**Henrikas PRANEVIČIUS, Šarūnas RAUDYS,
Algimantas RUDŽIONIS, Vytautas RUDŽIONIS,
Kastytis RATKEVIČIUS, Jūratė SAKALAUSKAITĖ,
Dalius MAKACKAS**

Agentinių sistemų modeliai

2008

UDK

Vadovėlis išleistas vykdant projektą „Informatikos ir matematikos doktorantūros studijų plėtra (InMaDra)“, finansuojamą Lietuvos Respublikos Vyriausybės ir ES Europos socialinio fondo. Paramos sutarties numeris: ESF/2004/2.5.0-03-383/BPD-157/ParS-12500-595.

Vadovėlį spausdinti rekomendavo KTU Informatikos (09P) doktorantūros kvalifikacijos komisija (2008 05 14, protokolo Nr. 33).

Leidinį maketavo
Redaktorė

ISBN

Turinys

1 skyrius

Daugiaagenčių sistemų mokymas

1.1. Įvadas

1.1.1. Pastoviai vykstantys aplinkos pokyčiai. Adaptavimasis daugiaagentėse sistemose

Daugelį tarpusavyje susijusių uždavinių sprendžiančios daugiaagentės sitemos gali būti sudarytos vadovaujantis *a priori* turimomis žiniomis.

Tačiau jos gali būti ir mokomos, tiek deklaratyviniu būdu, tiek ir realiais ar specialiai sukonstruotais pavyzdžiais. Viena iš pagrindinių šiuolaikinių daugiaagenčių sistemų savybių – tai sugebėjimas greitai reaguoti į aplinkos pokyčius ir prie jų prisitaikyti. Veikiamas netiesinių, chaotiškų, Gamtoje, bei žmogaus ūkinės veiklos iššauktų procesų, pasaulis nuolat, kasdien, kas metus keičiasi. Tas tapo pastebima naudojant naujas technologijas, juolab, kad jos minėtą kitimą žymia dalimi ir įtakoja. Pasaulis sparčiai kinta. įtakojamas labai spartaus naujų informacinių technologijų vystymosi ir aktyvaus žmogaus poveikio Gamtai pastarųjų dvejų dešimtmečių laikotarpiu aplinkos kitimas ypač paspartėjo. Stebimas spartus Žemės klimato kitimas. Prognozuojama, kad artimiausią penkiasdešimtį metų jis dar paspartės. Iš tradicinės finansų sistemos liko „šipuliai“, dabar ją nusako, ne tiek ekonomikos raida, kiek politika, bei ypač spartus informacinių ryšių, bei naujų technologijų vystymąsis. Visi „finansistai“ renka kiek įmanoma įvairesnę informaciją, visi naudoja galingus kompiuterius, panašius matematinius metodus ir juos realizuojančias programas, visi jie sprendžia panašius uždavinius, visi konkuruoja tarpusavyje. Tad naujų galingesnių kompiuterių ir tobulesnių matematinių metodų pasirodymas įtakoja rinką. Tokiu atveju, senesni, finansų rinką aprašantys, dėsniai, prognozavimui į priekį nebetinka. Tie patys procesai stebimi ir aptarnavimo, gamybos, energetinių išteklių gavimo sferose.

Paprastas, tačiau puikus, pavyzdys: streikas, avarija ar blogas oras aerouoste. Sąlygoms pasikeitus, klijentus, lėktuvus, aerouosto personalą aptarnaujanti DaS turi persijungti į naują darbo režimą, ir toliau funkcionuoti kiek įmanoma efekty-

viau. Tam ji turi pasirūpinti, kad gautų naują, tolimesnei jos veiklai reikalingą, informaciją, daug ką išmokti, išprognozuoti daugelį aerouosto darbą aprašančių parametrų. Žodžiu, artimiausiomis minutėmis, valandomis ar net dienomis DaS egzistuos pastovaus persimokymo sąlygomis, o vėliau turės grįžti į normalų darbo režimą, kuris irgi neliks nė kiek nepasikeitęs.

Gyvename informacijos antplūdžio šimtmetyje. Kasdien informacijos gaunama žymiai daugiau, nei galime apdoroti. Normalus verslininkas, o ir administratorius, praktiškai nesugeba sekti pasikeitimų įstatymuose, poįstatyminiuose aktuose, ekonominių politinių ir kitokio pobūdžio svyravimų. Tad šiame, informacijos antplūdžio, laikotarpyje žmonijai iškyla visa eilė naujų uždavinių, su kuriais anksčiau ji nėra susidūrusi. Tai informacijos išgavimo, apdorojimo, jos analizės, patikimumo, saugumo, slaptumo, nuosavybės, saugojimo techniniai ir juridiniai klausimai. Akivaizdu, kad jie dar neišspęsti ir kad kol kas mes esame dar labai toli nuo to.

Tad sugebėjimas adaptuotis – būtinas šiuolaikinių, pažangių DaS bruožas. Ekonomiškai pasiteisina tik pakankamai universalios, sugebančios prisitaikyti prie netikėtų aplinkos pasikeitimų, bei naujai iškylančių uždavinių, sistemos. Iš anksto užprogramuoti visus įmanomus pokyčius, ir intelektualius, aplinką charakterizuojančius ir informaciją apdorojančius, metodus, reikalingus sprendimams priimti, tampa nebeįmanoma. Tenka naudoti priemones, leidžiančias, pagal stebimus reiškinius keisti sprendimų priėmimo mechanizmus, juos tarpusavyje suderinti.

Sprendžianat pagal pavadinimą „daugiaagentė sistema“ – tai informaciją apdorojanti sistema, susidedanti iš daugelio elementų-agentų, kiekvienas iš kurių sprendžia savo individualų uždavinį. Kiekvienas iš agentų mokosi atskirai, tačiau siekdamai bendro tikslo savo sprendimus jie turi suderinti. Tad DaS-oje mokomasi dvejų dalykų: a) kiekvienas agentas mokosi individualiai (arba grupėse, jei agentų grupė sprendžia panašaus tipo uždavinius), b) DaS taip pat mokosi pati suderinti savo agentų veiksmus. Sekančiuose dvejuose skyriuose kalbėsim pagrindinai apie individualų ir grupinį agentų mokymąsi. DaS, kaip vientisos sistemos, mokymasis taip pat būtų galima nagrinėti iš bendrųjų pozicijų.

Aplinkai ir bendriems, visų DaS sudarančių elementų sprendžiamiems, uždaviniams pakitus, turi keistis ir individuali agentų veikla. Kiekvienas iš jų turėtų sparčiai mokytis spręsti naujus uždavinius. Turėtų keistis ir agentų bendrą darbą koordinuojantys algoritmai. Akivaizdu, kad persimokymas spręsti naujai iškilusius uždavinius yra individualus kiekvienai DaS-mai, o svarbiausia, kad tai priklauso nuo jos tipo, nuo aplinkos pasikeitimų charakterio.

Štai vienas iš pavyzdžių. Romoje, Universitete *La Sapienza* dirbtinio intelekto laboratorijoje buvo sukurti robotai, kurie buvo mokomi orientuotis erdvėje ir joje judėti. Kaipo mokymo informacija buvo naudojami specialiai tam paruošti video įrašai. Modeliuojant robotų veiklą jie puikiai susidorojo su video medžiaga, tačiau tapo bejėgiai, patekę į realią ervdę, kur jie jau nebesugebėdavo prie jos adaptuotis. Tam, kad rasti išeitį, teko atlikti didelių skaičiavimo resursų reikalaujančius eksperimentus, idant suprasti, kodėl taip atsitiko. Po ilgų bandymų

buvo nutarta mokymui naudojamus video įrašus „pagadinti“, t.y. uždėti ant jų „triukšmus“, mokant robotus sumažinti video vaizdų ryškumą. Pasirodė, kad turint labai aukštos kokybės video įrašus agentai prie jų pernelyg tobulai prisiderindavo, kitaip sakant, agentai „persimokydavo“. Todėl pateikę į pasikeitusias sąlygas (laboratorinę aplinką), kuri šiek tiek nuo video įrašų, bei juose pateiktų situacijų skyrėsi, intelektualūs agentai nebesugebėjo adaptuotis greitai. Specialiai įrašus „sugadinus“, pirminis išmokimas tapo blogesniu, tačiau pernelyg nepersimokę robotai lengviau sugebėdavo prisiderinti prie naujos aplinkos. Tai bendra visoms DaS-moms savybė.

Aukščiau aprašytas pavyzdys – tai nėra individualus pastebėjimas, specifiškas tik šio tipo uždaviniams. Mokant dirbtinius neuroninius tinklus ypatingai sėkmingo išmokimo atveju tarp atskirų neuronų susidaro labai stiprūs ryšiai, kuriuos pakeisti nėra lengva. Jei kažkokiu būdu pernelyg gerą išmokimą trukdyti, persimokymas spresti naują, pasikeitusį uždavinį taps lengvesniu.

Šio ir sekančių dviejų skyrių uždavinys yra išdėstyti bendrus persimokymo dėsningumus, kurių žinojimas leis projektuotojams geriau orientuotis mokymo, mokymosi ir persimokymo problematikoje. Todėl pradžioje mes išaiškinsime pagrindinius besiadaptuojančius sperndimų priėmimo kompiuterinius modelius – vienasluoksnius ir daugiasluoksnius perceptronus, pateiksime teorines žinias apie jų mokymo ir persimokymo algoritmus, pagrindines persimokymo savybes ir metodus, naudotinus persimokymo procesui pagreitinti. Minėtos priemonės, iš esmės susiveda į tikamų, mokymo procesą (mokymosi stilių) valdančių parametrų parinkimą, kurie priklauso tiek nuo sprendžiamų uždavinių, tiek nuo aplinkos pasikeitimų, ypač nuo jų dažnio ir stiprumo. Vėlesniuose skyreliuose pateiksime keletą daugiaagenčių sistemų schemų, tiksliau, DaS posistemės, sukurtos aplinkos pasikeitimų sekimui, informacijos apie jų pobūdį kaupimui, bei tinkamų mokymosi stilių nusakančių parametrų radimui.

1.1.2. Adaptyvi DaS-mos mokymosi stiliaus parametrus randanti posistemė

Kaip jau minėta anksčiau, DaS galime sudaryti keliais būdais.

Pirma, remiantis taikomojo uždavinio analize, jį formalizuojame, nustatome kiekvieno DaS agento ir jo veiklą charakterizuojančias funkcijas bei parametrus. Visa tai aprašome kaip ekspertinę sistemą.

Antras, sudėtingesnis, DaS sudarymo būdas, – pradžioje daryti kaip pirmuoju atveju, tačiau vietoj ekspertinės naudojame besimokančią pagal pavyzdžius sistemą. Šiuo atveju mokytis turėtų ne tiktai agentai, bet ir visą DaS veiklą aprašantys algoritmai.

Trečiuoju, dar sudėtingesniu, atveju DaS agentai ir struktūra adaptuojasi prie besikeičiančių uždavinių žmogui šiame proces net nedalyvaujant. Tai modernios dabarties ir ateities sistemos.

Nagrinėsim adaptyviąją daugiaagentę sistemą, kurios tikslas – išgyventi besikeičiančioje aplinkoje, išmokti „optimalius“ ją sudarančių agentų „mokymosi

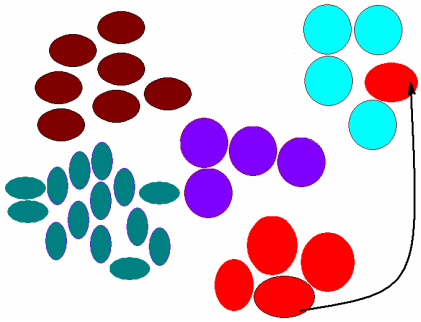
stilių“ aprašančius parametrus ir juos pateikti pagrindinei DaS sistemai. Išgyvenimas besikeičiančioje aplinkoje – tai esminis evoliucijos stimulas, o šio reiškinio analizė – pagrindinis teorinio nagrinėjimo objektas. Šio tipo DaS-mos sudarytos iš dviejų dalių: pirma sudaryta iš pagrindinį uždavinį sprendžiančių vykdančiųjų agentų, o antra - iš „pagalbinių“ agentų, kurie turi užtikrintį sistemos evoliuciją, meta-mokymąsi, kuriame išmokstami mokymo stiliaus parametrai, užtikrinantys spartų sistemos prisitaikymą prie netikėtų pokyčių. Apie pirmąją posistemę jau buvo kalbėta kituose skyriuose. Šiame is dvejuose sekančiuose skyriuose panagrinėsime antrąją, virtualiąją, sistemą.

1.1 pav. schematiškai pavaizduota daugiagentė, iš 35 agentų sudaryta, antroji sistema. Kiekvienas agentas turi spręsti jam skirtą individualų klasifikavimo (vaizdų atpažinimo) uždavinį. Prileiskime, kad agentai – tai vienasluoksniai perceptronai, kurie mokosi skirti objektus (daugiamačius vektorius) į dvi klases. Jie turi mokėti atpažinti $p-$mačius įejimo vektorius $\boldsymbol{x} = (x_1, x_2, ..., x_p)$ su maža klasifikavimo klaidos tikimybe. Šiame pavyzdyje darome prielaidą, kad agentų sprendžiami uždaviniai laikusi bėgant pasikeis, todėl agentai yra adaptyvūs, t.y. jie gali mokytis iš jiems pateikiamų pavyzdžių. Pasikeitus uždaviniui, jie turės prie įvykusio pokyčio prisitaikyti ir sugebėti teisingai atpažinti naujus vektorius.

Prileiskime, kad uždaviniui pasikeitus, kiekvienas iš perceptronų gauna po naują mokymo duomenų rinkinį: po N pirmai ir antrai klasėms priklausomus vektorių $x_1^{(1)}, x_2^{(1)}, ..., x_N^{(1)}, x_1^{(2)}, x_2^{(2)}, ..., x_N^{(2)}$. Naudodamasis, jam duotu duomenų masyvu perceptronas per trumpą laiką turi išmokti gerai klasifikuoti pasikeitusius duomenis. Perceptrono mokymosi greitis priklauso nuo daugelio aplinkybių ir jo mokymo algoritmą charakterizuojančių parametrų reikšmių, tokių kaip mokymo žingsnis, trokštamų išėjimų reikšmės, triukšmo lygis (iškraipymai), kuris kiekvienos mokymo iteracijos metu „uždedamas“ ant dalies atsitiktinai parinktų mokymo vektorių. Apie šiuos parametrus kalbėsime sekančiuose skyreliuose. Minėti parametrai nusako perceptrono „mokymosi stilių“. Priklausomai nuo šio stiliaus perceptronas gali mokytis ir greitai ir lėtai. Deja, nėra universalių, visiems atvejams nustatytų, mokymosi stiliaus parametrų, todėl sprendžiamiems uždaviniams keičiants stiliaus parametrai turi keistis taip pat.

Štai čia ir suformuluojame sekančiuose dvejuose skyriuose sprendžiamą uždavinį: kaip įvertinti minėtus mokymo parametrus. Suprantama, jei mokymosi stiliaus parametrai bus nustatomi netiksliai, sistemos agentai mokysis lėtai. Gali atsitikti, kad pati daugiaagentė sistema taps neveiksminga. Tam, kad agentai (mūsų atveju vienasluoksniai perceptronai) sugebėtų mokytis pakankamai greitai, sukurkime evoliucionuojančią mokymosi sistemą, kurios esmė yra ta, kad:

1) kiekvienas iš agentų turi savo specifinį mokymosi stilių aprašantį charakteristikų rinkinį,
2) jei konktretus agentas mokosi lėtai ir negali per jam užduotą trumą laiką išmokti klasifikuoti naujus vektorius su tikimybe mažesne nei $P_{\max}$, jis pašalinamas iš DaS,
3) žuvusio (pašalinto) agento vietą užima naujai sukurtas agentas-vaikas, kuris



1.1 pav. Daugiagentė sistema, kur 35 agentai susikirstyti į penkias grupes.

paveldi iš kažkurio, atsitiktinai parinkto, sėkmingai (svarbiausiai greitai) besimokančio agento jo mokymosi stiliaus parametrų rinkinį,

4) naujo agento „gimimo“ momentu paveldimų parametrų reikšmės mutuoja (truputį iškraipomos).

Tokiu būdu per ilgą virtinę atpažinimo uždavinių pasikeitimų Das-os agentai prisitaiko prie aplinkos pasikeitimų kitimo dažnių ir pasikeitimų stiprumų. Atkreipiame dėmesį, kad prisitaikymas vyksta daugelio agentų žūties sąskaita. Evoliucinio adaptavimosi idėja paimta iš Gamtos. Gyvūnai, ir mes, žmonės, pritaikėme savo mokymosi būdą prie daugelį tūkstantmečių vykstančių aplinkos pasikeitimų „stiliaus“. Kuriant realias, kompiuteriais modeliuojamas, daugiaagentes sitemas agentai nežūna, – jie tiesiog perprogramuojami. Tam, kad daugiaagentė sistema funkcionuotų, ji turėtų turėti nemažą kiekį „atsarginių“ agentų, kurie, kol agentas-vaikas mokysis ir pradės dirbti teisingai, galėtų atlikti „žuvusiųjų“, daugiaagentei sistemai būtinų, agentų funkcijas.

Kaip jau minėta, DaS prisitaikymas vyksta daugelio agentų žūties sąskaita. Minėto tipo daugiaagentės sistemos posistemė neužtikrina, kad esant ypač dideliems pasikeitimams visi agentai nežus. Dėl šios priežasties pasiūlyta visa eilė DaS patobulinimų. Pavyzdžiui, įvesta altruizmas, kai vienas agents padeda kitam, atliekamas agentų suskirstymas į grupes, „kompiuterinės emocijos“. Šie klausimai detaliau bus nagrinėjami sekančiuose skyreliuose. 1.1 pav. pavaizduotoje daugiagentėje sistemoje 35 agentų suskirstyti į penkias grupes: 4, 4, 5, 7 ir 14 agentų vienoje grupėje. Šioje DaS-je vienos grupės agentai padeda vienas kitam: jei vieno agento klasifikavimo klaidų dažnis žemesnis nei $P_{\max}$, tai jis ir kiti panašūs sėkmingi agentai padeda tam, kurio klaidų dažnis vos vos aukštesnis nei $P_{\max}$. Tokiu būdu sėkmingi agentai padeda „kolegai-agentui“. Jei vienoje grupėje agentų, galinčių „daryti vaikus“ liko tik vienas ar du, sėkmingos grupės agentas gali perduoti savo „genetinį kodą“ (mokymosi stilių nusakantį parametrų rinkinį) kitai vos nežūstančiai grupei, taip padėdamas jai išlikti. 1.1 pav. mes pademonstravome šią situaciją, kur agentas iš raudonos „grupės“ perdavė savo genetinį kodą žydrajai grupei. Toks parametrų perdavimas naudingas – sėkmingos grupės ge-

netinis kodas plinta kitoms grupėms, kurios pradeda greičiau mokytis ir tampa stipresnėmis. Rezultate, grupiniu principu sudaryta DaS tampa atsparesne ir „atlaiko“ stipresnius aplinkos pokyčius.

1.1.3. Neteisingos mokymosi direktyvos – būdas persimokymo procesui pagreitinti

Vėlesniuose skyreliuose parodysime, kad perceptroną mokant jo svoriai auga ir kad mokant minimizuojama nuostolių funkcija prisisotina, t.y. jos išvestinė svorių atžvilgiu pavojingai priartėja prie nulio. Tokiu atveju, pasikeitus aplinkai (atpažinimo uždaviniui, kurį agentas turi išmokti spręsti) persimokymo procesas labai sulėtėja ir per mažą mokymo iteracijų kiekį perceptronas gali nebespėti išmokti gerai atpažinti naujo tipo vektorius su klasifikavimo kaidų tikimybe, pažesne, nei $P_{\max}$.

Tam, kad pasikeitus uždaviniui perceptronas mokytųsi greičiau, reikia neleisti, kad prieš tai vykusio mokymosi metu perceptrono svoriai pernelyg išaugtų. Šio efekto galime siekti įvairiais būdais. Vienas jų – tai mokymas su „specialiai sumaišytomis mokymo direktyvomis“, kur turima galvoje, kad kiekvienos mokymo epochoje (tai $2N$ iteracijų, atitinkančių turimą $2N$ mokymo vektorių kiekį) atsitiktinai parenkamai mokymo vektorių daliai nurodoma neteisingas klasės numeris. Tokių būdu, vietoj to, kad parodžius vektorių $\boldsymbol{x}_{\mathrm{A}}$ ir judeti A kryptimi, šį sykį judėsime priešinga, $\overleftarrow{A}$, kryptimi. Vietoj to, kad neteisingai nurodyti klasės indeksą, prie mokymo vektoriaus komponenčių galime pridėti triukšmą, t.y. jas iškraipyti. Galime rasti keletą artimiausių vektoriaus $\boldsymbol{x}_{\mathrm{A}}$ kaimynų, $\boldsymbol{x}_{\mathrm{B}}, \boldsymbol{x}_{\mathrm{C}}, \boldsymbol{x}_{\mathrm{AD}}, \boldsymbol{x}_{\mathrm{AE}}$, ir juos „užvidurkinti“, pvz.

$$\boldsymbol{x}_{\mathrm{A\ naujas}} = 0.4 \times \boldsymbol{x}_{\mathrm{A}} + 0.2 \times \boldsymbol{x}_{\mathrm{B}} + 0.2 \times \boldsymbol{x}_{\mathrm{C}} + 0.1 \times \boldsymbol{x}_{\mathrm{D}} + 0.1 \times \boldsymbol{x}_{\mathrm{E}}$$

Galima mažinti skirtumus tarp priešingų klasių trokštamų išėjimų, t_1, t_2. Pavyzdžiui, vietoj trokštamų išėjimų $t_1 = 0, t_2 = 1$, naudoti $t_1 = 0.1 t_2 = 0.9$. Tai irgi padeda. Sekančiuose skyreliuose detaliau nagrinėsime visą eilę priemonių, ledžiančių valdyti perceptronų svorių augimo procesą ir padedančių paspartinti persimokymo procesą atpažinimo uždaviniui pakitus.

Vėliau kalbėsime, kad triukšmo, iškraipymų pridėjimas yra „įgimta“ gamtinių biologinių ir netgi socialinių sistemų savybė, kurią verta „pakartoti“ techninėse, informaciją apdorojančiose, sistemose tais atvejais, kai aplinka visą laiką nenumatytai keičiaisi. Skirtumų tarp trokštamų išėjimų t_1 ir t_2(stimuliavimo, sužadinimo) mažinimas irgi Gamtos gamtos pasiūlyta idėja. Sekačiuose skyriuose parodysime, kad modeliuojant DaS veiklą besikeičiančiose aplinkose tiek triukšmo (iškraipymų) pridejimo tiek stimuliavimo lygis seka aplinkos pokyčių stiprumą.

1.2. Dirbtiniais neuroniniais tinklais paremtų agentų mokymas

Patys populiariausi adaptyvūs sprendimų priėmimų algoritmų mokymo metodai yra dirbtiniai neuroniniai tinklai (DNT). DNT sudarymo idėja yra paimta iš Gamtos. Pats paprasčiausias DNT, vienasluoksnis perceptronas (VsP) – tai smegenų ląstelės, neurono, matematinis modelis. Be VsP, plačiai taikomi daugiasluoksniai perceptronai (DsP), spindulinių radialinių funkcijų (SRF), mokymo vektorių kvantavimo (MVK) neuroniniai tinklai. Šiame skyriuje, juos ir jų mokymo metodus ir nagrinėsime.

1.2.1. Sprendimų priėmimo uždaviniai daugiaagentėse sistemose

DaS-se dažniausiai atlikami klasifikavimu ar prognozavimu paremti sprendimai. Štai keletas pavyzdžių, a) kuria aviakompanija, kurio maršrutu suplnuoti klijentui kelionę iš Kauno į Hong-Kongą ir atgal, b) aplenkti kitą, panašias funkcijas atliekantį, agentą-robotą ar sekti jam iš paskos, c) kiek smarkiai ir kuria kryptimi ištiesti šiame eiliniame judesio etape roboto „ranką“ siekiant kuo greičiau pasiekti norimą detalę. Formalus būdas šio tipo uždaviniams spręsti yra kiekvieną objektą, procesą ar situaciją aprašyti „standartinių“ charakteristikų visuma, $x_1, x_2, ..., x_p$, – ateityje vadinkime ją požymių rinkiniu, požymių vektoriumi, ar jų sistema. Klasifikavimo uždavinys būtų: vektorių $\boldsymbol{x} = (x_1, x_2, ..., x_p)$, aprašantį robotų aibės buvimo vietą, jų tuometinius greičius ir pagreičius, aplinką, siekiamą tikslą aprašančius paramterus reikia priskirti vienai iš daugelio klasių, $\Pi_1, \Pi_2, ..., \Pi_K$. Prognozavimo uždavinyje norėsime nustatyti ne klasės numerį, bet kiekybinėmis reikšmėmis aprašomų požymių, y ir α, labiausiai tikėtinas vertes, pvz. *atstumas* $y_1 = 37$ cm, kampas $\alpha_2 = 49°$.

Paprasčiausias yra tiesinis, klasifikatorius, turintis priskirti vektorių $\boldsymbol{x}$ vienai iš K klasių. Jis suskaičiuos „svertines sumas“

$$suma_j = x_1 w_{j1} + x_2 w_{j2} + ... + x_p w_{jp} + w_{j0}, \tag{1.1}$$

kur $w_{j1}, w_{j2}, ..., w_{jp}, w_{j0}$ yra sprendimo (diskriminantinės) funkcijos svoriai ($j = 1, 2, ..., K$), ir sprendimą atlieks pagal $suma_j$ ($j = 1, 2, ..., K$) maksimumą.

Tiesiniu būdu prognozuojant požymį y_j irgi skaičiuosime svertines sumas

$$y_j = x_1 w_{j1} + x_2 w_{j2} + ... + x_p w_{jp} + w_{j0}, \quad (j = 1, 2, ..., R), \tag{1.2}$$

kur R yra prognozuojamų rodiklių skaičius.

Kad rasti svorius $w_{j1}, w_{j2}, ..., w_{jp}, w_{j0}$ pagal empirinius (eksperimentinius, aplinką aprašančius) duomenis yra žinoma kelios dešimtys statistinių, euristinių bei dirbtiniais neuroniniais tinklais paremtų metodų, kur paprastai minimizuojama pasirinkta nuostolių (vidutinės paklaidos) funkcija. Kai kada tiesiniai klasifikatoriai ar prognozės nėra patys geriausi sprendiniai, nes neleidžia pasiekti pakankamai aukšto klasifikavimo ar prognozavimo tikslumo. Tokiais atvejais naudojami

sudėtingesni, netiesiniai klasifikavimo ar prognozavimo metodai. Kai kada norimo tikslumo joks metodas pasiekti neleidžia. Galimos dvi priežastys: a) požymių sistema yra nepakankamai informatyvi, arba b) turimų duomenų kiekis (sprendimo taisyklei sudaryti naudojamų vektorių kiekis) nėra pakankamai didelis. Kai kuriais atvejais aplinka taip staigiai keičiasi, kad net neįmanoma surinkti pakankamai didelės apimties sprendimo taisyklei sudaryti reikalingų duomenų kiekį. Tokiais atvejais reikia stebėti (registruoti kompiuterių atmintyje) ilgos pasikeitimų sekos parametrus ir iš jos bandyti „išgauti“, reikalingą DaS sudaryti, valdyti, permokyti informaciją. Apie tai kalbėsime vėliau.

1.2.2. Vienasluoksnis perceptronas ir jo mokymas

Dirbtiniame neurone, JAV mokslininkų McCullock ir Pits dar 1943 metais pasiūlytame smegenų ląstelės, neurono, modelyje skaičiuojama pareitame skyrelyje jau minėta svertinė suma, *suma*, kuri paduodama į netiesinį, išėjime esantį, elementą, kurio signalas lygus arba artimas nuliui, jei *suma* stipriai neigiama, ir artimas vienetui, jei signalas *suma* yra didelė teigiama. Tad prieš patekdama į VSP išėjimą *suma* dar netiesiškai apdorojama: $išėjimas = f(suma)$. Funkcija $f(suma)$ vadinama aktyvavimo funkcija. Labai dažnai naudojama sigmoidinė funkcija, kintanti tarp nulio ir vieneto (žr. antrą paveikslėlį)

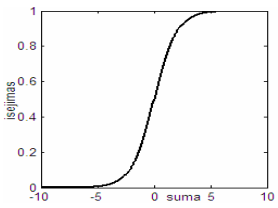
$$f(suma) = 1/(1 + e^{-suma}). \tag{1.3}$$

Naudojant minėtą sigmoidinę funkciją trokštami išėjimai galėtų būti intervale [0 1], pavyzdžiui: $t_1 = 0$ (pirmai klasei), $t_2 = 1$ (antrai klasei). Galimi ir kiti pasirinkimai, pvz. $t_1 = 0.1, t_2 = 0.9$, arba netgi $t_1 = 0.495, t_2 = 0.505$. Skirtumą $\Delta t = t_2 - t_1$ sąlyginai vadinsime stimuliavimu. Alternatyvūs terminai būtų: suždinimas, sujaudinimas, dirginimas (angl. *stimulation, arousal*). Jei stimuliavimas mažas, mokymas vyks naudojant tik labai trumpą aktyvavimo funkcijos $f(suma)$ dalį, kur *suma* reikšmė mažai nutolsta nuo nulio. Tad siauroje, aplink nulį esančioje zonoje, funkcija $f(suma)$ beveik tiesinė. Kaip jau minėta, dirbtiniai neuroniniai tinklai, paremti iš gamtos „pasiskolintomis“ idėjomis. Kad nežinomus svorius surasti, naudojame vidutinės kvadratų sumos funkciją, kurią vadinsime „nuostoliais“

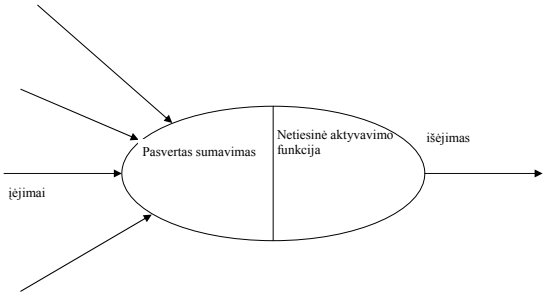
$$nuostoliai = \frac{1}{n}\sum_{i=1}^{2}\sum_{j=1}^{N}(t_{ij} - f(x_{1ij}w_1 + x_{2ij}w_2 + w_0))^2, \tag{1.4}$$

kuri šiuo atveju užrašyta dviejų požymių ($p = 2$), ir dviejų klasių atveju (šiuo, binariniu, atveju neuroninis tinklas turi tik vieną išėjimą) x_{sij} yra i-tosios klasės j-tojo mokymo vektoriaus s-tojo požymio reikšmė, $t_{ij} = t_i$ yra i-tosios klasės trokštamas išėjimas (vienodas visiems tos klasės mokymo vektoriams).

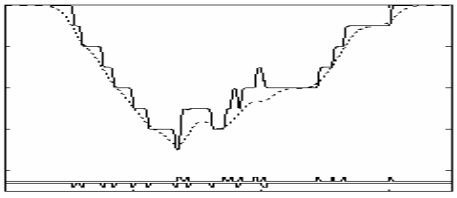
Nuostolių funkcijoje figūruoja netiesiškumas, $f(suma)$. Todėl ji turi daug ekstremumų (4 pav.). Dėl daugelio ekstremumų ieškomi svoriai negali būti rasti



1.2 pav. Netiesinė aktyvavimo funkcija.



1.3 pav. Dirbtinio neurono schema.



1.4 pav. Vienasluoksniu perceptronu paremto klasifikatoriaus nuostolių funkcija svorio w_1 atžvilgiu), kai turim po dešimt mokymo vektorių iš kiekvienos klasės.

analitiškai. DNT-se ieškomų svorių radimas paprastai atliekamas iteracijų būdu: *naujassvoris = senassvoris + pataisymas*. Populiariausias yra gradientinio nusileidimo (angl. *gradient descent*) metodas

$$\boldsymbol{w}_{naujas} = \boldsymbol{w}_{prieš\ \ tai} - \eta \frac{\partial nuostoliai}{\partial \boldsymbol{w}}, \tag{1.5}$$

kur parametras η vadinamas mokymo žingsniu (angl. learning step).

Punktyrine linija pavaizduotas grafikas 1.3 pav. yra dar gana glotnus – jis pavaizduotas mokymo pradžios situacijai, kai dar perceptrono svoriai nėra per daug išaugę. Kai mokymo metu svoriai išauga, pasvertosios sumos (1.1) išauga taip pat. Todėl daugumai mokymo vektorių aktyvavimo funkcijos (1.3) išėjimai priartėja prie vieneto ar nulio. Tokiu atveju nuostolių funkcija tampa laiptuota (ištisine linija 1.3 pav.). Dėl šios priežasties daugelyje vietų nuostolių funkcijos (1.4) išvestinės, $\frac{\partial nuostoliai}{\partial \boldsymbol{w}}$, tampa artimos nuliui. Kaip pasekmė – išaugus svoriams mokymas sulėtėja. Kai svorių, atitinkamai ir išvestinių, daug, išvestinių vektorius $\frac{\partial nuostoliai}{\partial \boldsymbol{w}}$, vadinamas gradientu, o mokymo taisyklė – gradientiniu minimizavimo algoritmu).

Perceptroną mokant svarbus momentas yra teisingai parinkti pradinį svorių vektorių, $\boldsymbol{w}_{\text{start}}$. Klasifikavimo uždavinyje, mokant VsP, t.y. augant mokymo iteracijų (epochų) kiekiui galima gauti kelis vienas po kito sekančius klasifikavimo algoritmus. Jei $\boldsymbol{w}_{\text{start}} = 0$, ir mokymo duomenų (abieju klasių) vidukis yra nulis, tai iš pat pradžių galime gauti patį paprasčiausią, t.y. Euklidinio atstumo klasifikatorių, kuris nežinomo vektoriaus $\boldsymbol{x}$ priskyrimą vienai iš klasių atlika pagal $\boldsymbol{x}$ Euklidinius atstumus iki klasių vidurkių vektorių, įvertintų pagal mokymo duomenis. Vėliau gauname reguliarizuotą ar standartinį tiesinį Fišerio klasifikatorius, t.y., vienus iš pačių populiariausių klasifikavimo metodų. Jei trokštami išėjimai, t_1 ir t_2, arti nulio ar vieneto, ir mokant turime labai mažai klasifikavimo klaidų. tai kaip jau minėjome, perceptrono svoriai išauga. Naudojant netiesinę aktyvavimo funkciją, jos užsilenkimai „pradeda jaustis“: VsP išėjimai $(1 + e^{-xijw+w_0})^{-1}$ pradeda artėti prie 1 arba 0. Tokiu atveju dideli atsilenkimai (nukrypimai nuo klases skiriančios hiperplokštumos), jeigu tokių mokymo duomenyse yra, pradeda mažai įtakoti mokymo procesą. Reiškia, VsP tampa robastiškas, t. y. atsparus dideliems mokymo duomenų nukrypimams (netikslumams). Tai labai puiki VsP savybė.

Jei mokyme naudoti ribines trokštamų išėjimų reikšmes, $t_1 = 0$ (pirmai klasei), $t_2 = 1$ (antrai klasei), tai esant mažam klasifikavimo klaidų kiekiui mokymo duomenyse, perceptrono svoriai gali ypač smarkiai išaugti. Esant labai dideliems svoriams, netiesinio VsP išėjimai $(1 + e^{-xijw+w_0})^{-1}$ visiškai priartės prie 1 arba 0. Tokiu atveju nuostolių funkcija (1.3) išreikš neteisingai suklasifikuotų mokymo vektorių dalį (dažnį), t.y. empirinę klasifikavimo klaidos tikimybę. Esant, kad ir nedidelei neteisingai klasifikuojamų mokymo vektorių daliai, svoriai pernelyg smarkiai neišaugs, nes neteisingai klasifikuojami vektoriai „temps svorių vektorių atgal“. Vėliau matysime, kad vykstant aplinkos pasikeitimams, ši perceptrono savybė yra teigima.

Mokant VsP dar ilgiau situacijose, kai mokyme klaidų nėra arba, kai nuo klasifikavimo hiperplokštumos labai toli esantys mokymo vektoriai (kad ir neteisingai klasifikuoti) jau nustoja įtakoti mokymo procesą, gauname atraminių vektorių klasifikatorių (angl. *support vector classifier*), kuris šiuo metu daugelio žmonių dar labai garbinamas ir laikomas bene geriausiu.

Be gradientinio yra daugybė sudėtingesnių antros eilės (skaičiuojama ne pirmos eilės, bet antros eilės išvestinės) mokymo metodų, kurie leidžiančia su mažesniu iteracijų kiekiu priartėti prie nuostolių funkcijos (1.4) minimumo. Greta šios puikios savybės antros eilės metodai turi ir tą trūkuma, kad „labiau skubėdami“, jie greičiau pakliūna į lokalinį minimumą ir esant svoriams dideliems, nuostolių funkcijos dugnui – plokščiam, o „šonams“ – statiems, dažnai nebesugeba iš tokio „netikusio“ lokalinio minimumo „pabėgti“. Egzistuoja visa eilė būdų, kaip tai bandyti padaryti (pabėgti iš blogo lokalinio minimumo). Tai *multistart* (daugkartinis mokymas pradedant nuo skirtingų pradinių svorių vektorių), triukšmo pridėjimas prie svorių, mokymo vektorių, trokštamų išėjimų reikšmių ir pan. Labai greitas mokymas taip pat veda prie to, kad kai kada perceptronas „beskubėdamas“ „prašoka“ paprastesnius klasifikavimo algoritmus (pvz. reguliarizuotą Fišerį), kurie esant nedideliam mokymo duomenų kiekui (o tas ypaš svarbu mokant besikeičiančiose sąlygose) gali pasirodyt labiausia tinkami. O prašokus, – kelio atgal jau nebėra. Lieka vėl mokyti iš naujo (multistart).

Aukščiau išdėstyta medžiaga rodo, kad vienasluksnis perceptronas yra gan paprastas algoritmas, tačiau turi daug universalumo savybių. Todėl VsP naudotinas sudarant daugiaagentines sitemas, funkcionuojančias besikeičiančiose aplinkose.

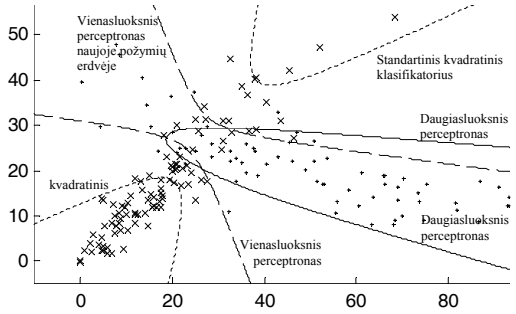
1.2.3. Netiesinės erdvės transformacijos

Vienasluoksnis perceptronas leidžia gauti tik tiesines klasifikavimo ar prognozavimo taisykles. Kai kuriuose praktiniuose uždaviniuse to nepakanka, – įėjimo požymių požiūriu taisyklė turi būti netiesinė. Sekančiuose skyreliuose kalbėsime, kad šio tipo problemas gali išspręsti daugiasluoksniai perceptroniai, spindulinių bazinių funkcijų, mokymo vektoriaus kvantavimo dirbtiniai neuroniniai tinklai. Vienasluoksnis perceptronas taip pat gali būti pritaikytas netiesinių uždavinių sprendimui. Tam prieš jį mokant *reikia netiesiškai transformuoti požymių erdvę*.

Bene paprasčiausias būdas sudaryti netiesinę taisyklę – tai vietoj originaliųjų p požymių, $x_1, x_2, ..., x_p$, naudoti daugiau jų, t.y. išvestinius, netiesinių transformacijų pagalba padarytus, požymius. Pavyzdžiui, prie turimų požymių pridėjus naujus: $(x_1)^2, x_1x_2, ..., x_1x_p, (x_2)^2, x_2x_3, ..., x_{p-1}x_p, (x_p)^2$ ir mokant $p(p+1)/2$ matavimų erdvėje galime gauti kvadratinę diskriminantinę funkciją (žr. 5-tą paveikslą).

Dažnai nauji požymiai daromi panašumo į mokymo vektorius pagrindu. Tegul $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n$ yra n mokymo vektorių, nesvarbu tai ar klasifikavimo, ar prognozavimo uždavinį sprendžiame. Tada kiekvienam mokymo (testiniam, taip pat) vektoriui galime suskaičiuoti n atstumų iki mokymo vektorių $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n$

$$z_j = D(\boldsymbol{x}, \boldsymbol{x}_j) = (\boldsymbol{x} - \boldsymbol{x}_j)(\boldsymbol{x} - \boldsymbol{x}_j)^T, \quad j = 1, 2, ..., n, \tag{1.6}$$



1.5 pav. Kvadratinės klasifikavimo taisyklės skiriamoji kreivė, sudaryta mokant VsP kvadratinių požymių erdvėje ir jos palyginimas su DsP ir standartinio kvadratinio klasifikatoriaus pagalba gautomis klases skiriančiomis netiesinėm ribom.

kur n yra mokymo vektorių kiekis, $\boldsymbol{x}_j$ yra j-tasis mokymo vektorius,

Atstumai $z_1, z_2, ..., z_n$ ir bus nauji požymiai. Šioje erdvėje galime sudaryti bet kurį iš žinomų tiesinių klasifikatorių. Jeigu būtų sudaromas Fišerio klasifikatorius, tai jis leistų gauti hiperplokštumą, vienodai nutolusią nuo visų n mokymo vektorių. Naudoti visus n požymių neracionalu. Praktiškai reikėtų sumažinti požymių kiekį, atrenkant tik pačius svarbiausius (žr. sekantį skyrių). Autoriaus nuomone, bene racionaliausias variantas, tai mokyti VsP.

Tiesioginis panašumo požymių naudojimas pasiteisina ne visada. Todėl dažnai naudojami netiesinės branduolio funkcijos pagalba papildomai apdoroti požymiai. Labai plačiai naudojamas būdas, pereiti į „netiesinę požymių erdvę“, tai nudoti branduolio funkcijas. Populiariausios yra Gausinės

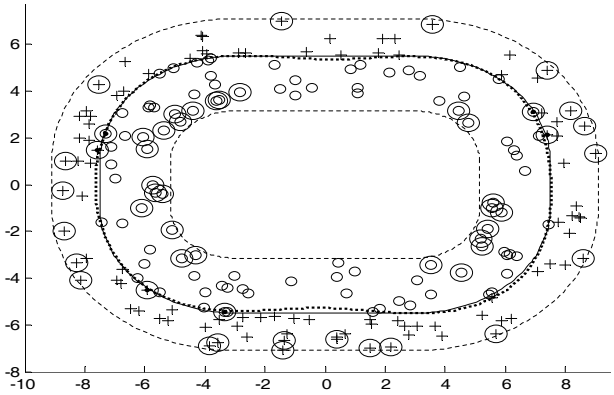
$$z_j = \kappa(\boldsymbol{x}, \boldsymbol{x}_j), = \exp(-\alpha D(\boldsymbol{x}, \boldsymbol{x}_j)) \tag{1.7}$$

ir polinominės branduolo funkcijos

$$z_j = \kappa(\boldsymbol{x}, \boldsymbol{x}_j), = (\boldsymbol{x}(\boldsymbol{x}_j)^T + 1)^r, \tag{1.8}$$

kur α yra atstumo matavimo skalės nustatymo (glotninimo) parametras, o r yra polinominių požymių eilės rodiklis. Aukščiau minėtu, kvadratinių požymių atveju turėjome: $r = 2$.

Praktikai labai svarbu parinkti tinkamą atstumo matavimo skalės (glotninimo) parametrą α. Transformacija (1.7) skirta sumažinti smarkiai nutolusių vektorių įtaką. Bet kuriuo atveju, klasifikatoriaus sudarymo uždavinys nėra lengvas, nes reikia ne tik sumažinti požymių kiekį, bet ir pasirinkti požymių tipą, nustatyti parametrą α, rasti kada optimaliai stabdyti perceptrono mokymą. 1.6 pav. pateikiamas pavyzdys, kuriame turime po šimtą kiekvienos iš dviejų klasių mokymo vektorių 2-matėje erdvėje ir klases skiriančioji kreivė, gauta VsP pagalba.



1.6 pav. Klasifikavimo ribos, gautos naudojant VsP 200 naujų požymių erdvėje.

1.2.4. Mokymosi greitis ir jį įtakojantys parametrai

Kuriant daugiaagentes sistemas darbui besikeičiančiose aplinkose labai svarbu, kad jas sudarantys agentai mokytųsi greitai ir per trumpą laiką (nedidelį mokymosi iteracijų skaičių) prisitaikytų prie pasikeitusių uždavinių. Praeitame skyrelyje matėme, kad pagrindiniai parametrai, įtakojantys vienasluoksnio perceptrono mokymo procesą, yra:

- mokymo žingsnis,
- pradinis svorių vektorius,
- trokštami išėjimai,
- mokymo iteracijų kiekis.

Be išvardytųjų parametrų mokymo greitį labai įtakoja duomenys. Kad mokymo procesas būtų spartus, svertinės sumos, $x_1 w_1 + x_2 w_2 + ... + x_p w_p + w_0$, neturi būti pernelyg didelės. Atskiriems mokymo vektoriaims šios sumos turi būti tiek teigiamos, tiek neigiamos. Dėl šios priežasties prieš pradedant perceptroną mokyti tikslinga mokymo duomenu vidurkį „nustumti“ į nulinį tašką. Tai galim padaryti suskačiuojant turimų duomenų vidurkių vektorių $\mathbf{M}$, ir jį atimti iš kiekvieno mokymo, bei testinio vektoriaus. Taip pat tikslinga dekoreliuoti įėjimo vektoriaus $\boldsymbol{x}$ požymius, padauginant mokymo ir testinius vektorius iš ortogonalios duomenų transformacijos matricos:

$$\boldsymbol{y} = \mathbf{T}(\boldsymbol{x} - \mathbf{M}). \tag{1.9}$$

Kadangi mokymo procese naudojamas viena mokymo žingsnio, η, reikšmė, yra bendra visiems požymiais (pasuktoje požymių erdvėje, tai naujasis požymių vektorius $\boldsymbol{y}$), tai labai svarbu suvienodinti visų požymių vektoriaus $\boldsymbol{y}$ komponenčių $y_1, y_2, ..., y_p$ dispersijas. Tai padaryti galime padauginus vektorių $\boldsymbol{y}$ iš diagonalinės matricos $\mathbf{d} = \mathbf{D}^{-1/2}$, kur matricos $\mathbf{D}$ diagonalinės elementai yra vektoriaus $\boldsymbol{y}$

komponenčių $y_1, y_2, ..., y_p$ dispersijos. Kalbant apie praktinius minėtos transformacijos aspektus prileiskime, kad skaičiavimus atliksime *Matlab'o*, programinės sistemos, skirtos darbui su matriciniais duomenimis, pagalba. Tegul $\mathbf{S}$ jau suskaičiuota duomenų kovariacinė matrica (vidutinė K klasių kovariacinė matrica, jei sprendžiam klasifikavimo uždavinį). Standartinė *Matlab'o* programa svd.m leidžia surasti minėtas matricos nuosavų vektorių matricą $\mathbf{T}$ ir nuosavų reikšmių matricą $\mathbf{D}$: [$\mathbf{T}$, $\mathbf{D}$, $\mathbf{U}$] = svd($\mathbf{S}$). Jeigu lyginant su požymių kiekiu, p, mokymo vektorių, n, nėra daug, matrica $\boldsymbol{S}$ gali būti artima išsigimusiai (dalis matricos $\mathbf{D}$ diagonalinių elementų bus lygūs nuliui arba bus labai artimi nuliui). Tokiu atveju vietoj matricos $\mathbf{S}$ reiktų naudoti reguliarizuotą jos variantą, $\mathbf{S}_{\mathrm{R}} = \mathbf{S} + \lambda\mathbf{I}$, kur λ yra reguliarizavimo konstantė (nelabai didelis bandymų ir klaidų metodu parenkamas teigiamas skaičius), o $\mathbf{I}$ yra $p \times p$ iš vienetukų diagonalėje sudaryta vienetinė matrica (*Matlab'e* $\mathbf{I}$ = eye(p)).

Taigi, praktiškai vietoj vektorių $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n$ naudotume vektorius $\boldsymbol{z}_1, \boldsymbol{z}, ..., \boldsymbol{z}_n$

$$\boldsymbol{z}_j = (\mathbf{D} + \lambda\,\mathbf{I})_{-1/2}\mathbf{T}(\boldsymbol{x}_j - \mathbf{M}), \quad j = 1, 2, ..., n. \tag{1.10}$$

Dažnai vietoj visų, p, naujųjų vektorių, $\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_n$, komponenčių tikslinga naudoti tik dalį iš jų. Tai ir pagreitintų skaičiavimus, ir padėtų spręsti „trumpų" mokymo duomenų problemą, kas ypač aktualu kuriant daugiaagentes sistemas, skirtas darbui besikeičiančiose aplinkose.

Duomenų transformacija (1.10) turi dar *dvi labai svarbias savybes*.

Pirmoji savybė. Po transformacijos (1.10) požymiai tampa nekoreliuoti ir beveik su vienodom (vienetinėm) dispersijom. Jei perceptrono pradinis svorių vektorius sudarytas iš nulių, tai po primos *batch* režime (svorių pataisymas atliekamas parodžius visus visų klasių mokymo vektorius) atliktos mokymo epochos gautas Euklidinio atstumo klasifikatorius sutampa su Fišerio klasifikatoriumi originalioje, $\boldsymbol{x}$ požymių erdvėje. Jei sprendžiamas regresijos uždavinys, tai tinkmai parinkus mokymo žingsnio parametrą η, po pirmos mokymo epochos $\boldsymbol{x}$ požymių erdvėje gautume prognozavimo lygtį, kurios visos komponentės proporcingos standartinei minimalių regresijos lygrie komponentėms. Tai geras varinatas – gauti labai neblogą klasifikavimo ar prognozavimo taisyklę jau pirmosios iteracijos metu.

Antroji savybė. Mokymo taisyklėje (1.5) naudojama mokymo žingsnio η reikšmė yra bendra visomis vektorių $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n$ ar $\boldsymbol{z}_1, \boldsymbol{z}, ..., \boldsymbol{z}_n$ komponenčių kryptimis. Jei požymių dispersijos šiomis kryptimis smarkiai skiriasi, η reikšmė vienoms kryptimis tampa pernelyg maža, o kitoms – per didelė. Tai negerai, nes mokymo procesas arba bus pernelug lėtas arba nestabilus (gali diverguoti). Transformacija (1.10) suvienodina dispersijas ir leidžia parinkti optimalią visoms kryptims mokymo žingsnio reikšmę. Tada mokymo procesas tampa greitu.

Kalbant apie dauelio agentų mokymą besiskeičiančiose aplinkose, reika stengtis atlikti tokias, lygties (1.10) tipo, duomenų transformacijas, kad įvykus atpažinimo ar prognozavimo uždavinių pasiketimams, požymiai tebeliktų nebekoreliuoti ir jų dispersijos pernelyg smarkiai nepasikeitų. Tai – neišspęstas ateities uždavinys.

Mokymo žingsnis valdo svorių modifikavimo greitį eilinės mokymo iteracijos (epochos) metu. Iš pirmo žvilgsnio atrodo, kad kuo didesnis žingsnis, – tuo mokymosi procesas bus spartesnis. Tai tiesa, bet tik iki tam tikro laipsnio. Jei mokymosi žingsnis pernelyg didelis, tai jau po pirmos iteracijos galime gauti pernelyg didelį svorių vektoriaus pokytį, kurio rezultate daugumos ar net visų svertinių sumų, $x_1 w_1 + x_2 w_2 + ... + x_p w_p + w_0$, absoliutinės reikšmės gali tapti pernelyg aukštos, kas ves prie to, kad sprendžiant klasifikavimo uždavinius perceptrono išėjimų reikšmės

$$f(c) = \frac{1}{1 + e^{-(x_1 w_1 + x_2 w_2 + ... + x_p w_p)}} \tag{1.11}$$

taps artimos nuliui arba vienetui, o funkcijos (1.11) išvestinės svorių $w_1, w_2, ... w_p$ ir w_0 atžvilgiu – visiškai priartės prie nulio. Tokiu atveju mokymas praktiškai sustos. Esant dideliam mokymo žingsniui, mokymo procesas gali ir diverguoti. Viena iš išeičių, – tai adaptyvus mokymo žingsnio valdymas mokymo proceso metu. Jei per tam tiktą mokymo iteracijų kiekį nuostolių funkcija pastoviai mažėja, mokymo žingsnis didinamas. Jei nuostolių funkcija paauga, tai mokymo žingsnis sumažinamas.

Pradinio svorių vektoriaus $\boldsymbol{w}_{\text{start}}$ komponentės taip pat labai įtakoja mokymo procesą. Jei regresijos uždavinyje vektoriaus $\boldsymbol{w}_{\text{start}}$ komponentės tiksliai sutampa su idealios tiesinės prognozavimo lygties svorių reikšmėmis, perceptoną mokyti reikia trumpai. Čia labai svarbiu tampa optimalaus sustojimo aspektas. Kuo tikslesnis yra $\boldsymbol{w}_{\text{start}}$, tuo anksčiau reiks mokymą nutraukti, tuo tikslesnė (laiku sustojus) gausis prognozavimo lygtis. Jei laiku nesustosim, o mokysimės iki nuostolių funkcijos minimumo radimo, geros pradinių svorių vektoriaus $\boldsymbol{w}_{\text{start}}$ reikšmės bus prarastos. Daugiaagentėse sistemose, pasikeitus sprenžiamam uždaviniui, dažnai pasikeitimo momento mes net nežinome ir mokymą tęsiame toliau. Svorių vektorius $\boldsymbol{w}_{\text{finiš}}$, gautas paskutiniu momentu prieš uždavinio pasikeitimą, tampa pradiniu (startiniu) mokant toliau. Todėl kuriant DaS-as į šį faktorių reikia atsižvelgti.

Atpažinimo uždavinyje vektorių $\boldsymbol{w}_{\text{start}}$ ir $\boldsymbol{w}_{\text{finiš}}$ reikšmės taip pat įtakoja tolimesnę mokymo proceso sėkmę. Šiuo atveju prisideda dar vienas, papildomas aspektas: be pradinio vektoriaus $\boldsymbol{w}_{\text{start,}}$ nustatančio klases skiriančiosios hiperplokštumos padėtį, didelę įtaką daro vektoriaus $\boldsymbol{w}_{\text{start}}$ komponenčių absoliutūs dydžiai: juk padauginus iš teigiamos konstantės γ, hiperplokštumos

$$x_1 \gamma w_{\text{start 1}} + x_2 \gamma w_{\text{start 2}} + ... + x_p \gamma w_{\text{start p}} + \gamma w_{\text{start 0}} = 0, \tag{1.12}$$

padėtis p-matėje erdvėje nepasikeis. Daugiklio γ reikšmė, tačiau, gali labai smakiai paveikti mokymosi proceso greitį. Jei vektoriaus $\boldsymbol{w}_{\text{start}}$ komponetės labai didelės, daugumos ar net visų svertinių sumų absoliutinės reikšmės gali tapti pernelyg aukštomis. Tuomet perceptrono išėjimų reikšmės taps artimos nuliui arba vienetui, o funkcijos (1.11) išvestinės svorių $w_1, w_2, ... w_p$ ir w_0 atžvilgiu – taps artimos nuliui. Mokymosi procesas praktiškai sustos. Jei pradinio vektoriaus $\boldsymbol{w}_{\text{start}}$

komponetės labai mažos, mokymas bus greitas, bet esant pakankamai didelei mokymo žingsnio vertei, gera, naudinga informacija, kurią savyje saugo vektorius $\boldsymbol{w}_{\text{start}}$, gali būti prarasta jau pirmos mokymo iteracijos metu. Tai dar viena neišspręsta dilema, kuri DaS sitemose, veikiančiose besikeičiančiose sąlygose, yra ypač aktuali.

Trokštami išėjimai yra ypatingai svarbūs mokant ir permokant klasifikavimo uždavinį sprendžiančius perceptronus. Jei trokštami išėjimai sutampa su aktyvavimo funkcijos ribinėmis reikšmėmis, tai mokant perceptroną, jo svoriai gali neapibrėžtai išaugti jei mokymo duomenys visiškai ar netgi tik beveik tiesiškai atsiskiria (empirinių klasifikavimo klaidų dažnis tampa lygus arba artimas nuliui). Kaip jau minėjome, išaugus svoriams mokymo greitis sumažėja.

Ką daryti, kad mokymo greitį padidinti? Naudojant aktyvavimo funkciją (1.3) jos išėjimo ribinės reikšmės yra 0 ir 1. Šiuo atveju jiems atitinkantys trokštami išėjimai yra $t_1 = 0$(pirmai klasei) ir $t_2 = 1$(antrai klasei). Tad efektyvus būdas neleisti svoriams pernelyg išaugti ir taip lėtinti mokymo procesą yra nenaudoti ribinių tokštamų išėjimų reikšmių, t.y. pasirinkti $t_1 = 0.1, t_2 = 0.9$, arba $t_1 = 0.4, t_2 = 0.6$. Reikėtų pasakyti, kad dirbtinis svorių sumažinimas pradeda trukdyti robastinio, minimalios klasifikatvimo klaidos bei atraminių vektorių klasifikatorių gavimą. Tad kad kai kuriuose atpažinimo uždaviniuose šis metodas gali tapti ir trukdžiu. Taipogi, „optimalios“ (mokymo greičio prasme) trokštamų išėjimų reikšmės turi kisti mokymo proceso metu. Tad problemų čia daug ir jos dar tik laukia savo sprendimo. Bet jas žinoti reikia.

Triukšmo pridėjimas. Alternatyvus metodas, skirtas svorių augimui sulėtinti, yra didinti empirinių klasifikavimo klaidų kiekį kiekvienoje iteracijoje prie atsitiktinai parinktų mokymo vektorių komponenčių ar prie trokštamų išėjimų reikšmių pridedant „triukšmą“, t.y, šias reikšmes iškraipant. Kiekvienos iteracijos metu prie atsitiktinai parinktų mokymo vektorių komponenčių pridedamas atsitktinis dydis, kurio vidurkis paprastai yra nulis, o dispersija λ. Tai, taip vadinamo, „balto“ triukšmo pridėjimas. Galime pridėti ir „spalvotą“ triukšmą. Efektyvus būdas - tai kiekvienam mokymo vektoriui (pažymekime jį $\boldsymbol{x}_m$) rasti du ar tris „artimiausius kaimynus“ (panašiausius į $\boldsymbol{x}_m$ tos pačios klasės mokymo duomenų vektorius, $\boldsymbol{x}_{m1}, \boldsymbol{x}_{m2}, \boldsymbol{x}_{m3}$) ir jų kryptimi R kartų atlikti vektoriaus iškraipymus. Tokiu atveju mokymo duomenų kiekis padidės R kartų. Įėjimo mokymo vektorių iškraipymas didina klaisifikavimo klaidų kiekį mokymo procese ir trukdo svoriams pernelyg išaugti. Triukšmą galime pridėti ne tik prie įėjimų, bet ir prie išėjimų. Taip mokant, karts nuo karto atsitiktinai parinkto mokymo vektoriaus klasės numeris nurodomas neteisingai. Kaip ir anksčiau triukšmo lygis parenkamas eksperimentuojant „bandymų ir klaidų“ metodo pagalba. Paminėtina, kad DaS-ose kiekvienam iš agentų ar jų grupių gali tekti naudoti skirtingo tipo mokymo duomenų iškraipymus.

Mokymo iteracijų kiekis įtakoja svorių dydį, klasifikatoriaus ar regresijos taisyklės tipą, sąlygoja persimokymo reiškinio atsiradimą. Kaip minėjome, optimalus jų kiekis priklauso nuo mokymo duomenų ir mokymo algoritmo charakteristikų, o taip pat nuo sprendžiamo uždavinio pasikeitimų dažnio, pasikeitimų

tipo ir jų stiprumo. Nepaisant to, kad laikotarpiai tarp uždavinio pasikeitimų tūkstančius kartų didesni, nei laikas reikalingas DaS permokymui, į DaS ir jos elementų mokymo iteracijų kiekį dėmesį kreipti yra būtina.

1.2.5. Vienasluoksnio perceptrono mokymas atpažinimo uždaviniui pasikeitus

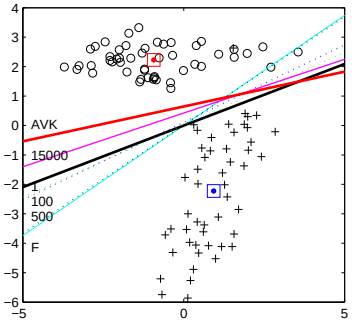
Tam, kad geriau įsiąmoninti sunkumus, iškylančius permokant vienalsuoksnį ir daugiasluoksnį perceptronus, šiame skyrelyje panagrinėsime atvejį, kai VsP paeiliui mokosi spręsti du skirtingus atpažinimo uždavinius. Mus domina kaip greitai perceptroną mokant mažėja klasifikacijos klaida mokantis pirmą ir vėliau – antrą uždavinius. Konkretumo dėlei panagrinėkime paprastą atveją, kur prileidžiame, kad kiekviena iš klasių – tai pagal normalųjį $\mathrm{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ pasiskirstymą išsidėsčiusiųjų vektorių grupė, kur i – tai klasės indeksas, $\boldsymbol{\mu}_i$ yra i-tosios klasės vidurkių vektorius, o $\boldsymbol{\Sigma}_i$ yra kovariacinė matrica. Sudarant (mokant) klasifikatorių laikoma, kad $\boldsymbol{\mu}_i$ ir $\boldsymbol{\Sigma}_i$ yra nežinomi.

Tegul klasifikavimo taisyklei sudaryti (perceptronui mokyti) turime po N 2-mačių vektorių iš kiekvienos klasės. Priimkime, kad pirmame uždavinyje $\boldsymbol{\mu}_1 = [1.090 \quad -2.259]$, $\boldsymbol{\mu}_2 = -\boldsymbol{\mu}_1$, $\boldsymbol{\Sigma}_1 = \begin{bmatrix} 0.99 & 1.29 \\ 1.29 & 4.01 \end{bmatrix}$, $\boldsymbol{\Sigma}_2 = \begin{bmatrix} 1.79 & 1.19 \\ 1.19 & 0.21 \end{bmatrix}$.

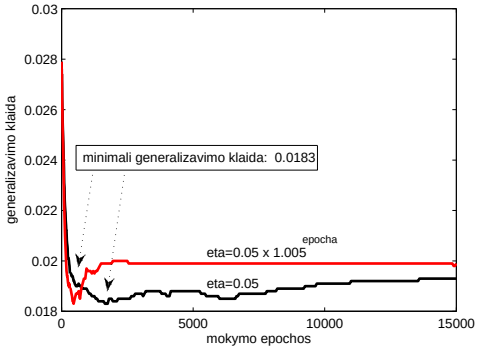
1.7 pav. pavaizdavome po 50 kiekvienos klasės vektorių, naudojamų mokymui. Klasių vidurkiai pažymėti kvadratukais. Perceptroną pradėjus mokyti nuo vektoriaus $\boldsymbol{w}_{\text{start}} = [0\ 0\ 0]$, po pirmos mokymo epochos *batch* režime gauname Euklidinio atstumo klasifikatorių, kurio „skiriamasis paviršius“ – tai vienetuku pažymėta tiesė. Po šimto ir penkių šimtų epochų naudodami mokymo žingsnį $\eta = 0.05$ gauname mažai besiskiriančias tieses 100 ir 5000, o po 15000 epochų – skaičiumi 1500 pažymėtą tiesę. Jei perceptroną mokant palaipsniu didinti mokymo žingsnį, $\eta_r = 0.05 \times 1.005^r$, po 1500 mokymo epochų priartėjame prie atraminių vektorių klasifikatoriaus (raudona tiesė, 1.7 pav. pažymėta raidėmis AVK). Jei vienasluoksnį perceptroną mokyti mažai besiskiriančiais trokštamais išėjimais, $t_1 = 0.4, t_2 = 0.6$ ir labai pamažu didinti mokymo žingsnį, $\eta_r = 0.05 \times 1.0001^r$, tai po 15000 epochų priartėsime prie standartinio tiesinio Fišerio klasifikatoriaus (žydra ir juoda punktyrinės linijos, pažymėtos raide F).

Matome, kad mokant VsP galime gauti įvairaus tipo tiesinius klasifikatorius. Vienasluoksnis perceptronas (daugiasluoksnis taip pat) – tai ne vienas klasifikavimo metodas, o nuo mokymo sąlygų labai priklausanti klasifikavimo algoritmų aibė. Klasifikavimo klaidos dinamika 15000 mokymo epochų metu pavaizduota 1.8 pav., kur raudonai pavaizduotas grafikas rodo, kad „protingai“ didinant mokymo žingsnį mokymo laiką galime žymiai sumažinti. Tačiau tam, kad įvaldyti perceptrono mokymą, reikia ne tik giliai teoriškai pažinti perceptrono mokymosi procesą, bet ir turėti nemažą praktinio darbo patirtį. Kai kada sakoma, kad perceptrono mokymas – tai ne mokslas, o menas.

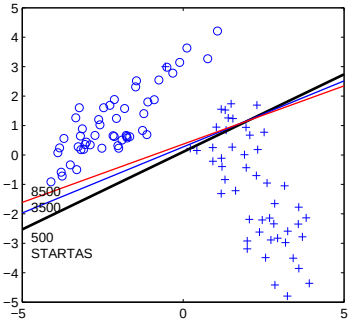
1.9 pav. pateikiame dviejų klasių mokymo duomenų vektorių išsibarstymą antrąjame atpažinimo uždavinyje. Čia taip pat matome tris skirtingas startines,



1.7 pav. Dviejų klasių mokymo duomenų vektorių išsibarstymas pirmąjame atpažinimo uždavinyje ir klases skiriančiosios tiesės: – po pirmos epochos, 100/500 – po 100 ir po 500 epochų, 15000 – po 15000 epochų, dvi mažai besiskiriančios tiesės F – tai Fišerio klasifikatoriaus ir VsP po specialaus mokymo 15000 epochų metu.



1.8 pav. Generalizavimo klaidos dinamika pirmąjame uždavinyje mokant su pastoviu mokymo žingsniu (juoda) ir eksponentiškai augančiu mokymo žingsniu (raudona).



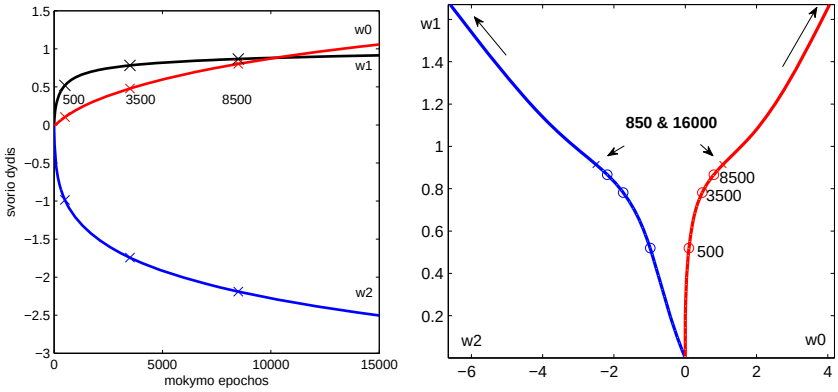
1.9 pav. Dviejų klasių mokymo duomenų vektorių išsibarstymas antrąjame atpažinimo uždavinyje ir startinės, klases skiriančiosios tiesės, gautos pirmojo VsP mokymo metu: 500 - po 500 epochų, 3500 – po 3500 epochų, 8500 – po 8500 epochų.

klases skiriančiosias tieses, $\boldsymbol{w}_{\text{start}}$, gautas pirmojo mokymo metu po 500, 3500 ir po 8500 mokymo epochų.

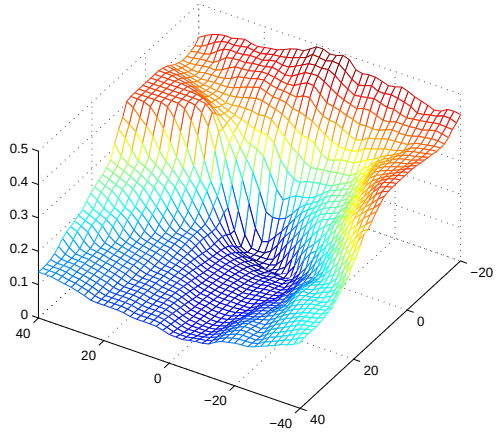
Viena iš svarbiausių savybių, trukdanti perceptronams persimokyti spręsti naują atpažinimo uždavinį yra jų svorių augimas prieš tai sekusio mokymosi metu. 1.10a pav. matome, kaip auga perceptrono svoriai, w_0, w_1 ir w_2, 15000 mokymo epochų metu naudojant duomenis, pavaizduotus 1.7 pav. 1.10b pav. matome, kaip perceptrono svorių augimas pavaizduotas dvimatėse, požymių $w_1 - w_2$ ir $w_1 - w_0$, erdvėse. Po pirmos mokymo epochos svorių dydžiai artimi nuliui, tačiau po 15000 epochų, – jie jau dideli (1.10b pav. pažymėti rodyklėmis). 1.10a pav. parodėme svorių dydžius po 500, 3500 ir po 8500 mokymo epochų, t.y. tuomet, kai jie buvo panaudoti kaip startiniai mokantis spręsti antrąjį atpažinimo uždavinį.

Kai mokymo žinsnis palaipsniui buvo didinamas ($\eta_r = 0.05 \times 1.005^r$), svoriai augo sparčiau. Per pirmąsias 850 mokymo epochų jie pakartojo 15000 epochų svorių kitimo trajektoriją, gautą mokant su pastoviu mokymo žingsniu ($\eta_r = 0.05$), o vėliau augo praktiškai tiesiškai (žr. viršutinę 10b paveikslo dalį). Po 15000 epochų mokymo žingsnis nuo 15000 epochų jau buvo išaugęs net iki $\eta_{15000} =$ 1.5484e+031, o svoriai – iki: $w_1 = 12.5$ ir $w_2 = -52.9, w_0 = 34.2$. Tiesiškas, tarpusavyje proporcingas, svorių augimas yra specifinė netiesiniais perceptronais pagrįstų klasifikatorių savybė. Vėliau pamatysime, kad ją įvaldžius perceptronų mokymo ir permokymo procesus galima valdyti. Tai labai svarbu kuriant daugiagentes sitemas, funkcionuojančias besikeičiančių aplinkų sąlygomis. Mokymo pradžioje, deja, tiesiškumo neturime ir mokymo procese daugiamatis svorių vektorius turi nemažai „pavinguriuoti‘, kol jis išeina „į finišo tiesiąsias“ (žr., dvimates ir trimates svorių vektorių kitimo diagramas 10b ir 11 paveiksluose.

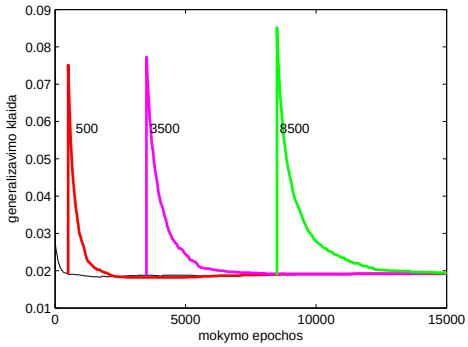
1.9 pav. turėjome dviejų klasių vektorių išsibarstymą antrąjame, jau pasikeitusiame, atpažinimo uždavinyje. 12 paveikslas rodo, kaip kinta klasifikavimo klaidų tikimybės, perceptrono mokymą tęsiant tris kartus naudojant jau antro



1.10 pav. Svorių augimo dinamika pirmąjame uždavinyje mokant su pastoviu ir augančiu (dešinio paveikslo viršutinė dalis) mokymo žingsniais.



1.11 pav. Nuostolių funkcijos (1.4)paviršius svorių w_1 ir w_0, kintančių intervaluose [-20: 1.5: 40] ir [- 40: 2: 40], erdvėje; $w_2 = -6.2$.



1.12 pav. Generalizavimo klaidos dinamika po atpažinimo uždavinio pasikeitimo mokant VsP su pastoviu mokymo žingsniu, kai mokymas buvo pradedamas nuo svorių vektoriaus gauto mokant perceptroną L epochų su pirmojo uždavinio duomenimis, $L = 500, 3500$ ir 850.

uždavinio duomenis. Kiekviename iš trijų mokymo sesijų pradiniai svorių vektoriai $\boldsymbol{w}_{\text{start 1}}, \boldsymbol{w}_{\text{start 2}}$ arba $\boldsymbol{w}_{\text{start 3}}$ buvo skirtingi. Jie buvo gauti mokantis 500, 3500 arba 8500 epochų pirmojo atpažinimo uždavinio duomenimis. 9 paveikslas rodo, kad pradinių, klases skirainčiujų tiesių padėtys, nors ir nedaug, bet skyrėsi. Mūsų analizėje svarbiausia, kad svorių vektorių $[w_1, w_2, w_0]$ dydžiai buvo skirtingi:

$$\begin{aligned}
\boldsymbol{w}_{\text{start 1}} &= [0.5190 \quad -0.9851 \quad 0.1041], \\
\boldsymbol{w}_{\text{start 2}} &= [0.7819 \quad -1.7436 \quad 0.4782], \\
\boldsymbol{w}_{\text{start 3}} &= [0.8666 \quad -2.1898 \quad 0.8025].
\end{aligned}$$

1.12 pav. rodo, kad sąlygoms pasikeitus, greičiausiai perceptronas mokosi, kai svoriai maži (pirmas variantas). Mokymasis lėčiausias, kai svoriai dideli (trečias variantas). Jeigu laikotarpis tarp uždavinių pasikeitimo didelis ir daugiagentės sistemos perceptronai „mokosi visą gyvenimą“ (angl. *life long learning*), tai jie gali persimokyti ir nebesugebėti prie pasikeitusių sąlygų prisitaikyti. Kuriant daugiagentes sistemas, numatomas darbui atpažinimo uždavinių keitimosi sąlygomis, reikia imtis specialių priemonių, kad minėto persimokymo išvengti.

Ką daryti, kad išvengti persimokymo reiškinio sukeltų problemų ir pagreitinti pakartotiną perceptrono mokymosi procesą? Skyrelyje 2.4 „Mokymosi greitis ir jį įtakojantys parametrai“ kalbėjome apie ankstyvo sustojimo reikalingumą, trokštamų išėjimų reikšmių skirtumo mažinimą, triukšmo prie mokymo vektoriau komponenčių pridėjimą, klasės indekso sąmoningą „sumelavimą“. Dar vienas iš būdų, naudojamų svorių augimui slopinti, – tai prie nuostolių funkcijos (1.4) pridėti specialų, reguliarizacijos (angl. *weight decay*) narį, $+\lambda \sum_{s=1}^{p} w_s^2$:

$$\textit{nuostoliai} = \frac{1}{n} \sum_{i=1}^{2} \sum_{j=1}^{N} (t_{ij} - f(\sum_{s=1}^{p} x_{sij} w_s + w_0))^2 + \lambda \sum_{s=1}^{p} w_s^2, \tag{1.13}$$

kur (1.13) lygtyje λ yra reguliarizacijos parametras, dažniausiai parenkamas eksperimentuojant „bandymų ir klaidų“ metodo pagalba.

Kuriant daugiaagentes sistemas reguliarizacijos nario pridėjimas techniškai lengvai įgyvendinimas. Be to, esant įtarimui, kad atpažinimo uždavinys pasikeitė, galim specialiai visus svorius proporcingi sumažinti. Iš principo, kiekvienam, svarbias funkcijas atliekančiam, agentui galime naudoti, ne vieną, o kelis, skirtingai parametrais valdomus mokymo algoritmus, kur galutinį sprendimą atlieka einamuoju laiko momentu geriausas (sėkmingiausias) algoritmas.

1.2.6. Daugiasluoksniai perceptronai

DsP architektūra.

Daugelis tyrėjų šiuo metu laikosi nuomonės, kad mąstančių organizmų smegenyse, informacijos apdorojimui, bei sprendimų priėmimui taikomi daugiasluoksniai perceptronai (DsP) bei spindulinių bazinių funkcijų (SBF) neuroniniai tinklai. Kaip ir vienasluoksnis perceptronas, daugiasluoksnis jo variantas taip pat yra gamtos inspiruota, informaciją apdorojanti sistema. Jame atskiri neuronai yra surikiuoti į sluoksnius. Įėjimo signalai $x_1, x_2, \ldots, x_p$ patenka į taip vadinamojo „paslėptojo“ sluoksnio neuronus. Paslėptųjų sluoksnių gal būti vienas arba net keli. Žemesnio sluoksnio neuronų išėjimo signalai keliauja į aukštesnįjį sluoksnį, kol galų gale pasiekia išėjimo sluoksnį. Pastoviai daugelį kartų sumuojami, netiesiškai transformuojami, signalai gali suformuluoti sudėtingas sąvokas, reikalingas tolimesniam sprendimų priėmimui pagal informaciją, esančią DsP įėjimuose. 1.6 pav. pateikiame DsP su vienu paslėptu sluoksniu schemą.

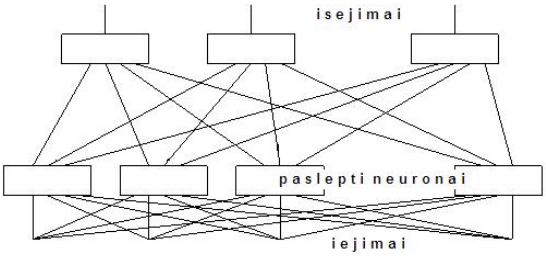
1.13 pav. pavaizduotasis perceptronas turi p įėjimų, K išėjimų. Kiekviename išėjime stovi po vieną VsP, į kurį patenka informacija iš paslėptų neuronų. Pateikus į perceptrono įėjimą vektorių $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)$, visuose paslėpto sluoksnio neuronuose suskaičiuojamos pasvertos sumos

$$s_v = \sum_{v=1}^{p} x_{v1v} + w_0, \quad j = 1, 2, ...h, \tag{1.14}$$

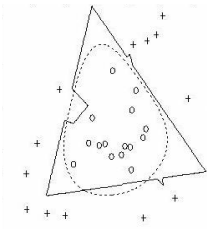
kurios paduodamos į atitinkamas aktyvacijos funkcijas, kurių išėjimai, $f(s_v)(j = 1, 2, ...h)$, tampa DsP išėjimo sluoksnio įėjimais.

Kaip ir VsP, daugiasluoksnis perceptronas gali spręsti klasifikavimo ir prognozavimo uždavinius. Lyginant su VSP, daugiasluoksniame perceptrone mokant kiekvienam p-mačiam įėjimo vektoriui $\boldsymbol{x}$ turime pateikti K-matį trokštamų išėjimų reikšmių turintį vektorių $\boldsymbol{y} = (y_1, y_2, \ldots, y_K)$. Jeigu sprendžiame prognozavimo uždavinį, tai išėjimo sluoksnyje paprastai aktyvavimo funkcija tiesinė, o vektoriaus $\boldsymbol{y}$ komponentės, – tai realūs skaičiai.

Jei sprendžiame klasifikavimo uždavinį, tai išėjimo neuronų tiek, kiek turime klasių. Jei naudojama sigmoidinė aktyvavimo funkcija (1.3), tai perceptroną mokant trokštamų išėjimų vektoriaus komponentės būna 0 arba 1, kur vienetukas



1.13 pav. DsP su vienu paslėptu sluoksniu schema.



1.14 pav. Dvi duomenų klasės ir apmokyto perceptrono skiriamas paviršius (punktyrinė linija) ir paviršius, gautas padauginus paslėpto sluoksnio neuronų svorius iš 1000.

paprastai dedamas išėjime, atitinkančiame mokymo vektoriaus klasės numerį. $\boldsymbol{y} = (y_1, y_2, \ldots, y_K)$.

1.14 pav. yra pavaizduotas jau apmokyto daugiasluoksnio perceptrono, turinčio du įėjimus, šešis paslėptus neuronus ir vieną išėjimą su netiesiniu VsP jame, skiriamas paviršius (punktyrinė linija). Kadangi paslėptajame sluoksnyje turime netiesinę, lėtai augančią, aktyvavimo funkciją, tai skiriamasis paviršius – glotni kreivė. Tačiau, jegu specialiai, dirbtiniu būdu, paslėpto sluoksnio svorius padauginsime iš 1000, kiekvieno iš paslėptų neuronų pasvertųjų sumų reikšmės padidės 1000 kartų. Tokiu atveju aktyvavimo funkcijų reikšmės priartės prie 0 arba 1. Rezultate, klasių skiriamosios ribos dalys bus sudarytos iš susikertančių tiesių (hiperplokštumų daugiamačiu atveju). Tad gausime „gabalais tiesinį“ klases skiriantįjį paviršių (ištisinė linija 1.14 pav.). Matome, kad perceptrono svoriai, bei paslėptame sluoksnyje esanti netiesinė aktyvavimo funkcija vaidina labai svarbų vaidmenį nustatant klases skiriančiojo paviršiaus formą. Šis aspektas yra labai svarbus permokant DsP, kai atpažinimo uždavinys pasikeičia. Jei primojo mokymo metu paslėptojo sluoksnio neuronai bus pernelyg išaugę, tai sekančio mokymo metu juos bus nelengva pakeisti, ypač, kad DsP-se labai žymų vaidmenį vaidina nuostolių funkcijos lokaliniai minimumai.

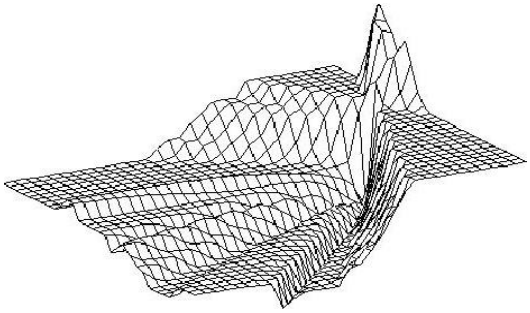
Mokymosi ir lokalinių minimumų klausimai.

Kaip minėjome, daugiasluoksnio perceptrono mokyme minimizuojama jau daug sudėtingesnė, nei (1.4) nuostolių funkcija. Sudėtingumas išauga dėl to, kad šiuo atveju, vietoj vieno išėjimo, turime K jų. Be to, daugiasluoksniame perceptrone atsiranda dar vienas, paslėptasis, sluoksnis. Todėl DsP nuostolių funkcijoje sumavimas vyksta ne tik pagal mokymo vektorius, bet ir pagal išėjimo ir paslėptojo sluoksnių neuronus. Nuostolių funkcijų išraiškos ir perceptronų mokymo taisyklės nuosekliai aprašytos profesorių Antano Veriko ir Gintauto Dzemydos paskaitų konspektuose, tad mes čia jau nebesikartosime. Paminėsim, kad pakankamai aiškus aprašas, kaip sudaryti ir mokyti perceptronus yra pateiktas *Matlab'e* (*Neural networks toolbox*). Vartotojui, pakanka būti susipažinusiam su DNT architektūra, jų naudojimo principais. Tada *Matlab'e* esantys dirbtiniai neuroniniai tinklai tampa jam nebesunkiai „įkandami".

Bene pats svarbiausias aspektas mokant daugiasluoksnį perceptroną, yra tas, kad turime minimizuoti sudėtingos formos, turinčią daugybę lokalinių (vietinių) minimumų, nuostolių funkciją. Nuostolių funkcija yra sudaryta iš daugybės „griovių", plokščiu dugnu ir stačiomis sienomis, kaip pavaizduota 1.15 pav. Grioviai išeina iš vieno centro, netoli kurio, vienam iš „griovių" dažniausiai ir būna lokalinis arba globalinis minimumas.

Mokymą paprastai pradedame nuo mažų svorių. Mokyti nuo „nulinių", svoriu, kaip rekomendavome dirbant su vienasluoksniu perceptronu, čia nebegalime. Jei taip darytume, visi paslėpti neuronai taptų vienodi, nes mokydami kiekvienam neuronui naudotumėm vieną ir tą pačią taisyklę, kiekvienam paslėptam neuronui pateiktumėm vienus ir tuos pačius duomenis. Žiūrint iš vienos pusės – tai gerai, kad pradedam nuo mažų svorių, nes šiuo atveju nesąmoningai realizuojame prielaidą, kad nuostolių funkcijos minimumas yra nedidelių svorių zonoje. Jei ši prielaida teisinga, turėtume gauti išlošimą. Žiūrint iš kitos pusės, pradėjus mokyti nuo mažų svorių, dalis paslėptųjų neuronų „pakliūna į tą pati lokalinį griovį" ir mokymo metu susivienodina su kitu, į jį panašiu, neuronu. Tą mes patyrėme mokydami su duomenimis pateiktais 1.14 pav. Tam, kad neuronai nesupanašėtų, ir gautumėm pakankamo sudėtingumo klases skiriamąjį paviršių buvome priversi generuoti svorius labai plačiame jų reikšmių intervale, didinti paslėptųjų neuronų kiekį. Dad ir šiuo atveju susiduriame su dilema, kokio dydžio turi būti pradinių svorių reikšmės, kad mokymas vyktų ir greitai, ir dalis paslėpųjų neuronų nesusivienodintų.

Kaip ir vienasluoksniame, taip ir daugiasluoksniame percptrone, mokant perceptrono svoriai kurį laiką „pasisukinėja apie centrą" ir po to, pakliuvę į griovį, jo jau laikosi, – auga į visas puses. 1.15 pav. ne dviejų, o santykinai kelių, požymių erdvėje pavaizduota, kaip auga penki jo svoriai. Aiškiai matome, kad mokymo proceso pabaigoje dalies svorių komponentės auga tiktai tiesiškai. Augimo greitis palaipsniui mažėja. Priežastis ta, kad augant svoriams auga ir pasvertosios sumos. Dėl to gradientas mažėja ir svorių „pataisymai" tampa vis mažesni ir mažesni. Mokymas sulėtėja. Ypatingai tai pavojinga, kai atpažinimo uždavi-

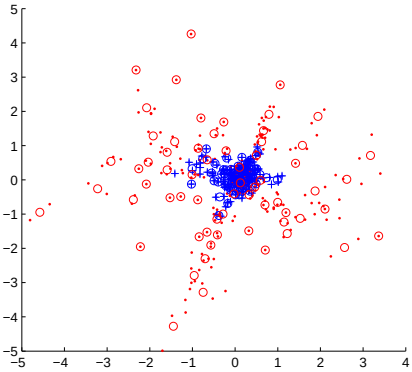


1.15 pav. Daugiasluoksnio perceptrono nuostolių funkcija.

1.16 pav. Daugiasluoksnio perceptrono svorių augimas mokymo metu.

nys pasikeičia ir esant svoriams dideliems, DsP neuronai jau nebesugeba greitai reaguoti į įvykusius pokyčius.

Kad sušvelninti lokalinių minimumų problemą, mokymas atliekamas ne vieną, o keletą kartų, kiekvieną sykį pradedant nuo skirtingų pradinių svorių. Tai, – taip vadinamas, „*multistart*“ režimas. Kadangi mokant svoriai auga, to pasekmėje gradientas mažėja. Kad užtikrinti greitesnį mokymąsi, mokymo žingsnis mokymo eigoje turėtų būti didinamas. Bet čia gali „išlįsti“ kiti pavojai, pvz. galime peršokti į kitą minimumą, Dažnai vietoj gradientą minimizuojančio mokymo algoritmo naudojami sudėtingesni, antros eiles metodai (žr. *Matlab'o* neuroninių tinklų paketą). Tada mokymas pagreitėja, bet atsiranda didesni šansai pakliūti į lokalinį minimumą. Dėl tos priežasties greitesniems, antros eilės, metodams multistart režimas reikalingas labiau. Kuriant DaS-as kiekvienas agentas turėtų turėti ne vieną, o kelias, lygiagrečiai besimokančias, mokymo schemas.



1.17 pav. Dvi elektros variklių klasės ir blokinių (klaster) analizės metu rasti centrai klasėse.

1.2.7. Mokymo vektorių kvantavimo ir spindulinių bazinių funkcijų ir neuroniniai tinklai

Mokymo vektorių kvantavimo neuroniniai tinklai.

Tiek vaizdų atpažinimo teorijoje, tiek duomenų analizėje (angl. *data mining*), tiek dirbtiniame intelekte populiarios vieno ar k-artimiausių kaimynų taisyklės. Vieno artimiausio kaimyno klasifikavimo ar prognozavimo taisyklėse sprendimai priimamai pagal klasifikuojamo (arba prognozuojamo) vektoriaus $\boldsymbol{x}$ atstumą (panašumą) iki artimiausio mokymo vektoriaus $\boldsymbol{x}_{\text{artimiausias}}$: vektorius $\boldsymbol{x}$ priskiriamas tai pačiai klasei, kuriai priklauso jo artimiausias kaimynas, $\boldsymbol{x}_{\text{artimiausias}}$. Panašiai elgiamasis ir prognozuojant.

Kai mokymo vektorių n labai daug, atpažinimo procesas yra lėtas: reikia suskaičiuoti n atstumų. Vienas iš būdų – tai mokymo vektorius sugrupuoti į kompaktiškas grupes, kuriose vektoriai išsidėstę arti vienas kito. Tegul grupių kiekis visose klasėse bus $r (r \ll n)$. Jų centrus pažymėsime $\boldsymbol{C}_1, \boldsymbol{C}_2, ..., \boldsymbol{C}_n$. Kiekvienai klasei priklausančius centrus galime surasti naudodamiesi *blokinių* (*klaster) analizės* metodus. Vektoriaus $\boldsymbol{x}$ klasifikaciją (prognozavimą taip pat) atliksime pagal artimiausio jam centro klasės numerį. Gretimus, skirtingoms klasėms priklausančius centrus, skirsime hiperplokštumomis (faktiškai Euklidinio atstumo klasifikatoriais). Tad atlikdami sprendimo priėmimo procesą jau minėtu būdu turėsime gabalais tiesinį klasifikatorių. 1.17 pav. dvimatėje specialiai klasių struktūrą išryškinančių požymių erdvėje mes pavaizdavome 200+200 gerų ir blogų elektros variklių vibracijas atspindinčių vektorių ir blokinių (klaster) analizės būdu surastus centrus, pažymėtus skrituliukais. Atkreipiame dėmesį, kad kažkurie blokinių (klasterių) centrai sudaryti tik iš vieno mokymo vektoriaus. Tuo atveju vektorius (taškas) sutampa su klasterio centru (skrituliuku). Skaitytojui paliekame pačiam nubrėžti poklases skiriančiasias tieses ir iš jų sudaryti bei popieriuje pažymėti gabalais tiesinę, dvi klases skirianiančiąją, laužtinę liniją.

Pateiktame pavyzdyje ieškant centrų dėmesys į klasių atskiriamumą nebuvo kreipiamas. Mokymo vektorių kvantavimo neuroniniai tinklai tai daro. Naudodami specialią nuostolių funkciją jie tiksliau randa centrus ir leidžia gauti geresnį klasifikatorių. Minėti centrai susikirsto daugiamatę požymių erdvę į atskiras, hiperplokštumomis aprėžtas sritis – kvantus. *Matlab'o* neuroninių tinklų pakete yra programos, leidžiančios sudaryti mokymo vektorių kvantavimo neuroninius tinklus. Kuriant mokymo vektorių kvantavimo neuroniniais tinklais paremtas DaS-mas, permokymo metu klasiu klasterių centrai galėtų būti prieš tai buvę centrai, kurie laikui bėgant yra modifikuojami. Skaitytojui, norinčiam šio tipo algoritmus taikyti DaS-mų kūrimui, reikėtų susipažinti ir su adaptyvaus rezonanso teorija (angl. *adaptive resonance theory*) paremtais atpažinimo metodais, kur laikui bėgant palaipsniui formuojami nauji neuronai (klasteriai, poklasių centrai), kai nauji, „nepanašūs į prieš tai buvusius", mokymo vektoriai patenka „į akiratį". Senieji, jau nebenaudojami, centrai palaipsniui „užmirštami". Sprendimų priėmimo metodologija primena dirbtinių imuninių sistemų (angl. *artificial immune systems*) metodus, kurie pastarųjų dviejų dešimtmečių laikotarpyje tapo labai populiarūs sprendžiant įvairius informatikos uždavinius.

Spindulinių bazinių funkcijų neuroniniai tinklai.

Žiūrėdami į 17 paveikslą matome, kad kai kurie iš blogųjų elektros motorų centrai (raudoni skrituliukai) patenka tarp tankiai išsidėsčiosių geruųjų motorų (mėlyni taškai). Mokymo vektorių neuroninis tinklas nekreipia dėmesio į vektorių kiekius kiekvienoje grupėje, jų spindulius (tūrius). Spindulinių bazinių funkcijų neuroninis tinklas tai daro. Čia kiekvieno mokymo vektoriaus indėlis į klasifikavimo procesą priklauso ne tik nuo grupes centro padėties, bet ir nuo to, kiek mokymo vektorių yra pogrupyje, koks vektoriaus $\boldsymbol{x}$ atstumas iki grupės centro $\boldsymbol{C}_r$.

Spindulinių bazinių funkcijų (SBF) neuroninio tinklo išėjimo sluoksnyje kiekvienai klasei skaičiuojama pasvertoji suma:

$$išėjimas_i(\boldsymbol{x}) = \frac{1}{N_i}\sum_{j=1}^{N_i} v_{ij}\exp(-(\boldsymbol{x}-\boldsymbol{C}_{ij})(\boldsymbol{x}-\boldsymbol{C}_{ij})^T/\,\lambda_{ij}), \tag{1.15}$$

kur N_i yra centrų kiekis i-tosios klasės j-tojoje grupėje, o parametrai $\boldsymbol{C}_{ij}, \lambda_{ij}$, ir v_{ij} su indeksas „ij" atitinkamai charkterizuoja ij-tosios grupės „centrą", „spindulį" ir „apriorinę tikimybę".

Norint rasti nežinomas parametrų $\boldsymbol{C}_{ij}, \lambda_{ij}$, ir v_{ij} $(i = 1, 2, ...K;\ \ j = 1, 2, ..., N_i)$ reikšmes, spindulinių bazinių funkcijų neuroninis tinklas mokomas iteraciniu būdu minimizuojant pasirinktą nuostolių funkciją. Pradinės nežinimųjų parametrų reikšmės randamos blokinių (klaster) analizės būdu. Blokinių (klasterių) centrai charakterizuoja pradines $\boldsymbol{C}_{ij}$ reikšmes, vektorių kiekis grupėje – parametrą v_{ij}, o mokymo vektorių grupėje vidutinis nukrypimas nuo „centro" $\boldsymbol{C}_{ij}$, – parametrą λ_{ij}. *Matlab'o* neuroninių tinklų paketas turi tam reikalingą programinę

įrangą. Iš vienos pusės spindulinių bazinių funkcijų neuroninis tinklas panašus statistinį Parzeno lango klasifikatorių, kadangi skaičiuoja „branduolio funkciją“ $\exp(-atstumas(\boldsymbol{x}, \boldsymbol{C}_{ij})$. Kadangi SBF neuroniniame tinkle centrų mažiau nei mokymo vektorių, tai atpažinimo fazėje SBF daug greitesnis, nei Parzeno lango klasifikatorius, tačiau SBF reikalauja daug daugiau kompiuterinių resursų parametrams išmokti. Už viską reika mokėti. Mašinų mokymo (*machine learning*) ir dirbtinių neuroninių tinklų mokslinėse bendruomenėse yra netgi toks sparnuotas posakis „*there is no free lunch*“ (žr. internetą). Yra netgi teorema, teigianti, kad neturėdamas informacijos, manipuliacijomis jos neišgausi. Lygindami SBF ir mokymo vektoriaus kvantavimo neuroninius tinklus matome, kad abu naudoja centrus $\boldsymbol{C}_{ij}$, tačiau SBF tinklai yra „subtilesni“ – papildomai vertina parametrus λ_{ij} ir v_{ij}, bei vektoriaus $\boldsymbol{x}$ atstumą nuo centro $\boldsymbol{C}_{ij}$, taip pat kitų, gretimų, grupių įtaką. Todėl jis lėtesnis. Taikant adaptyvaus rezonanso teoriją naujo tipo centrams išmokti, o seniems – pamiršti SBF neuroninius tinklus sėkmingai galima naudoti kuriant DaS-mas, funkcionuojančias pastoviai vykstančiame apliknos kitime.

Priminsime, kad kaip ir perceptronai, taip ir spindulinių bazinių funkcijų neuroniniai tinklai yra paremti Gamtoje vykstančio informacijos proceso apdorojimu. Žmogus, saugodamas atmintyje daugybę atsiminimų, pamažu juos sugrupuoja, pamiršdamas netipinius, jam atrodančius nesvarbiais, dalykus. Tai dažniausia vyksta naktį, sapnuojant. Atmintyje lieka tik tipiniai dalykai, sakykime, „centrai“, kaip ir SBF neuroniniuose tinkluose. Tokiu būtu saugomos informacijos panaudojimas, jos „ištraukimas iš atminties“ tampa greitesniu. Žmogaus atmintyje kaupiamos informacijos sistematizavimo procesas dar nėra iki galo aiškus, tačiau daugelis mokslininkų linkę galvoti, kad jis nemaža dalimi primena SBF neuroniniuos tinkluose vykdomus informacijos apdorojimo procesus.

1.3. Adaptyvios daugiaagentės sistemos

1.3.1. Genetiniai mokymo algoritmai

Tradiciniais pirmos ar antros eilės optimizavimo metodais paremtų daugiasluoksnių perceptronų, radialinių bazinių funkcijų, mokymo vektorių kvantavimo neuroninių tinklų mokymas labai jautrus lokalinių minimumų buvimui. Genetiniai optimizavimo algoritmai – tai patobulinti atsitiktinės paieškos algoritmai. Savo veikimo principą jie „pasiskolino“ iš Gamtos. Lyginant su pirmos ar antros eilės optimizavimo metodais genetiniai algoritmai daug lėtesni, tačiau jie sėkmingiau sprendžia lokalinių minimumų problemas. Augant kompiuterių skaičiavimo greičiams ir atminčiai, jie tampa vis populiaresniais. Mes juos naudosime adaptyvių daugiaagenčių sistemų mokyme, kur genetinės evoliucijos pagalba bus nustatomi pereceptronų mokymą apibrėžiantys parametrai. Šiame skyrelyje paaiškinsime genetinių algoritmų veikimo principą.

Mokant dirbtinius neuroninius tinklus reikia nustatyti jų svorių reikšmes. Akivaizdu, kad šių svorių reikšmes galime užrašyti naudodami ne tik dešimtainę, bet

ir dvejetainę sistemas. Bet kuris iš svorių, pvz. w_r-tasis, gali būti užkoduotas vienetukų ir nuliukų seka. Pavyzdys – geriausias mokytojas. Štai sekos, užkoduojančios kelis svorius, pavyzdys:

1001000101101000010111001101000111100010101010

Šią seką suskirstykime į dalis

1001000 101101 00001 011100 1101000 11110 0010 101010

Sekdami genetikos terminais visą seką pavadinkime *genetiniu kodu*, o atskiras to kodo dalis, - *genomomis*. Viena ar dvi genomos galėtų aprašyti vieną dirbtinio neuroninio tinklo svorį arba jo mokymosi stiliaus parametrą, pavyzdžiui, mokymo greitį, η, arba skirtumą tarp trokštamų išėjimų, $t_2 - t_1$. Genetiniame mokymo algoritme, kaip ir paprasčiausiame atsitiktinės paieškos optimizavimo metode, pradžioje atsitiktinai sugeneruojama aibė, sakykim, n_{gen}, genetinių kodų. Turint mokymo duomenis, jiems paskaičiuojama nuostolių funkcija, kuri galėtų būti kad ir standartinė nuostolių funkcija (sakykim (1.4) ar (1.13)). Genetinių algoritmų teorijoje, ši, minimizuojamoji (arba maksimizuojamoji) funkcija vadinamas tvirtumo (angl. *fitness*) vardu. Surikiuokime tvirtumo funkcijos reikšmes pagal jų dydį ir iš jų išrinkime n_{ger} geriausiųjų. Tai genetinių kodų grupė, kuriems bus leista generuoti naujus genetinius kodus-vaikus. Iš dviejų atsitiktinai iš geriausiųjų grupės parinktų kodų genomų sudarome vieną ar kelis naujus genetinius kodus. Žemiau pateikti du skirtingi genetiniai kodai, priklausantieji „geriausiems“:

1001000 101101 00100 011101 1101000 11000 1010 110010
1001011 100000 00101 011111 1101000 11110 1010 110101

Sekančiose dvejose eilutėse du, iš ankstesniųjų kodo gabaliukų atsitiktinai suformuoti, kodai:

1001000 100000 00100 011111 1101000 11000 1010 110101
1001011 100000 00101 011101 1101000 11110 1010 110010

Ši operacija vadinama „kryžminimu“ (angl. *cross-over*). Sekant Gamta, dar įvedama mutacija, kuri palengvina išbristi iš blogų lokalinių ekstremumų (minimumų mūsų nagrinėjamu atveju). Žemiau pateikiamame pavyzdyje mutacijos pažymėtos mėlynai.

1001001 100000 00100 011011 1101000 11000 1010 110101
1001011 100100 00101 011101 1101010 11110 1010 110010

Tokiu būdu sudaroma nauja genetinių kodų grupė, kurioje, jau antrame mokymosi proceso cikle, iš naujo vertinamos tvirtumo funkcijos reikšmės, atrenkami labiausiai vykę genetiniai kodai, vykdomos kryžminimo operacijos, generuojamos atsitiktinės mutacijos. Kartojant šias procedūras daugelį kartų mokymosi procesas tęsiasi. Jo metu tvirtumo funkcijos reikšmės vis auga, svoriai vis tikslinami.

Jei mokymosi algoritmo parametrai, $n_{\text{gen}}, n_{\text{ger}}$, mutacijų dažnis ir kiti, nustatomi neteisingai, tobulėjimo (mokymosi) procesas gali sustoti ir anksčiau.

Mes aprašėme paprasčiausią variantą. Šį algoritmą tobulinant galėtume: į parametrų, kuriuos reiktų išmokti įtraukti ir $n_{\text{gen}}, n_{\text{ger}}$, mutacijų dažnį, suskirstyti genetinius kodus į grupes, kuriose kryžminimo operacija vyksta tik tarp toje grupėje esančių genetinių kodų. Retais atvejais kryžminimo operacija būtų leidžiama ir tarp genetinių kodų, esančių skirtingose grupėse. Tai taip pat Gamtos evoliucijos pasiūlyta idėja. Kai kuriose genetinių algoritmų modifikacijose vietoj vienintelio tvirtumo kriterijaus, naudojama keletas jų. Šioje modifikacijoje, tam, kad pakliūti į geriausiųjų grupę, genetinis kodas turi patenkinti ne vieną, o keletą ar net visus kriterijus. Pavyzdžiui, sprendžiant daugiaagenčių sistemų sudarymo klausimus kriterijais galėtų būti:

- klasifikavimo klaidos tikimybė,
- mokymosi laikas,
- atpažinimo laikas,
- kompiuterio atmintis, reikalinga duomenims, tarpiniams rezultatams saugoti.

Pastarojo dešimtmečio laikotarpyje atsirado kombinuotieji mokymo metodai (angl. *memetic algorithms*), kur mokymo procese kartu dalyvauja ir genetiniai ir gradientiniai arba antros eilės optimizavimo algoritmai. Po kiekvieno genetinio mokymo ciklo seka adaptacinis, kur tvirtumo funkcijos skaičiuojamos tik po įvykusio adaptacinio mokymosi. Paveldimos tik genetiniame mokymesi įgytos svorių reikšmės. Tai irgi Gamos evoliucijos pasiūlyta idėja.

Besimokančios daugiagentės sistemos priklauso nuo visos eilės mokymo procesą aprašančių parametrų. Anksčiau minėjome, kad tai ir mokymo žingsnis, ir skirtumai tarp trokštamų išėjimų, $\Delta t = t_2 - t_1$, triukšmo lygis, pridedamas prie atsitiktinai parinktų mokymo vektorių kiekvienos epochos metu, mokymo vektorių dalis, α, kuriems tas triukšmas bus uždedamas, ir t.t. Mineti parametrai įtakoja agentų mokymosi greitį, o taip pat konkretizuoja klasifikavimo (arba prognozavimo) metodus, įtakoja jų darbo tiklsumą. Nežinant aplinkos pasikeitimų pobūdžio, ką tik minėtus parametrus tiksliai nustatyti neįmanoma. Parametrus galima išmokti naudojant genetinius mokymo algoritmus, apie ką sekančiuose skyreliuose ir kalbėsime.

1.3.2. Daugiaagentės sistemos, jų architektūra ir jų mokymosi ypatumai

Daugiaagentė sistema veiklai besikeičiančioje aplinkoje

Veikiamas netiesinių, chaotiškų, Gamtoje, bei žmogaus ūkinės veiklos iššauktų procesų, pasaulis nuolat, kasdien, kas metus keičiasi. Tas tapo pastebima naudojant naujas technologijas, juolab, kad žymia dalimi jos minėtą aplinkos kaitą ir įtakoja. Tapo būtina tuos kitimus analizuoti, prognozuoti jų tendencijas. Kintant

aplinkai, kinta ir sprendžiami uždaviniai, kinta matuojamos charakteristikos bei jas aprašantys požymiai. Dėl šios priežasties duomenys tampa nebehomogeniški. Tad jų kiekio didinimas, paremtas ilgesniu stebėjimo laiku, veda prie to, kad naudojant standartinius metodus stebimi dėsningumai gali būti prarasti. Pats kitimo procesas vyksta netolygiai. Pažymėtina, kad skirtingos charakteristikos gali kisti įvairiu greičiu. Tad tampa neįmanoma nustatyti optimalų daugiamačių stebėjimų surinkimo laiko intervalą.

Esant tokiom aplinkybėm, vienu metu galime kurti keletą sprendimo priėmimų algoritmų, besiskiriančių ne tik stebėjimo laikotarpiu, bet ir požymiais, pagal kuriuos atliekama tiriamų reiškinių prognozė arba klasifikavimas. Kalbant dirbtinio intelekto terminais, *kiekvieną adaptyvią*, mokomą pagal riboto kiekio mokymo duomenų kiekį, klasifikavimo ar prognozavimo *sistemą galima būtų laikyti adaptyviu intelektualiu agentu.* Šiuo principu veikiančią duomenų analizes sistemą galima būtų vadinti daugiaagente sistema. Juk pagal neformalų apibrėžimą, daugiaagentėmis vadinamos iš daugelio intelektualių agentų sudarytos sistemos, kurios *veikia kartu, siekdamos bendro tikslo.*

Šiame skyrelyje aprašysime vieną iš galimų, pačios Gamtos evoliucijos proceso pasiūlyto, daugiaagentę perceptrono mokymosi stiliaus parametrus nustatančią sistemą. Kadangi aplinka keičiasi, tai nebegalime naudoti optimalumo kriterijų, kuriuos naudojome aukščiau aprašytose statinėse (nekintančiose) klasifikavimo bei prognozavimo sistemose. Gamtoje veikia tik išlikimo kriterijus. Mes jį ir naudosime. Šiame skyrelyje apsiribosime klasifikavimo uždavinio analize.

Tegul minėto tipo daugiaagentę sistemą (DaS) sudaro maksimaliai L vieno tipo adaptyvių intelektualių agentų, besimokančių spręsti klasifikavimo į dvi kategorijas (klases) uždavinius. Žemiau aprašomoms iliustracijoms iš daugelio žinomų, klasifikavimo uždaviniui spręsti skirtų, metodų pasirinkome vienasluoksnį perceptroną. Pasirinkimo argumentai sekantys:

a) VsP yra adaptyvus algoritmas,
b) jis turi daugelį universalumo savybių,
c) atlikdamas klasifikaciją netiesiškai transformuotoje požymių erdvėje jis gali spręsti ir labai sudėtingus netiesinius atpažinimo uždavinius.

Tai, kad agentų kiekis, L, apribotas „iš viršaus“, simbolizuoja, kad kurdami šią daugiaagentę atpažinimo sistemą darome prielaidą, kad Gamtoje turimi resursai yra riboti.

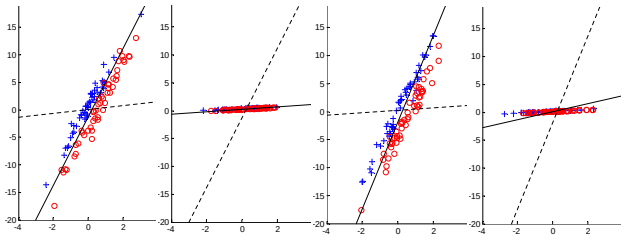
Anksčiau kalbėjome apie daugiaagenčių sistemų tipus, jų sprenžiamus uždavinius. Šiame skyriuje mes postuluosime, kad sistema veikia staigių, netikėtų aplinkos pasikeitimų sąlygomis. Kažkuriuo laiko momentu aplinkos sąlygos nelabai arba netgi žymiai pakinta ir kiekvienas iš DaS-mą sudarančių agentų, įskaitant ir sistemos darbą koordinuojančius agentus, turi persimokyti spręsti jiems keliamas funkcijas. Be to sistema turi veikti nenustrūkstamai. Tad laiko tarpas, skirtas DaS pesimokyti, yra „nulinis“: sistema ir toliau turi vykdyti savo funkcijas, bet siekdama savo darbą tobulinti privalo savo svorius, koeficientus, apibrėžiančius kiekvieno agento funkcijas, palaipsniui adaptuoti prie įvykusių aplinkos pokyčių.

Prileidžiame, kad uždaviniams pasikeitus kiekvienas iš agentų gauna (tobulesnės sitemos atveju, agentas pats juos „susiranda“) jo uždavinį nusakančius naujus mokymo vektorius, kiekvieno iš jų klases nusakančius indeksus, ir pradėdamas nuo prieš tai buvusių perceptrono svorių mokymąsi tęsia ir toliau. Tam, kad persimokymo procesas vyktų „neskausmingai“, jis būtų greitas, kiekvienas iš agentų turi naudoti „optimalias“ jo mokymosi procesą valdančių parametrų reikšmes. Kiekvienam agentui indivialiai nusistayti optimalias mokymo stilių aprašančių parametrų reikšmes neįmanoma, nes tam jis neturi pakankamo informacijos kiekio. Šis uždavinys tampa įmanomu, jei klausimas yra sprendžiamos visos DaS-os (ar kelių panašių sistemų), funkcionuojančios labai ilgą laiko tarpą, daugybės aplinkos pokyčių skalėje. Ilgai sebėdama aplinkos kaitą, mokymo stiliaus parametrus valdanti sistema jau gali atlikti kažkokius tai apibendrinimus, liečiančius aplinkos pokyčių charakterį, jų stiprumą, dažnį.

Kad supaprastinti aiškinimą darme prielaidą, kad DaS-je turime specialią posistemę, kurios tiklas – valdyti pagrindinių agentų mokymo stilių nusakančius parametrus. Posistemei sudaryti darome dar vieną prielaidą, kad agentai ir jų sprenžiami uždaviniai šiek tiek panašūs ir visiems posistemės agentams pakinta vienu metu. Nesėkmingas agentas gali būti pakeičiams kitu tos pačios posistemės agentu. Tad bendra agentų savybė – panašūs jų mokymosi ir persimokymo stilių nusakantys parametrai, tokie, kaip skirtumas tarp trokštamų išėjimų, $\Delta t_s = t_2 - t_1$, vadinamas stimuliavimu, mokymo vektorių dalis, α_s kuriems kiekvienos mokymo epochos metu atliekamas trokštamų išėjimų sukeitimas (toliau vadinamas korupcijos lygiu), mokymo žingsnis, $\eta_s, (s = 1, 2, ..., L)$, ir t.t. Be to dar darome prielaidą, kad mokymosi stiliaus parametrai kiekvienam agentui yra individualaūs ir evolicijos metu genetiškai perduodami agentams-vaikams.

Labai svarbus aspektas optimaliems (ar bent esantiems arti optimalių) mokymo stilių nustatančių parametrų reikšmėms surasti yra ilga aplinkos pasikeitimų seka. Nepaisant to, kad nei pasikeitimų laiko, nei jų tipo ar stiprumo iš anksto numatyti neįmanoma, išanalizavus gan ilgą aplinkos pokyčių seką kažkokią naudingą informacija išgauti galima. Šiame skyriuje nagrinėsime kaip ją išgauti ir kaip ją panaudoti DaS-mos mokymosi efektyvumui pagerinti.

Praktiškai aplinkos pokyčiai vyksta chaotiškai. Siekdami nustatyti ir skaitytojui išaiškinti galimybę „ištraukti“ iš ilgos pasikeitimų sekos DaS-ai darbui pagerinti reikalingą informaciją, analizuosime paprastesnes determinuotas aplinkos pokyčių sekas. 2.5 skyrelyje, kalbėdami apie vienasluoksnio perceptrono mokymąsi atpažinimo uždaviniui pasikeitus, nagrinėjome du dvejų Gausinių klasių (daugiamačių normaliųjų populicijų) atskyrimo uždavinius. Žemiau analizuosime labai ilgas besikeičiančių uždavinių serijas. Siekdami atsiriboti nuo „pašalinių“ veiksnių poveikio ir susikoncentruoti tik ties mokymo stiliaus parametrų išmokimu, mūsų pateikiamuose pavyzdžiuose klasifikavimo uždavinio analizę „palengvinsime“: laikysime, kad priešingų klasių kovariacinės matricos vienodos ir beveik nekinta ilgoje atpažinimo uždavinių sekoje. Kas kiekvienas 100 mokymo epochų aplinką pakeičiame, t.y. modelyje klasifikavimo uždavinys pakeičiamas nauju. 1.18 pav. matome keturių uždavinių seką.



1.18 pav. Vieno iš agentų sprendžiamų atpažinimo uždavinių sekos dalis. Uždaviniui pakitus percptronas mokosi toliau naudodamas seną, prieš tai buvusį, svorių vektorių (punktyrinės linijos).

Iš paveikslėlio matome, kad nepaisant neinformatyvių pradinių svorių vektorių, pirmuosius tris atpažinimo uždavinius agentas išmoko spręsti teisingai, tačiau mokantis ketvirtąjį, – jis to padaryti nebesuspėjo. Kaip minėjome, atpažinimo uždavinyje mokymo procesą apsunkina ne tik nevykusi pradinės skiriančiosios plokštumos padėtis, bet ir perceptrono svorių dydžiai.

DaS posistemio, skirto altruizmo parametrui išmokti, architektūra

Agentų, paremtų dirbtiniais neuroniniais tinklais, mokymosi greitį įtakoja teisingai nustatyti mokymosi stiliaus parametrai. Mokymosi stilius – tai ir yra tai, kuo adaptyvūs intelektualūs agentai tarpusavyje skiriasi. Štai keli iš daugelio pavyzdžių:

a) mokymo duomenų sekos ilgiu, ($s-$tasis agentas mokomas n_s ilgio, daugiamačių vektorių seka, surinkta pačio paskutinio laikotarpio metu),

b) požymiais, naudojamais klasifikavimui (tiek jų dimensiškumu, tiek jų „sąstatu“),

c) reguliarizavimo (korupcijos lygio) parametru (tam, kad agentas greičiau mokytųsi, t.y. kad jo svoriai neaugtų pernelyg greitai, ir agentas galėtų sparčiau prisitaikyti prie pasikeitusių aplinkos sąlygų, α_s atsitiktinai kiekvienoje mokymo iteracijoje parinktoje mokymo vektorių dalyje trokštamų išėjimų reikšmės būdavo sukeistos),

d) skirtumu tarp trokštamų išėjimų, $\Delta t_s = t_{2\mathrm{s}} - t_{1\mathrm{s}}$, kuris taip pat įtakoja perceptrono svorių augimą ir jo persimokymo greitį.

Kad šiuos ar panašius parametrus „išmokti“ ilgos aplinkos pasikeitimų sekos metu, konstruojame specializuotą DaS posistemę, kur agentas per $iter_{\max}$ mokymo epochų, privalo išmokti spręsti naują klasifikavimo uždavinį darydamas mažiau nei $P_{\text{leistinas}}$ klasifikavimo klaidų. Vienas iš pagrindinių reikalavimų – mokytis reikia greitai.

Tegul DaS posistemė maitinama realiais, DaS funkcionavimo metu pateikiamais, uždaviniais. Darome prielaidą, kad kiekvienas iš agentų sprendžia panašius

ir panašiai kintančius atpažinimo uždavinius. Pasikeitus uždaviniui kiekvienas iš agentų gauna savo mokymo duomenis ir mokosi toliau. Kadangi uždavinys pasikeitė, klasifikavimo klaida po pasikeitimo staigiai išauga. 1.12 pav. tai mes jau matėme. Jame atpažinimo uždavinys (dvimačių duomenų išsibarstymo pasisukimo kampą nustatantis parametras, θ) pasikeitė nežymiai. Dėl to klasifikavimo klaidos šuolis nebuvo didelis. Kintant atpažinimo uždaviniams pagal 1.18 pav. pavaizduotą schemą, klasifikavimo klaida šokteli jau beveik iki 50%, tačiau per 100 mokymo epochų, tris kartus iš keturių, klasifikavimo klaidų dažnis spėjo sumažėti beveik iki nulio. Jeigu agentas laiku nespėjo savo klasifikavimo klaidos dažnio sumažinti iki $P_{\text{leistinas}}$, iš populiacijos jis pašalinamas. Kad populiacija (daugiaagentė sistema) nežūtų ilgą laiką vykstančio aplinkos pokyčių (atpažinimo uždavinių kaitaliojimo) metu, žuvus agentui, jis pakeičiamas nauju, sprendžiančiu tą patį uždavinį, kaip ir „žuvęs“ agentas, tačiau naujo agento mokymosi stiliaus parametrai jau kiti. Pradedamas mokytis nuo „nulinių“ svorių naujas agentas iš atsitiktinai parinkto „gan sėkmingo“ agento paveldi pastarojo „mokymosi stilių“. Priedo, paveldimi parametrai mutuoja. Naujo agento (vaiko) leistinas mokymosi laikas pailginamas iki $\beta_{\text{vaik}} \times iter_{\max}$ iteracijų ($\beta_{\text{vaik}} > 1$). Tokiu būdu, ilgainiui populiaciją sudaro tik tie agentai, arba jų palikuonys, kurie praeityje mokėsi pakankamai greitai.

Jei atpažinimo uždavinio pokyčiai nėra labai dideli, gan dažna situacija kai visi agentai mokosi sėkmingai, įvykdo jiems keliamus reikalavimus ir nežūna. Tokiu atveju evoliucinis mokymasis sustoja. Jei atpažinimo uždaviniai kinta pakankamai smarkiai, dažniausiai tik dalis iš agentų mokosi sėkmingai. Šio tipo situacija pavaizduota 1.19 pav. Viršuje (a) pateikta dviejų klasių, apie kurias kalbėta komentuojant 1.18 pav., vektorių pasukimo kampą charkterizuojančio parametro, θ, raida pirmųjų 35 atpažinimo uždavinio kitimų metu. Pradžioj, kol kitimo procesas nusitovės, „pasisukimo parametras“ θ syravo nedaug, paskiau svyravimas (tarp reikšmių $\theta_{\min} = 0.1887$ ir $\theta_{\max} = 6.3$) pamažu didėjo ir nekito 5000 kitimų serijos metu. 1.19b pav. pavaizduota, kaip pokyčių serijos pradžioje kito bendras agentų (mėlyna) ir agentų-vaikų (b) kiekis. Matome, kad kai pokyčiai tampa dideli, prasideda laikotarpis, kai dauguma agentų žūna. Reiškia, tuo metu mokymo stiliaus parametrų evoliucija vyksta smarkiai. Iš to, kas pasakyta, seka, kad *agentų žūtis yra būtina sąlyga evoliucijai vykti ir mokymo parametrams rasti.* Šia prasme DaS darosi panaši į skruzdėlių kolonijų ir spiečių sistemas (angl. *ant colony and swarm systems*), pastaraisiais metais vis populiarėjančių daugiaagenčių sistemų atmainą. Įvertinant tai, kad visą laiką dalis agentų bus „žuvę“, kuriant DaS posistemę mokymo stiliaus parametrams rasti, agentų joje turi būti daugiau, nei realioje daugiaagentėje sistemoje.

Jei aplinkos pasikeitimai ypač stiprūs ir reti, tada perceptronų svoriai pernelyg išauga ir persimokymo uždavinys pasunkėja. Atskirais momentais visi populiacijoje esantys agentai gali nebespėti išmokti naują atpažinimo uždavinį ir dėl to „žūti“. Tokiais atvejais žūna ir agentų populiacija, o perceptrono mokymo stilių aprašančių parametrų nustatymo uždavinys žlunga. Tad sudėtingais atvejais privalu kurti tobulesnę DaS posistemės struktūrą (architektūrą), kurioje:

a) agentų kiekį populiacijosje reikia didinti (skaitlingesnėje populiacijoje, didesnis agentų kiekis išlaiko testą $P_s < P_{\text{leistinas}}$, todėl didenė populiacija tampa ir įvairesne,ir „tvirtesne“),

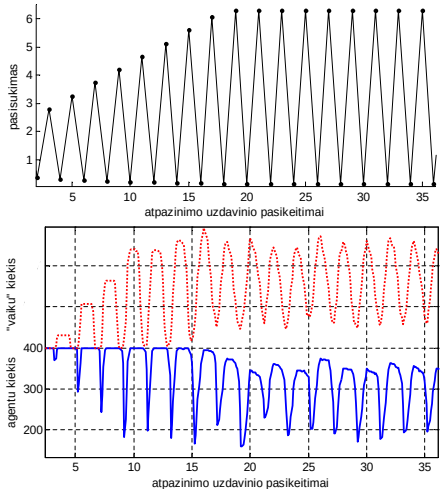
b) pačius agentus susikrstyti į grupes, kur kiekvienos grupės viduje agentai padeda vieni kitiems, o išimtiniais atvejais ir žūstančiųjų agentų grupei.

Kiekvienos grupes sėkmingas, normą $P_{\text{leistinas}}$ įvykdęs, s-tasis agentas dalį resursų kaupia sau („sunkesniems laikams“, kai aplinkos pasikeitimai ypač dideli), o kitą resursų dalį, κ_s, (resursais vadinisme klasifikavimo klaidų dažnio skirtumą tarp ribos $P_{\text{leistinas}}$ ir jo faktinio klasifikavimo klaidų dažnio, P_s) gali „paaukoti bendram grupės labui“ – t.y. tam, kad padėtų išlikti nesėkmingiesiems agentams. Paprastai padedama tik agentams, kuriems „mažiausiai trūksta“ iki normos, $P_{\text{leistinas}}$. Jei DaS posistemėje viena grupė labai silpna ir joje lieka tik vienas ar du sėkmingi agentai, kurie „gali daryti“ agentus-vaikus, pati sėkmingiausioji, tvirčiausioji, skaitlingiausioji grupė perduoda savo „genetinį kodą“ bežūstančios grupės agentams-vaikams ir tokiu būdu neleidžia tai grupei „pradingti“. Faktiškai ši pagalba susiveda į panašios į geriausiąją naujos grupės susidarymą. Kadangi mūsų nagrinėjamoje DaS posistemėje kiekvienas agentas nedidelę savo resusrsų dalį kaupia „sunkiems laikams“, tai šiuo būdu įvedama inercija, sumažinanti tikimybę tvirtiems, greitai besimokantiems, agentams atsitiktinai žūti ypač didelių aplinkos pasikeitimų metu. Aprašytoji agentų populiacijos architektūra padeda išsilaikyti grupinei DaS mokymo paramtrų įvertinimo posistemės struktūrai ir sėkmingai funkcionuoti didesnių aplinkos pasikeitimų sąlygomis. 1.19 pav. pateiktasis agentų ir agentų-vaikų dinamikos grafikas ir atitinka grupinę struktūrą, kur 400 agentų buvo suskirstyta į 16 grupių, po 25 agentus kiekvienoje iš jų.

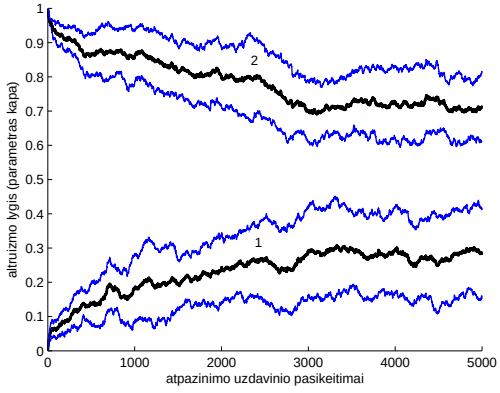
Aprašytoji sistema buvo skirta „altruizmo“ parametrų, κ_s, įtakojančių DaS virtualaus posistemio veiklą, reikšmėms išmokti. 1.20 pav. matome, kaip kito parametro κ_s vidurkis $\bar{\kappa}$ ir $\bar{\kappa}\pm$ *standartinis nuokrypis* 5000 atpažinimo uždavinių kitimo metu dviem atvejais:

1) pradinės parametrų $\kappa_s(s = 1, 2, ..., 400)$ reikšmės buvo išsibarsčiusos intervale [0 0.03] ir evoliucijos eigoje išaugo iki 0.3,
2) pradinės κ_s reikšmės buvo išsibarsčiusos intervale [0.97 1.00] ir evoliucijos eigoje sumažėjo iki 0.7.

Aprašytoji posistemė leidžia išmokti DaS-mą stiprinantį parametrą, κ_s. Be to, evoliucijos eigoje išsivysto šių reikšmių diversikacija (įvairovė), padedanti agentų populiacijai evoliucionuoti greičiau. Minėto tipo posistemė gali būti panaudota, kad išmokti ir kitus, DaS perceptronų mokymo greitį įtakojančius, parametrus. Tai – a) mutacijų lygis, b) sąlygos (reikalavimai), kai agentas žūna ir to rezultate daromas agentas-vaikas, c) kada sėkminga grupė padeda silpniausioms grupėms, ir dar daugelis kitų, kolektyvinę DaS agentų veiklą reguliuojančių, parametrų.



1.19 pav. a) duomenų pasukimo kampą charkterizuojančio parametro, θ, kitimas 35 atpažinimo uždavinio pokyčių metu; b) - bendro agentų (mėlyna) ir agentų-vaikų kiekio evoliucija (raida) pirmųjų aplinkos pokyčių laikotarpiu.



1.20 pav. Altruizmo parametro κ evoliucija 5000 atpažinimo uždavinių pokyčių metu.

Stimuliavimo reikšmės nustatymas evoliucinio algoritmo pagalba

Pirmoji aptariamos DaS posistemės funkcija – tai pagrindinių (vykdačiųjų) DaS agentų – perceptronų mokymo stiliaus parametrų valdymas. Kalbėdmi apie perceptronus minėjome, kad vienas iš pagrindinių, jų svorių dydį apsprendžiančių, parametrų yra skirtumas tarp trokštamų išėjimų, $\Delta t = t_2 - t_1$. Žemiau pademonstruosime, kad grupinės architektūros parametrų įvertinimo posistemė sugeba parametrą Δt įvertinti, kai aplinkos *pokyčių stiprumas kinta* laiko bėgyje. Tam pasirinksime jau nagrinėtą aplinkos kitimų modelį, kai vykstant eiliniam aplinkos pokyčiui abi klasės pasisuka prieš laikrodžio rodyklę arba atgal (žr. 18 paveikslą), o pasisukimo laipsnį nusakantis parametras θ šokinėja tarp $\theta_{\max}$ ir $1/\theta_{\max}$, kur $\theta_{\max}$ nekinta arba lėtai varijuoja tarp reikšmių $\theta_{\max 1} = 3.5$ ir $\theta_{\max 2} = 5.4$, kaip parodyta 1.21a pav.

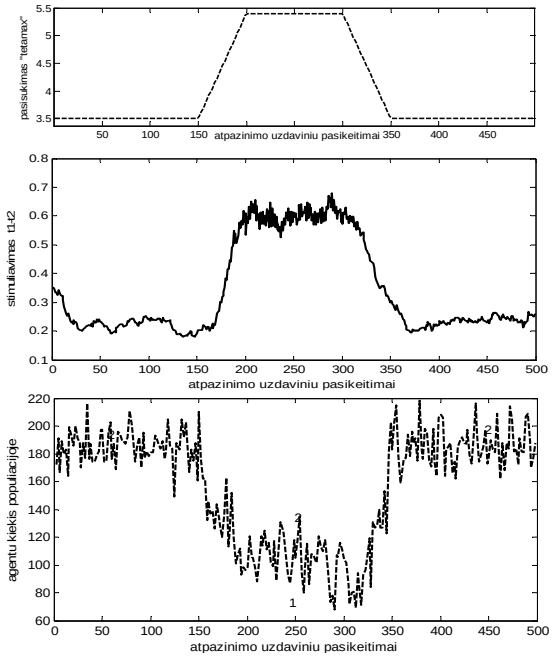
1.21b pav. pateikiame eksperimento, atlikto su iš 400, į 8 grupes susikrstytų, agentų DaS „optimalios“ stimuliavimo reikšmės įvertinimo rezultatus. Matome, kad Δt reikšmė beveik atkartoja atpažinimo uždavinių kitimo stiprumą (21a paveikslas). Iš to seka, kad, optimalios trokštamų išėjimo reikšmės priklauso nuo sprendžiamų atpažinimo uždavinių kitimo. 21c paveikslas rodo, kad agentų kiekis DaS posistemėje taip pat priklauso nuo atpažinimo uždavinių pokyčių stiprumo: esant didžiausiems pokyčiams, agentų kiekis populiacijoje ženkliai sumažėja. Pačiu sunkiausiu laikotarpiu jų kiekis nukrito iki 70-ties. Jei atpažinimo uždavinių pokyčių stiprumas išaugtų dar labiau, žūtų visi agentai. Tai reikštų, kad netgi daugiagrupinė DaS struktūra neužtikrina išgyvenamumo ypač didelių aplinkos pokyčių laikotarpyje. Sekančiame skyrelyje matysime, kad Gamta yra radusi dar vieną būdą, kaip paspartinti agentų populiacijos mokymąsi.

Kompiuterinių emocijų panaudojimas stimuliavimo ir neteisingai nustatytų mokymo direktyvų daliai išmokti

Pataruųjų metų laikotarpyje tyrėjai, dirbantys dirbtinio intelekto srityje, daug dėmesio pradėjo skirti „kompiuterinėms emocijoms“. Psichologai, kur emocijos yra viena iš pagrindinių tyrimo sričių, nesutaria dėl emocijų apibrėžimo. Kiekviena tyrėjų srovė jas interpretuoja vis savaip. Tad terminas „kompiuterinės emocijos“ nėra nusistovėjęs. Tačiau daugelis psichologų ir informatikos specialistų sutaria, kad emocijos padeda mokymuisi. Šiame skyrelyje naudosim vieną iš daugelio galimų emocijų „apibrėžimų“, kuris taipogi „pasiskolintas“ iš žmonių elgesio analizės.

Kertinis šiame skyrelyje postuluojamų „kompiuterinių emocijų“ punktas yra stimuliavimas, Δt, skirtumas tarp trokštamų išėjimų. Ką tik matėme, kad jis iš esmės priklauso nuo aplinkos pokyčių stiprumo: pokyčiams esant stipresniem, s stimuliavimas pastebimai išauga.

Tad ir emocijas apibrėšime sekančiu būdu: jei kurį laiką mokymasis greitėjo, „emocijos“ stimuliavimą didina. Jei kurį laiką mokymasis lėtėjo, emocijos stimuliavimą mažina. Šio tipo elgesį galime stebėti vaikų mokymęsi: jei dalykas (disciplina) sekasi, mokinys „save-sužadina“, skiria tam dalykui didesnį dėmesį,



1.21 pav. a) – dviejų klasių, pasukimo kampą charkterizuojančio parametro, $\theta_{\max}$, kitimas 500 atpažinimo uždavinio pokyčių metu, b) – stimuliavimo lygio prisiderinimas prie aplinkos pokyčių stiprumo, c) – agentų kiekio populiacijoje raida 500 atpažinimo uždavinių kaitos laikotarpyje.

noriau mokosi. Jei mokytis nesiseka, noras tą dalyką mokytis „kažkaip savaime dingsta“, emocijos jį sumažina. Tad „užfiksuojame“:

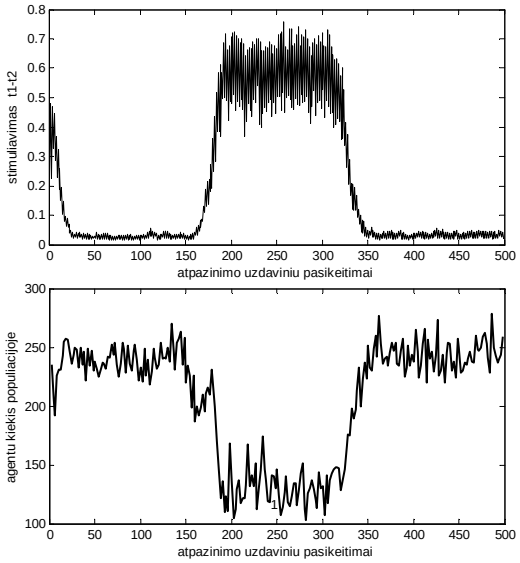
- jei n_{inercija} mokymo epochų metu $s-$tojo agento mokymasis buvo sėkmingas (klasifikavimo klaidų dažnis sumažėjo daugiau nei ξ karto), skirtumas Δt_s didinamas, jį padauginant iš koeficiento γ_{emoc} ($\xi > 1, \gamma_{\text{emoc}} > 1$),
- jei n_{inercija} mokymo epochų metu mokymasis buvo nesėkmingas (klasifikavimo klaidų dažnis išaugo daugiau nei ξ karto), skirtumas Δt_s mažinamas, jį padalinant iš koeficiento γ_{emoc},
- jei n_{inercija} mokymo epochų metu klasifikavimo klaidų dažnis pakito labai nežymiai, skirtumas tarp trokštamų išėjimų, Δt_s, neikeičiamas.

Žemiau pateikiamoje iliustracijoje atpažinimo uždaviniai kito pagal 1.21a pav. parodytą schemą. Visiems 400, į 8 grupes susikrstytiems, agentams, buvo nustatytos vienodos pradinės stimuliavimo parametro reikšmės: $\Delta t_s = 0.5$ ($s = 1, 2, ..., 400$). Kompiuterines emocijas charakterizuojantis koeficientas γ_{emocs}, buvo paveldimas iš sėkmingai besimokančio agento. Eksperimento pradžioje šios reikšmės buvo sugeneruotos atsitktinai intervale (1.01, 2), o kiti du, emocijas apsprenžiantys, parametrai buvo nekeičiami: $n_{\text{inercija}} = 10, \xi = 1.01$.

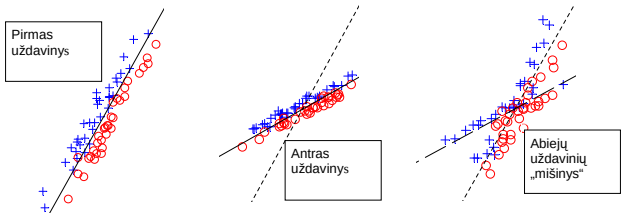
Šiame, kaip ir ankstesniame, eksperimente stimuliavimo reikšmės buvo paveldimos, tačiau papildomai, kas dešimtą mokymo epochą jas buvo galima pakeisti priklausomai nuo mokymosi sėkmės ir paveldėtų $s-$tojo agento emocijų γ_{emocs} reikšmių. 1.22a pav. pateiktas grafikas rodo, kad kaip ir ankstesniame eksperimente, stimuliavimo reikšmės priklauso nuo atpažinimo uždavinio kitimų stiprumo. Šis grafikas taip pat rodo, kad papildomo grįžtamo ryšio įvedimas paspartina mokymosi procesą: vėlinimas nustatantΔt_sreikšmes sumažėja. Žiūrint į agentų išlikimo grafikus matome, kad sunkiausiais populiacijai momentais daugiau agentų „atlaiko“ stipriausius atpažinimo uždavinio pokyčius (minėtam eksperimente vietoj 70 jų liko net 105, t.y. pusantro karto daugiau). Tai reiškia, kad stimuliavimo valdymas per paveldimas emocijas padeda grečiau rasti „optimalias“ trokštamų išėjimų reikšmes.

Adaptavimasis prie aplinkos, kai laikas tarp aplinkos pasikeitimų nėra pastovus

Iki šiol nagrninėtuose pavyzdžiuose, paprastumo dėlei darėme prielaidą, kad laiko tarpai (mokymo epochų kiekiai) tarp atpažinimo uždavinių pasikeitimų buvo fiksuoti, o uždaviniui pakitus, perceptronas buvo mokomas naudojant tik naujus duomenis. Šiame skyrelyje, nagrinėsime sudėtingesnį atvejį, kai laiko tarpai tarp pokyčių nėra pastovūs. Atpažinimo uždaviniui pakitus, kurį laiką mokymasis vyksta naudojant ir senus, ir naujus duomenis. Jeigu mokymui naudosime daug senų duomenų, perceptronas gali likti prie jų pernelyg pristaikęs ir klasifikuojant naujus, jau pakitusius, duomenis daryti daug klaidų. 23 paveikslo dešinėje matome situaciją, kai mokyme dalyvauja ir seni ir nauji duomenys.



1.22 pav. a) - stimuliavimo lygio prisiderinimas prie aplinkos pokyčių stiprumo, b) - agentų kiekio populiacijoje raida 500 atpažinimo uždavinių kaitos laikotarpyje.



1.23 pav. Mokymo duomenys, gaunami įvykus atpažinimo uždavinio pokyčiui.

Dėl minėto „senų ir naujų duomenų persikirtimo“ mokymo procesą reikia organizuoti taip, kad perceptronui mokyti naudojamų duomenų seka būtų kuo trumpesnė. Tuomet galimybė, kad kartu bus naudojami ir seni ir nauji duomenys sumažės. Žiūrint iš kitos pusės, jei mokymo seka bus pernelyg trumpa, jos gali neubeužtekti patikimai klasifikavimo taisyklei sudaryti.

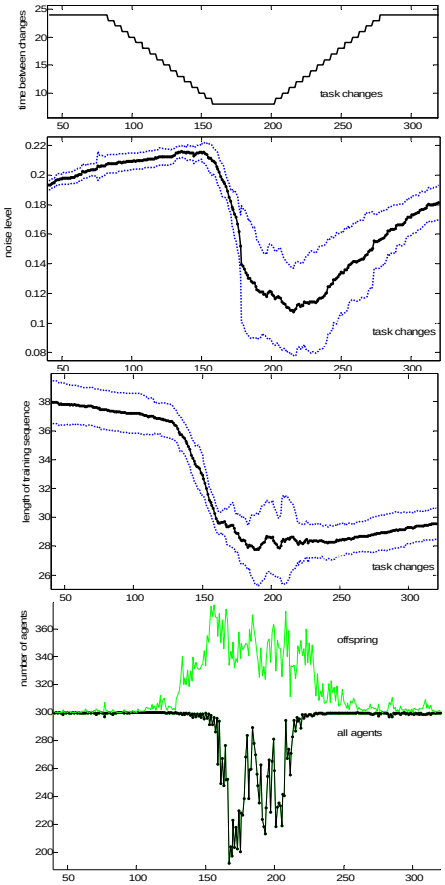
Šioje atpažinimo uždavinio formuluotėje *mokymo sekos ilgis ir būtų vienas iš daugelio perceptrono mokymosi stilių nusakančių parametrų.* Minėtą dilemą išspręsti nėra lengva, nes nežinome nei atpažinimo pasikeitimo laiko momento, nei pokyčio stiprumo. Iš principo, dalį klausimų būtų galima išspresti ir teoriškai darant daugybę prielaidų apie pasikeitimus, jų dažnį, tipą, stiprumą, požymių, $x_1, x_2, ..., x_p$, pasiskirstymus ir t.t. Teorinis sprendinys būtų naudingas, jei visos prielaidos būtų teisingos ir atpažinimo uždavinių ir jų keitimųsi parametrai būtų žinomi. Ralybėje taip nėra. Parodysime, kaip minėto tipo problemas galime sėkmingai spręsti evoliucionuojančios DaS posistemės pagalba. Paprastumo dėlei darykime prielaidą, kad uždavinių pokyčiai vyksta kaip ir ankstesniuose trijuose pavyzdžiuose: duomenys sukinėjasi „prieš“ ir „pagal laikrodžio rodyklę“ pagal jau anksčiau aprašytąją schemą. Šį kartą pasisukimo amplitudes nusakantis parametras, θ, laike nekinta. Kinta tik laikas (epochų skaičius) tarp dviejų gretimų atpažinimo uždavinio pokyčių.

Kaip ir ankstesniuose trijuose DaS parametrų nustatymo pavyzdžiuose, ilgos aplinkos pokyčių serijos metu nesėkmingi agentai pakeičiami naujais. Kolektyviai besimokydama daugiagentė posistemė sprendžia savo išlikimo klausimą, t.y. siekia išmokti pakitusius atpažinimo uždavinius, aplamai su mažesne, nei $P_{\text{leistinas}}$ klasifikavimo klaidų tikimybe. Išliekančių agentų mokymosi stiliaus charakteristikos laikui bėgant prisitaiko prie pasikeitimų dinamikos. Kaip tokios posistemės veikimo pavyzdį 1.24 pav. pateikiame:

a) laikotarpių (mokymo epochų kiekį), tarp atpažinimo uždavinių keitimosi, kitimo grafiką;
b) neteisingų mokymo direktyvų dalies, korupcijos parametro α_s, vidurkio per visus 400 agentų ($\pm$ *standartinis nukrypimas*, pažymėta mėlynai) kitimą 330-ties uždavinių metu;
c) vidutinį kiekvieno agento mokymosi sekos ilgio kitimą;
d) bendrą „gyvų“ agentų daugiaagentėje sistemoje (juoda), bei naujų, dar „vaikais“ esančių, agentų (žalia) kiekio kitimą.

Tam, kad galėtume aiškiau pademonstruoti paveldimų mokymosi stiliaus charakteristikų priklausomybę nuo aplinkos pokyčių stiprumo, aprašomame eksperimente laikas tarp dviejų vienas po kito sekančių, atpažinimo uždavinių pasikeitimų kito pagal iš anksto nustatytą schemą, kaip pavaizduota viršutiniame iš 24 paveikslo grafikų (24a).

Laiko tarp dviejų, vienas po kito sekančių, atpažinimo uždavinių pokyčių dėsningumą mes (teoretikai tyrėjai) žinome, bet pati daugiaagentė sistema to tai nežino. Pasikeitus atpažinimo uždaviniui daugiagentė sistema tęsia mokymosi



1.24 pav. Daugiaagentės atpažinimo sistemos dinamika 330 aplinkos pokyčių metu.

procesą nekeisdama pradinių svorių vektorių ir naudodama dalį senų ir naujus (jau pasikeitusius) duomenis. Kaip ir ankstesniuose pavyzdžiuose, esant trumpesniems laikotarpiams tarp pokyčių, daugiau agentų žūna, populiacija tampa mažesne, daugiau turime agentų-vaikų. Matome, kad paveldimos mokymosi stiliaus charakteristikos, vidutinis mokymo sekos ilgis ir iškraipytų mokymo direktyvų dalis, kinta kartu su laiko tarpais tarp dvejų, vienas paskui kitą sekančių, atpažinimo uždavinių pokyčių. Visa tai rodo, kad mūsų nagrinėjama adaptyvi evoliucionuojanti daugiaagentė sistema yra veiksni: darbo eigoje pati gali nusistatyti jos funkcionavimui reikalingus parametrus.

1.3.3. Baigiamosios pastabos

Daugiaagenčių sistemų modeliavimai parodė, kad agentų populiacijų atsparumas didesniems aplinkos pokyčiams labai žymia dalimi priklauso nuo agentų populiacijos dydžio. Kuo populiacija didesnė, tuo ji turi didesnę galimybę atlaikyti pačius stipriausius aplinkos (atpažinimo uždavinio) pokyčius. Todėl, sekant Gamtos evoliucijos pasiūlytoms idėjomis „savaime susiformuoja“ *optimizavimo kriterijus*. Tai – *populiacijos dydis didžiausių aplinkos pokyčių laikotarpyje*. Matome, kad reikia kurti tokias parametrų nustatymo posistemes, kuriose esant net dideliems aplinkos pokyčiams, žuvusiųjų agentų kiekis nebūtų pernelyg didelis. Čia reikia atminti, kad situacijose, kai agentai nebežūna, evoliucija sustoja taip pat. Štai čia, šioje vietoje, ir glūdi dar viena, iki šiol pilnai dar neišspręsta, dilema: DaS, skirtos mokymo stiliaus parametrų radimui, dydžio nustatymas. Jei sistemos dydis bus per didelis, mokymas gali sulėtėti, nes smarkiausiai besikeičiančių atpažinimo uždavinių laikotarpiais žus per maža agentų dalis. Jei nagrinėjamoje posistemėje agentų bus per mažai, minėtais laikotarpiais gali žūti visi agentai.

Šiuose trijuose skyriuose išaiškinome visą eilę klasifikavimo ir prognozavimo metodų, paremtų statistiniais metodais ir dirbtiniais neuroniniais tinklais. Svarbiausią dėmesį skyrėme ne tiek detalių išdėstymui (jos pakankamai gerai išdėstytos laisvai prieinamoje arba perkamoje standartinėje duomenų analizės programinėje įrangoje), bet konceptų, glūdinčių šių metodų sudaryme, aiškinimui ir gautų rezultatų interpretavimui, bei rezultatų patitikimumo įvertinimui.

Tam, kad teisingai panaudoti atpažinimo, prognozavimo metodus sudarant daugiaagenčių sistemų atskirų agentų sprendimo priėmimo taisykles, reikalingas nemažas patyrimas, praktika kalbantis su uždavinį suformulavusiais asmenimis, susipažinimas su užsakovo sprendžiamomis problemomis, nuoširdus įsigilinimas į užsakovo keliamas mokslines/praktines hipotezes.

Vienas iš kertinių akcentų pabrėžiamų šiuose skyreliuose, tai poreikis žinoti teisingus perceptronų mokymo stilių nusakančius parametrus, kad DaS sugebėtų greitai prisitaikyti prie pasikeitusių uždavinių. Tam kiekviena daugiaagentė sistema turi turėti ją aptarnaujančią, jos agentų mokymosi ir persimokymo procesus valdančią, posistemę. Aprašėme keturis iš paprasčiausių DaS posistemių, skirtų perceptronų mokymosi stilių nusakančių parametrų įvertinimui iš ilgų aplinkos pokyčių serijų stebėjimų. Eksperimentiniai tyrimai parodė, kad DaS tampa at-

sparesnėmis dideliems aplinkos pokyčiams, jei jos agentai pasiskirsto į nevienodo dydžio grupes, kuriose jie vienas kitam padeda išlikti. Bendravimas tarp grupių labai apribotas: tik esant kritinei situacijai, pati stipriausia grupė gali padėti jau bežūstančiajai, ypač neskaitlingai, grupei tuo, kad vienas ar keli iš jos sėkmingiausiųjų agentų, perduoda savo mokymosi stiliaus parametrus nesėkmingoje grupėje daromiems agentams-vaikams. Tokiu būtu agentų grupės populiacijoje žūna rečiau.

Jei pasikeitimai maži, perceptronų mokymo stilių nusakančių parametrų įvertinimo problema žymiai supaprastėja. Čia reikia pabrėžti, kad gyvename nuolatinės konkurencijos sąlygomis. Tam, kad sukurti naują konkurentabilią produkciją, reikia, kad ji savo teigiamomis savybėmis išsiskirtų iš kitų. Sugebėjimas greitai prisitaikyti prie pasikeitusių aplinkos sąlygų ir yra viena iš DaS skatinintų teigiamų savybių. Kaip jau ne kartą minėta, greta pagrindines funkcijas vykdančių DaS-mos agentų darbui ypač didelių netikėtų aplinkos pasikeitimų sąlygomis turėtų būti kuriamos ir lygiagrečios, perceptronų mokymosi parametrus valdančios DaS-os. Jose virtualių agentų turėtų būti daugiau nei pagrindinių, vykdančiųjų agentų. Pagrindinė tokios posistemės funkcijų būtų „prižiūrėti“ pagrindinę sistemą, rūpintis, kad „jai nieko netrūktų“, t.y. joje kiekvienai funkcijai, kiekvienu laiko momentu būtų pakankamai veiksmingų agentų, kad agentų mokymosi parametrai visą laiką būtų tinkamai nustatyti.

Kalbant apie ateitį, rikia pabrėžti, kad laikui bėgant, augant kompiuterių galingumams ir jiems pingant, DaS sprenžiamų uždavinių ratas sparčiai augs. Daugeliu atvejų daugiaagentes sistemas stengiamasi daryti kiek įmanomai universalesnes. Tad jų adaptavimosi prie besikeičiančios aplinkos reikalavimai visą laiką tik stiprės. Viena iš krypčių, kuria DaS-os vystysis, bus *mokymasis priimti kolektyvinius sprendimus*. Šioje paradigmoje yra mokoma didelis kiekis panašias funkcijas atliekančių agentų, priklausančių vienai ar kelioms grupėms. Agentai, jų grupės tarpusavyje bendrauja, konkuruoja, o reikiamu momentu priima kolektyvinius sprendimus. Daugeliu atveju kolektyviniai sprendimai būna patikimesni. Kad šio tipo daugiaagentės sprendimų priėmimo sistemos veiktų efektyviau, reikia, kad jas sudarntys agentai, jų grupės būtų kiek įmanoma įvairesni, naudotų skirtingus požymius, mokymosi būdus, skirtingo ilgio mokymo duomenų istorijas ir t.t.. Kolektyvinio sprendimo metodai taip pat yra svarbus, dar ne iki galo ištirtas, dalykas. Neištirtų problemų, kurių nagrinėjimas ateityje taps daugelio informatikos specialistų darbo sritimi, yra daug. Kompiuterių panaudojimo sritims plečiantis, problemų kiekis vis augs. Pasikisrtytų sistemų (angl. *distributed systems*), visur esančio intelekto ir skaičiavimų (angl. *ubiquitous intelligence and computing*), dirbtinės gyvybės (angl. *artificial life*), dirbtinių imuninių sistemų (angl. *artificial immune systems*) ir panašių sparčiai besivystančių dirbtinio intelekto sričių, raida labai tampriai susijusi su daugiaagenčių sistemų kūrimo teorija ir praktika.

Išvardytos informatikos sritys padeda kurti ne tik naujus, intelektualius, žmogaus veiklą palengvinančius produktus, bet mokslinių tyrimų rezultatai, sukaupti minėto tipo tyrimuose gali būti panaudoti kaip matematiniai-kompiuteriniai mo-

deliai daugelyje Gamtos, socialinių, ekonomikos, psichologijos mokslų sričių tyrimuose. Matėme, kad daugelis dirbtinių neuroninių tinklų ir daugiaagenčių sistemų kūrimo idėjų yra „pasiskolintos“ iš Gamtos. Moksliniai tyrimai išvardytose informatikos srityse jau pradeda šią „skolą“ gražinti. Kaipo pavyzdį paminėsime šiame skyriuje nagrinėtas daugiaagentes sistemas, skirtas perceptrono mokymosi stilių aprašančių parametrų įvertinimui. Matėme, kad triukšmo pridėjimas prie trokštamų išėjimų (neteisingų, specialiai sugadintų) mokymo direktyvų atsiradimas tampa naudingu instrumentu, sustiprinančiu agentų populiaciją vykstant dideliems aplinkos pasikeitimams. Tad nėra ko stebėtis, kad pokario laikotarpiuse ar pasikeitus santvarkai (kaip pavyzdžiui iš „subrendusio socializmo“ žengiant į kapitalizmą Rytų Europos šalyse) kriminologinė situacija, korupcija išauga. Iš DaS tyrimų seka, kad tai normalus, visuomenės adaptavimąsi prie naujų sąlygų greitinantis, reiškinys. Kol kas remiantis tik adaptyvių daugiaagenčių, vienasluoksniais perceptronais paremtų, sistemų tyrimais taip kategoriškai teigti nereikėtų, bet DaS sistemų sudarymo ir nagrinėjimo metodus galima ir netgi būtina panaudoti ir šio tipo socialinių reiškinių analizei.

Kitas pavyzdys liestų poreikį suskaldyti agentų populiaciją į grupes, kur benradarbiavimas tarp grupių yra iš esmes ribojamas. Tūkstantmečių laikotarpyje išsivystė valstybės, religijos, ekonominės sitemos, kur benradarbiavimas tarp grupių ribotas, kur greta didelių grupių sėkmingai egzistuoja ir mažosios. DaS tyrimai parodė, kad tokia strategija padeda adaptyvių agentų populiacijai (kalbant plačiąja prasme, žmonijai) išgyventi pačius stipriausius aplinkos pokyčius. Gal šio tipo modeliavimo tyrimus reikėtų „pernešti“ ir į politiką, ekonomiką, religijos tyrimus? Taip, kad daugiaagenčių sistemų tyrimai apima daug platesnę žmogaus veiklos sferą, nei buvo galima manyti.

Literatūra

[1] **Wooldridge M., Jennings N. R.**, *Intelligent Agents: Theory and Practice* // Knowledge Engineering Review, 10 (2), 1995.

[2] **Shoham, Y**. *An Overview of Agent-oriented Programming. In Software Agents*, ed J. M. Bradshaw. Menlo Park, Calif.: AAAI Press. 1997.

[3] **Nwana, H. S.** *Software Agents: An Overview.* Knowledge Engineering Review, 11(3): 205-244, 1996.

[4] **Newell, A.** *The Knowledge Level. Artificial Intelligence.*87-127, 1982.

[5] *Agents. American Association for Artificial Intelligence.* [interaktyvus] [žiūrėta 2007 m. gruodžio 7 d.]. Prieiga per internetą: <http://www.aaai.org/AITopics/html/agents.html>.

[6] **Genesereth, M. R., and Ketchpel, S. P.** *Software Agents.* Communications of the ACM 37(7): 48-53, 147, 1994.

[7] **Dastani M., Hulstijn J.** *Leendert van der Torre, BDI and QDT: a comparison based on classical decision theory*// In Proceedings of the National Conference on Artificial Intelligence (AAAI'01).

[8] **Rao, A., and Georgeff, M.** *Modeling rational agentswithin a bdi architecture.* In Proceedings of Second International Conference on Knowledge Representation and Reasoning (KR'91), 473-484.1991. Morgan Kaufmann.

[9] **Xiaobing Zhao, Jayendran Venkateswaran, and Young Jun Son**, *Modeling Human Operator Decision-Making in Manufacturing Systems Using BDI Agent paradigm* // in Proceedings of 2005 IIE annual conference, May 16, Atlanta, Georgia, 2005 (6pages).

[10] **Brazdil, P.** *Agents and Multi-Agent Systems*[interaktyvus]. 2001. [žiūrėta 2007 m. gruodžio 12 d.]. Prieiga per internetą: <http://www.niaad.liacc.up.pt/ pbrazdil/Ensino/madsad/agents.html>.

[11] **DeLoach, S. A.** *Multiagent Systems Engineering: A Methodology And Language for Designing Agent Systems* // Presented at Agent-Oriented Information Systems (AOIS) '99, 1999.

[12] **G. Weiß and S. Sen,** eds., *Adaptation and Learning in Multiagent Systems.* Berlin: Springer Verlag, 1996.

[13] **Vlassis, Nicos**. *A Concise Introduction to Multiagent systems and Distributed AI* [interaktyvus]. [Amsterdam, Holand]: University of Amsterdam, 2003 [žiūrėta 2007 m. lapkričio 14 d.]. Prieiga per internetą: <http://remote.science.uva.nl/ vlassis/cimasdai/cimasdai.pdf>.

[14] **K.S. Decker**. *Distributed problem solving: A survey,* IEEE Transactions on Systems, Man, and Cybernetics, vol. 17, pp. 729-740.

[15] **Jose M Vidal**. *Fundamentals of Multiagent Systems with NetLogo Examples* // 2007.

[16] **Jelle R. Kok**. *Coordination and Learning in Cooperative Multiagent Systems* // 2006

[17] **Warren**. *The Inteligent Software Agents Lab - The Robotics Institute - Carnegie Mellon University*, 2001 [interaktyvus] [žiūrėta 2007 m.

gruodžio 12d.]. Prieiga per internetą: <http://www-2.cs.cmu.edu/ softagents/warren.html>.

[18] *ASAP Theme: Multi-Agents* [interaktyvus]. [Žiūrėta 2007 m. gruodžio 13 d.] Prieiga per internetą: <http://www.asap.cs.nott.ac.uk/themes/ma.shtml>.

[19] **Brazdil, P.** *Architectures of Multi-Agent Systems* [interaktyvus] [žiūrėta 2007 m. gruodžio 12d.]. Prieiga per internetą: <http://www.niaad.liacc.up.pt/ pbrazdil/Ensino/madsad/arch_agents.html>.

[20] **DeLoach, S. A.** *Multiagent Systems Engineering: A Methodology And Language for Designing Agent Systems* // Presented at Agent-Oriented Information Systems (AOIS) '99, 1999.

[21] **Kurbel K., Schereber D.** *AGENT-BASED DIAGNOSTICS IN SUPPLY NETWORKS.pdf* [interaktyvus] [žiūrėta 2008 m. sausio 11 d.] Prieiga per internetą: <http://www.iacis.org/iis/2007_iis/PDFs/Kurbel_Schreber.pdf >.

[22] **Zhongzhi Shi Shiqiang Zhong Bo Wang.** *Agent-based E-Commerce* // Laboratory of Intelligent Information Processing Institute of Computing Technology, Chinese Academy of Sciences Beijing.

[23] **Mougayar, W.** *The future of agent-based commerce on the Web.* CYBER Management Inc. 1997.

[24] **Odell, James.** *Agents: Technology and Usage* (Part 1). Iš Distributed Computing Architecture/e-Business Advisory Service Executive Report [interaktyvus]. 2000, vol. 3, no. 4 [žiūrėta 2007 m. gruodžio 20d]. Prieiga per internetą: <http://www.upv.es/sma/teoria/introd/Cutter00-04.

[25] *A Trustworthy Agent Based Online Auction System.* [interaktyvus] [žiūrėta 2008 m. sausio 10d.] Prieiga per internetą: <http://www.cis.umassd.edu/ hxu/Projects/UMD/TrustworthyAuction/ Trustworthy-Auction-System.ppt>

[26] **Stanley, Tracey.** *Intelligent Searching Agents on the Web* [interaktyvus]. [Leeds, UK]: Library at the University of Leeds, January 23rd 1997 [žiūrėta 2008 m. vasario 8 d.]. Prieiga per internetą: <http://www.ariadne.ac.uk/issue7/search-engines/>.

[27] **Etzioni, O., and Weld, D. S.** *Intelligent Agents on the Internet: Fact, Fiction, and Forecast.* IEEE Expert 10(4): 44-49, 1995.

[28] **Wooldridge, M. J., and Jennings, N. R.** *Agent Theories, Architectures, and Languages: A Survey.* In Intelligent Agents: ECAI-94 Workshop on Agent Theories, Architectures, and Languages, eds. M. J. Wooldridge and N. R. Jennings, 1-39. Berlin: Springer-Verlag. 1995.

[29] **Maes, P.** *Modeling Adaptive Autonomous Agents.* In Artificial Life: An Overview, ed. C. G. Langton, 135-162. Cambridge, Mass.: MIT Press. 2006.

[30] **Genesereth, M. R., and Ketchpel, S. P.** *Software Agents.* Communications of the ACM 37(7): 48-53, 147. 2005.

[31] *Kompiuterijos terminų aiškinamasis žodynas* [interaktyvus]. [Žiūrėta 2008 m. sausio 10 d.] Prieiga per internetą: <http://aldona.mii.lt/pms/terminai/term/z2odynas.html>.

[32] *KQML.* Iš Webopedia Computer Dictionary [interaktyvus]. [Žiūrėta 2007 m. gruodžio 19 d.]. Prieiga per internetą: <http://www.webopedia.com/TERM/K/KQML.html>.

[33] **S. Haykin.** *Neural Networks: A comprehensive foundation.* 2nd edition. Prentice-Hall, Englewood Cliffs, NJ, 1999.

[34] **S. Raudys.** *Statistical and Neural Classifiers: An integrated approach to design.* Springer. London, 2001.

[35] **S. Raudys, A. K. Jain.** Small sample size effects in statistical pattern recognition: recommendations for practitioners. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-13(3):252-264, 1991.

[36] **S. Raudys.** Evolution and generalization of a single neurone. I. SLP as seven statistical classifiers. *Neural Networks*, 11(2):283-296, 1998.

[37] **S. Raudys** Trainable Fusion Rules. I. Large sample size case. *Neural Networks*, Vol. 19 1506-1516, 2006.

[38] **S. Raudys.** An adaptation model for simulation of aging process. *Int. J. of Modern Physics, C.* 13: 1075-1086, 2002.

[39] **S. Raudys, A. Mitasiunas.** Multi-agent system approach to react to sudden environmental changes. *Machine Learning and Data Mining. Lecture Notes in Artificial Intelligence*, Springer-Verlag Berlin Heidelberg, Vol. 4571, pp. 810–823, 2007.

[40] **S. Raudys.** Effect of synthetic emotions on agents' learning speed and their survivability. *Artificial Life. Lecture Notes in Artificial intelligence*, Springer, Vol. 3630, pp. 1 – 10, 2005.

[41] **S. Raudys.** Social organization of evolving multiple classifier system functioning in changing environments. *Adaptive and Natural Computing Algorithms.* Le*cture Notes in Computer Science,* Vol. 4431, pp. 711–720 Springer-Verlag, Berlin Heidelberg New York, 2007.

[42] **S. Raudys, A. Hussain, V. Justickis, A. Pumputis A. Augustinaitis.** Functional model of criminality: simulation study. *CONTEXT'05. Lecture Notes in Artificial intelligence*, Springer, Vol. 3554, pp. 410 – 423, 2005.

[43] **S. Raudys, A. Pumputis.** Group interests of agents functioning in changing environments. *Multiagent Systems. Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin Heidelberg New York 3690: 559 – 563, 2005.

[44] **J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur,E. Thelen.** Autonomous mental development by robots and animals. *Science*, 291(5504): 599, 2001.

[45] C. Dixon, H.-C. Gago, M. Fisher, W. van der Hoek, Temporal logic of knowledge and their applications in security, Electronic Notes in Theoretical Computer Science, 186, 27-42, 2007.

[46] E.A. Emerson, Temporal and modal logic, ch. 16 in Handbook of Theoretical Computer Science, ed. J. van Leeuwen, Elsevier Science Publishers B.V., 996-1072, 1990.

[47] R. Fagin, J. Y. Halpern, Y. Moses and M.Y. Vardi , Reasoning about knowledge, MIT Press , Cambridge MA, 1995.

[48] M. Fisher, R. H. Bordini, B. Hirsch, P. Torroni, Computational logics and agents: a roadmap of current technologies and future trends, Computational Intelligence, 23(1), 61- 91, 2007.
[49] J. Y. Halpern, R. van der Meyden, M.Y. Vardi, Complete axiomatizations for logics of knowledge and time, SIAM Journal of Computing, 674-703, 2004.
[50] J. Y. Halpern, Y. Moses, Knowledge and common knowledge in a distributed environment, J. of the ACM, 37(3), 549-587, 1990.
[51] J. Y. Halpern, Y. Moses, A guide to completeness and complexity for modal logics of knowledge and belief, Artificial Intelligence, 54, 311-379, 1992.
[52] J. Y. Halpern, L.D. Zuck, A little knowledge goes a long way: knowledge based derivations and correctness proof of family of protocols, Journal of ACM, 39(3), 449 - 478, 1992.
[53] J. Hintikka, Knowledge and belief, Cornell University Press, Ithaca, N. Y., 1962.
[54] W. van der Hoek, Logical foundations of agent-based computing, Lecture Notes on Artificial Intelligence, 2086, 50-73, 2001.
[55] W. van der Hoek, B. van Linder, J.-J. Meyer, An integrated modal approach to rational agents, Foundations of Rational Agency, Kluwer, Dordrecht, 37-75, 1999.
[56] W.van der Hoek, L. C. Verbrugge, Epistemic logic: a survey , Game Theory and Applications, 8, Nova Science Publishers, New York, 53-94, 2002.
[57] S. Kripke, A semantical analysis of modal logic I: normal modal propositional calculi, Zeitschrift fur Mathematishe Logik und Grundlagen der Mathematik, 9, 67-96, 1963.
[58] R. van der Meyden, Common knowledge and update in finite environment, Inftormation and Compuation, 140(2), 115-157, 1998.
[59] J.- J. Ch. Meyer, W.van der Hoek, Epistemic logic for AI and computer science, Cambridge University Press, 1995.
[60] S. Norgėla, Logika ir dirbtinis intelektas, TEV, Vilnius, 2007.
[61] S. Norgėla, J. Sakalauskaitė, Neklasikinės logikos informatikams, TEV, Vilnius, 2007.
[62] A.S. Rao, M. P. Georgeff, A model-theoretic approach to the verification of situated reasoning systems, in: Proc. of 13-th International Joint Conference on Artificial Intelligence, Chambery, France, 318-324, 1993.
[63] A.S. Rao, M. P. Georgeff, Formal models and decision procedures for multi-agent systems, Technical note 61, Australian Artificial Intelligence Institute, 1995.
[64] A.S. Rao, M. P. Georgeff, Decision procedures for BDI logics, J. of Logic and Computation, 8(3), 293-343, 1998.
[65] M. Wooldridge, Reasoning about rational agents, MIT Press, Cambridge MA, 2000.
[66] **Antoniou, A.** *Digital Filters.* McGraw-Hill, Toronto, 2000.
[67] **Bahl, L. R., Jelinek, F.** *Decoding for channels with insertions, deletions,*

substitutions, with applications to speech recognition. IEEE Trans. Informatiuon Theory, vol. 21, pp. 404-411, 1975.

[68] **Jelinek, F.** *Continuous speech recognition by statistical methods.* Proceedings of IEEE, vol. 64, Nr. 4, pp. 532-556, 1976.

[69] **Baum, L.E.** *An inequality and associated maximization technique in statistical estimation for probabilistic functions of a Markov process.* Inequalities, vol. 3, pp. 1-8, 1972.

[70] **Cavicchi, T.** *Digital Signal Processing.* John Wiley, New York, 1999.

[71] **Deller, J., Hansen, J., Proakis, J.** *Discrete-Time Processing of Speech Signals.* 2nd. edition, Wiley-IEEE Press, Washington, 1999.

[72] **Fcrguson, J.D.** *Hidden Markov Analysis: An Introduction in Hidden Markov Models for Speech.* ed. J. D. Ferguson. Princeton. NJ: Institute for Defense Analyses. pp. 8- 15, 1980.

[73] **Furui, S.** *Speaker-independent isolated word recognition using dynamic features of speech spetctrum.* IEEE Transactions on Acoustics, Speech, Signal Processing, vol. 34, No 1, 1986, pp. 52-59, 1986.

[74] **Furui, S.** *Digital Speech Processing, Synthesis and Recognition.* Marcel Dekker Inc., New York, 2001.

[75] **Hermansky, H., Morgan, N., Bayya, A., Kohn, P.** *RASTA-PLP Speech Analysis.* ICSI Technical Report, 1991.

[76] **Huang X., Ariki Y., Jack, M.A.** *Hidden Markov Models for Speech Recognition.* Edinburgh Univ. Press, Edinburgh, 1990.

[77] **Huang, X., Acero, A., Hon, H.** *Spoken Language Processing: A Guide to Theory, Algorithm and System Development.* Prentice-Hall, New York, 2001.

[78] **Juang. B.H.** *On the Hidden Markov Model and Dynamic Time Warping for Speech Recognition-A Unified View.* AT&T Technical Journal. Vol. 63. pp. 1213-1243, 1984.

[79] **Jurafsky, D., Martin, J.** *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice-Hall, New York, 2000.

[80] **Makhoul, J.**. *Linear prediction: A tutorial review.* Proceedings of the IEEE, Vol. 63, No. 4., 1975, pp. 561 -580, 1975.

[81] **Markel D., Gray, Jr. A.H.**. *Linear Prediction of Speech.* Springer Verlag, Berlin, 1976.

[82] **McTear, M.F.**. *Spoken Dialogue Technology: Enabling the Conversational Interface.* ACM Computing Surveys, 34(1):90-169, 2002.

[83] **Painter, T., Spanias, A.** *Perceptual coding of digital audio.* Proceedings of the IEEE, Vol. 88, Issue 4, pp. 451 - 515, 2000.

[84] **Proakis, J.G., Manolakis, D.G.** *Digital signal processing: principles, algorithms and applications.* Macmillan , New York, 1996.

[85] **Rabiner, L.R., Levinson, S.E., Sondhi, M.M.**. *On the application of vector quantization and hidden Markov models to speaker-independent, isolated word recognition.* Bell Systems Technical Journal, vol. 62, Nr. 4, pp.

1075-1105, 1983.

[86] **Rabiner, L.R**. *A tutorial on hidden Markov models and selected applications in speech recognition.* Proceedings of the IEEE, Volume 77, Issue 2, pp. 257 - 286, 1989.

[87] **Shenoi, B. A.**. *Introduction to Digital Signal Processing and Filter Design.* John Wiley, New York, 2000.

[88] **Shikano, K.** *Spoken word recognition based upon vector quantization of input speech.* Transactions of Committee on Speech Research, Acoust. Soc. Of Japan, S82-60, 1982.

[89] **Viterbi, A.J.**. *Error bounds for convolutional codes and an asymptotically optimal decoding algorithm.* IEEE Transactions on Information Theory, vol. 13, pp. 260-269, 1967.