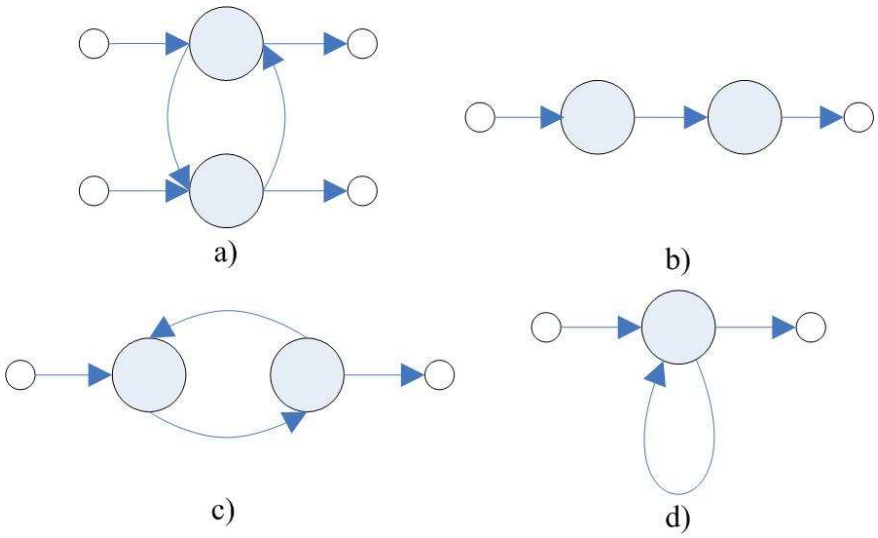
5. Hiperbolinio tangento (sigmoidinė bipoliarinė) funkcija

$$f(u) = \tanh(u) = \frac{e^{u} - e^{-u}}{e^{u} + e^{-u}} = \frac{e^{2u} - 1}{e^{2u} + 1} \tag{6}$$

2.3. Neuroninių tinklų struktūrų charakteristikos

Neuroninius tinklus sudarančių neuronų tarpusavio jungimo schemos pagal ryšius yra skirstomos taip:

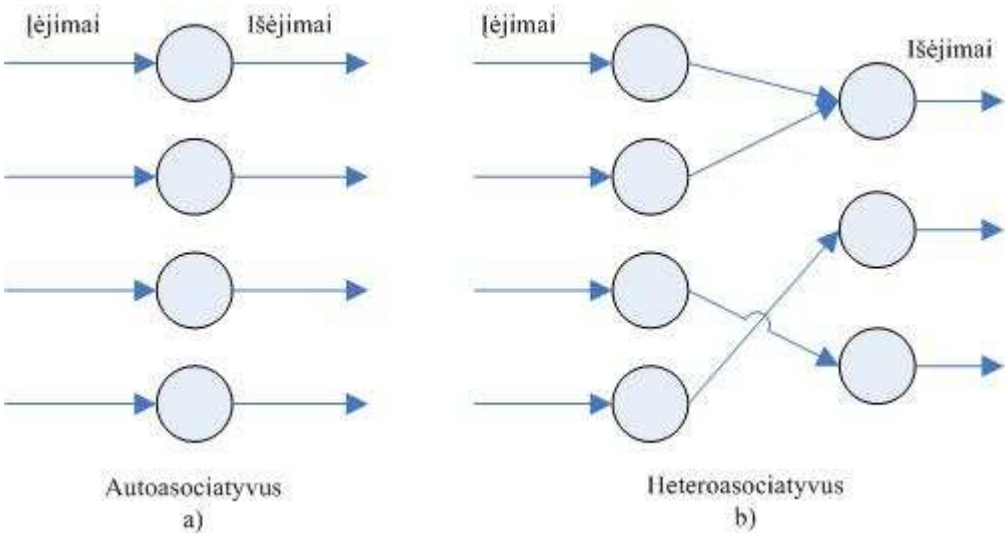
- *intra-sluoksniniai ryšiai*, kurie jungia to paties sluoksnio neuronus tarpusavyje (2 a pav.);
- *inter-sluoksniniai ryšiai*, kurie jungia skirtingų sluoksnių neuronus ir dar yra skirstomi į:
  - *nuoseklius* (2 b pav.);
  - *grįžtamuosius* (2 c pav.);
  - *rekurentinius* (2 d pav.) [14].



2 pav. ***Neuroninių tinklų jungimo schemos***

Paprastai neuroninį tinklą sudarantys neuronai yra skirstomi į keletą sluoksnių. Neuronų sluoksnis, kuris turi tiesioginius ryšius su įėjimais yra vadinamas *įėjimo sluoksniu*. O sluoksnis, kuriame yra išėjimo neuronai yra vadinamas *išėjimo sluoksniu*. Likę neuronų sluoksniai yra vadinami *paslėptaisiais sluoksniais*. Sluoksniai yra numeruojami pradedant nuo įėjimo sluoksnio ir baigiant išėjimo sluoksniu. Jeigu visi neuronai turi tiesioginius ryšius su kitais neuronais, toks neuroninis tinklas vadinamas pilnųjų ryšių neuroninių tinklu [14].

Atsižvelgus į neuroninių tinklų įėjimo ir išėjimo sluoksniuose esančių neuronų skaičių bei naudotų neuronų sluoksnių kiekį, galima išskirti du pagrindinius neuroninių tinklų jungimo tipus (3 pav): *autoasociatyvus*, kuriame įėjimo neuronai kartu yra ir išvesties neuronai (pvz. Hopfildo tinklai); *heteroasociatyvus*, kuriame yra skirtingos įėjimo ir išėjimo neuronų aibės (pvz. daugiasluoksniai perceptronai, Kohoneno tinklai) [3].



3 pav. ***Autosociatyvūs ir heteroasociatyvūs neuroninių tinklų tipai***

Atsižvelgiant į tai ar neuroniniame tinkle yra grįžtamojo ryšio jungčių, galima išskirti dvi neuroninių tinklų architektūras:

- *Vienos krypties* (feedforward) tinkluose tarpusavyje jungiami tik skirtinguose iš eilės einančiuose sluoksniuose esantys neuronai. Visa informacija juose teka viena kryptimi iš vieno sluoksnio į kitą ir pats mokymasis pasireiškia jungčių svorių pokyčiu. Šio tipo neuroniniai tinklai neįsimena buvusių savo būsenų.
- *Tinkluose su grįžtamuoju ryšiu* (rekurentiniai tinklai) informacija gali sklisti kilpa tame pačiame sluoksnyje arba gali būti perduodama iš aukštesnio sluoksnio į žemesnį. Šio tipo tinklai įsimena praėjusias būsenas ir sekančios priklauso ne tik nuo įėjimo signalo, bet ir prieš tai buvusių tinklo būsenų. Kuomet neuroniniame tinkle yra grįžtamieji ryšiai apmokymo procesas visuomet yra labiau komplikuotas, kadangi galima susidurti su ciklišku mu ar rekursija [3].

2.4. Neuroninių tinklų apmokymas

Patraukliausia neuroninio tinklo savybė yra jo gebėjimas mokytis. Mokymasis leidžia modifikuoti atsaką aplinkai. Neuroniniai tinklai yra apmokomi tam, kad gavę įėjimo vektorių aibę *X* jie sugebėtų pateikti tinkamą atsako vektorių aibę *Y*; arba tam, kad tinklas perprastų vidinius duomenų aibės *X* dėsningumus bei struktūrą. Tinklo apmokymui naudojama aibė *X* yra vadinama *apmokymo aibe*, o šios aibės elementas *x* yra vadinamas *pavyzdžiu* [3].

Apmokymo procesu yra siekiama suformuoti tokius ryšius, kurie geriausiai atitiktų *pagrindinę funkciją* tarp įeities ir išeities vektorių. Kiekvieną kartą neuroniniam tinklui gavus pavyzdį yra matuojama paklaida tarp išpranašautos ir tikrosios atsako reikšmės. Vėliau mokymo algoritmas atnaujina jungčių svorius taip, kad būtų pasiektas paklaidų paviršiaus minimumas.

Mokymosi sugebėjimą neuroniniam tinklui suteikia mokymosi algoritmas. Mokymo algoritmai pagrinde skirstomi į tris grupes:

- *Mokymasis su mokytoju*. Mokymo pavyzdžiai yra sudaryti iš įėjimo vektoriaus *x* ir trokštamo atsako vektoriaus *y*. Mokymas vykdomas tol, kol neuroninis tinklas išmoksta susieti įėjimo vektorių *x* su atitinkamu atsako vektoriumi *y*; pavyzdžiui neuroninis tinklas gali mokytis aproksimuoti funkciją $y = f(x)$, kuri išreikštų mokymo pavyzdžių (x, y) aibę.
- *Mokymasis be mokytojo*. Neuroniniam tinklui yra pateikiami tik įėjimo vektoriai *x*. Tinklas mokosi tam tikrų visos jam pateiktos įėjimo vektorių aibės vidinių savybių.
- *Sustiprintas mokymas*. Tai yra aukščiau paminėtų dviejų mokymosi algoritmų

kombinacija. Sustiprintas mokymo metu neuroniniam tinklui yra paduodamas įėjimo vektorius *x* ir žiūrima į tinklo sugeneruotą atsako vektorių. Jeigu jis įvertinamas „gerai",tada tinklui yra suteikiamas „atlygis", t.y. egzistuojantys jungčių svoriai yra padidinami; kitu atveju tinklas yra „nubaudžiamas" – jungčių svoriai, kurie įvertinami „netinkamai nustatytais", yra sumažinami. Sustiprintas mokymas yra mokymas su kritiku, o ne su mokytoju.

Mokymasis nėra individualaus neurono sugebėjimas. Tai yra kolektyvinis viso neuroninio tinklo procesas ir mokymosi algoritmo rezultatas. Sinapsinių jungčių svorių matrica *W* turi globalaus šablono reikšmę. Jos visuma išreiškia neuroninio tinklo „žinias".

Neuroniniuose tinkluose yra laikomasi mokymosi iš pavyzdžių metodo. Mokymosi iš pavyzdžių teorija nagrinėja tris pagrindinius praktinius klausimus:

- Talpumą;
- Pateikiamų pavyzdžių sudėtingumą;
- Skaičiavimų sudėtingumą.

Talpumas apibrėžia, kiek šablonų neuroninis tinklas gali įsiminti, kokias funkcijas jis gali aproksimuoti bei kokios yra jo sprendimų galimybių ribos.

Pavyzdžių sudėtingumas nurodo reikiamų išmokti šablonų skaičių, kuris garantuotų tinkamą duomenų apibendrinimą. Per mažas šablonų skaičius gali sukelti per didelio prisitaikymo prie apsimokymo duomenų efektą, kurio metu tinklas gerai veikia dirbant tik su apmokymo duomenų rinkiniais ir neefektyviai, kai testuojami su nepriklausomais pavyzdžiais.

Skaičiavimų sudėtingumas labiausiai remiasi į laiką, kurio reikia algoritmui, kad jis besiremdamas mokosi pavyzdžiais priimtų tinkamą sprendimą. Dauguma mokymosi algoritmų pasižymi didžiuliu skaičiavimų sudėtingumu. Ypač aktualiu tapo efektyvių neuroninių tinklu mokymo algoritmų kūrimas.

Yra keturios pagrindinės mokymosi taisyklės:

1. Klaidų taisymo;
2. Boltzmann;
3. Hebb'o;
4. Konkurencingojo mokymosi (competitive).

Klaidų taisymo taisyklės

Mokymosi su mokytoju metu, neuroninis tinklas kiekvienam įvesties pavyzdžiui generuoja atitinkamą atsaką *y*, tačiau jis gali skirtis nuo trokštamo atsako $\bar{y}$. Klaidų taisymo mokymosi taisyklė naudodamasi paklaidos signalu $(\bar{y} - y)$ koreguoja sinapsinių ryšių svorius taip, kad būtų sumažinta ši paklaida.

Pradedant daugiasluoksnio neuroninio tinklo apmokymą paprastai sinapsinių jungčių svoriams yra suteikiamos pradinės atsitiktinės reikšmės. Duomenims perėjus neuroninį tinklą yra skaičiuojamos paklaidos tarp apskaičiuotų ir trokštamų rezultatų. Visų neuroninio tinklo svorių korekcija yra daroma taip, kad kuo greičiau sumažinti paklaidų paviršių, dar kitaip šis procesas vadinamas gradiento mažinimu (gradient descent).

Šiame procese labai svarbi yra papildoma didinamoji konstanta vadinama *mokymo žingsniu*, kuri gali padidinti arba sulėtinti nusileidimą gradientu. Apmokymo procesas labai priklauso nuo šios konstantos pasirinkimo. Esant per dideliam mokymosi žingsniui, algoritmas įgyja per mažą globalaus minimumo skiriamąją gebą, o jei mokymo žingsnis per mažas, tada algoritmas per lėtai konverguos ir perilgai truks mokymo procesas. Taip pat mažas mokymosi žingsnis sudėtinguose modeliuose, turinčiuose didelį skaičių lokalių minimumų, gali būti problema, nes modelis gali patekti į lokalųjį minimumą ir dėl mažo šuolio gali iš jo nebeištrūkti.

Boltzmann‘o mokymosi taisyklės.

Boltzmann‘o mašinos yra simetriški rekurentiniai tinklai, sudaryti iš binarinių mazgų (+1 reiškia „įjungta“ ir -1 reiškia „išjungta“). Simetriškumas pasireiškia tuo, kad tinklo *i*-tojo neurono sinapsinės jungties su *j*-tuoju neuronu svoris yra lygus *j*-tojo neurono sinapsinės jungties su *i*-tuoju neuronu svoriu ($\omega_{ij} = \omega_{ji}$). Matomų neuronų poaibis sąveikauja su aplinka, tuo tarpu likę nematomi neuronai ne. Kiekvienas neuronas yra stochastinis vienetas, generuojantis atsaką remiantis statistiniu Boltzmann‘o pasiskirstymo mechanizmu. Boltzmann‘o mašinos veikia dviem režimais:

1. Suvaržytu, kai aplinkos faktorių veikiamiems matomiems neuronams yra paskiriamos tam tikros būsenos;
2. Laisvu, kai matomi ir nematomi neuronai turi veikimo laisvę.

Boltzmann‘o mokymasis yra stochastinė mokymosi taisykle. Boltzmann‘o mokymosi taisyklių tikslas yra taip sureguliuoti sinapsinių jungčių svorius, kad matomų neuronų būsenos atitiktų tam tikrą trokštamą tikimybinį pasiskirstymą. Neuronų sinapsinių svorių ω_{ij} pokytį remiantis Boltzmann‘o mokymosi taisykle galima apskaičiuoti sekančia 7 formule:

$$\Delta\omega_{ij} = \alpha(\overline{\rho}_{ij} - \rho_{ij}) \tag{7}$$

kur α yra mokymosi greitis, o $\overline{\rho}_{ij}$ ir ρ_{ij} yra koreliacija tarp *i*-tojo ir *j*-tojo neuronų būsenų, kuomet tinklas veikia suvaržytu ir laisvu režimu. $\overline{\rho}_{ij}$ ir ρ_{ij} reikšmės paprastai nustatomos remiantis Monte Carlo eksperimentais, kurie užtrunka gana ilgai.

Boltzmann‘o mokymasis gali būti traktuojamas kaip tam tikras klaidų taisymo atvejis, kuriame klaida vertinama ne pagal tai, kaip atitinka norimas atsakas ir gautasis, o pagal dviejų neuronų atsakų koreliacijos skirtumą, veikiant suspaustame ir laisvame režimuose.

Hebb‘o mokymosi taisyklės.

Hebb‘o mokymosi postulatas yra seniausia mokymosi taisyklė. Hebb‘as suformulavo ją besiremdamas neurobiologinių tyrimų rezultatais. Tyrimų rezultatai parodė, kad jeigu sinapsinio ryšio abiejuose galuose esantys neuronai yra sinchroniškai ir pakartotinai aktyvuojami, tada sinapsės svarumas yra padidinamas. Matematiškai Hebb‘o taisyklė yra išreiškiama 8 formule:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha y_j(t) x_i(t) \tag{8}$$

kur x_i ir y_j yra *i*-tojo ir *j*-tojo neuronų atsakų reikšmės, w_{ij} sinapsinio ryšio tarp jų svoris, α mokymosi greitis. Svarbi taisyklės savybė yra ta, kad mokymasis vyksta tik lokaliai ir sinapsių svorių pokyčiai priklauso tik nuo jos galuose esančių dviejų neuronų veiklos. Toks mokymasis labai supaprastina painią mokymosi proceso grandinę.

Konkurencinio mokymosi taisyklės

Kitaip nei Hebb‘o mokymesi, kuriame leidžiama daugeliui išėjimo neuronų būti sužadintiems vienu metu, konkurencinio mokymo išėjimo neuronai konkuruoja tarpusavyje dėl aktyvacijos. Tik vienas išėjimo neuronas gali būti aktyvus konkrečiu momentu. Šis reiškinys yra vadinamas „viskas priklauso nugalėtojui“.

Konkurencinis mokymasis pagrinde naudojamas įvesties duomenų grupavimui ar kategorizavimui. Tinklas panašias struktūras sugrupuoja ir pateikia kaip vieną neuroną. Grupavimas yra atliekamas automatiškai, atsižvelgiant į duomenų koreliaciją.

Paprasčiausias konkuruojančio mokymosi tinklas yra sudarytas iš vieno išėjimo sluoksnio. Kiekvienas išėjimo neuronas i jungiasi su visai įėjimo neuronais sinapsiniais ryšiais su svoriais ω_{ij}, kur *j=1,2,...,n*. Kiekvienas išėjimo neuronas taip yra slopinančiomis jungtimis sujungtas su visais likusiais išėjimo sluoksnio neuronais bei turi žadinančią jungtį pats su savimi. Konkuravimo rezultate tiktai vienas neuronas i' su didžiausiu aktyvavimo signalu tampa nugalėtoju. Paprasta konkurencinio mokymosi taisyklė, kuri nurodo sinapsinių jungčių svorio pokyčio apskaičiavimą išreiškiama 9 formule:

$$\Delta\omega_{ij} = \begin{cases} \alpha\left(x'_j - \omega_{i'j}\right), & i = i' \\ 0, & i \neq i' \end{cases} \tag{9}$$

Kur $\Delta\omega_{ij}$ *i*-tojo išėjimo sluoksnio neurono sinapsinio ryšio su *j*-tuoju įėjimo neurono svorio pokytis; α - mokymosi greičio konstanta.

Galime pastebėti, kad yra atnaujinami tik neurono „nugalėtojo“ sinapsių svoriai. Šios mokymosi taisyklės tikslas yra pakoreguoti „nugalėtojo“ svorius taip, kad jie būtų artimesni įvesties pavyzdžiui. Kaip matote iš konkurencinio tinklo taisyklės, tinklas nesustos mokytis tol, kol mokymosi greičio konstanta nebus lygi 0. Tam tikras įėjimo pavyzdys skirtingose iteracijose gali sužadinti skirtingus išėjimo neuronus. Taip užtikrinamas mokymosi sistemos stabilumas.

Jeigu po baigtinio mokymosi iteracijų kiekio pakartotinai į tinklą patenkantys pavyzdžiai nebekeičia savo kategorijos, sistema vadinama stabilia. Vienas iš būdų pasiekti sistemos stabilumą yra mokymosi greičio konstantos palaipsnis mažinimas iki 0.

1 lentelėje galite išvysti įvairių mokymosi algoritmų ir su jom susijusių neuroninių tinklų architektūrų suvestinę. Tiek mokymasis su mokytoju, tiek mokymasis be mokytojo naudoja klaidos taisymo, Hebb‘o ir konkurencinio mokymosi taisykles. Klaidos taisymu grįstos mokymosi taisyklės naudojamos vienakrypčių tinklų apmokymui. Hebb‘o taisyklės gali būti naudojamos visų tipų tinklų architektūroms. Tačiau kiekvienas mokymosi algoritmas yra sukurtas konkrečios neuroninio tinklo architektūros mokymui. Kiekvienas algoritmas yra skitas tam tikrų užduočių sprendimui ir jas galite paskutiniame lentelės stulpelyje.

1 lentelė. ***Geriausiai žinomi mokymosi algoritmai***

Paradigma	Mokymosi taisyklė	Architektūra	Mokymosi algoritmas	Užduotis
Mokymasis su mokytoju	Klaidos korekcijos (Error-correction)	Vienasluoksnis arba daugiasluoksnis perceptronas	Perceptrono mokymosi algoritmai, Adalinas ir Madalinas	Šablonų klasifikacija, Funkcijų aproksimacija, Prognozavimas, kontrolė
	Boltsmano	Rekurentinė	Boltsmano mokymosi algoritmas	Šablonų (pattern) klasifikacija
	Hebbian	Daugiasluoksnė vienakryptė	Tiesinė diskriminanto analizė	Duomenų analizė, Šablonų klasifikacija
	Konkurencinė (competitive)	Konkurencinė	Mokymosi vektoriaus kvantavimas (LVQ)	Vidinis klasės suskirstymas, Duomenų suspaudimas
		ART tinklai	ARTMap	Vidinis klasės suskirstymas (within-class categorization), Šablonų klasifikacija
Mokymasis be mokytojo	Klaidos korekcijos (Error-correction)	Daugiasluoksnė vienakryptė	Sammon‘o projekcija	Duomenų analizė
	Hebbian	Vienakryptė arba konkurencinė	Principinė komponentų analizė	Duomenų analizė, duomenų suspaudimas
		Hopfield tinklai	Asociatyvus mokymasis	Asociatyvinė atmintis
	Konkurencinė	Konkurencinė	Vektorių kvantavimas	Kategorizacija,

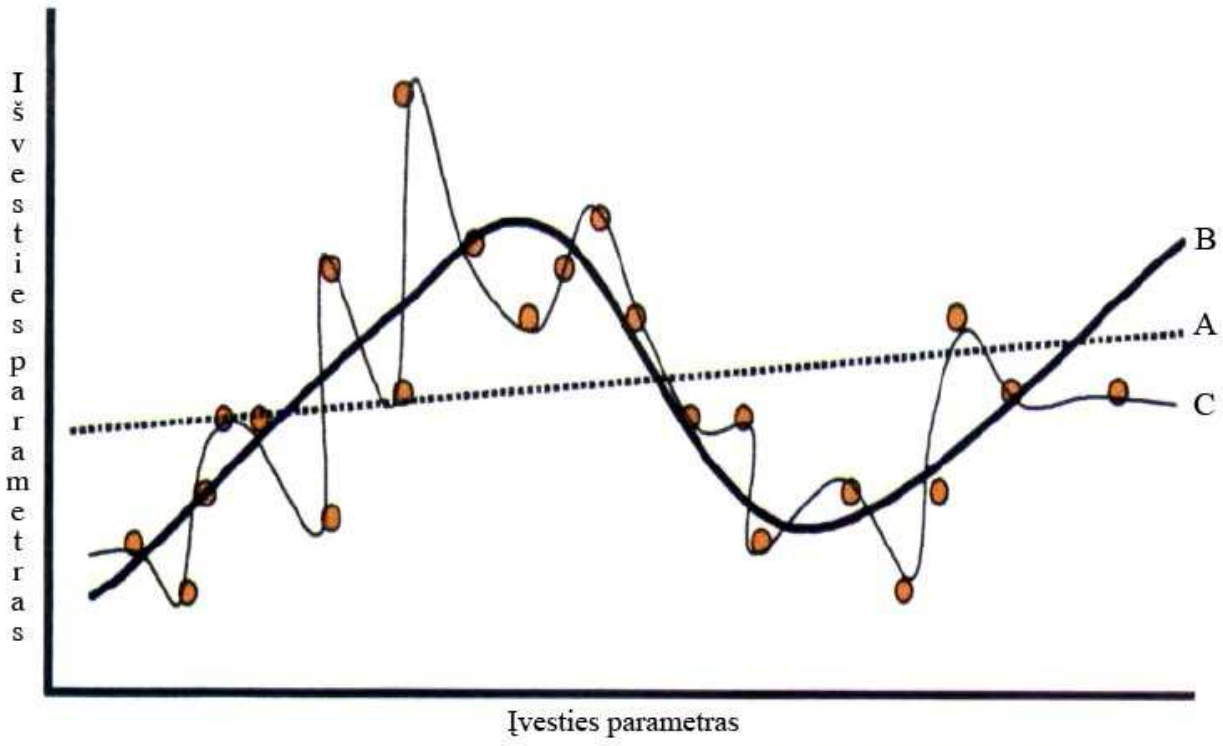
				Duomenų suspaudimas
		Kohonen‘o SOM	Kohonen‘o SOM	Kategorizacija, duomenų analizė
		ART tinklai	ART1, ART2	Kategorizacija
Hibridinė	Klaidos korekcijos ir konkurencinė	RBF tinklai	RBF mokymosi algoritmas	Šablonų klasifikavimas, funkcijų aproksimavimas, prognozė, kontrolė

Automatizuotas mokymas

Neuroninio tinklo mokymas yra iteratyvus procesas, kurį mokymo algoritmas vykdo automatiškai. Mokymas susideda iš šių žingsnių:

- Įvestis: apmokymo duomenys yra įvedami į modelį. Programa apdoroja duomenis ir atlieka iteratyvų tinklo jungčių svorių korekciją.
- Mokymo paklaidos: šio proceso metu yra minimizuojamos mokymo paklaidos.
- Optimalus mokymas: optimaliai apmokytas neuroninio tinklo modelis apibrėžia ryšius, kurie tiksliai išreiškia bendrą koreliaciją tarp įvesties ir išvesties parametrų. Jeigu modelis yra permokytas, tada bendras sąryšis gali būti nenustatytas ir tuo pačiu jis negali būti atstovaujamas modelio ryšių svorių rinkiniu. Permokytas modelis gali modeliuoti pavyzdžius gerai, tačiau gali skirtis nuo tikrojo sąryšio.

Tai galite pamatyti 4 pav., kuriame sužymėtų duomenų bendras sąryšis matomas glotnioje linijoje B. Linija A rodo, kad modelis yra apmokomas, o iš C matosi, kad jis jau yra permokytas. Permokytas modelis, su sudėtingais, didelio laipsnio ryšiais gali gerai tikti apmokymo duomenims, tačiau gavus kitus duomenis gali susidurti su dideliais prognozavimo netikslumais.



4 pav. ***Neuroninio tinklo mokymasis [11]***

Daugiasluoksnių neuroninių tinklų modeliai su minimaliu paslėptų sluoksnių bei apdorojimo elementų skaičiumi yra atsparūs permokymui.

Mokymo tikslas

Apmokymo paklaidos minimizavimas nėra mokymo proceso uždavinys. Juo laikomas siekis minimizuoti klaidą, kai neuroninis tinklas naudoja duomenis, kurių nenaudojo apmokyme. Tai reiškia, kad norint korektiškai apmokyti daugiasluoksnį neuroninį tinklą, reikia turimus duomenis padalinti į du poaibius – apmokymo aibę ir testavimo aibę. Testavimo aibė yra skirta apmokymo duomenimis apmokyto neuroninio tinklo apibendrinimo paklaidai įvertinti. Apskaičiuotas paklaidos vidurkis tarp išpranašautų ir tikrųjų testavimo duomenyse esančių atsako reikšmių nurodo ar modelis yra tinkamai apmokytas.

Laikui bėgant apmokymo ir apibendrinimo paklaidų įverčiai mažėja, tačiau modeliui persimokius jos vėl pradeda augti. Tai atsitinka todėl, kad modelis vietoj bendro duomenų sąryšio įsimena apmokymo duomenų šabloną [15].

Neuroninio tinklo stabilumas ir konvergencija

Neuroninių tinklų elgsena juos mokinant yra apibūdinama neuroninio tinklo stabilumo bei konvergencijos sąvokomis. Praktiškai naudingi gali būti tik stabilūs ir konverguojantys neuroniniai tinklai. Tinklo stabilumui įrodyti naudojamos trys teoremos:

- Coheno-Grosbergo teorema;

- Coheno-Grosbergo-Kosko teorema;
- ABAM teorema.

O tuo tarpu neuroninio tinklo konvergencija gali būti įrodyta dvejopai:

- su tikimybe 1;
- vidutinių kvadratų metodu.

2.5. Vienasluoksnis perceptronas

1958 m. Rosenblatt pasiūlė vieną iš pirmųjų neuroninio tinklo modelių. Jis buvo pavadintas vienasluoksniu perceptronu (toliau vadinamas tiesiog perceptronu). Perceptronas buvo sudarytas iš vieno neurono su slenkstine aktyvavimo funkcija. Įėjimo signalų ir atitinkamų jungčių svorių sandaugų sumai viršijus neurono slenksčio reikšmę, būdavo generuojamas atsako signalas lygus 1, o priešingu atveju atsako signalas būdavo lygus 0. Šiuo metu vietoj slenkstinės aktyvavimo funkcijos dažniausiai naudojamos apribota tiesinė, sigmoidinė bei gauso aktyvavimo funkcijos.

Perceptrono jungimo struktūra yra vienakryptė ir jis susideda iš dviejų sluoksnių. Pirmasis sluoksnis yra vadinamas įėjimo sluoksniu, nes jam yra tiekiami įėjimo duomenys. Šio sluoksnio neuronai yra sujungiami su antrojo – išėjimo sluoksnio neuronais. Perceptrono apmokymo algoritmas:

1. Perceptrono, turinčio n įėjimų ir m išėjimų, ryšių bei slenksčių inicializavimas. Visiems perceptrono ryšių svoriams ω_{ij} bei slenksčių svoriams ω_{0j}, kur $i = 1, 2, ..., n; j = 1, 2, ..., m$, priskiriamos atsitiktinės mažas reikšmes. Vidinis nekintamas signalas Θ_j yra nustatomas -1.
2. Įėjimo pavyzdžių pateikimas. Perceptronui yra paduodamas įėjimo pavyzdys x ir pagal formulę 10 yra apskaičiuojamas kiekvieno perceptrono neurono atsakas y_j.

$$y_j = f\left(\sum_{i=1}^{n} x_i \omega_{ij} - \omega_{0j}\right), \quad kai \ f(u) = \begin{cases} 1, u > 0 \\ 0, u \le 0 \end{cases} \tag{10}$$

3. Tikrinimas. Kiekvienam neuronui yra skaičiuojama klaida tarp esamo y_j ir norimo gauti atsako $\bar{y}_j$.

$$Err_j = \bar{y}_j - y_j \tag{11}$$

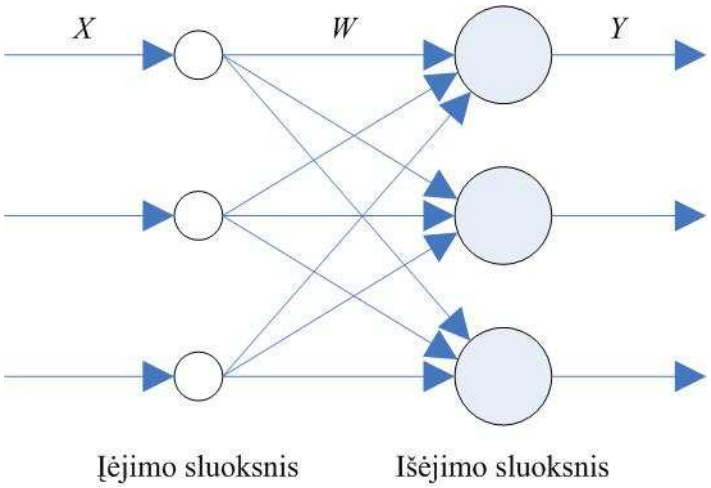
4. Ryšių bei slenksčių adaptavimas. Prie esamų ryšių bei slenksčių svorių reikšmių pridedama:

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \alpha \cdot x_i \cdot Err_j \tag{12}$$

$$\omega_{0j}(t+1) = \omega_{0j}(t) + \alpha \cdot Err_j \quad (13)$$

kur α yra mokymosi greičio konstanta, kuri priklauso intervalui *[0,1]*.

5. Žingsniai 2 – 4 kartojami tol, kol klaida Err taps pakankamai maža.



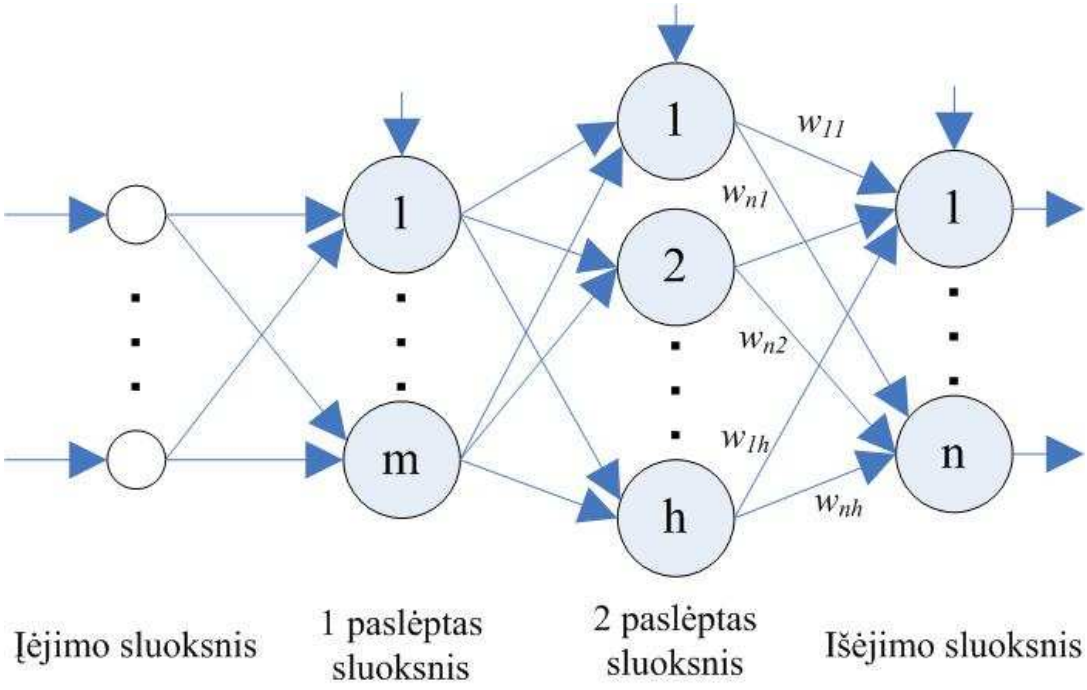
5 pav. ***Vienasluoksnis perceptronas***

1960 m. Rossenblatt įrodė, kad perceptronų mokymo algoritmas suras sprendinį bet kokiam tiesiškai atskiriamam uždaviniui per baigtinį laiko intervalą Tačiau taikydami perceptroną sprendžiant realius klasifikavimo uždavinius susiduriame su šiomis problemomis:

1. Blogos apibendrinimo savybės.
2. Perceptronas negali įsisavinti netiesiškai išskiriamų klasių.
3. Ilgai trunkantis mokymas.

2.6. Daugiasluoksnis perceptronas

Kad būtų apeiti perceptronų tiesinio išskiriamumo apribojimai buvo sukurti daugiasluoksniai perceptronai. Tipinę daugiasluoksnio perceptrono konfigūraciją galite išvysti 6 pav. Kaip matote neuronai yra suskirstyti į paslėptuosius ir išėjimo sluoksnius. Įėjimo sluoksnyje paprastai neuronai nėra realizuojami. 6 pav. pavaizduotame daugiasluoksniame perceptrone kiekvieno sluoksnio neuronai yra pilnai sujungti tik su sekančio sluoksnio neuronais. Praktikoje tarp neuronų taip pat leidžiamos ir acikliškos jungtys. Kiekvienas neuronas daugiasluoksniame perceptrone turi netiesinę aktyvacijos funkciją, kuri dažniausiai yra tolydžiai diferencijuojama. Dažniausiai vartojamos yra simgoidinės bei hiperbolinio tangento aktyvavimo funkcijos. Buvo įrodyta, kad daugiasluoksnis perceptronas, turintis nors du paslėptus sluoksnius ir pakankamą paslėptų neuronų skaičių, yra pajėgus aproksimuoti pakankamai sudėtingą sąryšį.



6 pav. ***trijų sluoksnių daugiasluoksnis perceptronas***

Daugiasluoksnio perceptrono mokymas atbulo sklidimo algoritmu

Tariama, kad svoriai surišti su kiekvieno sluoksnio neuronais formuoja to sluoksnio svorių matricą W^s kur $s \in [1, S]$ sluoksnio numeris (1 numeriu žymimas *1-asis* paslėptas sluoksnis, o *S*-tasis numeris priklauso išėjimo sluoksniui) (įėjimo sluoksnis neturi neuronų, todėl jis neturi ir svorių matricos). Šis atbulo sklidimo metodas siūlo rasti geriausias svorių reikšmes.

Duota mokymo pavyzdžių aibė *{(X(k),* $\bar{Y}(k)$*), kur* $k \in [1, K]$ yra apmokymo pavyzdžio numeris*}* ir atbulinio sklidimo mokymas pradedamas suteikiant tinklo ryšių svoriams mažas atsitiktines reikšmes, rekomenduojama patenkančias į intervalą *[-1/n, 1/n]*, kur *n* yra įėjimo signalų skaičius[3]. Po to paduodant į daugiasluoksnį perceptroną visus *K* pavyzdžius yra apskaičiuojami atsakai $y_m^s(k)$, čia *m* yra *s-tajame* sluoksnyje esančio neurono numeris. Tada yra apskaičiuojama *susumuota kvadratinė klaida*:

$$E = \sum_{k=1}^{K} \sum_{m=1}^{M} \left(e_m(k)\right)^2 = \sum_{k=1}^{K} \sum_{m=1}^{M} \left(\bar{y}_m(k) - y_m^S(k)\right)^2 \qquad (14)$$

Čia e_m yra išėjimo sluoksnio *m-tojo* neurono klaidos signalas, y_m^S išėjimo sluoksnio *m-tojo* neurono atsakas, $\bar{y}(k)$ trokštamas signalas, *M* išėjimo sluoksnyje esančių neuronų skaičius.

Tikslas yra pakoreguoti jungčių svorius taip, kad būtų minimizuota susumuota kvadratinė klaida *E*. Ir čia susiduriame su netiesiniu mažiausių kvadratų optimizavimo uždaviniu. Kurio

sprendimui radimui dažniausiai naudojama gradientinio nuolydžio (gradient descent) metodas. Šis metodas mažina svorius priešinga susumuotos kvadratinės klaidos *E* gradientui kryptimi. Taigi svorio pokytis ΔW^s turi neigiamą kryptį ir jis apskaičiuojamas sekančia 15 formule:

$$\Delta W^s = -\alpha \frac{dE}{dW^s} \tag{15}$$

kur W^s yra *s-tojo* sluoksnio svorių matrica, α yra mokymosi greičio konstanta, o susumuotos kvadratinės klaidos *E* dalinę išvestinę svorio ω_{ij}^s atžvilgiu galime rasti taip:

$$\begin{aligned} \frac{\partial E}{\partial \omega_{ij}^s} &= -2\sum_{k=1}^{K} \frac{\partial E}{\partial u_i^s(k)} \cdot \frac{\partial u_i^s(k)}{\partial \omega_{ij}^s} = -2\sum_{k=1}^{K} \left[\delta_i^s(k) \cdot \frac{\partial}{\partial \omega_{ij}^s} \sum_m \omega_{im}^s y_m^{s-1}(k) \right] = \\ &= -2\sum_{k=1}^{K} \delta_i^s(k) \cdot y_j^{s-1}(k) \end{aligned} \tag{16}$$

Kur $u_i^s(k)$ *s-tojo* sluoksnio *i-tojo* neurono aktyvacijos potencialas, y_m^{s-1} yra *s-1* sluoksnio m-tojo neurono apskaičiuotas atsakas. Dydis δ_i^s išėjimo sluoksniui išreiškiamas sekančiai:

$$\delta_i^s(k) = [\bar{y}_i(k) - y_i^s(k)] f'\left(u_i^s\right) \tag{17}$$

Kur f' neurono aktyvacijos funkcijos išvestinė. O visiems likusiems sluoksniams dydis δ_i^s apskaičiuojamas pagal šią formulę:

$$\begin{aligned} \delta_i^s(k) &= \frac{\partial E}{\partial u_i^s(k)} = \sum_{m=1}^{M} \frac{\partial E}{\partial u_m^{s+1}(k)} \cdot \frac{\partial u_m^{s+1}(k)}{\partial u_i^s(k)} = \\ &= \sum_{m=1}^{M} \left[\delta_m^{s+1}(k) \cdot \frac{\partial}{\partial u_i^s(k)} \sum_{j=1}^{J} \omega_{mj}^s f\left(u_j^s(k)\right) \right] = f'\left(u_i^s(k)\right) \cdot \sum_{m=1}^{M} \delta_m^{s+1}(k) \cdot \omega_{mi}^s \end{aligned} \tag{18}$$

18 formulė yra klaidos atgalinio sklidimo formulė, kuria apskaičiuojama delta klaida pradedant išėjimo sluoksniu ir baigiant pirmuoju paslėptuoju sluoksniu. Žemesnio sluoksniui esančiam neuronui delta klaida δ_i^s skaičiuojama sudedant aukštesniame sluoksnyje esančių neuronų apskaičiuotų delta klaidų δ_m^{s+1} ir jungčių, vedančių į atitinkamus aukštesnio sluoksnio neuronus, svorių sandaugas.

Daugiasluoksniam perceptronui pateikus visus K mokymo pavyzdžių, galima apskaičiuoti bendrą svorio ω_{ij}^s pokytį $\Delta\omega_{ij}^s$, kuris yra gaunamas susumavus kiekvieno mokymo pavyzdžio atveju apskaičiuotą šio svorio pokytį. O pats svoris atlikus vieną mokymo ciklą yra apskaičiuojamas pagal žemiau esančią formulę:

$$\omega_{ij}^{s}(t+1) = \omega_{ij}^{s}(t) + \alpha \sum_{k=1}^{K} \delta_i^s(k) y_j^{s-1}(k) \quad (19)$$

Svoriai yra atnaujinami kiekvieno mokymosi ciklo pabaigoje. Mokymosi ciklas yra procesas, kurio metu vienas arba keletas mokymo pavyzdžių yra perduodami neuroniniu tinklu ir yra paskaičiuojama klaida *E*. Aukščiau pateiktose formulėse ciklas užtrunka tol, kol klaida *E* bus apskaičiuota visiems *K* mokymo pavyzdžių.

Tačiau praktikoje siūloma naudoti papildytą svorių korekcijos formulę 20 [17]:

$$\omega_{ij}^{s}(t+1) = \omega_{ij}^{s}(t) + \alpha \sum_{k=1}^{K} \delta_i^s(k) y_j^{s-1}(k) + \mu\left[\omega_{ij}^{s}(t) - \omega_{ij}^{s}(t-1)\right] + \varepsilon_{ij}^{s}(t) \quad (20)$$

Dešinėje lygties pusėje antras sumos narys yra klaidų kvadratų vidurkio gradientas ω_{ij}^{s} atžvilgiu. Trečias sumos narys žinomas kaip momentas, kuris leidžia adaptyviai pritaikyti žingsnio dydį. Kada gradiento vektorius einančiuose vienu po kito cikluose turi tą pačią kryptį, tada žingsnis yra didinamas (įgaunamas momentas). Jeigu viename paskui kitą einančiame cikle gradiento vektorius vis keičia kryptį zigzagais, tada momentas sureguliuos gradiento kryptį taip, kad būtų sumažinta vidutinė kvadratinė paklaida.

Čia yra du parametrai, kuriuos reikia pasirinkti: mokymosi greitis α ir momento konstanta μ. Abu parametrai turi priklausyti intervalui *[0,1]*. Praktikoje mokymosi greičiui α dažniausiai yra suteikiama maža reikšmė $0 < \alpha < 0{,}3$. Mmomento konstantai μ suteikiama didesnė reikšmė $0{,}6 < \mu < 0{,}9$.

Paskutinis 20 formulės narys ε_{ij}^{s} yra mažas atsitiktinis triukšmo dydis, kuris turi mažą reikšmę, kai antras ir trečias formulės nariai turi dideles reikšmes. Kada mažiausios vidutinės kvadratinės paklaidos paieška patenka į lokalųjį minimumą, tada labai sumažėja gradiento vektoriaus bei momento reikšmės ir mokymosi algoritmas gali iš jo nebeištrūkti. Tokioje situacijoje triukšmas gali padėti mokymosi algoritmui išlipti iš lokalaus minimumo ir tęsti optimalaus globalaus sprendimo paiešką.

2.7. SOM (Savaime susitvarkantys žemėlapiai)

SOM (Self-Organizing Map) yra vienasluoksnis, konkuruojantis neuroninis tinklas, kuris surikiuoja neuronus tam tikra tvarka [17]. Jis konvertuoja daugiamačių duomenų netiesinius statistinius tarpusavio ryšius į paprastai taisyklingą dvimatį iš mazgų sudarytą tinklelį.

Taigi SOM‘as tuo pačiu suspaudžia informaciją, kartu išsaugant svarbiausius duomenų elementų tarpusavio ryšius. Taigi galima sakyti jis atlieka tam tikrą abstrakciją.

SOM‘as formaliai gali būti apibrėžtas kaip netiesinis, surūšiuotas, glotnus (smooth) daugiamačių įeities duomenų vertimas į mažamačio (low-dimensional) masyvo elementus.

Vienas iš labiausiai naudojamų neuroninių tinklų modelių yra SOM (Self-Organizing Map) kurį pateikė ir išvystė Teuvo Kohonen (1982, 1990, 1993).

SOM‘as sudarytas iš dviejų sluoksnių – įėjimo sluoksnio ir išėjimo sluoksnio, dar vadinamo būdingu žemėlapiu, kuris vaizduoja išėjimo vektorius. Visų įėjimo sluoksnio *n* neuronų jungčių su išėjimo sluoksnio *j*-tuoju neuronu svoriai formuoja *n*-matavimo svorių vektorių ω_j.

Išėjimo neuronai apmokymo proceso metu yra specializuojami reaguoti į tam tikroms grupėms (klasterius) priklausančius įėjimo vektorius ir išreikšti tipines įėjimo vektorių savybes. SOM‘as sugeba išgauti iš daugiamačių duomenų abstrakčią informaciją ir išreikšti ją vienmatėje, dvimatėje ar trimatėje erdvėje.

Neuronai išėjimo sluoksnyje yra konkuruojantys. Yra įvesta horizontali sąveika tarp kaimyninių neuronų, t.y. neuronai turi stiprias žadinančias jungtis su savim ir kelias žadinančias jungtis su tam tikro spindulio kaimynais. Neuronai, esantys už šio spindulio, yra slopinami slopinančiomis jungtimis arba yra visai neįtakojami. Dar galima taisyklė „nugalėtojas pasiima viską“, kuri nurodo, kad tik vienas neuronas po konkuravimo dėl įėjimo vektoriaus tampa nugalėtoju. Jis atstovauja šio vektoriaus klasei ir būdingiems bruožams.

SOM‘as transformuoja panašumus tarp įėjimo vektorių į topologinį neuronų artumą, išreiškiant tai topologiniu žemėlapiu. Panašūs vektoriai yra išreiškiami artimais neuronais išėjimo erdvėje. Atstumas tarp neuronų išėjimo erdvėje yra reikšminga tinklo savybė.

Yra dvi SOM‘o naudojimo galimybės. Pirmoji – naudoti jį mokymosi be mokytojo metodu, kai įėjimo vektoriams priklausančios klasės yra nežinomos. Antroji galimybė yra po mokymosi be mokytojo atlikti mokymąsi su mokytoju. Šiam tikslui buvo sukurti LVQ (learning vector quantization) algoritmai (Kohonen 1990). Po mokymosi be mokytojo pabaigos, LVQ algoritmai kalibruoja išėjimo neuronus, priskirdami juos žinomoms klasėms ir yra atliekamas mokymasis su mokytoju, kuris papildo žemėlapį žiniomis apie išėjimo klases, regionus, grupes ir klasių žymes.

SOM mokymasis be mokytojo

Po kiekvieno įėjimo vektoriaus patekimo į SOM‘ą, yra išrenkamas neuronas – laimėtojas ir jo kaimyninėje srityje Nt yra padidinami jungčių svoriai, o neuronų, esančių už šios srities, jungčių svoriai išlieka tie patys. Rekomenduojama atlikti ne mažiau kaip 500 kartų daugiau mokymosi ciklų nei yra išėjimo neuronų.

Sinapsių svorių vektoriai siekia aproksimuoti įėjimo vektorių tankumo funkciją. Sinapsių svoriai ω_j konverguoja eksponentiškai į vektorių grupių centrus ir visas žemėlapis išreiškia tam

tikrą įėjimo duomenų pasiskirstymo tikimybę.

1. Kiekvieno išėjimo sluoksnio neurono svorių vektoriui priskirti mažas atsitiktines reikšmes.
2. Laiko momentu t gauti naują įėjimo vektorių ***x***.
3. Kiekvienam neuronui *j* suskaičiuoti atstumą *dj* (*n*-matėje erdvėje) tarp įėjimo vektoriaus ***x*** ir svorių vektoriaus $\boldsymbol{\omega_j(t)}$. Euklidinėje erdvėje jis yra skaičiuojamas taip:

$$d_j = \sqrt{\sum \left(x_i - \omega_{ij}\right)^2} \tag{21}$$

4. Neuronas *k*, kurio svorių vektorius yra arčiausiai vektoriui ***x*** yra laikomas nugalėtoju ir jis tampa kaimyninės srities *Nt* centru.
5. Visų neuronų, priklausančių kaimyninei sričiai *Nt*, svorių vektorių pakeitimas:

$$\begin{aligned} &\omega_j(t+1) = \omega_j(t) + \alpha\left(x - \omega_j(t)\right), \quad \textit{jeigu } j \in Nt, \\ &\omega_j(t+1) = \omega_j(t), \quad \textit{jeigu } j \notin Nt \end{aligned} \tag{22}$$

Visi žingsniai nuo 2 iki 5 yra kartojami visiems mokymo pavyzdžiams. Palaipsniui yra mažinamas Nt ir mokymosi greitis α. Šita procedūra turi būti kartojama su tais pačiais mokymo duomenimis iki SOM‘o konvergavimo.

LVQ algoritmai mokymuisi su mokytoju

SOM‘o mokymasis be mokytojo turėjo problemų įėjimo vektorių, kurie patenka į dviejų išėjimo vektorių sričių ribą, atskyrime ir žinomų klasių žymių priskyrime neuronams. Taigi šitų problemų sprendimui buvo pasiūlyti LVQ algoritmai LVQ1, LVQ2 ir LVQ3.

LVQ1 algoritme svorių vektoriai yra priskiriami klasėms, kurios turi savo žymes. Pradinės svorių vektorių reikšmės yra nustatomos SOM algoritmu. Po to išėjimo vektoriai yra pažymimi klasių, kurioms jie priklauso, žymėmis. Tada yra atliekama svorių korekcija žemiau pateiktomis formulėmis:

$$\begin{aligned} &\omega_j(t+1) = \omega_j(t) + \alpha(t)\left(x(t) - \omega_j(t)\right), \ \textit{jeigu tinklas gerai priskiria vektorių } \mathbf{x} \\ &\textit{klasei } c_j, \textit{atstovaujamai } j - \textit{tuoju neuronu} \\ &\omega_j(t+1) = \omega_j(t) - \alpha(t)\left(x(t) - \omega_j(t)\right), \\ &\textit{jeigu } \mathbf{x} \textit{ buvo klaidingai priskirtas klasei} - \textit{neuronui} \\ &\omega_i(t+1) = \omega_i(t), \textit{ visiems i besiskiriantiems nuo } j \end{aligned} \tag{23}$$

kur $\alpha(t)$ laikui bėgant monotoniškai mažėja

LVQ2 algoritmas koreguoja laimėjusio neurono tiesioginių kaimynų jungčių svorius. Jeigu c_i yra laimėjusio neurono klasė, tačiau vektorius ***x*** priklauso klasei c_j, tada neuronų *i* ir *j* svorių apskaičiavimui yra naudojamos šios formulės:

$$\begin{aligned}
&\omega_i(t+1) = \omega_i(t) - \alpha(t)\big(x(t) - \omega_i(t)\big),\\
&\omega_j(t+1) = \omega_j(t) + \alpha(t)\big(x(t) - \omega_j(t)\big),\\
&\omega_k(t+1) = \omega_k(t), \ \textit{likusiems k neuronų}
\end{aligned} \tag{24}$$

LVQ3 algoritmas naudojamas, kada atstumas tarp įėjimo vektoriaus bei *i*-tojo ir *j*-tojo neuronų svorių vektorių yra vienodi, t.y. patenka į „langą“ tarp jų.

$$\begin{aligned}
&\omega_i(t+1) = \omega_i(t) - \alpha(t)\big(x(t) - \omega_i(t)\big),\\
&\textit{jeigu } \mathbf{x} \textit{ patenka į langą ir } \mathbf{x} \textit{ priklauso klasei } c_j\\
&\omega_j(t+1) = \omega_j(t) + \alpha(t)\big(x(t) - \omega_j(t)\big),\\
&\text{kai k} \in \{\text{i, j}\} \textit{ ir } \mathbf{x} \textit{ patenka į langą ir } \mathbf{x} \textit{ priklauso klasei } c_j\\
&\omega_k(t+1) = \omega_k(t) + \alpha(t)\big(x(t) - \omega_k(t)\big),\\
&\text{kai k} \in \{\text{i, j}\} \textit{ ir } \mathbf{x} \textit{ patenka į langą ir i bei j priklauso tai vienai klasei}
\end{aligned} \tag{25}$$

Visi šie trys LVQ algoritmai daugelį statistinių šablonų aptikimo uždavinių atlieka panašiu tikslumu, tačiau kiekvienas iš jų priklauso skirtingai filosofijai. LVQ1 ir LVQ3 apibrėžia tvirtesnį procesą, kuriame neuronų svorių vektoriai yra tariamai pastovesni. LVQ1 mokymosi greitis gali būti apytiksliai optimizuotas greitesnei konvergencijai pasiekti. LVQ2 reliatyvūs atstumai tarp svorių vektorių ir klasių kraštų yra optimizuoti, tačiau nėra garantijos, kad svorių vektoriai yra optimaliai išsidėstę klasių pasiskirstymo apibūdinimo atžvilgiu. Patartina LVQ2 naudoti tik su mažu mokymosi greičiu bei apribotu mokymosi žingsnių skaičiumi.

3. DARBO APRAŠYMAS

3.1. Ligos diagnozavimas

Ligos diagnozės nustatymas nėra trivialus uždavinys, ypač kai liga turi keletą atmainų ir/ar stadijų, kurių simptomai sąlyginai panašūs, todėl gydytojams labai pravestų tam tikroms ligų grupėms specializuoti e-patarėjai. Taip pat ir eiliniams žmonėms, abejojantiems dėl savo sveikatos būklės ir dėl tam tikrų priežasčių neturintiems galimybių apsilankyti pas gydytojus, būtų naudinga turėti savo sveikatos pasitikrinimo internetu galimybę. Šio uždavinio sprendimui sėkmingai gali būti panaudotos adaptyvios e-mokymosi sistemos, naudojančios dirbtinių neuroninių tinklų metodus.

Kadangi susirgimo prognozavimas yra įmanomas tik pasitelkus tikrus gydymo įstaigų duomenis apie pacientų susirgimų požymius, todėl buvo ieškota tikrų duomenų. Iš visų rastų buvo pasirinkti dermatologijos ligų duomenys, kuriuos galite rasti Kalifornijos Universiteto svetainėje adresu: http://www.ics.uci.edu/~mlearn/MLSummary.html. Pasirinkti duomenys yra suskirstyti į šešias klases, t.y. ligas, jie turi 366 pavyzdžius, iš kurių kiekviename yra po 34 atributus. Duomenyse kai kuriems pavyzdžiams trūksta kelių atributų reikšmių.

Siekdamas įrodyti neuroninių tinklų tinkamumą dermatologinių ligų diagnozės nustatymui aš atlikau eksperimentą su dviem skirtingom neuroninių tinklų architektūrom. Dėl plačių pritaikymo galimybių buvo pasirinktas daugiasluoksnis perceptronas bei SOM‘as.

Kadangi standartiniame *MatLab‘e* neužteko neuroniniams tinklams skirtų įrankių, buvo nuspręsta ieškoti papildomų į *MatLab‘ą* įdiegiamų programinių paketų, galinčių pilnai atskleisti daugiasluoksnio perceptrono bei SOM‘o galimybes.

3.1.1. Įrankiai daugiasluoksnio perceptrono modeliavimui

Ieškomam daugiasluoksnio perceptrono modeliavimo įrankiui buvo keliami šie reikalavimai:

- Vartotojas privalo turėti mokymosi greičio bei momento konstantų nustatymo galimybę.
- Turi būti leidžiama pasirinkti norimą neuronų aktyvacijos funkciją.
- Programinė įranga privalo turėti duomenų normalizavimo funkciją.
- Programa turi būti nemokama.
- Privaloma draugiška vartotojo sąsaja.

Buvo rasta 5 daugiasluoksnį perceptroną realizuojantys programiniai paketai. Vienas iš jų buvo mokamas (*WinNN*), kitam trūko draugiškos vartotojos sąsajos (*NEURON*), o *Ann* ir *MnSim* nusileido *NetLab‘ui* paruoštuos dokumentacijos kokybe. Taigi buvo pasirinktas Aston‘o Universitete kurtas *NetLab*. Jame yra realizuoti ne vien tiktai daugiasluoksniai perceptronai, bet ir RBF tinklai, Laplaso aproksimavimo karkasas, Markovo grandinės, K-vidurkių grupavimas ir t.t. *NetLab‘as* yra integruojamas į *MatLab‘ą* (nesenesnės kaip 5 versijos).

3.1.2. Įrankiai SOM modeliavimui

Ieškomai SOM programinei įrangai buvo keliami šie reikalavimai:

- Turi būti realizuotas tiek Paketinio žemėlapio (Batch Map), tiek inkrementinio. mokymosi algoritmas.
- Vartotojas privalo turėti mokymosi greičio bei kaimyninės srities nustatymo galimybę.
- Programinė įranga turi mokėti dirbti su pavyzdžiais, kuriuose trūkta tam tikro atributo reikšmės.
- Įranga turi turėti U-matricos, laimėtojo trajektorijos tiek ant pagrindinio žemėlapio, tiek ant komponentų planų atvaizdavimo galimybę.
- Programinė įranga privalo turėti duomenų normalizavimo funkciją.
- Programa turi turėti žemėlapio žymėjimo klasių žymėmis galimybę.
- Programa privalo būti nemokama.

Buvo pasirinkti 3 vieni populiariausių programiniai įrankių, kurie netrukus bus apžvelgti:

SOM_PAK

Tai yra pirmas ir viešai platinamas programinis paketas, skirtas SOM‘o realizacijai. Jis buvo sukurtas dar 1990 m. Helsinkio Universiteto kompiuterių ir informacijos mokslų laboratorijoje. Prieš tai 1989 m. ten pat buvo sukurta panašus paketas *LVQ_PAK* skirtas mokymosi vektorių kvantavimui. Šiuose paketuose yra detaliai išdėstyti SOM ir LVQ metodai, bei pateiktos jų realizacijos. Programas *SOM_PAK* ir *LVQ_PAK*, jų išeities kodą bei pilną dokumentaciją galite rasti interneto svetainėje: http://www.cis.hut.fi/research/software.shtml.

SOM_PAK‘e yra realizuotas tik standartinis inkrementinio mokymosi algoritmas. Vartotojo sąsaja yra operacinės sistemos komandinė eilutė. SOM‘o topologija gali būti pasirinktinai stačiakampis arba šešiakampis, o žemėlapio dydis ir vektorių matavimas yra neribojamas, vienintelis apribojimas yra pačio kompiuterio resursai. Modelių inicializacija gali būti atsitiktinė arba išilgai dviejų pagrindinių duomenų pasiskirstymo ašių. Inicializacija gali būti

automatiškai kartojama ir po nedidelio testavimo, išrinkus geriausią žemėlapį, apmokymas gali būti tęsiamas. Kaimyno funkcija gali būti „burbulo“ arba Gauso formos. *SOM_PAK*‘e realizuotas algoritmas moka dirbti su nepilnais duomenimis. Paketas gali atvaizduoti komponentų planus su trajektorijomis ir U-matricą.

SOM Toolbox

SOM Tollbox buvo sukurtas 1996 m. taip pat Helsinkio Universiteto kompiuterių ir informacijos mokslų laboratorijoje. Šis įrankis yra integruojamas į *MatLab‘ą* (ne žemesnę kaip 5 versiją). *SOM_Toolbox* galima nemokai parsisiųsti iš interneto svetainės http://www.cis.hut.fi/software.shtml.

Šiame programiniame pakete yra įgyvendinti inkrementinio SOM‘o ir paketinio žemėlapio (Batch Map) algoritmai. Žemėlapis gali būti stačiakampio arba šešiakampio formos, o jo dydis yra neribojamas. Galima atsitiktinė inicializacija arba išilgai dviejų pagrindinių duomenų pasiskirstymo ašių. Įėjimo vektoriai gali būti automatiškai normalizuojami. Galima pavaizduoti komponentų planus su trajektorijomis, U-matricą, taip pat mokymosi pavyzdžių įverčių neuronuose histogramas.

Nenet (Neural Networks Tool)

Šį programinį produktą 1997 m. sukūrė Nenet Team. Jis ketino tapti savo laikmečio vartotojui draugiškiausia dirbtinius neuroninius tinklus realizuojanti programa. Programą galima rasti interneto svetainėje http://www.mbnet.fi/~phodju/nenet/Nenet/General.html. Ji yra skirta tik *Microsoft Windows* operacinei sistemai.

SOM‘o inicializavimo fazėje yra atliekamas duomenų normalizavimas. Nenet‘e yra realizuotas tik inkrementinis SOM‘o algoritmas. Inicializacija gali būti automatinė arba nustatyta principinėmis ašimis. Yra realizuotos plačios vaizdavimo galimybės: galima sudaryti komponentų planus su trajektorijomis, U-matricas, įėjimo vektorių trimates histogramas bei aktyvių neuronų koordinates.

Išvados

SOM_PAK buvo kurtas didelių ir sudėtingų skaičiavimų reikalaujančių profesionalių užduočių sprendimui, tačiau jis yra labai nedraugiškas vartotojui. Nenet yra paprastai naudojama, turi geras grafinio vaizdavimo funkcijas, tačiau patartina ją naudoti tik mažoms problemos spręsti. *SOM Toolbox* yra kompromisas, turintis lankstumo ir nesunkiai naudojamas, tuo pačiu galintis spręsti pakankamai sudėtingas problemas.

3.1.3. Ligos diagnozavimas daugiasluoksniu perceptronu

Turimuose medicininiuose duomenyse yra 6 ligos, taigi ligos diagnozavimui buvo kurtas perceptronas su 6 išėjimo neuronais. Gautuose duomenyse ligos klasė buvo nurodoma skaičiumi nuo 1 iki 6, taigi reikėjo pertvarkyti duomenis – vietoj vieno ligos klasę nurodančio atributo reikėjo sukurti 6 (t.y. po vieną kiekvienai ligai). Sukurtų atributų reikšmė buvo nustatoma priklausomai nuo duoto duomenų pavyzdžio ligos klasės – jei ji buvo priskirta 1 klasei, tada pirmajame sukurtame atribute buvo įrašomas 1, o likusiuose 5 po 0; jeigu liga buvo priskirta 2 klasei, tada antrajam sukurtam atributui buvo priskiriamas 1, o likusiems 5 po 0; ir t.t.. Šitaip pertvarkęs duomenis galėjau *i-tąjį* išėjimo neuroną mokyti atpažinti *i-tajam* atributui priskirtą ligą.

Ligos diagnozavimo atlikimui pasirinkta naudoti dviejų sluoksnių perceptroną, kurio sukūrimui *NetLab‘as* turi funkciją *mlp(nin, nhidden, nout, func, prior),* kur *nin* yra įėjimo neuronų skaičius, *nhidden* paslėptame sluoksnyje esančių neuronų skaičius, *nout* yra išėjimo neuronų skaičius, *func* aktyvavimo funkcija, kuri reikšmės gali būti *linear*, *logistic* ir *softmax;* o *prior* yra mokymosi greičio konstanta.

Dvisluoksnio perceptrono apmokymui *NetLab‘as* turi funkciją *netopt(net, options, x, t, alg),* kur *net* yra perceptronui priskirto kintamojo vardas; vektoriaus *options* reikšmėmis galima valdyti apmokymo algoritmą. Svarbiausi šio vektoriaus atributai yra: 14, kuris nurodo apmokymo ciklų skaičių; 17 – jis nurodo mokymosi momento dydį; 18, kuris nurodo mokymosi greitį. Komandos *netopt* parametras *x* yra įėjimo vektorių matrica, o y – trokštamų atsakų vektorių matrica. Parametru *alg* yra nurodomas mokymosi algoritmas. Galimos jo reikšmės – graddesc (gradiento nuolydžio algoritmas), hmc (Monte Carlo algoritmas) ir t.t. Funkcijos *netopt* rezultatas yra perceptronas su optimizuotais duotiems apmokymo duomenims sinapsių svoriais.

Apmokius dvisluoksnį perceptroną galime atlikti ligos diagnozę. Tam reikalinga dar viena *NetLab‘o* funkcija *mlpfwd(net, x),* kurios reikšmė yra perceptrono atsakas. Šios funkcijos argumentas *net* yra perceptronui priskirto kintamojo vardas, *x* įėjimo duomenų vektorius. Taigi į apmokytą perceptroną paduodant įėjimo vektorių su ligos simptomais, gauname atsako vektorių. Kadangi kiekvienas išėjimo neuronas yra susietas su konkrečios ligos prognozavimu, tai ligos diagnozę nurodys didžiausias išėjimo atsakas.

3.1.4. Ligos diagnozavimas SOM‘u

SOM‘o apmokymas *SOM Toolbox‘e* gali būti vykdomas komanda. *sMap = som_make(sD,'training','long','algorithm','batch','init','randinit'),* kurioje *sD* yra įėjimo duomenų

matrica, *training long* nurodo, kad turi būti vykdomas ilgas mokymo procesas (dar galimas *short* – trumpas ir *medium* – vidutinis); *algorithm batch* nurodo mokymo algoritmą, šiuo atveju paketinį (dar galimas *seq* – nuoseklus); *init randinit* nurodo, kad SOM‘o inicializavimui turi būti naudojamas atsitiktinio inicializavimo metodas.

Atlikus mokymosi be mokytojo procesą turi būti atliekamas klasių sužymėjimas. Žymėjimas yra atliekamas komanda *sMap = som_autolabel(sMap,sD,'vote')*, kur *sMap* yra SOM žemėlapiui priskirtas kintamasis, *sD* yra apmokymo duomenų matrica, o *vote* yra žymėjimo taisyklė.

Norint nustatyti ligos diagnozę reikia į SOM‘o įėjimo sluoksnį paduoti įėjimo vektorių su reikiamais medicininiais atributais. Šio vektoriaus neurono-laimėtojo žymė ir tampa ligos diagnoze. Laimėjusį SOM žemėlapio neuroną galime nustatyti komanda *bmus = som_bmus(sMap,a)*, kur *sMap* yra SOM žemėlapiui priskirtas kintamasis, o *a* įėjimo duomenų vektorius.

3.2. E-patarėjo padedančio pasirinkti tinkamą profesiją modelis

Vienas iš darbo uždavinių yra sukurti e-patarėją galimybėms socialinės atskirties terpėje pasirinkti. Šis e-patarėjas turi palengvinti profesijos pasirinkimą. Duomenų, kurie parodytų kokių savybių reikia konkrečių profesijų darbuotojams, rasti nepavyko. Todėl buvo nutarta kurti e-patarėją, kuris sugebėtų pats aptikti dėsningumus gautuose iš aplinkos duomenyse. Šio uždavinio sprendimui ypač gerai tinka dirbtiniai neuroniniai tinklai.

Įgyvendinant e-patarėją yra realizuotas dvisluoksnis perceptronas. Įėjimo neuronų yra 29, t.y. tiek pat kiek ir klausimų, nustatančių žmogaus gebėjimus. Visi klausimai turi tris atsakymo variantus: „taip“, „galbūt, tačiau nepranokstu kitų“ ir „ne“. Šiem atsakymams yra suteikiamos reikšmės nuo 3 iki 1 atitinkamai. Gebėjimo nustatymui skirti klausimai yra paimti iš tinklapio http://www.profesijupasaulis.lt/main/V.HTM. Paslėptame neuroniniame sluoksnyje yra pasirinkta naudoti 50 neuronų. Perceptrono išėjimo sluoksnyje yra 45 neuronai, kurių kiekvienas atitinka vieną profesiją. Profesijos ir jų trumpi apibūdinimai yra paimti iš interneto svetainės http://www.profesijupasaulis.lt/abecedni/abecedni.htm.

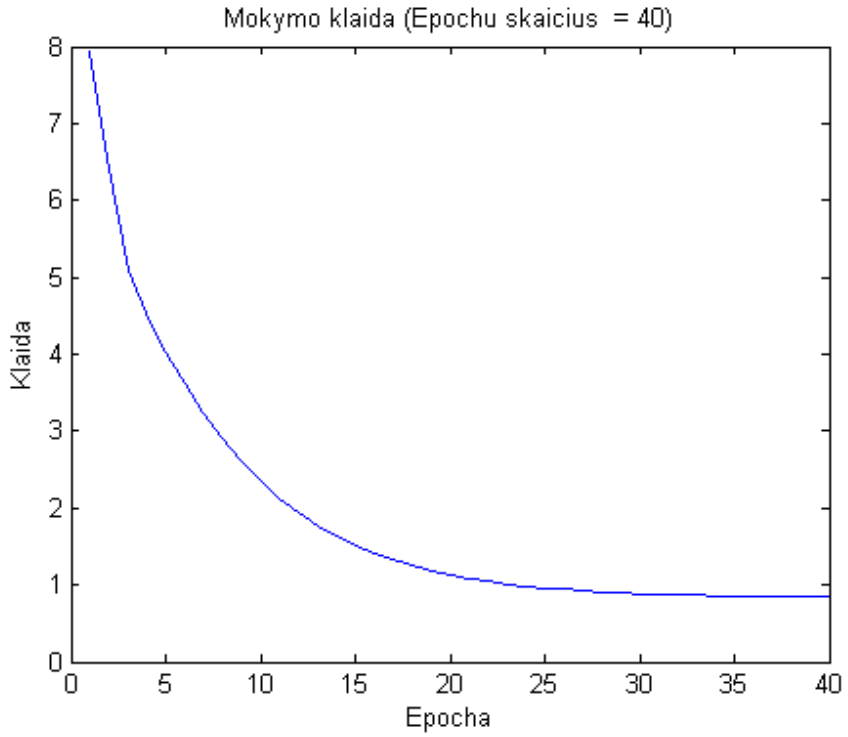
E-patarėjas turėtų būti realizuotas kaip interneto svetainė, todėl jo įgyvendinimui buvo pasirinktas PHP. Kadangi serveryje, kuriame e-patarėjas yra patalpintas, nėra jokios DBVS, todėl buvo nutarta perceptrono svorius bei gaunamus iš išorės duomenis kaupti tekstinėse bylose.

4. DARBO REZULTATAI

4.1. Ligos diagnozavimas daugiasluoksniu perceptronu

Siekiant atlikti tinkamą ligos diagnozavimą, pirmiausia reikėjo nustatyti optimalius dvisluoksnio perceptrono parametrus. Šitam uždaviniui pasiekti buvo atlikta eilė bandymų.

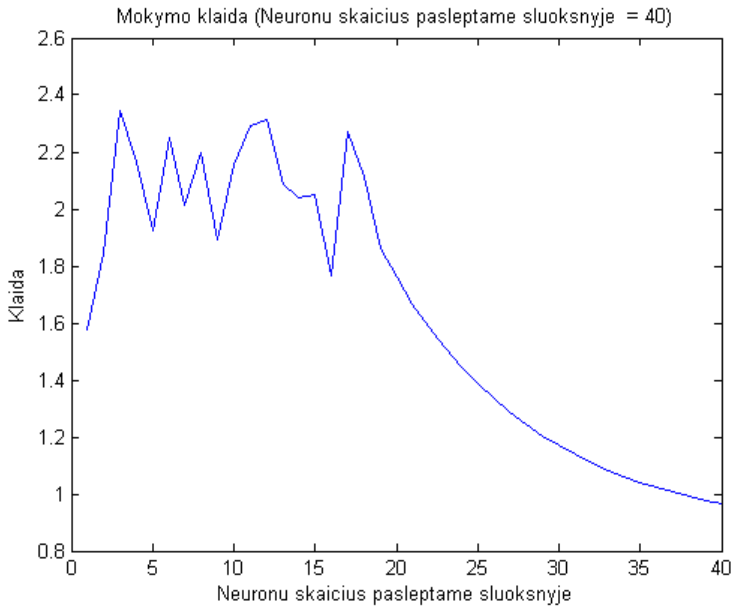
Pirmiausia buvo atliktas eksperimentas, nustatantis kaip priklauso vidutinė kvadratinė klaida nuo epochų skaičiaus. Tam tikslui buvo sukurtas algoritmas, kuris apskaičiuoja mokymosi klaidą esant skirtingam mokymosi epochų skaičiui bei nubraižo grafiką, pavaizduojantį mokymosi klaidos priklausomybę nuo mokymosi epochų skaičiaus. 7 pav. matome, kad mūsų nagrinėjamu atveju su medicininiais duomenimis mokymosi vidutinė kvadratinė klaida po 35-tos epochos išsilygina. Galime daryti išvadą, kad perceptrono apmokymui optimalus epochų skaičius yra 35.



7 pav. ***Vidutinės mokymosi klaidos priklausomybė nuo epochų skaičiaus***

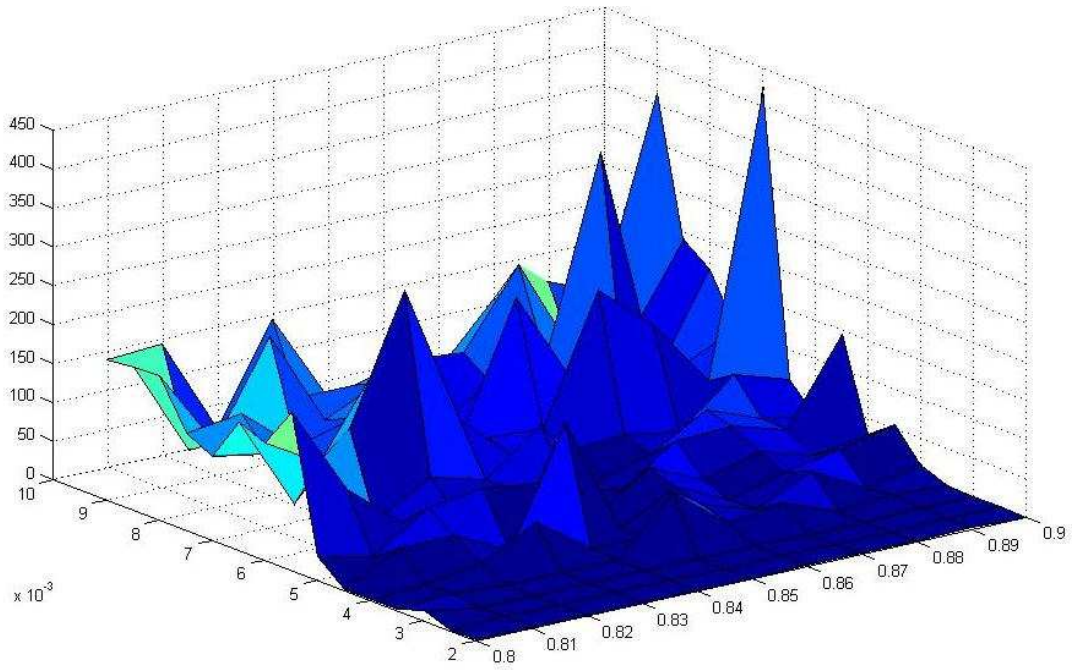
Taip pat buvo svarbu pažiūrėti koks turėtų būti optimalus neuronų kiekis paslėptajame sluoksnyje. Tam uždaviniui spręsti buvo sukurtas algoritmas ir atliktas bandymas, kurio metu perceptronas buvo apmokomas su skirtingu neuronų skaičiumi. Kaip matote 8 pav. mokymosi procesas tiksliausias, kai neuronų skaičius paslėptame sluoksnyje yra artimas įėjimų skaičiui

(šiuo atveju turime 34 įėjimo signalus).



8 pav. ***Vidutinės mokymosi klaidos priklausomybė nuo neuronų skaičiaus***

Ankstesnių bandymų metu buvo pastebėta, kad mokymasis yra stabilus, kai mokymosi greitis yra pakankamai mažas, tačiau jį padidinus, mokymasis tampa nestabiliu. Taigi siekiant įsitikinti kaip mokymosi proceso vidutinė kvadratinė klaida priklauso nuo mokymosi greičio α ir momento dydžio μ buvo sukurtas algoritmas bei jo pagalba atliktas eksperimentas. Pav.9 galite išvysti šio tyrimo rezultatą. Apatinė kairioji ašis nurodo mokymosi greitį, o dešinioji momentą. Kaip galite pastebėti mokymosi kvadratinė klaida yra maža, kol mokymosi greitis yra labai mažas – $\alpha \in [0.002,0.0035]$. Jos reikšmei padidėjus vidutinė mokymosi klaida pradeda labai svyruoti ir iš to sprendžiame, kad mokymosi procesas tampa nestabiliu. Taip pat galima nuspręsti, kad mokymosi klaida prie tinkamo mokymosi greičio yra stabili bet kokiam mokymosi momento dydžiui $\mu \in [0.8,0.9]$.



9 pav. ***Vidutinės kvadratinės klaidos priklausomybė nuo mokymosi greičio ir momento dydžio***

Atlikus eilę bandymų buvo nustatyti dermatologijos ligas diagnozuojančio dvisluoksnio perceptrono optimalūs parametrai: paslėptame sluoksnyje turi būti 40 neuronų, geriausias mokymosi greitis $\alpha = 0.0025$, mokymosi epochų skaičius 35. o mokymosi momentas $0.8 \le \mu \le 0.9$.

Galiausiai turint reikalingus duomenis buvo sukurta programa *dermet_diagnostika* (1 priedas) veikianti *MatLab* pakete, bei turinti komandinės eilutės sąsają su vartotoju. Kaip matote 10 pav. vartotojas privalo suvesti atributų reikšmes, o 11 pav. vartotojui pateikiama diagnozė su 99,99% tikimybe. Bandymo metu buvo pastebėta, kad ši programėlė tikrai gana tiksliai sugeba nustatyti diagnozę įėjimo duomenims, kurie nebuvo panaudoti apmokymui.

```
Command Window
Enter values from 0 to 3 for some clinical atributes:
erythema: 2
scaling: 0
definite_borders: 1
itching: 3
koebner_phenomenon: 1
polygonal_papules:
```

10 pav. ***ligos diagnozės programos vartotojo sąsaja***

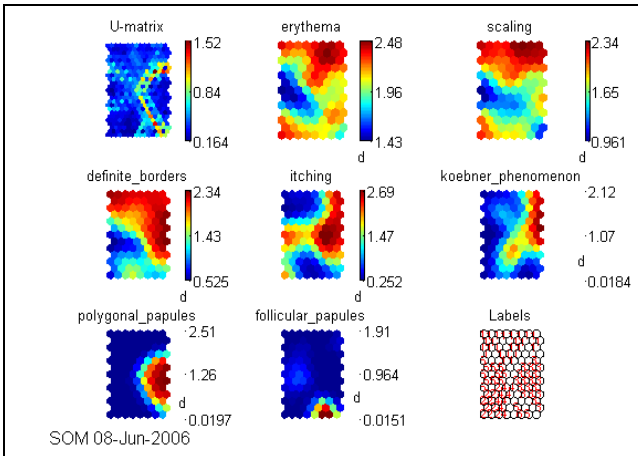
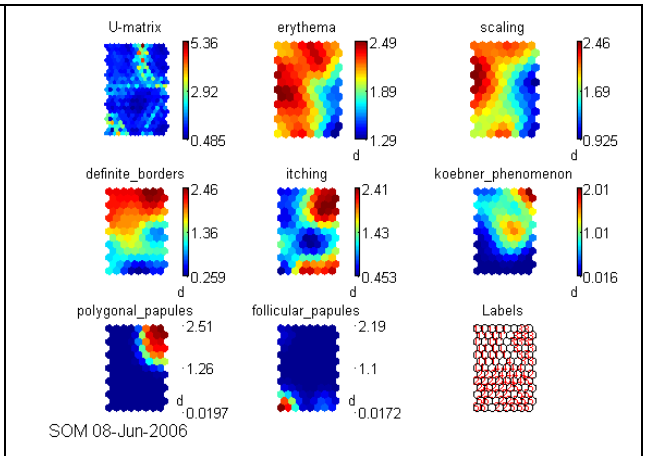
```
Your diagnozis with 0.99998 probality is - psoriasis
>>
```

11 pav. ***Ligos diagnozė***

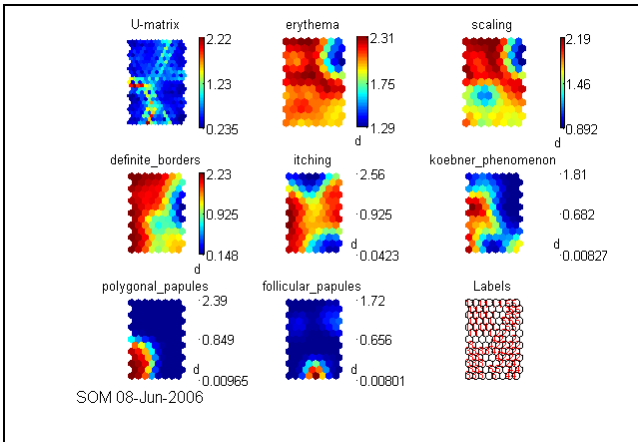
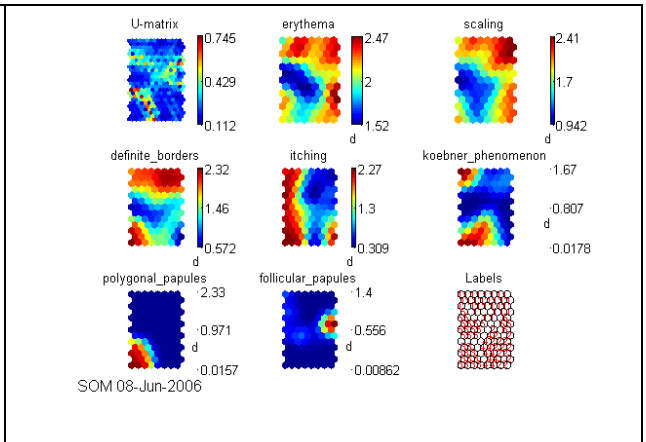
4.2. Ligos diagnozavimas SOM‘u

Siekiant atlikti tinkamą ligos diagnozavimą SOM žemėlapiu, taip pat kaip ir perceptrono atveju, pirmiausia reikėjo nustatyti optimalius SOM apsimokymo parametrus. Šitam uždaviniui pasiekti buvo atlikta eilė bandymų.

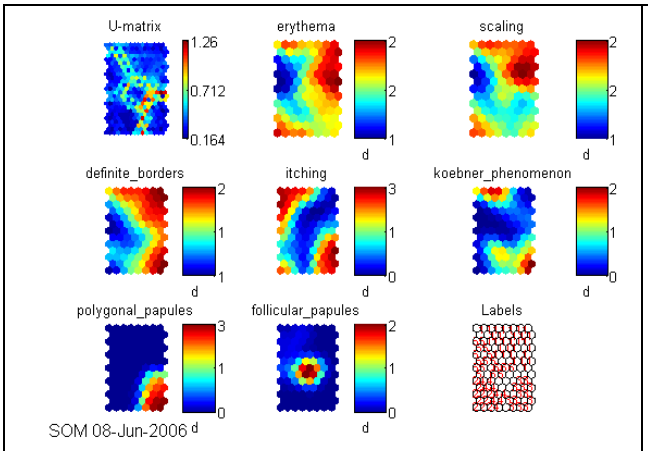
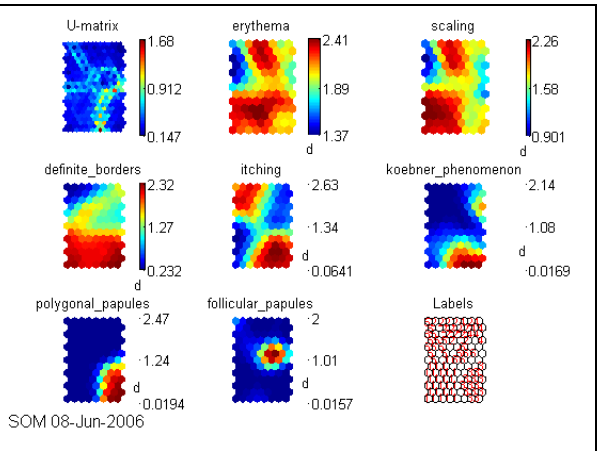
Pirmiausia buvo atliekami bandymai su įvairiais apmokymo normalizacijos metodais, t.y. buvo žiūrima kaip nuo normalizacijos metodo priklauso kvantavimo klaida bei topografinė klaida. Kaip matote 12, 13, 14 pav. geriausius rezultatus parodė duomenų normalizacija logistiniu metodu:

	
a. Normalizacijos metodas: Range; Kvantavimo klaida: 0,886 Topografinė klaida: 0,022	b. Normalizacijos metodas: Var; Kvantavimo klaida: 3,230 Topografinė klaida: 0,033

12 pav. ***SOM žemėlapiai***

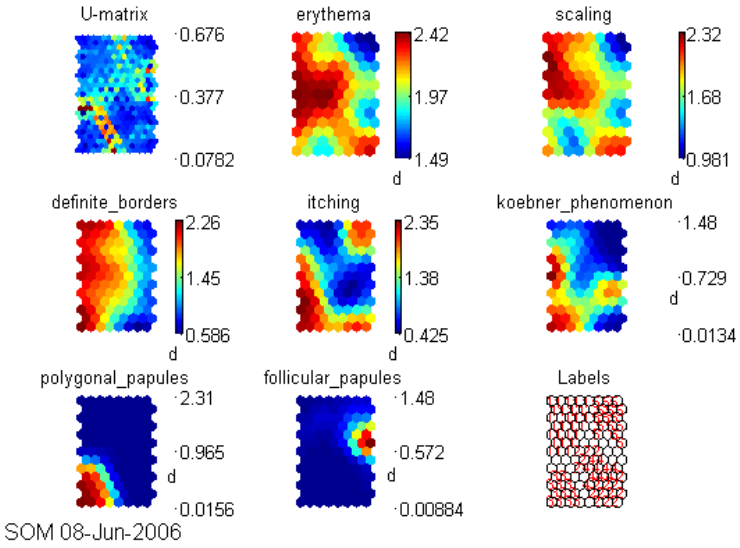
	
a. Normalizacijos metodas: Log; Kvantavimo klaida: 1,248 Topografinė klaida: 0,003	b. Normalizacijos metodas: Logistic; Kvantavimo klaida: 0,636 Topografinė klaida: 0,016

13 pav. ***SOM žemėlapiai***

	
a. Normalizacijos metodas: histD; Kvantavimo klaida: 0,898 Topografinė klaida: 0,016	a. Normalizacijos metodas: histC; Kvantavimo klaida: 0,906 Topografinė klaida: 0,019

14 pav. ***SOM žemėlapiai***

Aukščiau esantys bandymai buvo atliekami su paketiniu mokymosi algoritmu. Pamėginus atlikti bandymą su logistiniu normalizavimo metodu bei inkrementiniu mokymo metodu, kvantavimo klaida gauta lygi 0,651, o topografinė klaida 0,025 (15 pav.).

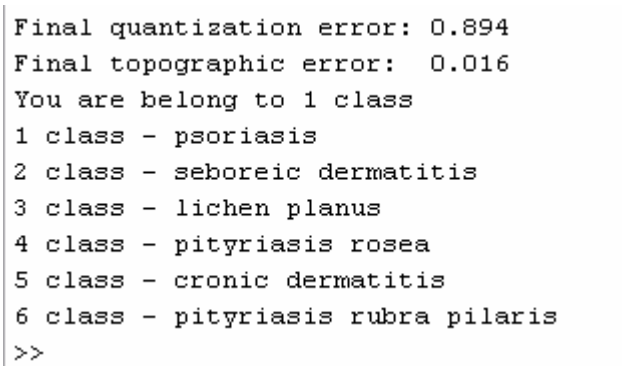


15 pav. ***SOM žemėlapiai***

Taigi palyginus gautus rezultatus buvo nuspręsta, kad labiausiai šiam ligos diagnozavimo uždaviniui tinka logistinis duomenų normalizavimo algoritmas bei paketinis mokymosi modelis. Ir taip pat pastebėta, kad ne visiems žemėlapio neuronams yra priskiriamos klasės žymės.

Pasinaudojus bandymų metu gauta informacija bei *SOM Toolbox* programiniu paketu,

buvo sukurta ir SOM‘o pagrindu veikianti ligos diagnozės nustatymo programa (16 pav.). Šioje programoje analogiškai, kaip ir perceptrono atveju, reikia įvesti prašomus medicininius duomenis. Tada gautam įėjimo vektoriui yra išrenkamas SOM žemėlapio neuronas-laimėtojas, kurio žymė ir nurodo ligos diagnozę.

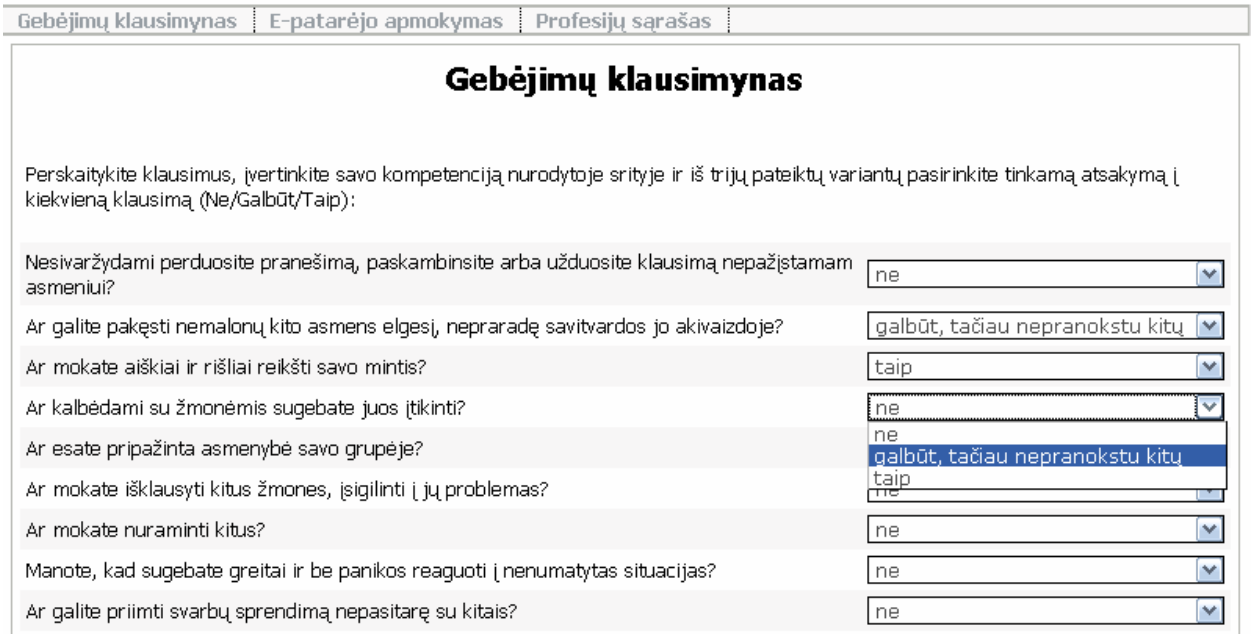


16 pav. ***Ligos diagnozė***

4.3. E-patarėjas profesijos pasirinkimui

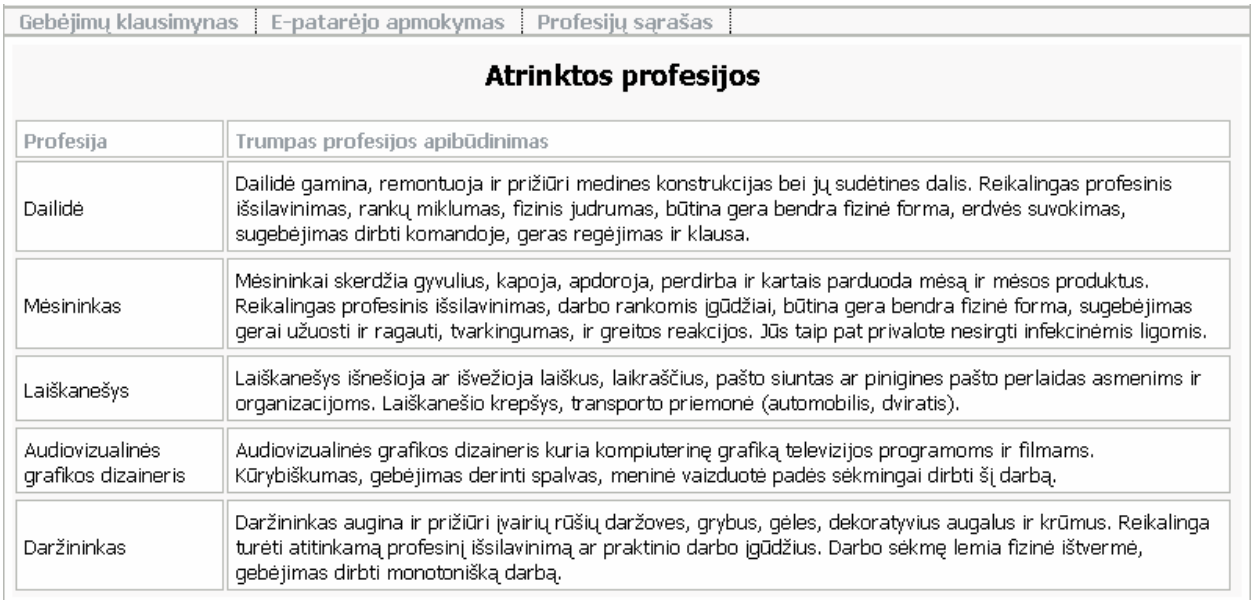
Pasirinkus dvisluoksnio perceptrono architektūrą buvo sukurtas e-patarėjas padedantis pasirinkti tinkamą profesiją. Jį galite rasti adresu http://www.equal-greitkelis.lt/100/3

Įėjus į sistemą Jus pasitinka e-patarėjo kvietimas užpildyti gebėjimų klausimyną (17 pav.). Klausimyną sudaro 29 klausimai, kuriais galima nustatyti žmogaus tinkamumą tam tikroms profesijoms. Kiekvienas klausimas turi tris atsakymo variantus – „ne“, „galbūt, tačiau nepranokstu kitų“ bei „taip“.



17 pav. ***Gebėjimų klausimynas***

Užpildžius gebėjimų klausimyną bei paspaudus mygtuką „Įvertinimas“, e-patarėjas surenka žmogaus gebėjimus ir paduoda juos perceptronui kaip įeities duomenis. Perceptrono kiekvieno išėjimo neurono atsako reikšmė parodo žmogaus tinkamumą neurono atstovaujamai profesijai. Kuo didesnis neurono atsakas, tuo žmogus su savo gebėjimais yra tinkamesnis atitinkamai profesijai. E-patarėjas atrenka 5 tinkamiausias profesijas ir pateikia jas bei jų trumpą apibūdinimą klausimyną pildžiusiam asmeniui (18 pav.).



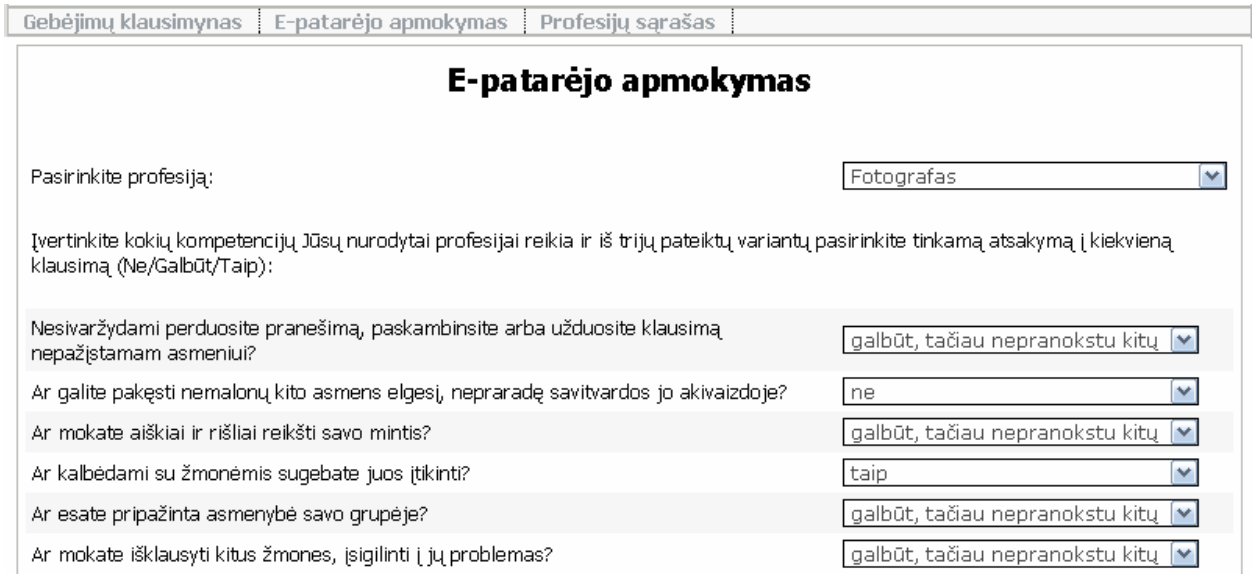
Profesija	Trumpas profesijos apibūdinimas
Dailidė	Dailidė gamina, remontuoja ir prižiūri medines konstrukcijas bei jų sudėtines dalis. Reikalingas profesinis išsilavinimas, rankų miklumas, fizinis judrumas, būtina gera bendra fizinė forma, erdvės suvokimas, sugebėjimas dirbti komandoje, geras regėjimas ir klausa.
Mėsininkas	Mėsininkai skerdžia gyvulius, kapoja, apdoroja, perdirba ir kartais parduoda mėsą ir mėsos produktus. Reikalingas profesinis išsilavinimas, darbo rankomis įgūdžiai, būtina gera bendra fizinė forma, sugebėjimas gerai užuosti ir ragauti, tvarkingumas, ir greitos reakcijos. Jūs taip pat privalote nesirgti infekcinėmis ligomis.
Laiškanešys	Laiškanešys išnešioja ar išvežioja laiškus, laikraščius, pašto siuntas ar pinigines pašto perlaidas asmenims ir organizacijoms. Laiškanešio krepšys, transporto priemonė (automobilis, dviratis).
Audiovizualinės grafikos dizaineris	Audiovizualinės grafikos dizaineris kuria kompiuterinę grafiką televizijos programoms ir filmams. Kūrybiškumas, gebėjimas derinti spalvas, meninė vaizduotė padės sėkmingai dirbti šį darbą.
Daržininkas	Daržininkas augina ir prižiūri įvairių rūšių daržoves, grybus, gėles, dekoratyvius augalus ir krūmus. Reikalinga turėti atitinkamą profesinį išsilavinimą ar praktinio darbo įgūdžius. Darbo sėkmę lemia fizinė ištvermė, gebėjimas dirbti monotonišką darbą.

18 pav. ***Atrinktos profesijos***

Kadangi duomenų, kurie parodytų kokių gebėjimų reikia dirbant tam tikroms profesijoms, nebuvo rasta, todėl buvo realizuota e-patarėjo mokymosi iš gautų duomenų galimybė. 19 pav. galite išvysti kaip yra apmokomas e-patarėjas. Pirmiausia yra pasirenkama viena profesija iš siūlomų sąrašo. Vėliau reikia sužymėti pasirinktai profesijai tinkamus gebėjimus. Tada e-patarėjas kaip įeities vektorių siunčia perceptronui gebėjimus, o kaip trokštamą atsaką pasirinktą profesiją. Perceptronas gavęs įeities duomenis paskaičiuoja atsakus. Tada yra skaičiuojamas skirtumas tarp apskaičiuoto ir trokštamo atsako. Tada remiantis gauta paklaida yra pakoreguojami išėjimo sluoksnio neuronų sinapsių svoriai. Vėliau atbulinio sklidimo metodu yra apskaičiuojami bei pakoreguojami paslėptojo sluoksnio neuronų sinapsių svoriai.

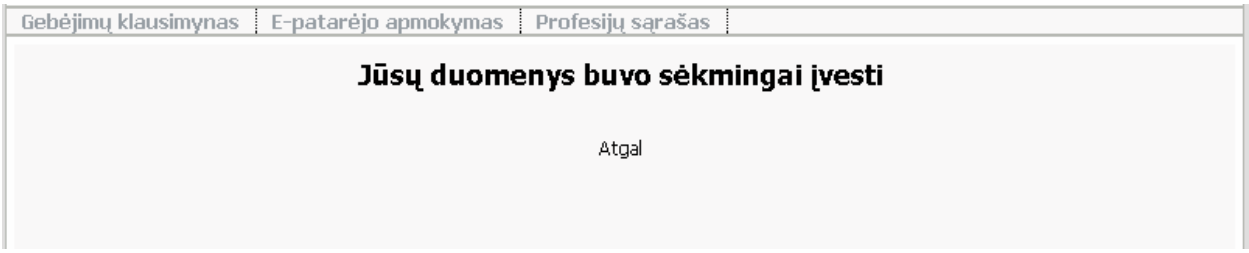
Pakoreguoti svoriai yra išsaugomi į tekstines bylas, ir sekantį kartą bandant nustatyti tam tikriems gebėjimams tinkančią profesiją, e-patarėjas jau naudojasi naujais duomenimis. Taigi, galime teigti, kad šitokiu būdu e-patarėjas mokosi.

Visi įvedami gebėjimai bei jiems tinkančios profesijos taip pat yra kaupiami bylose *inputs* ir *outputs.* Sukauptus duomenis būtų galima panaudoti tolimesniems tyrimams.



19 pav. ***E-patarėjo apmokymas***

Sėkmingai perceptrono mokymui panaudojęs apmokymo duomenis, e-patarėjas praneša vartotojui, kad jo duomenys buvo sėkmingai įvesti (20 pav.).



20 pav. ***Duomenų išsaugojimas***

Pasirinkę meniu punktą *Profesijų sąrašas* Jūs galite išvysti visas šiuo metu e-patarėjui žinomas profesijas, bei jų trumpą apibūdinimą (21 pav.). Šiuo metu e-patarėjas žino tik 45 profesijas. Tačiau yra įmanoma realizuoti ir e-patarėjo apmokymą naujomis profesijomis. Norint sukurti naują profesiją reikėtų sukurti dar vieną neuroną perceptrono išėjimo sluoksnyje ir prireikus dar vieną neuroną paslėptame sluoksnyje. Tada visų perceptrono neuronų jungčių svoriams suteikti atsitiktines mažas reikšmes ir, panaudojant visus prieš tai sukauptus apmokymo duomenis, apmokyti perceptroną nuo pradžių.

Gebėjimų klausimynas | E-patarėjo apmokymas | Profesijų sąrašas

Profesijų sąrašas

Profesija	Trumpas profesijos apibūdinimas
Aktorius	Aktoriaus darbas - vaidinti įvairius vaidmenis teatre, televizijoje, radijuje ar kine. Tai kartu ir sunkus darbas, ir menas. Aktoriaus karjeros pažanga priklauso nuo jo talento ir sugebėjimų. Siekiantiems šio darbo reikalingas aktorinio meno išsilavinimas. Gebėjimas aiškiai ir raiškiai kalbėti, gera atmintis, gebėjimas susikaupti, improvizuoti, kantrybė tvarkinga išvaizda, gebėjimas dirbti komandoje padėtų sėkmingai dirbti darbą.
Antenų montuotojas	Antenų montuotojas montuoja, remontuoja ir keičia televizijos ar radijo antenas, palydovines lėkštes ir kitas komunikacijos sistemas. Siekiantys dirbti šį darbą turi turėti specialų profesinį išsilavinimą. Techninė nuovoka, gebėjimas priimti savarankiškus sprendimus, greita reakcija, gera rankų judesių koordinacija padės sėkmingai dirbti šį darbą. Antenų montuotoju negali dirbti bijantys aukščio asmenys.
Apdailininkas	Apdailininkas paruošia medinius, metalinius, tinkuotus paviršius dažyti ir juos dažo įvairias dažų mišiniais, klijuoja įvairius apmušalus. Reikalingas profesinis pasirengimas. Sėkmingą darbą lemia praktinis mąstymas, gera dėmesio koncentracija, fizinė ištvermė, gera akių, rankų ir pirštų judesių koordinacija, gebėjimas skirti spalvas.
Apdailininkas dažytojas	Apdailininkas dažytojas dažo, lakuoja, glaisto ir kitais būdais apdailina įvairius daiktų paviršius. Reikalinga atitinkama profesinė kvalifikacija. Sėkmingą darbą lemia fizinė ištvermė, kantrybė, gera akių, rankų ir pirštų judesių koordinacija, fizinė ištvermė, gebėjimas skirti spalvas. Šį darbą sunkiau dirbti žmonėms, turintiems rimtų regos, judėjimo sutrikimų, alergiškiems bei sergantiems kvėpavimo takų ligomis.

21 pav. ***Profesijų sąrašas***

IŠVADOS

Atlikus ligos diagnozės nustatymo keliais dirbtinių neuroninių tinklų metodais tyrimą bei realizavus e-patarėją galimybėms socialinės atskirties terpėje pasirinkti, padarytos šios išvados:

1. Atlikta teminės literatūros analizė parodė, kad e-patarėjas galimybėms socialinės atskirties terpėje pasirinkti šiuo metu yra aktualus, o jo įgyvendinimui tikslingiausia naudoti dirbtinius neuroninius tinklus, kadangi jie yra labai galinga priemonė nustatant didelio sudėtingumo sąryšius.
2. Atlikus ligos diagnozavimo bandymus su keliais neuroninių tinklų metodais, buvo nustatyta, kad neuroniniai tinklai yra gera priemonė ligos diagnozės nustatymui. Taip pat pastebėta, kad tikslingiau naudoti daugiasluoksnį perceptroną, negu SOM, nes pastarasis žemėlapio žymėjimo metu ne visiems neuronams priskiria klasės žymę.
3. Sukūrus e-patarėją padedantį pasirinkti tinkamą profesiją, buvo nustatyta, kad jo įgyvendinimui labai gerai tinka daugiasluoksnis perceptronas. Taip pat nutarta, kad tinkamai apmokytas šis e-patarėjas gali būti realiai pritaikomas.

LITERATŪRA

1. MacKay D.J.C. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003, 628 p.
2. Rabunal J. R., Dorado J. *Artificial Neural Networks in Real-life Applications*. Idea Group Publishing, 2006, 375 p. ISBN 1-59140-904-7
3. Kasabov N. K. *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering.* Cambridge: The MIT Press, 1996, 550 p. ISBN 0262112124
4. Zhang G. P. *Neural Networks in Business Forecasting*. Idea Group Publishing, 2004, 350 p. ISBN 1591402158
5. Hastie T., Tibshirani R., Friedman J. *The Elements of Statistical Learning.* Springer-Verlag, 2001
6. *Artificial intelligence applications and innovations.* Boston: Springer Science, 2004, 484 p. ISBN 1-4020-8151-0
7. Wang L., Fu X. *Data Mining with Computational Intelligence.* Springer, 2005, 276 p.
8. Haykin S. *Kalman Filtering and Neural Networks.* John Wiley & Sons, Inc., 2001, 284 p. ISBN 0-471-22154-6
9. Kohonen T. *Self-Organizing Maps.* Springer, 2001, 501 p. ISBN 3-540-67921-9
10. Berry M. J.A., Linoff G. S. *Data Mining Techniques For Marketing, Sales, and Customer Relationship Management.* Indianapolis: Wiley Publishing, Inc. 2004, 643 p.ISBN 0-471-47064-3
11. Ablameyko S., Goras L., Gori M., Piuri V. *Neural Networks for Instrumentation, Measurement and Related Industrial Applications*. IOS Press, 2003, 329 p. ISSN 1387-6694
12. Korb K. *Introduction: Machine Learning as Philosophy of Science*.// Minds & Machines, 2004, Nr. 14, p. 433-440
13. Mjolsness E., DeCoste D. *Machine learning for science: state of the art and future prospects.//* Science, 2001, Nr. 293
14. Navakauskas D., Paulikas Š., Urbanavičius V., Martavičius R. *Šiuolaikinės SSA priemonės* [interaktyvus] . Vilnius: Vilniaus Gedimino technikos universitetas [žiūrėta 2006 06 18]. http://www.el.vtu.lt/ssa/index.html
15. Furrer D., Thaler S. *Neural-network modeling*.// Advanced Materials & Processes, 2005, Nr. 163, p. 42-46
16. Puočiauskas M. *Atsitiktinių signalų identifikavimas naudojant dirbtinius neuroninius tinklus.* VU, 2001
17. Hu Y.H. *Handbook of neural network signal processing*. CRC Press, 2002, ISBN: 0-8493-2359-2
18. EurLex. *Komisijos komunikatas tarybai ir Europos parlamentui. 2005 m. ES tvaraus vystymosi strategijos peržiūra: pirmoji apžvalga ir ateities gairės* [interaktyvus]. Briuselis. [žiūrėta 2006 05 28] *http://eur-lex.europa.eu/smartapi/cgi/sga_doc?smartapi!celexplus–!prod!DocNumber&type_doc=COMfinal&an_doc=2004&nu_doc=38&lg=lt.*
19. Gricius, G., D.Drungilas, A.A.Bielskis. *(2006) Nutolusio objekto valdymas e.laboratorijoje per kompiuterių tinklą, Technologijos mokslo darbai Vakarų Lietuvoje, V tomas, ISSN 1822-4652, Klaipėdos universiteto leidykla, 2006, p. 172-176*
20. Baziukaitė D., Bielskis A. A., Ramašauskas O. *Applying Adaptive Learning principles for the E-studies.//* Lithuanian Mathematical Works, 2002 T.42, spec. No**,** Vilnius Institute of Mathematics and Informatics, p. 214-218.
21. Bielskis A. A. *Using Knowledge Classifiers to Speed up E-Learning,*// Vadyba: Proceedings of West Lithuania Business College, 2004, p. 12-15.
22. Lukauskas V. *Implementing of an Adaptive Intelligent Component into E-Learning Environment by Using SOM Neural Networks*: Magistro tezės, KU, 2005
23. Vaitkus, M. *Making an E-Tutor More Intelligent.* Magistro tezės, KU,. 2004

24. Bielskis, A.A., P.Bielskis, A.Rima, N.Romas, R.Strigūnas. *Analysis of methods for e-business data mining using SOM networks, //* Technologijos mokslo darbai Vakarų Lietuvoje, Klaipėdos universiteto leidykla, 2006, V tomas, p. 152-159, ISSN 1822-4652.
25. *Gidas į profesijų pasaulį.* [žiūrėta 2006 05 21] *http://www.profesijupasaulis.lt/*

1 PRIEDAS

```
function dermet_diagnostika;
clear all;
clc;
duomenu_byla = 'dermetologijos_duomenys.dat';
duomenys = load(duomenu_byla);   % load training file
clear duomenu_byla;
[K,MN]=size(duomenys);
nin = 33; %iejimo signalu skaicius
nout = 6;  %isejimo signalu skaicius

atsakas  = duomenys(:,nin+2:nin+nout+1);
duomenys = duomenys(:,1:nin);


nhidden=40;
alpha = 0.001;      % Weight decay
mom = 0.8;
ncycles = 40;       % Number of training cycles.
% Set up MLP network
net = mlp(nin, nhidden, nout, 'softmax',alpha);
options = zeros(1,18);
options(1) = 1;               % Print out error values
options(14) = ncycles;
options(18) = alpha;
options(17) = mom;


disp('Enter values from 0 to 3 for some clinical atributes:');
labels=[];
labels{1}='erythema';
labels{2}='scaling';
labels{3}='definite_borders' ;
labels{4}='itching' ;
labels{5}='koebner_phenomenon' ;
labels{6}='polygonal_papules' ;
labels{7}='follicular_papules';
labels{8}='oral_mucosal_involvement' ;
labels{9}='knee_and_elbow_involvement' ;
labels{10}='scalp_involvement' ;
labels{11}='family_history' ;
labels{12}='melanin_incontinence' ;
labels{13}='eosinophils_in_the_infiltrate' ;
labels{14}='PNL_infiltrate' ;
labels{15}='fibrosis_of_the_papillary_dermis' ;
labels{16}='exocytosis' ;
labels{17}='acanthosis' ;
labels{18}='hyperkeratosis' ;
labels{19}='parakeratosis' ;
labels{20}='clubbing_of_the_rete_ridges' ;
labels{21}='elongation_of_the_rete_ridges' ;
labels{22}='thinning_of_the_suprapapillary_epidermis' ;
labels{23}='spongiform_pustule' ;
labels{24}='munro_microabcess' ;
labels{25}='focal_hypergranulosis' ;
labels{26}='disappearance_of_the_granular_layer' ;
labels{27}='vacuolisation_and_damage_of_basal_layer' ;
labels{28}='spongiosis' ;
labels{29}='saw_tooth_appearance_of_retes' ;
labels{30}='follicular_horn_plug' ;
labels{31}='perifollicular_parakeratosis' ;
```

```
labels{32}='inflammatory_monoluclear_inflitrate' ;
labels{33}='band_like_infiltrate' ;

testduomenys=[];
for i=1:10,
inputas = -1;
while (isempty(inputas)|inputas<0 |inputas>3 )
  inputas=input([labels{i} ': ']);
end
testduomenys=[testduomenys inputas];
inputas=-1;
end

disp('Enter values from 0 to 1 for some clinical atributes:');
i=11;
inputas = -1;
while (isempty(inputas)|inputas<0 |inputas>1 )
  inputas=input([labels{i} ': ']);
end
testduomenys=[testduomenys inputas];
inputas=-1;

disp('Enter values from 0 to 3 for some Histopathological Attributes:');
for i=12:33,
inputas = -1;
while (isempty(inputas)|inputas<0 |inputas>3 )
  inputas=input([labels{i} ': ']);
end
testduomenys=[testduomenys inputas];
inputas=-1;
end


[net,options,errlog] = netopt(net, options, duomenys, atsakas, 'graddesc');
y = mlpfwd(net, testduomenys);
max=0;
for i=1:6,
   if y(i)>max ,
      disp(num2str(y(i)));
      max=y(i);nmax=i;
   end
end
ligos=[];
ligos{1}='psoriasis';
ligos{2}='seboreic dermatitis';
ligos{3}='lichen planus';
ligos{4}='pityriasis rosea';
ligos{5}='cronic dermatitis';
ligos{6}='pityriasis rubra pilaris';

disp(['Your diagnozis with ',num2str(y(nmax)) ,' probality is - ', ligos{nmax}]);
```