Klaipėdos universitetas

Šarūnas Raudys

ŽINIŲ IŠGAVIMAS IŠ DUOMENŲ

Vadovėlis

KLAIPĖDOS UNIVERSITETO LEIDYKLA

Klaipėda, 2008

UDK 004.9:519.2(075.8)
Ra255

Šarūnas Raudys. ŽINIŲ IŠGAVIMAS IŠ DUOMENŲ

Vadovėlyje pateikiamos matematine statistika, dirbtinių neuroninių tinklų teorija paremtos žinios, leidžiančios suprasti ir įsisavinti modernius duomenų analizės ir žinių atpažinimo duomenų bazėse būdus, naujus dirbtinio intelekto metodus. Dėstomi šiuolaikiniai duomenų analizės metodai, paremti daugiamate matematine statistika, vienasluoksniais ir daugiasluoksniais perceptronais, spindulinių bazinių funkcijų ir mokymo vektorių kvantavimo neuroniniais tinklais, genetiniais algoritmais, sprendimų medžiais. Ypatingas dėmesys kreipiamas į preliminarų duomenų apdorojimą prieš juos analizuojant, į metodų sudėtingumo, turimų duomenų kiekio ir analizės tikslumo ryšį. Aiškinamos mokymo greičio ir išvadų patikimumo problemos. Taip pat mokoma pritaikyti ir praktiškai naudotis skaitiniais ir kompiuterinio modeliavimo metodais, planuoti ir atlikti eksperimentus, juos tinkamai interpretuoti, gauti motyvuotas išvadas. Teorinė vadovėlio medžiaga naudotina kartu su Klaipėdos universiteto Gamtos ir matematikos mokslų fakulteto pateiktu paskaitų kursu 908M03 „Statistinė duomenų analizė ir žinių išgavimas“ virtualioje mokymo aplinkoje (http://vma.ku.lt/moodle/course/index.php).

Vadovėlio leidimą remia Europos Sąjungos struktūriniai fondai − projektas „Informacinių technologijų srities magistrantūros studijų programų modernizavimas, plėtra ir mobilumo užtikrinimas“,
SFMIS kodas: BPD2004-ESF-2.5.0-03-05/0063

ISBN 978-9955-18-345-7

TURINYS

Duomenų analizė – tai radimas taško, iš kurio reikėtų pažvelgti į duomenis, kad pastebėtum juose esantį dėsningumą.

PRATARMĖ

Informacijos antplūdžio laikotarpis žmonijos istorijoje

Gyvename informacijos antplūdžio šimtmetyje. Kasdien mes, žmonės, informacijos gauname gerokai daugiau, nei galime apdoroti. Dažnai tai tampa trikdžiu. Gaunamą informaciją dėl jos pertekliaus pradedame ignoruoti. O argi tai gerai? Galime būti apkaltinti: „Štai ši informacija paskelbta viešai, ir Jūs būtinai privalote ją žinoti.“ Pavyzdys: sakoma, kad įstatymų nežinojimas nuo atsakomybės neatleidžia; o jie (tie įstatymai, įstatymo lydimieji aktai) keičiasi, tad normalus verslininkas, o ir administratorius, praktiškai nesugeba visų jų sekti; juolab kad dažnai jie prieštarauja vienas kitam.

Didieji prekybos centrai, draudimo bendrovės ir socialinio draudimo įstaigos, aviabilietų pardavimo agentūros, teisėtvarkos įstaigos kiekviena atskirai turi arba gali turėti daugybę informacijos apie kiekvieną iš mūsų. O jei jų informaciją „sujungtume“, tai ko gero, visai prarastume privatumą. Jei ta informacija būtų panaudota „geriems tikslams“, mūsų gyvenimas palengvėtų. O jei blogiems?

Tad šiuo informacijos antplūdžio laikotarpiu žmonijai kyla nemaža naujų uždavinių, su kuriais anksčiau ji nėra susidūrusi. Tai informacijos išgavimo, apdorojimo, analizės, patikimumo, saugumo, slaptumo, nuosavybės, saugojimo techniniai ir juridiniai klausimai. Akivaizdu, kad jie dar neišspręsti, ir kol kas mes esame dar labai toli nuo to. Daugelis mokslinių disciplinų nagrinėja minėtus klausimus. Šioje knygoje kalbėsime tik apie vieną iš darbo su dideliais informacijos kiekiais aspektų, būtent – apie duomenų analizę, galimą žinių išgavimą iš didelių informacijos masyvų. Šiuo metu pasaulyje (ypač JAV ir

Anglijoje) yra labai daug duomenų analizės (angl. *data mining*) firmų, sprendžiančių įvairiausias tiek mažų firmų, tiek ir labai didelių problemas.

Statistinė duomenų analizė ir žinių išgavimas iš duomenų masyvų – plati, įvairialypė disciplina. Panaršę po internetą, netgi lietuvišką, pamatysime, kad skirtingi autoriai aprašo/nagrinėja įvairius duomenų analizės, žinių išgavimo būdus. Žemiau bus aprašoma tik dalis iš daugelio žinomų metodų. Visų jų nagrinėjimas apimtų daug tomų. Šioje knygoje bus kalbama bene apie pagrindinę metodų dalį. Tai klasifikavimo, prognozavimo algoritmai, duomenų nevienalytiškumo klausimai, nuolat kintančių dėsningumų radimas. Ypatingas dėmesys bus skiriamas analizės rezultatų interpretavimui, gautų modelių patikimumo, pasitikėjimo gautais rezultatais įvertinimui. Tai ypač svarbus analizės etapas, nes jei analizės rezultatai nepatikimi, jie gali būti ir klaidingi. Tokiu atveju vietoj naudos galime turėti ir bėdos.

Ši knyga – tai daugiamečio mokslinio tiriamojo bei praktinio duomenų analizės darbo, bendravimo su įvairių Lietuvos ir kitų šalių universitetų studentais ir dėstytojais, kolegomis Matematikos ir informatikos institute bei kitose mokslinio tyrimo įstaigose, jų žinių siekimo ir kūrybiškumo rezultatas. Žmonių, turėjusių įtaką – dešimtys, o gal ir du šimtai. Už jų pagalbą ir bendravimo džiaugsmą esu jiems labai dėkingas.

Autorius

Vilnius
Matematikos ir informatikos institutas
2008 m.

SANTRUMPŲ IR TERMINŲ ŽODYNĖLIS

DAS – daugiaagentė sistema – tai iš daugelio intelektualius uždavinius sprendžiančių elementų (agentų, programų), turinčių bendrą tikslą, sudaryta sistema.

DNT – dirbtiniai neuroniniai tinklai – tai iš adaptyvių elementų, imituojančių gyvų organizmų smegenų ląstelių (vadinamų neuronais) veiklą, sudaryta informaciją apdorojanti sistema.

DsP – daugiasluoksnis perceptronas – tai daugybės vienasluoksnių perceptronų, išdėstytų sluoksniais, tinklas, kur informacija iš vieno sluoksnio perceptrono perduodama kito sluoksnio perceptronams.

EAK – euklidinio atstumo klasifikatorius – tai klasifikavimo metodas, atliekantis sprendimą pagal Euklido atstumą iki klasių „centrų“, t. y. iš mokymo duomenų įvertintų tų klasių vidurkių vektorių.

RBF – spindulinės bazinės funkcijos – tai dirbtinis neuroninis tinklas, sudarytas iš specializuotų elementų, dirbtinių neuronų, kurių kiekvienas charakterizuoja įėjimo vektoriaus $\boldsymbol{x}$ panašumą į specifinius „centrus“ $\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_m$.

VKN – vidutinis kvadratinis nuokrypis – tai kvadratinė šaknis iš vidutinės paklaidų kvadratų sumos.

VsP – vienasluoksnis perceptronas – tai vienos smegenų ląstelės, vadinamos neuronu, informacijos apdorojimo matematinis modelis.

EP_n^{A} – klasifikavimo metodo A laukiama generalizavimo paklaidos tikimybė.

EP_{R} – empirinė klasifikavimo klaidos tikimybė, konkretaus klasifikavimo algoritmo, sudaryto pagal mokymo duomenis (MD), klasifikavimo klaidų dažnis, įvertinant mokymui jau naudotus duomenis.

P_{B} – Bajeso klasifikavimo klaidos tikimybė – tai idealaus klasifikavimo algoritmo, sudaryto žinant tikruosius įėjimo vektorių $\boldsymbol{x}$ pasiskirstymo tankius ir klasių apriorines tikimybes, klaidos tikimybė.

P_{∞}^{A} – metodo A asimptotinė klaidos tikimybė – tai klasifikatorius, sudarytas naudojant duomenų pasiskirstymo modelį A, kurio parametrai tiksliai žinomi, o ne įvertinti iš mokymo duomenų.

$\Phi(c)$ – standartinio normaliojo tankio, kurio vidurkis 0 ir dispersija 1, kumuliatyvinė pasiskirstymo funkcija, t. y. tikimybė, kad $x \leq c$.

Klasifikavimas – tai vektoriaus $\boldsymbol{x}$ priskyrimas vienai iš klasių.

Prognozavimas – tai funkcinės priklausomybės $y = f(\boldsymbol{x})$ radimas iš eksperimentinių duomenų.

Požymių (įėjimo) vektorius $\boldsymbol{x}$ – tai objektą ar reiškinį aprašančių charakteristikų rinkinys.

PIRMAS SKYRIUS. DUOMENŲ ANALIZĖS PROBLEMA. NUOLATINIAI POKYČIAI IR ADAPTAVIMASIS PRIE JŲ

1.1. Įvadas. Duomenų analizės problema. Nuolatiniai aplinkos pokyčiai

1.1.1. Vartotojo krepšelio uždavinys

Praktiškai pats seniausias ir bene populiariausias duomenų analizės uždavinys – tai vartotojo krepšelio uždavinys: milžiniškų duomenų masyvų apie nupirktas, įsigytas, grąžintas prekes didžiausiuose prekybos tinkluose analizė. Tikslas – iš minėtų duomenų išgauti žinias, reikalingas prekybai tobulinti. Prekybos tinkluose kasdien apsilanko dešimtys tūkstančių ar net milijonai pirkėjų. Paprastai jie moka kreditinėmis kortelėmis. Tad tampa įmanoma sekti jų pirkimų istoriją, jų įpročius ir tai panaudoti geresniam pirkėjų aptarnavimui, prekybos tinklo įvaizdžio kėlimui. O geras įvaizdis, patrauklumas prekybos tinklams reikalingas. Kai kuriose šalyse labai populiarios mažosios parduotuvėlės. Kiekvienos iš jų savininkas kartu yra ir pardavėjas. Jis pažįsta pirkėjus asmeniškai, žino jų problemas, bėdas ir pan. Šis savo gerbiamam ir mylimam pirkėjui parūpins jo mėgstamą prekę, pakalbės apie šeimą ir artimuosius, aptars bendrus pažįstamus, miestelio problemas. Ar rasime tokią galimybę „Akropolyje?“ Ne. Bet štai Prancūzijos prekybos tinklo vadovybė nori tuos senus pensininkus persivilioti pas save. Jų pasamdyta duomenų analizės tarnyba ištiria senų žmonių kelerių metų pirkimus, paseka, kokias prekes ir po kiek jų jie pirko, kokiomis savaitės dienomis, kuriuo paros metu, po vieną ar keliese. Visa ši informacija prekybos tinklo vadovams padeda šio tipo pirkėjams reikalingas prekes išdėstyti vienoje vietoje, įrengti ten kavinę, žodžiu, – „pensininkų klubą“. Ir štai pensininkai jau turi susitikimų vietą, kur „visi pažįstami“ susirenka ir greta kitų visus juos dominančių klausimų gali aptarti savo pirkinius, o galų gale – nusipirkti daugiau, negu buvo planavę. Prekybos tinklo vadovai dėl to tik „trina rankas“ ir duomenų analizės firmai ruošiasi pateikti dar didesnį užsakymą.

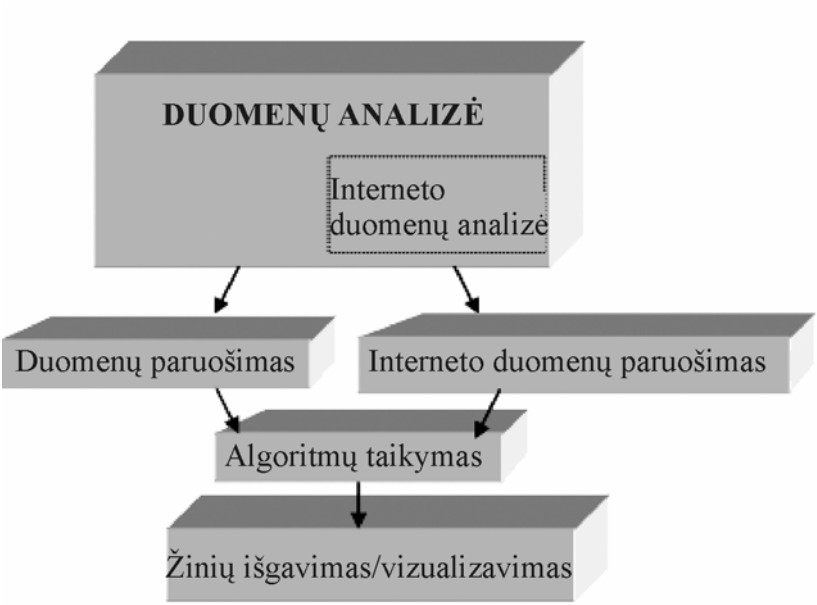
Štai keletas klausimų, kurie domina prekybos tinklų vadovus: Kaip išdėstyti prekes? Kokias, kada, kokiais intervalais, po kiek jas užsakinėti? Kokiais kiekiais jas fasuoti? Kokios prekių rūšys perkamos vienu pirkimu? Kokia žmonių kategorija vienaip ar kitaip elgiasi? Kada ir kokią nuolaidų akciją paskelbti? Kaip ir kur ją reklamuoti? Jei jiems pasisektų gauti pakankamai išsamią informaciją iš konkuruojančio prekybos tinklo, tai jiems būtų įdomu, kodėl dalį pirkinių pirkėjas įsigyja ne pas juos, o kitur. Vakarų šalių, kur nepatikusių ar nekokybiškų prekių grąžinimas yra įprastas dalykas, prekybos tinklai nori susekti atvejus, kur vienas pirkėjas, kasininkas arba jų „tandemas“ vykdo grąžinimo operacijas gan dažnai. Neretas atvejis, kai „pirkėjas“ prekę vienoje parduotuvėje įsigyja pigiau, o kitoje, ją grąžindamas, gauna didesnę kainą. Prekybos tinklas, tokius reiškinius pastebėjęs, pradeda juos sekti, samdo detektyvus ir t. t.

Visiems šiems uždaviniams spręsti reikia duomenų. Dalies reikalingų duomenų prekybos tinklai neturi. Todėl Vakarų šalyse populiarios firmos, kurios renka, tikrina ir pardavinėja duomenis.

1.1.2. Interneto duomenų analizė

Kitas, bet jau daug naujesnis, uždavinys – tai interneto duomenys, kurie lengvai prieinami ir kaupiami jau daugelį metų. Todėl nauja duomenų analizės mokslo šaka – interneto duomenų gavyba (angl. *Web Data Mining*) – dabar tapo labai populiari. Analizuojant interneto duomenis, atliekami tie patys veiksmai kaip ir su kitais duomenimis, tačiau tam turintys tikti analizės žingsniai yra saviti ir unikalūs. Interneto duomenų analizė susideda iš toliau išvardytų procesų (žr. 1 pav.):

- duomenų surinkimas,
- duomenų apdorojimas, švarinimas (šis etapas interneto duomenų analizėje yra unikalus),
- duomenų analizės algoritmų taikymas žinioms išgauti,
- interneto duomenų vizualizavimas.



1 pav. Interneto duomenų analizės procesas

Interneto duomenų analizė yra bendro duomenų analizės proceso dalis, tačiau kai kurie etapai skiriasi.

Galima išskirti tris pagrindines interneto duomenų gavybos kryptis:

- turinio duomenų gavyba (angl. *Web Content Mining*),
- struktūrinė duomenų gavyba (angl. *Web Structure Mining*),
- vartojimo duomenų gavyba (angl. *Web Usage Mining*).

Turinio duomenų gavyba yra duomenų išgavimas iš interneto duomenų dokumentų turinio. Taikymo sritis – vartotojų paieškos užklausų turinio tobulinimas ir pagreitinimas.

Struktūrinė duomenų gavyba yra susijusi su internetinės svetainės struktūra / tipologija; ji nagrinėja ryšius tarp atskirų svetainės puslapių.

Vartojimo duomenų gavyba yra duomenų gavybos metodų taikymas, naudojamas vartojimo šablonams atskleisti iš turimų (istorinių) interneto svetainių serveryje laikomų duomenų. Pvz., randamos dažniausiai pasikartojančios puslapių sekos – juose sudedama vertingiausia informacija.

1.1.3. Nuolatiniai aplinkos pokyčiai ir adaptavimasis prie jų

Pasaulis keičiasi. Ypač tai pastebime dabar, labai spartaus naujų informacinių technologijų vystymosi ir aktyvaus žmogaus poveikio gamtai laikotarpiu. Dabartiniu metu stebimas gan spartus Žemės klimato kitimas. Numatoma, kad artimiausią penkiasdešimtį metų klimatas keisis dar sparčiau. Iš tradicinės finansų sistemos liko „šipuliai“, dabar ją nusako ne tik ekonomikos raida, bet ir politika bei spartus informacinių ir ryšių technologijų vystymasis. Visi „finansininkai“ gauna tą pačią informaciją, visi naudoja galingus kompiuterius, panašius matematinius metodus, visi konkuruoja tarpusavyje. Tad naujų galingesnių kompiuterių ir tobulesnių matematinių metodų pasirodymas daro įtaką rinkai. Taigi senesni, finansų rinką aprašantys, duomenys analizei, prognozavimui nebetinka. Reikia naujo požiūrio į duomenų analizę. Nesant pakankamai duomenų, apibūdinančių tiriamus reiškinius po pokyčių, nestebėtų praeityje, kai kuriais atvejais taikomi metodai, leidžiantys reikalingus duomenis imituoti, tiriamą problemą modeliuojant kompiuteriu. Tad atsiranda naujos mokslo šakos, tokios kaip dirbtinė (*artificial*) ekonomika, dirbtinė istorija, dirbtinė chemija, dirbtinė gyvybė, dirbtinės visuomenės, sintetinė biologija ir pan.

Akivaizdu, jei tiriama probleminė sritis laikui bėgant nesikeičia, matematiniam klasifikavimo ar prognozavimo algoritmui (taisyklei) sudaryti reikia surinkti kuo daugiau duomenų. Tada galėsime tiksliau aprašyti tikrovę. Todėl ir algoritmas bus tikslesnis. Bet ką daryti, jei tiriama probleminė sritis kinta? Tada reikia naudoti mažiau ir tik pačius naujausius duomenis. Kuo mažiau duomenų, tuo paprastesnis turi būti naudojamas algoritmas. Štai ir viena iš „algoritmo patikimumo“

problemų, kurią nagrinėsime šiame kurse, – kaip rasti „aukso vidurį“, kaip esant ribotam (nedideliam) duomenų kiekiui sudaryti reikiamo sudėtingumo klasifikavimo ar prognozavimo taisykles.

Duomenų analizės metodų sukurta labai daug. Šioje knygoje aptarsime pagrindinius. Kad būtų aiškiau, praleisime kai kurias statistikos subtilybes, darbą su praleistais stebėjimais, įvairiatipiais požymiais, saviorganizuojančius (Kohoneno) ir rekurentinius neuroninius tinklus. Kalbėdami apie *sprendimo medžius*, genetinius algoritmus, spindulinių bazinių funkcijų neuroninius tinklus, bendram jų supratimui įgyti paminėsime tik jų sudarymo principus ir pagrindines idėjas. Šiais klausimais nukreipsime skaitytoją į panašiai besispecializuojančių kitų autorių paskaitų konspektus ir paskelbtas monografijas. Pagrindinis dėmesys šioje knygoje kreipiamas į duomenų analizės rezultatų patikimumą, būdus, kaip išvengti klaidingų išvadų, trikdančių ne tik duomenų analitiką, bet ir jo užsakovą.

1.2. Pagrindiniai uždaviniai

1.2.1. Klasifikavimo uždavinys. Jo statistinis formulavimas

Dažnai susiduriame su klasifikavimo (atpažinimo) uždaviniu. Turime objektą ar procesą, kurį pagal tam tikrus jo matavimus galime priskirti vienai iš daugelio klasių. Pavyzdžiui, ryt *lis* ar *nelis*. Akivaizdu, jog tam, kad būtų atsakyta į šį klausimą, reikia daug ką žinoti: Koks šiandien oras? Koks oras aplinkinėse šalyse? Žodžiu, reikia turėti išsamią šios ir pastarųjų dienų informaciją. Tai būtų: oro temperatūra, slėgis, vėjo greitis gan plačiame Europos regione, ciklono / anticiklono charakteristikos, metų laikotarpis ir pan. Formaliai tai būtų daugiamatis vektorius $\boldsymbol{x}$ (čia ir toliau juodu šriftu žymėsime vektorius, matricas), kurio komponentės $x_1, x_2, \ldots, x_p$ yra parametrai (charakteristikos, požymiai), aprašantys šios dienos, valandos ir praėjusio laikotarpio informaciją. Taigi konkretų vektorių $\boldsymbol{x}$ reikia priskirti vienai iš dviejų klasių. Geras meteorologas, turėdamas šią informaciją, galbūt ir tiksliai prognozuos rytdienos orą. Tam jis panaudos turimas teorines žinias, patyrimą, intuiciją. Mūsų uždavinys – panagrinėti, kaip šį už-

davinį (požymių vektoriaus $\boldsymbol{x}$ priskyrimą vienai ar kitai klasei) reikėtų spręsti formaliai, taikant duomenų analizės metodus.

Sprendžiant šį uždavinį į kuriamą klasifikavimo taisyklę reikėtų įtraukti ką tik minėtas „teorines žinias“, „patyrimą“ ir „intuiciją“. Panagrinėkime, kaip šią problemą sprendžia statistika. Čia daroma prielaida, kad vektorius $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)^T$ yra atsitiktinis. Ženklas „^{T}“ reiškia, kad vektorius eilutė $(x_1, x_2, \ldots, x_p)$ yra transponuota, t. y. jos elementai $x_1, x_2, \ldots, x_p$ išdėstyti kaip stulpelis. Be to, daroma prielaida, kad šis vektorius pasiskirstęs pagal normalųjį pasiskirstymą. Tai reiškia, kad p-mačio vektoriaus pasiskirstymo dėsnis yra charakterizuojamas p vidurkių $\mu_1, \mu_2, \ldots, \mu_p$ ir p dispersijų $\sigma_1^2, \sigma_2^2, \ldots, \sigma_p^2$. Tačiau to dar neužtenka. Tarp p vektoriaus komponenčių yra $p(p\text{-}1)/2$ koreliacijų – ρ_{ij}, $i, j = 1, 2, \ldots, p$, $j \neq i$. Kad kompaktiškai surašytume visą šią informaciją, naudosime matricų užrašymą. Tai darome ir dėl to, kad vėliau, spręsdami duomenų analizės uždavinius, vartosime *Matlab'o* programavimo (uždavinių užrašymo) kalbą, kuria labai patogu dirbti su vektoriais ir matricomis.

Taigi vektoriaus $\boldsymbol{x}$ vidurkį žymėsime juoda graikiška raide $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_p)^T$, dispersijas ir koreliacijas surašysime į matricą $\boldsymbol{\Sigma}$, susidedančią iš p eilučių ir p stulpelių (iš viso p^2 elementų). Taigi vektoriaus $\boldsymbol{x}$ pasiskirstymą, jeigu jis normalusis, galime aprašyti dviem matriciniais parametrais – $\boldsymbol{\mu}$ ir $\boldsymbol{\Sigma}$. Tai bus labai svarbu, kai spręsime praktinius uždavinius taikydami *Matlab'ą*. Toliau pateikiame užrašymą kaip vektorių stulpelį ir matricą:

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \ldots \\ \mu_p \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & ,,, & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & \ldots & \sigma_{p1} \\ \ldots & \ldots & \ldots & \ldots \\ \sigma_{p1} & \sigma_{p2} & \ldots & \sigma_{pp} \end{bmatrix}, \qquad (1.1)$$

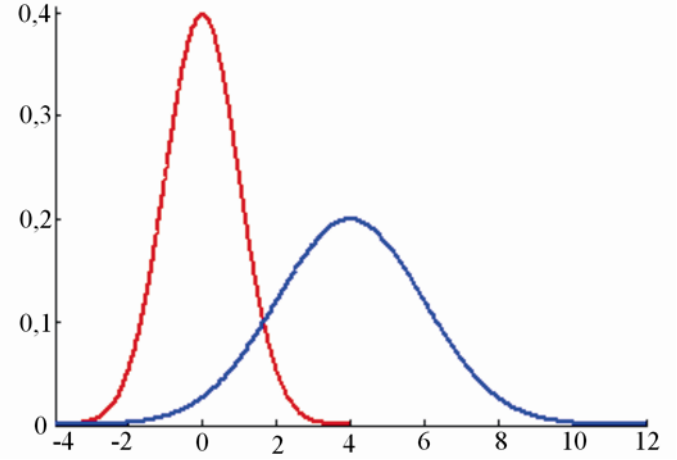
kur pažymėta

$\sigma_{ii} = \sigma_i^2$,

$\sigma_{ij} = \sigma_i \times \sigma_j \times \rho_{ij}$, $i, j = 1, 2, \ldots, p$; $j \neq i$.

Normalųjį, dažnai jį vadinsime *gausiniu*, pasiskirstymo tankį žymėsime $\varphi(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$.

Statistikos atveju informacija apie uždavinį yra: vienos klasės vidurkių vektorius $\boldsymbol{\mu}_1$ ir kovariacinė matrica $\boldsymbol{\Sigma}_1$; antros klasės vidurkių vektorius $\boldsymbol{\mu}_2$ ir kovariacinė matrica $\boldsymbol{\Sigma}_2$. Supaprastintu (vienmačiu) atveju (tada $p = 1$) tuos du skirtingų klasių tankius galime pavaizduoti grafiškai. Iš toliau pateikiamo 2 pav. matome, kad pirmai (raudonai) klasei x priskirsime tada, kai jis neviršija 1,8. Priešingu atveju – x priskirsime antrai klasei (kai $x > 1{,}8$ antros klasės tankis didesnis).



2 pav. Dviejų klasių tankiai

Populiariai aiškindami sprendimo priėmimo taisyklę, darėme prielaidą, kad abiejų normaliųjų pasiskirstymų parametrai $\boldsymbol{\mu}_1$, $\boldsymbol{\Sigma}_1$ ir $\boldsymbol{\mu}_2$, $\boldsymbol{\Sigma}_2$ yra žinomi. Tikrovėje taip nėra. Šiuos parametrus „sužinosime“ (įvertinsime) tik surinkę eksperimentinius (empirinius) duomenis ir pritaikę matematinės statistikos metodus. Ką tik minėtą sprendimo būdą galima taikyti ir kitiems, panašiems į šį, klasifikavimo uždaviniams spręsti. Praktiniuose uždaviniuose požymių (vektoriaus $\boldsymbol{x}$ komponenčių) skaičius būna dešimtys, šimtai, o kai kuriais atvejais siekia ir dešimtis tūkstančių. Klasių gali būti ne tiktai dvi, bet keletas ar net

dešimtys, o pavieniais atvejais ir daugiau. Štai keletas pavyzdžių. Variklio gedimų diagnozavimas pagal jo keliamą triukšmą, kurį apibūdinti pasirinktas jo spektrinio tankio atskaitymas. Vėžinių susirgimų diagnostika pagal tūkstančius paciento genų „išraiškų“ (angl. *gene expression*). Įmonės bankroto prognozavimas (bankrutuos, nebankrutuos, neaišku) pagal jos (įmonės) ir „gamybinės aplinkos“ ekonominius duomenis.

1.2.2. Prognozavimo uždavinys. Jo formulavimas

Klasifikavimo uždavinyje pagal turimą vektorių $\boldsymbol{x}$ reikia prognozuoti kokybinę kintamojo y reikšmę, pvz., 1 arba 2 (dviejų klasių atveju). Skaičių atpažinimo atveju kintamasis y priimtų vieną iš 11 reikšmių: 0, 1, 2, ... , 8, 9 arba „nežinau“. Praktikoje aptinkame nemažai uždavinių, kur prognozuojamas kintamasis y yra tolydinė reikšmė, pavyzdžiui, kietumas, trapumas, temperatūra, slėgis, kuro sunaudojimas, pelnas ir pan. Tai prognozavimo uždavinys, kuriame reikia rasti priklausomybę $y = f(x)$. Funkcinė priklausomybė ir yra tos žinios, kurias reikia „išgauti“ iš eksperimentinių (empirinių) duomenų.

Štai pavyzdys. Trijuose Lietuvos miestuose žiemą kasdien buvo registruojami trys parametrai: x_1 – vėjo greitis, x_2 – oro temperatūra ir y – dujų sunaudojimas mikrorajone, turinčiame 10 000 gyventojų (perskaičiuota, normalizuota reikšmė). Buvo spėjama, kad tada (dar tarybiniais laikais) šaltu oru gyventojai butus šildosi dujomis. Priklausomybei $y = f(x)$ aprašyti pasirinkta paprasta tiesinė lygtis: $y = x_1 w_1 + x_2 w_2 + w_0$. Nežinomiems koeficientams w_0, w_1 ir w_2 rasti turėjome $n = 270$ stebėjimų (po 90 dienų kiekviename iš 3 miestų), x_{1j}, x_{2j}, y_j, $j = 1, 2, \ldots , 270$, o koeficientų w_0, w_1 ir w_2 ieškojome minimizuodami kvadratinių nuokrypių sumą (vidutinį kvadratinį nuokrypį, *VKN*).

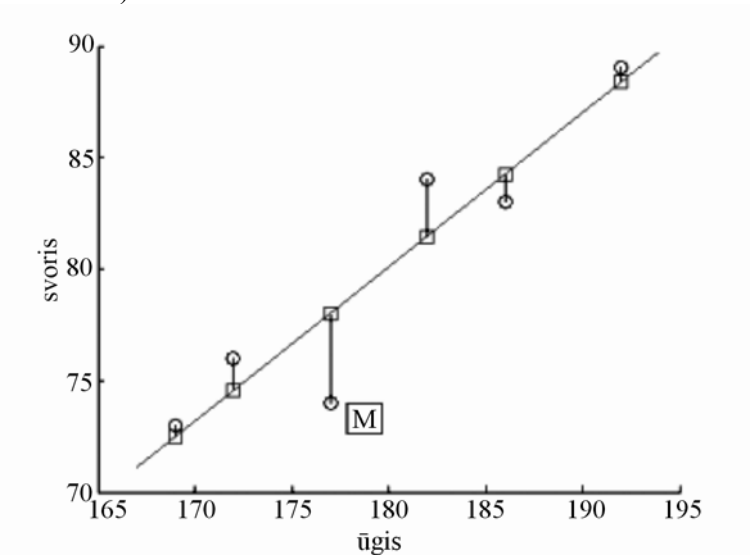
$$VKN = \frac{1}{n}\sum_{j=1}^{n}(y_j - (x_{1j}w_{1j} + x_{2j}w_{2j} + w_0))^2 \quad . \tag{1.2}$$

Kiti prognozavimo uždavinių pavyzdžiai: a) cheminės reakcijos rezultato prognozavimas atsižvelgiant į joje dalyvaujančių medžiagų koncentraciją, slėgį, temperatūrą; b) širdies operacijos rezultato (kiek

metų pacientas išgyvens) prognozavimas atsižvelgiant į paciento būklę aprašančius parametrus ir lėšų, numatomų išleisti operacijai atlikti, kiekį; c) kurios nors kompanijos akcijų kurso prognozavimas atsižvelgiant į šalies ir pasaulio ekonominę situaciją, politinius įvykius ir pan. Kad geriau pajustume „prognozavimo kokybės kriterijaus“ (1.2) esmę, 3 pav. jį iliustruosime grafiškai. Tarkime, vasarą šešetas jaunų žmonių besiginčydami sugalvojo rasti priklausomybę tarp ūgio ir svorio. Kiekvienas pasisvėrė, išsimatavo ūgį. Štai jų duomenys:

Ūgis:	177	169	186	172	182	192
Svoris:	74	73	83	76	84	89

3 pav. pateikiame šešių lentelėje minėtų matavimų rezultatus (mėlyni rutuliukai).



3 pav. Prognozavimo uždavinio iliustracija

Kad rastume lygties $y = x_1 w_1 + w_0$ (x_1 aprašo ūgį, y – svorį) koeficientus, diferencijavę pagal nežinomus koeficientus w_1 ir w_0 lygtį (1.2), prilyginę abu diferencialus nuliui ir išsprendę dviejų lygčių sistemą, gauname: $w_1 = \sigma_{12}/\sigma_{22}$, $w_0 = \mu_2 - \mu_1 w_1$. Parametrai σ_{12}, σ_{22}, μ_1, μ_2

mums nežinomi, bet juos galime įvertinti iš ką tik minėtų empirinių duomenų. Atlikę skaičiavimus gauname: $w_0 = -44{,}2827$, $w_1 = 0{,}6908$. Štai ir prognozavimo lygtis:

$$y = 0{,}69\, x_1 - 44{,}3.$$

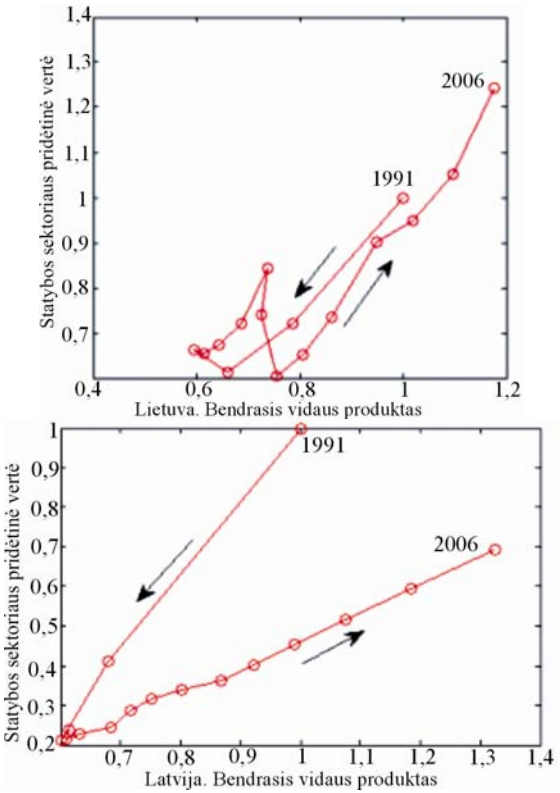
3 pav. taip pat pateikėme ir prognozuotas reikšmes (raudoni kvadratukai), nubrėžėme prognozavimo lygtį (raudona tiesė) ir nurodėme paklaidas. Tai mėlynos tiesės, jungiančios išmatuotas ir prognozuotas reikšmes. Būtent jų kvadratų sumą mums ir reikėjo minimizuoti, turint tik vieną prognozuojamą kintamąjį, prognozavimo lygtis – tiesė. Jei kintamųjų yra du – uždavinys bus sprendžiamas trimatėje erdvėje. Šiuo atveju prognozuosime su plokštumos pagalba. Kai rodiklių (požymių), pagal kuriuos sudaroma prognozės lygtis, yra daugiau nei du, prognozuojame jau naudodami hiperplokštumą. Terminas „hiperplokštuma“ atsirado anksčiau nei „hypermaxima“.

Pateikta iliustracija aiškiai rodo, kad vieno individo (pažymėto raide M) empiriniai matavimai akivaizdžiai yra nukrypę nuo stebimo dėsningumo. Galima manyti, kad jie dėsningumą iškreipia. Logiška išvada: šį stebėjimą galbūt reikėtų ignoruoti arba bent kokiu nors būdu jo įtaką sumažinti. Čia šį faktą paminėjome tik kaip vieną iš daugelio problemų, kylančių sprendžiant klasifikavimo ir prognozavimo uždavinius. Tai vadinamoji robastiškumo, arba „didelių nuokrypių“, problema. Apie šią problemą kalbėsime vėliau. Kita problema – tai duomenų reprezentatyvumas. Šeši stebėjimai – dar ne statistika. Kiek jų reikia, kaip juos rinkti ir kaip neapsirikti renkant „tendencingai“ – tik iš tam tikros grupės žmonių? Ką daryti, jei laikui bėgant pasaulis, o kartu ir duomenys, keičiasi? Koks ryšys tarp reikalingo dėsningumui nustatyti duomenų kiekio ir prognozavimo lygties sudėtingumo? Tai aibė klausimų, dažnai netgi neišspręstų problemų, kurios daro didelę įtaką duomenų analizės patikimumui. Apie tai bus kalbama vėliau.

1.2.3. Blokinių sudarymo (klasterizavimo) uždavinys

Dažna klaida, daroma analizuojant empirinius duomenis, yra nesąmoningai keliama prielaida, kad duomenys yra vienalyčiai (homogeniški). 3 pav. iliustruoti duomenys yra vienalyčiai, t. y. viena į atskiras dalis nesusiskaidžiusių duomenų grupė. 4 pav. pateikiame statybos

sektoriaus pridėtinės vertės (*y*) ašis – bendro vidaus produkto (*x* ašis) kitimo 1992–2006 metais dvimatį pasiskirstymą Lietuvoje ir Latvijoje. Čia abu veiksniai buvo išreikšti kaip santykis su jų reikšmėmis 1991 metais (1991 metų reikšmės prilygintos vienetui).



4 pav. Statybos sektoriaus pridėtinė vertė atsižvelgiant į bendrąjį vidaus produktą

Atkreiptinas dėmesys į tai, kad pačioje pradžioje duomenys analizei buvo pateikti kaip „metiniai procentiniai pokyčiai“. Todėl jokio dėsningumo nebuvo matyti. Abiejų veiksnių kaitą pateikę ne kaip metinius pokyčius, o kaip jų reikšmių kitimą pamečiui, matome, kad 1991–2006 metais faktiškai vyko du procesai: pirmus kelerius metus

abu rodikliai mažėjo, o paskui abu pradėjo augti. Tai lengva pastebėti iš pateiktų duomenų, ypač kai atskiri taškai sujungti tiesėmis ir pažymėti metai. Jei nagrinėjamų rodiklių (požymių) būtų daugiau, šį dėsningumą pastebėti būtų sunku. Blokinių (*klaster*) analizė ir skirta duomenų susiskaidymui į grupes pastebėti. Ir klasifikuojant, ir prognozuojant iškylančios problemos yra panašios.

Pateiktas pavyzdys puikiai iliustruoja bene pagrindinę duomenų analizės savybę: kad analizė būtų sėkminga, reikia pasirinkti tinkamą „atskaitos tašką“ (žr. knygos viršelį). Šiame pavyzdyje tai buvo: a) perėjimas prie duomenų santykinių reikšmių nagrinėjimo (lyginant su 1991 metais), b) duomenų suskaidymas į dvi dalis.

1.2.4. Vienasluoksnis perceptronas ir jo mokymas

Dirbtinis neuronas, t. y. dar 1943 metais pasiūlytas JAV mokslininkų McCullock'o ir Pits'o smegenų ląstelės, neurono modelis, susideda iš daugelio įėjimų, kurie yra sumuojami padauginus juos iš tam tikrų, svoriais vadinamų, koeficientų – tai „pasvertas“ sumavimas. Gautas rezultatas, *suma*, nukreipiamas į netiesinį, išėjime esantį, elementą, kurio išėjimo signalas lygus arba artimas nuliui, jei signalas *suma* stipriai neigiamas; ir artimas vienetui, jei signalas *suma* yra didelis ir teigiamas. Šia prasme neurono modelis yra labai panašus į 1.2.2 poskyryje aprašytą tiesinę prognozavimo taisyklę, kuri bendru (p požymių) atveju būtų užrašoma taip:

$$t = x_1 w_1 + x_2 w_2 + \ldots + x_p w_p + w_0. \qquad (1.3)$$

Nežinomų koeficientų w_0, w_1, w_2 ieškojome minimizuodami kvadratinių nuokrypių sumą (vidutinį kvadratinį nuokrypį, VKN). Tai minimaliųjų kvadratų metodas. Dabartiniu metu pasaulyje plačiai naudojami dirbtiniai neuroniniai tinklai (DNT), paremti iš gamtos „pasiskolintomis“ idėjomis. Čia žodį *pasiskolinti* vartojame todėl, kad tyrimų rezultatai, gauti analizei naudojant DNT, gali būti taikomi tiriant gyvųjų organizmų smegenų ląsteles, neuronus, taip pat patį jų „mąstymo“ (informacijos apdorojimo) procesą. Taip skola gali būti „grąžinta“.

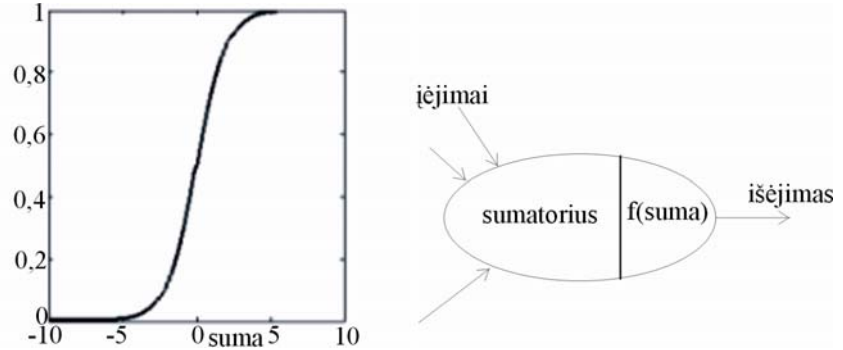
Paprasčiausiame DNT vienasluoksnio perceptrono (VsP) modelyje įėjimo signalai irgi sumuojami, padauginus juos iš atitinkamų koe-

ficientų – w_1, w_2, ... , w_p (žr. 5 pav.), beje, dirbtiniuose neuroniniuose tinkluose šie koeficientai vadinami *svoriais*. Skirtingai nuo ankstesnio modelio, pasverta suma

$$suma = x_1w_1 + x_2w_2 + ... + x_pw_p + w_0,$$

prieš patekdama į VsP išėjimą, dar netiesiškai apdorojama:

$$išėjimas = f(suma).$$



5 pav. Netiesinė aktyvavimo funkcija. Dirbtinio neurono schema

Norėdami rasti nežinomus svorius, vietoj kvadratinių nuokrypių sumos naudojame šiek tiek sudėtingesnę funkciją, kurią vadiname „nuostoliais“, kitaip sakant, tai „vidutinė“ paklaida:

$$nuostoliai = \frac{1}{n}\sum_{j=1}^{n}(t_j - f(x_{1j}w_{1j} + x_{2j}w_{2j} + w_0))^2 \quad , \qquad (1.4)$$

DNT atveju t vadinamas „trokštamu išėjimu“.

Jei t priima diskretines reikšmes, pvz., „nulį“ ar „vienetą“, turime klasifikavimo uždavinį. Jei t priima tolydines reikšmes, priedo $f(suma) = suma$, turime prognozavimo uždavinį.

Jei t yra vienodas visiems mokymo vektoriams (pvz., $t = 0{,}5$), turime blokinių sudarymo (klasterizavimo) į dvi klases uždavinį.

Nuostolių funkcijoje figūruoja netiesiškumas, $f(suma)$, taigi nuostolių funkcija turi daug ekstremumų (minimumų), ir ieškomi svoriai negali būti rasti analitiškai šią funkciją diferencijuojant. DNT'uose ieškomi svoriai paprastai randami iteracijų būdu:

$$naujas\ svoris = senas\ svoris + pataisymas.$$

Apie tai kalbėsime kituose skyriuose.

ANTRAS SKYRIUS. TIESINIAI PROGNOZAVIMO ALGORITMAI

2.1. Regresijos lygtis. Minimaliųjų kvadratų metodo taikymas tiesinei prognozavimo lygčiai gauti. Parametrai, naudojami tikslumui įvertinti

2.1.1. Minimaliųjų kvadratų metodas

Kad rastume nežinomus tiesinės prognozavimo lygties koeficientus, turime pasirinkti nuostolių funkciją. Anksčiau minėjome kvadratinių nuokrypių sumos funkciją (žr. 1.2 formulę)

$$nuostoliai = \frac{1}{n}\sum_{j=1}^{n}(y_j - suma(x_j))^2 \ , \qquad (2.1)$$

kur s*kirtumas*

$$(y_j, suma(x_j)) = y_j - (x_{1j}w_{1j} + x_{2j}w_{2j} + ... + x_{pj}w_{2j} + w_0) \ ,$$

ir naudojome kvadratinę nuostolių funkciją (juoda kreivė 6 pav.). Iš esmės įmanomos ir kitokios funkcijos. Apie jas kalbėsime kitame poskyryje.

Svertinę sumą

$$suma(x_j) = x_{1j}w_{1j} + x_{2j}w_{2j} + ... + x_{pj}w_{2j} + w_0$$

galime užrašyti daug trumpiau, būtent vektoriniu pavidalu:

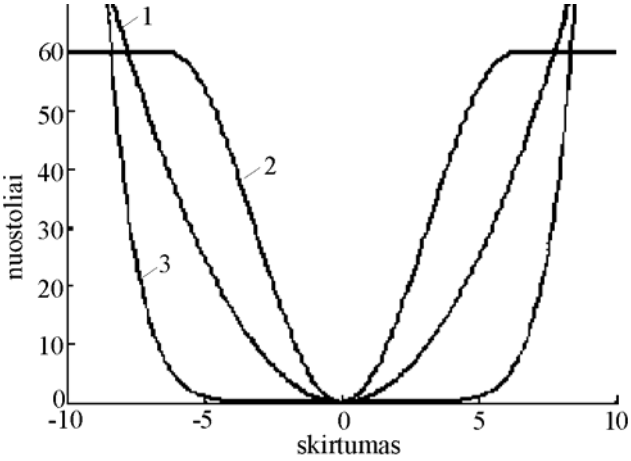
$$suma\ (x_j) = \boldsymbol{v}^T \boldsymbol{z}_j \ ,$$

kur (p+1)-matis vektorius $\boldsymbol{z}$ – tai vektorius $\boldsymbol{x}$, papildytas „vienetu“, t. y.

$$\boldsymbol{z} = [x^T, 1]^T \ ,$$

o (p+1)-matis vektorius $\boldsymbol{v}$ – tai vektorius w, papildytas w_0, t. y.

$$\boldsymbol{v} = [\boldsymbol{w}^T, w_0]^T .$$



6 pav. 1 (kvadratinė, juoda), 2 (robastinė, mėlyna) ir 3 (minimaksinė, raudona) nuostolių funkcijos

Čia ir kituose šio vadovėlio puslapiuose, kad lengviau galima būtų skirti linijas, kompiuterio ekrane reikėtų paanalizuoti spalvotus grafikus, pateiktus Klaipėdos universiteto Gamtos ir matematikos mokslų fakulteto virtualioje mokymosi aplinkoje (paskaitų kursas 908M03 „Statistinė duomenų analizė ir žinių išgavimas“, http://vma.ku.lt/moodle/course/index.php).

Naudodami žymėjimą su vienetų stulpeliu, ir duomenyse, kuriuose ieškome geriausios vektoriaus $\boldsymbol{v}$ reikšmės, teks pridėti stulpelį, sudarytą iš vienetų. Dabar ir mokymo duomenis užrašysime kaip vektorių stulpelį ir matricą:

$$\mathrm{D}y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}, \mathrm{D}z = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} & 1 \\ x_{21} & x_{22} & \dots & x_{2p} & 1 \\ \dots & \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{np} & 1 \end{bmatrix}. \qquad (2.2)$$

Taikant matricinį užrašymą, aukščiau minėta kvadratinių nuokrypių sumos funkcija reiškiama labai paprastai:

$$nuostoliai = (\mathbf{D}y\text{-}\mathbf{D}\boldsymbol{z}\times\boldsymbol{v})^T\times(\mathbf{D}y\text{-}\mathbf{D}\boldsymbol{z}\times\boldsymbol{v}). \quad (2.3)$$

Aiškumo dėlei išraiškoje (2.3) įdėjome daugybos ženklą, kuris šiuo konkrečiu atveju reiškia matricų daugybą. Toliau šioje knygoje taupydami vietą ir atsižvelgdami į aplinkybes svorių vektorių žymėsime pakaitomis tai $\boldsymbol{v}$, tai $\boldsymbol{w}^T$, w_0.

Norint rasti vektorių $\boldsymbol{v}$ (faktiškai $\boldsymbol{w}$ ir w_0) išraiškos (2.3) išvestinę pagal $\boldsymbol{v}$ – (p-1)-matį vektorių, vadinamą gradientu, reikia jį prilyginti nuliui. Praleisdami matematinius išvedimus, užrašysime gautą rezultatą:

$$\boldsymbol{w} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_{xy},\ w_0 = \mu_2 - \mu_1\boldsymbol{w}, \quad (2.4)$$

kur šiuo atveju (kai vektorius $\boldsymbol{v}$ ieškomas iš empirinių duomenų) $\boldsymbol{\Sigma}$ yra duomenų

$\mathbf{D}\boldsymbol{x}$ = $\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}$ kovariacinės matricos įvertis;

$\boldsymbol{\Sigma}^{-1}$ yra apverstoji $\boldsymbol{\Sigma}$ matrica;

$\boldsymbol{\Sigma}_{xy}$ yra kovariacija tarp vektoriaus $\boldsymbol{x}$ ir t.

Matricinis užrašymas yra labai patogus skaičiavimus atliekant *Matlab'u*. Skyriuje „Laboratoriniai darbai" skaičiavimai, atliekami pagal formulę (2.4), bus išaiškinti detaliau.

Pirmojo skyriaus 3 pav. prognozavimo uždavinys buvo iliustruotas vienmačiu atveju (kai p=1). Ten matėme mėlynas linijas, jungiančias išmatuotas ir prognozuotas reikšmes. Būtent jų kvadratų sumą ir minimizavome, prilyginę nuostolių (2.3) išvestinę nuliui.

2.1.2. Parametrai, naudojami prognozavimo tikslumui įvertinti

Sudarant prognozavimo lygtį pagal empirinius duomenis, visada kyla klausimas: koks yra prognozavimo tikslumas? Kad į jį atsakytume, turime pasirinkti kriterijų (matą), pagal kurį spręsime apie tikslumą. Paprasčiausias matas – tai vidutinė kvadratinė paklaida (*VKP*)

$$VKP = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - suma(x_j))^2} \quad . \tag{2.5}$$

Čia būtina atkreipti dėmesį į tai, kaip, kokiu būdu skaičiuojama vidutinė kvadratinė paklaida. Šią paklaidą galime skaičiuoti naudodami empirinius duomenis, kuriuos pasitelkę sudarėme ir prognozavimo lygtį. Toks būdas lengvas, tačiau jame glūdi galima apgaulė. Mat sudarant prognozavimo lygtį pagal empirinius duomenis, prie jų prisiderinama, ir skaičiuojant pagal formulę (2.5) gautas rezultatas bus „optimistinis“, ypač jei tų empirinių duomenų yra nedaug. Faktiškai prognozavimo paklaida gali būti daug didesnė. Siekdami išaiškinti šio prisiderinimo prie mokymo duomenų esmę, toliau pateiksime keletą teoriškai nustatytų faktų.

Visų pirma kalbėsime apie „tikrąją prognozavimo paklaidą“, kurią žymėsime σ_n. Tai paklaida, įvertinta iš nepriklausomų, mokymo procese nenaudotų duomenų. Teoriškai nustatyta, kad tikroji prognozavimo paklaida priklauso nuo mokymo duomenų kiekio n (dėl to ir prie jos dedame apatinį indeksą n), nuo požymių kiekio p ir nuo minimalios prognozavimo paklaidos σ_0. Minimali prognozavimo paklaida σ_0 – abstrakcija. Tai prognozavimo paklaidos riba, jei mokymo duomenų kiekis būtų begalybė. Teoriškai nustatyta, kad jei duomenys yra pasiskirstę pagal normalųjį dėsnį ir požymių p yra daug, tai galioja apytikslė lygybė:

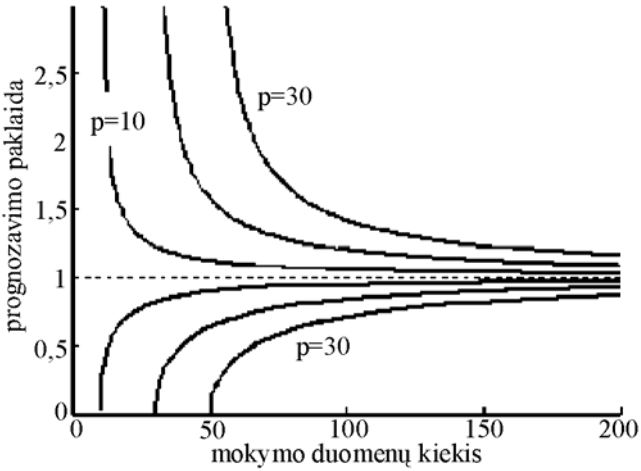
$$\sigma_n \approx \sigma_0 \sqrt{\frac{n}{n-p}} \quad . \tag{2.6}$$

Lygtis (2.6) rodo, kad kuo daugiau duomenų, tuo faktinė prognozavimo paklaida σ_n yra arčiau savo ribinės (minimalios) reikšmės σ_0.

Lygtis (2.6) taip pat rodo, kad prognozavimo tikslumas labai priklauso nuo skirtumo *n-p*. Jei *p* artimas *n*, tikslumas gali būti labai žemas. Jei *p>n,* anksčiau aprašytu standartiniu būdu prognozavimo lygties iš viso negalėsime sudaryti: nesugebėsime apversti iš mokymo duomenų įvertintos kovariacinės matricos **S**.

Šį „nesugebėjimą“ lengva paaiškinti trimačiu atveju (*p*=3). Tarkime, turime tik tris mokymo vektorius. Trimatėje erdvėje jie guli plokštumoje. Vadinasi, kryptyje, statmenoje tai plokštumai, visų trijų mokymo taškų išsibarstymas bus lygus nuliui. Kalbant matematiniais terminais, tai rodo, kad duomenų kovariacinė matrica bus „išsigimusi“, o jos determinantas bus lygus nuliui, Panašiai galvodami, turėdami šimtą mokymo vektorių šimtamatėje erdvėje, susidursime su ta pačia problema: kovariacinė matrica bus išsigimusi, ir prognozavimo lygties sudaryti negalėsime.

7 pav. pateikiame faktinės prognozavimo paklaidos σ_n priklausomybės nuo mokymo duomenų kiekio grafiką (viršutinės trys kreivės, pažymėtos raudonai), kuris buvo apskaičiuotas pagal formulę (2.6), kai σ_0=1, o požymių kiekis *p* = 10, 20 arba 50.



7 pav. Tikrosios (raudona) ir įsivaizduojamos (mėlyna) prognozavimo paklaidų priklausomybė nuo mokymo duomenų kiekio

Kaip minėjome, prognozavimo paklaida σ_{mok}, įvertinta su mokymui jau panaudotais duomenimis, yra „optimistiška“ – jos reikšmės mažesnės, nei turėtų būti iš tikrųjų. Štai apytikslė formulė jai apskaičiuoti:

$$\sigma_{mok} \approx \sigma_0 \sqrt{\frac{n-p}{n}} . \qquad (2.7)$$

7 pav. mėlyna spalva pateikiame *įsivaizduojamos prognozavimo paklaidos* σ_{mok}, priklausomybės nuo mokymo duomenų kiekio, grafiką, kuris buvo apskaičiuotas pagal (2.7) formulę. Matome, kad mokymo duomenų kiekiui augant, faktinė ir įsivaizduojamoji prognozavimo paklaidos labai priartėja viena prie kitos. Tačiau esant mažam mokymo duomenų kiekiui, jos labai skiriasi. Šiuos skirtumus ignoruodami galime labai apsirikti: apgauti save ir kitus. Praktiniame darbe (ir teoriniuose nagrinėjimuose) tai visada reikia atsiminti.

Vidutinė kvadratinė paklaida viena beveik nieko nesako apie prognozavimo tikslumą. Ją reikia lyginti su prognozuojamo rodiklio, požymio y, kitimu. Jei požymio y vidutinis kvadratinis nuokrypis (kvadratinė šaknis iš dispersijos) $\sigma_y >> \sigma_n$, tai reikia manyti, kad prognozavimas yra geras. Žodis *geras* – subjektyvi sąvoka. Kad būtų įvertinta, ar prognozavimas geras, reikia gautą paklaidą lyginti su prognozavimo uždaviniui keliamais reikalavimais, su kitų tyrinėtojų rezultatais.

Populiarus kriterijus (matas), naudojamas prognozavimo tikslumui įvertinti, yra koreliacijos koeficientas. Čia skaičiuojamas koreliacijos koeficientas tarp tikrųjų rodiklio y reikšmių y_{tj} ir prognozuotųjų y_{pj} ($j = 1, 2, \ldots, n$):

$$\rho = \frac{\frac{1}{n}\sum_{j=1}^{n}(y_{tj} - \bar{y}_{tj})(y_{pj} - \bar{y}_{pj})}{\sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_{tj} - \bar{y}_{tj})^2}\sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_{pj} - \bar{y}_{pj})^2}}, \qquad (2.8)$$

kur $\bar{y}_{tj}$ ir $\bar{y}_{tj}$ yra kintamųjų y_{tj} ir y_{pj} vidurkiai.

Koreliacijos koeficientas kinta tarp -1 ir +1. Jei koreliacija $\rho = 1$, tai y_{tj} ir y_{pj} yra vienareikšmiškai priklausomi. Jei $\rho = -1$, tai y_{tj} ir y_{pj} yra vienareikšmiškai neigiamai priklausomi (vienam iš šių rodiklių didėjant, kitas vienareikšmiškai mažėja). Jei $\rho = 0$ ir prognozuojamasis rodiklis pasiskirstęs pagal normalųjį dėsnį, tai priklausomybės tarp y_{tj} ir y_{pj} nėra.

Jei rodiklių pasiskirstymai nėra normalieji, tai maža koreliacija dar nerodo, kad jie nepriklausomi. Štai 4 pav. apačioje pavaizduotas Latvijos statybos sektoriaus pridėtinės vertės ir bendrojo vidaus produkto pasiskirstymas rodo, kad rodikliai beveik vienareikšmiškai priklausomi, nors formaliai žiūrėdami tik į koreliacijos koeficientą matysime, kad jis nėra aukštas. Tai dar kartą atkreipia mūsų dėmesį į tai, kad duomenų analizė turi būti apgalvota. Tai yra kūrybinis procesas.

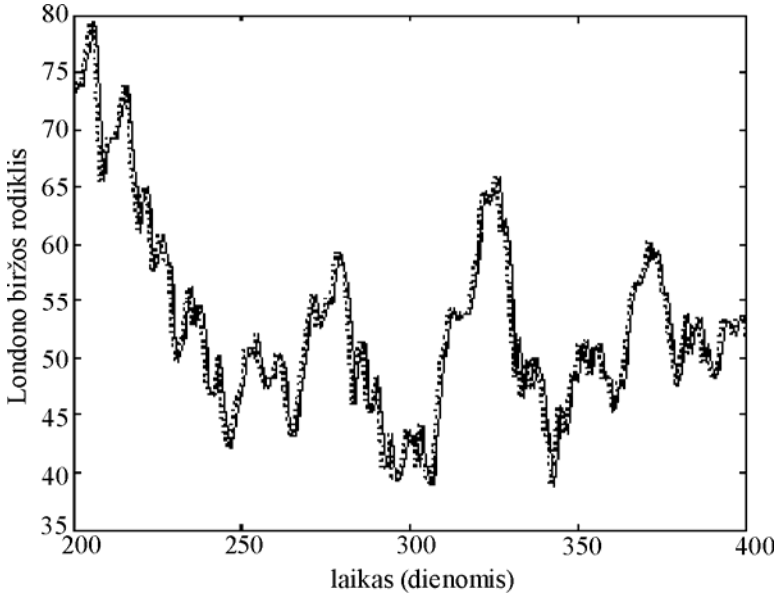
Papildoma problema kyla, kai prognozuojamos laiko eilutės, kur gretimi (laiko arba erdvės požiūriu) daugiamačiai stebėjimai tarpusavyje yra statistiškai priklausomi. Tada prognozavimas pagal principą „prognozuojamo stebėjimo reikšmė lygi prieš tai buvusiai reikšmei“ gali duoti geresnį rezultatą nei formaliai atlikta prognozė, gauta taikant sudėtingą metodą. Kaip šio reiškinio iliustraciją 8 pav. pateikiame vieno iš Londono akcijų biržos rodiklių kitimą dviejų šimtų dienų laikotarpiu (juoda).

Per vieną dieną jo suvėlinta reikšmė (raudona) labai stipriai koreliuoja su tikrąja reikšme. Šiuo atveju prognozavimas pagal principą „prognozuojamo stebėjimo reikšmė lygi prieš tai buvusiai reikšmei“ duos aukštą koreliaciją ir mažą prognozavimo paklaidą. Neįsigilinę, spręsdami tik iš formalių rodiklių (koreliacijos ir vidutinio kvadratinio nuokrypio), manysime, kad labai gerai prognozuojame. O iš tikrųjų „gera prognozė“ atsirado tik dėl tiriamų duomenų specifikos.

2.1.3. Reguliarizuota regresija

2.1.2 poskyryje matėme, kad jei mokymo (empirinių) duomenų yra mažai, palyginti su požymių skaičiumi, tai prognozės lygties tikslumas yra žemas. Ką daryti, jei duomenų iš tikrųjų maža ir daugiau gauti yra labai sunku arba net neįmanoma? Formulės (2.6) ir (2.7) rodo, kad jei pasisektų sumažinti požymių skaičių neprarandant jų informatyvumo, tai prognozavimo tikslumas galėtų išaugti. Tačiau

sumažinti požymių skaičių neprarandant naudingos informacijos nėra lengva. Apie tai dar kalbėsime vėliau.



8 pav. Akcijų biržos rodiklio kitimas 200 dienų laikotarpiu (juoda) ir jo per vieną dieną suvėlinta reikšmė (taškais, raudona)

Vienas iš efektyvių būdų pagerinti prognozavimo kokybę yra taikyti „geresnį“ kovariacinės matricos įvertinimą, pvz., prie iš mokymo duomenų įvertintos kovariacinės matricos **S** diagonalinių elementų pridėti po mažą teigiamą konstantą, tarkime, λ. Tada gauname „reguliarizuotą“ matricą:

$$S_R=S+\lambda\begin{bmatrix}1 & 0 & \dots & 0\\ 0 & 1 & \dots & 0\\ \dots & \dots & \dots & \dots\\ 0 & 0 & \dots & 1\end{bmatrix}. \qquad (2.9)$$

Prisiminę pavyzdį su 3 vektoriais trimatėje erdvėje, minėtą 2.1.2 poskyryje, galime įsivaizduoti, kad nario su λ pridėjimas formulėje (2.9) „praplatina“ taškų pasiskirstymą kryptyje, statmenoje ką tik minėtai trijų mokymo taškų plokštumai“. Kalbant matematiniais terminais, tai rodo, kad kovariacinės matricos įverčio $\mathbf{S}_R$ determinantas tampa nebelygus nuliui. Tokiu atveju regresijos lygties koeficientų vektorius $\boldsymbol{w}$ randamas pagal modifikuotą formulę:

$$\boldsymbol{w}_R = \left[\mathbf{S} + \lambda \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{bmatrix} \right]^{-1} \mathbf{S}_{xy},\ w_0 = \bar{\boldsymbol{x}}_2 - \bar{\boldsymbol{x}}_1 \boldsymbol{w}_R, \qquad (2.10)$$

kur $\mathbf{S}_{xy}$ yra vektoriaus $\boldsymbol{\Sigma}_{xy}$ įvertinimas, o $\bar{\boldsymbol{x}}_2$ ir $\bar{\boldsymbol{x}}_1$ – vektorių $\boldsymbol{\mu}_2$ ir $\boldsymbol{\mu}_1$ įvertinimai.

Dar vienas būdas „pagerinti regresijos lygties įvertinimą – tai prie n mokymo vektorių pridėti $k \times n$ vektorių, gautų iš turimų n vektorių, prie kiekvieno vektoriaus kiekvienos komponentės pridedant po atsitiktinį dydį, pasiskirsčiusį pagal normalųjį dėsnį, kurio vidurkis yra nulis ir dispersija λ. Teoriškai galima parodyti, kad kai $k \to \infty$, abu ką tik aprašyti reguliarizacijos metodai yra adekvatūs. Taikant šiuos metodus praktiškai reikia parinkti geriausią reguliarizacijos parametro λ reikšmę. Žinoma, kad didėjant mokymo duomenų kiekiui, λ turi mažėti. Tačiau kadangi duomenų charakteristikos iš anksto nežinomos, optimalios λ reikšmės parinkimas tampa problemiškas.

2.1.4. Robastinė regresija

3 pav. pateikta prognozavimo uždavinio iliustracija rodo, kad vieno (M-tojo) individo empiriniai matavimai akivaizdžiai nukrypę nuo stebimo dėsningumo. Be abejonės, šį stebėjimą reikėtų truputėlį ignoruoti kokiu nors būdu jo įtaką sumažinant. Tai vadinamoji *robastiškumo (didelių nuokrypių) problema*.

Vienas iš būdų, kaip sumažinti didelių nuokrypių įtaką, yra vietoj kvadratinės nuostolių funkcijos naudoti kitą, kurioje didelių nuokrypių indėlis dirbtinai sumažinamas. 6 pav. mėlyna spalva pavaizduota nuostolių funkcija (2 kreivė) esant dideliems skirtumams tarp tikrųjų ir prognozuotųjų kintamojo *y* reikšmių užsilenkia ir „prisisotina“. Štai jos analitinė išraiška *j*-tajam mokymo vektoriui:

$$nuostoliai = \Psi(y_j - \boldsymbol{v}^T \boldsymbol{z}_j), \tag{2.11}$$

$$\text{kur } \Psi(c) = \begin{cases} 60, & \text{jei } c > 2\pi \\ 30(1-\cos(0.5c)), & \text{jei } c < 2\pi \end{cases}. \tag{2.12}$$

Funkcijomis (2.11) ir (2.12) aprašomą suminę, iš *n* mokymo vektorių apskaičiuotą, nuostolių funkciją minimizuoti analitiškai problemiška. Reikia naudotis iteracinėmis procedūromis, jau minėtomis 1.2.4 poskyryje. Funkcijos (2.12) išvestinė lygi nuliui, kai nuokrypis $abs(y_j - \boldsymbol{v}^T \boldsymbol{z}_j) > 2\pi$, taigi mokymo vektoriai, kurių nuokrypiai didesni nei $nuokrypis_{\max} > 2\pi$, mokymo procese nebedalyvauja. Vienas iš praktinių uždavinių, kylančių sprendžiant robastinės regresijos uždavinius, yra pasakyti, kas yra dideli nuokrypiai. Juk iš anksto $nuokrypis_{\max}$ nėra žinomas. Tokiu atveju problema sprendžiama „bandymų ir klaidų metodu“: bandoma keletas skirtingų $nuokrypis_{\max}$ reikšmių. Prie robastinės regresijos mes dar grįšime, kai kalbėsime apie vienasluoksnį perceptroną, kurio mokymas ir yra pagrįstas iteraciniu mokymusi.

2.1.5. Minimaksinė regresija

Robastinėje regresijoje labai didelių nuokrypių indėlis slopinamas. Praktikoje pasitaiko uždavinių, kai norima minimizuoti blogiausią atvejį, t. y. maksimalųjį nuokrypį. Tai vadinamasis *minimaksinis uždavinio formulavimas*. Trokštamą rezultatą galima pasiekti, jei nuostolių funkcija $nuostoliai = \Psi(y_j - \boldsymbol{v}^T \boldsymbol{z}_j)$ labai augtų, didėjant $abs(y_j - \boldsymbol{v}^T \boldsymbol{z}_j)$.

6 pav. raudona spalva pavaizduota nuostolių funkcija (3 kreivė), kuri esant tam tikriems skirtumams tarp tikrųjų ir prognozuotųjų kintamojo *y* reikšmių pradeda labai stačiai kilti. Štai jos analitinė išraiška $\mathbf{z}_j$-tajam mokymo vektoriui:

$$nuostoliai_j = (y_j - \mathbf{v}^T \mathbf{z}_j)^S, \tag{2.13}$$

kur laipsnio rodiklis *s* turi būti nemažas, bet ir ne per didelis lyginis sveikasis skaičius, pvz., $s = 8$.

Minimizuoti funkcija (2.13) aprašomą suminę, *n* mokymo vektorių apskaičiuotą, nuostolių funkciją analitiškai neišeina, nes ji daugiaekstremali. Kaip ir robastinės regresijos atveju, tenka naudotis iteracinėmis procedūromis. Čia taip pat reikia žinoti, kada nuostolių funkcija turi išsilenkti į viršų. Problema taip pat sprendžiama *bandymų ir klaidų metodu*: bandoma keletas skirtingų *s* reikšmių. Prie minimaksinės regresijos taip pat grįšime, kai kalbėsime apie vienasluoksnį perceptroną.

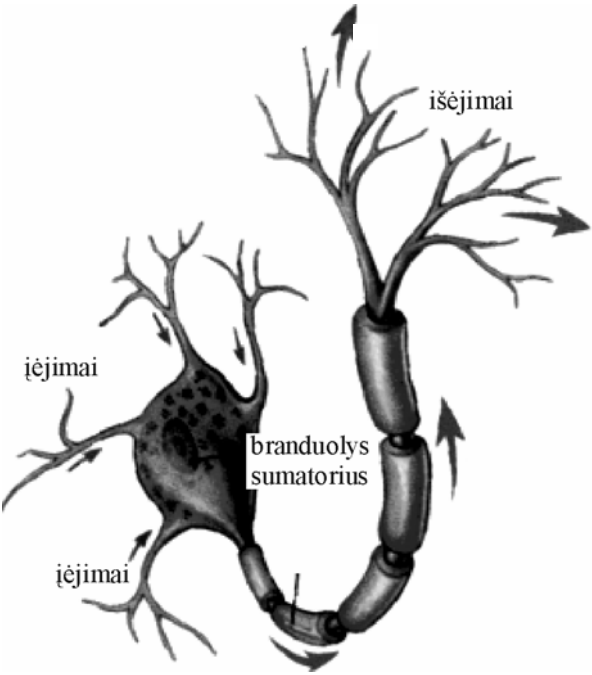
2.2. Vienasluoksniu perceptronu paremto regresijos algoritmo evoliucija iteraciniame mokymo procese

2.2.1. Vienasluoksnis perceptronas – gamtos inspiruotas informaciją apdorojantis elementas

Smegenys sudarytos iš trilijonų sąlyginai paprastų vienetų – neuronų, kurie yra susijungę į didžiulį tinklą. Sprendžiant iš biologinių tyrimų, neuronai atlieka gana nesudėtingą funkciją, tai yra perduoda elektrinį impulsą kitiems neuronams. Kai neuronas gauna impulsą iš greta esančiųjų, jo reakcija priklauso nuo gauto signalo stiprumo ir nuo šio neurono jautrumo tam neuronui, iš kurio gautas signalas. Kai kurie neuronai į dalį impulsų iš viso nereaguoja. Kiti neuronai, kuriems ateinantis signalas yra pakankamai stiprus, gali šalia ar toliau esantiems neuronams persiųsti atitinkamo proporcingo stiprumo impulsą. Taip susidaro sklindančių impulsų banga, kol galiausiai gaunamas pakankamas kiekis vienu metu aktyvuotų neuronų, kurie gali būti

suvokiami kaip mintis, jausmas, poreikis (nurodymas atlikti vieną ar kitą veiksmą, pvz., pabėgti).

1.2.4 poskyryje buvo minėta, kad vienasluoksniu perceptronu (VsP), pačios gamtos inspiruotu informaciją apdorojančiu elementu, galime spręsti ir prognozavimo, ir klasifikavimo, ir blokinių sudarymo (klasterizavimo) uždavinius. Jis turi daug universalumo požymių. Šiame poskyryje pamatysime, kad jis gali realizuoti šešis regresijos sudarymo metodus, o vėliau – ir septynis žinomus bei plačiai naudojamus klasifikavimo algoritmus. Ir tai dar ne riba.



9 pav. Smegenų ląstelė, neuronas, – vienasluoksnio perceptrono prototipas

2.2.2. Vienasluoksnio perceptrono mokymas

Siekiant perceptroną panaudoti klasifikavimo ar prognozavimo uždaviniui spręsti, reikia žinoti jo svorius. Tam reikia jį išmokyti. Panagrinėkime dirbtinio neurono, t. y. vienasluoksnio perceptrono, stan-

dartinę nuostolių funkciją, kai jis naudojamas prognozavimo uždaviniui spręsti:

$$nuostoliai = \frac{1}{n}\sum_{j=1}^{n}(y_j - (x_{1j}w_{1j} + \ldots + x_{pj}w_{pj} + w_0))^2 =$$

$$= \frac{1}{n}\sum_{j=1}^{n}(y_j - \mathbf{z}_j\mathbf{v}_j)^2 . \qquad (2.14)$$

Bene pats paprasčiausias būdas VsP mokyti yra *delta taisyklė*: po kiekvienos iteracijos (svorių pataisos) daroma kita pataisa, proporcinga gradientui, tam tikrai nuostolių funkcijos išvestinei pagal visas p+1 vektoriaus $\boldsymbol{v}$ komponentes. Užrašant matricų pavidalu, tai gana paprasta išraiška:

$$\frac{\partial nuostoliai}{\partial \boldsymbol{v}} = \frac{2}{n}\sum_{i=1}^{n}(y_i - \mathbf{v}^T\mathbf{z}_i)\,\mathbf{z}_i^T = -\frac{2}{n}\sum_{i=1}^{n}(y_i\mathbf{z}_i^T - \mathbf{v}^T\mathbf{z}_i\ \ \mathbf{z}_i^T) =$$

$$= -2\left(\frac{1}{n}\sum_{i=1}^{n}y_i\mathbf{z}_i^T - \mathbf{v}^T\frac{1}{n}\sum_{i=1}^{n}\mathbf{z}_i\ \ \mathbf{z}_i^T\right). \qquad (2.15)$$

Tada mokydami totalinio gradiento metodu, kur svorių pataisymai atliekami atsižvelgiant į visus n mokymo procese dalyvaujančius vektorius, turėsime:

$$\boldsymbol{v}_{\text{naujas}} = \boldsymbol{v}_{\text{prieš tai}} - \eta\ \frac{\partial nuostoliai}{\partial \boldsymbol{v}} =$$

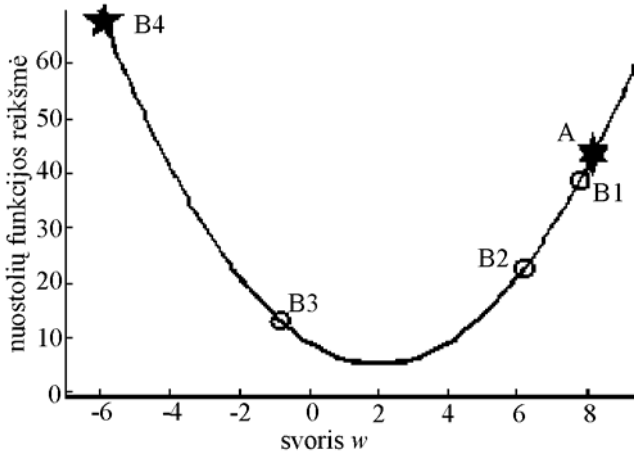
$$= \boldsymbol{v}_{\text{prieš tai}} + 2\eta\left(\frac{1}{n}\sum_{i=1}^{n}y_i\mathbf{z}_i^T - \mathbf{v}^T\frac{1}{n}\sum_{i=1}^{n}\mathbf{z}_i\ \ \mathbf{z}_i^T\right), \qquad (2.16)$$

kur parametras η vadinamas *mokymo žingsniu*.

Totalinio gradiento mokymo algoritmo veikimą iliustruosime vienmačiu pavyzdžiu, kur reikia rasti optimalią vieno svorio w reikšmę. 10 pav. pavaizduota kvadratinė nuostolių funkcija. Mokyti pradedama nuo $w_{(0)}$=8,3. Tada nuostoliai – 43,5 (taškas A). Kadangi nuostolių funkcijos išvestinė taške A yra teigiama, tai sekant algoritmu

(2.16) svoris bus mažinamas. Jei mokymo žingsnis η mažas, bus gautas nedidelis postūmis – iš taško A pateksime į B1. Jei η būtų didesnis, gautume didesnius postūmius ir pereitume į taškus B2, B3, B4. 10 pav. rodo, kad pastaruoju atveju, kai η labai didelis, nuostoliai (2.14) gali netgi padidėti (taškas B4). Tad atveju, kai mokymo žingsnis pernelyg didelis, mokymo algoritmas gali diverguoti, ir mes numatyto tikslo nepasieksime. Jei mokymo žingsnis per mažas, mokymo procesas bus per lėtas, ir tikslo (nuostolių funkcijos minimumo) galime taip pat nepasiekti. Tokiu atveju η reikšmę reikėtų didinti.

Pradedant mokyti reali situacija (nuostolių funkcijos pavidalas) nežinoma, taigi reikiamo mokymo žingsnio pasirinkimas yra viena iš dirbtinių neuroninių tinklų mokymo problemų. Bene labiausiai vykęs ir šiuo metu gan populiarus metodas yra adaptyvus mokymo žingsnio valdymas. Jei kurį laiką mokymas sėkmingas (nuostolių funkcija mažėja), η didinamas. Jei nuostolių funkcija kurį laiką auga, η iškart mažinamas. Ši mokymo žingsnio valdymo taktika pasiteisina perceptroną mokant spręsti ir atpažinimo uždavinį, taip pat mokant ir daugiasluoksnį perceptroną.



10 pav. Nuostolių funkcijos minimizavimas gradientiniu būdu

2.2.3. Keturios normaliuoju pasiskirstymu paremtos regresijos

Panagrinėkime dažnai praktiškai duomenų analizėje naudojamą duomenų transformaciją: kiekvienas mokymo duomenų požymis, $x_{\text{senas}\,j}$, įskaitant ir prognozuojamąjį y, yra „normalizuojamas“ taip, kad jo vidurkis būtų lygus nuliui, o dispersija – vienetui. Tai lengva padaryti transformuojant taip:

$$x_{\text{naujas}\,j} = (x_{\text{senas}\,j} - \bar{x}_{\text{senas}\,j})/ s_{\text{senas}\,j} \ , \qquad (2.17)$$

kur $\bar{x}_{\text{senas}\,j}$ – j-tojo požymio mokymo duomenų vidurkis, o

$s^2_{\text{senas}\,j}$ – j-tojo požymio mokymo duomenų dispersija.

Po minėtos duomenų transformacijos $\frac{1}{n}\sum_{i=1}^{n} y_i \boldsymbol{x}_i^T = \mathbf{S}_{y\boldsymbol{x}}$ reikš nebe kovariacijų, o koreliacijų vektorių tarp y ir vektoriaus $\boldsymbol{x}$ komponenčių. Matrica $\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i \ \boldsymbol{x}_i^T = \mathbf{S}_{\boldsymbol{xx}}$ reikš iš mokymo duomenų įvertintą vektoriaus $\boldsymbol{x}$ koreliacinę matricą.

Tarkime, kad perceptrono mokymą pradedame nuo „nulinio“ vektoriaus, t. y. vektoriaus, kurio visos komponentės lygios nuliui, – $\boldsymbol{v}_{(0)} = [0\ 0\ \ldots\ 0\ 0]^T = \mathbf{0}$. Kadangi $\boldsymbol{v}_{(0)} = 0$, po pirmos mokymo iteracijos gausime:

$$w_{(1)} = w_{(0)}\, 2\eta(\frac{1}{n}\sum_{i=1}^{n} y_i \boldsymbol{z}_i^T) = -2\eta\ \mathrm{S}_{yx}, \qquad w_{0(1)} = 0. \qquad (2.18)$$

Gautą svorio išraišką lygindami su svoriais, gautais standartinės regresijos būdu (formulė (2.4): $w = \boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_{yx}$, $w_0 = 0$, kai $\boldsymbol{\mu}_x = 0$ ir $\boldsymbol{\mu}_y = 0$; šį kartą nagrinėjame atveją, kai duomenys centruoti), matome, kad formulėje (2.18) nebėra kovariacinės (šiuo atveju koreliacijų) matricos $\mathbf{S}_{xx}$. Vadinasi, po pirmos iteracijos perceptronas „ignoruoja“ koreliacijas tarp vektoriaus $\boldsymbol{x}$ komponenčių, bet teisingai įvertina koreliacijas tarp prognozuojamojo rodiklio y ir vektoriaus $\boldsymbol{x}$ komponenčių $x_1, x_2, \ldots, x_p$. Šią

regresiją vadinsime *„naiviąja“*. Tokią regresiją galime sukonstruoti, remdamiesi „sveiku protu“: kiekvieno svorio dydis priklauso nuo koreliacijos. Paprasta ir teisinga, bent iš pirmo žvilgsnio.

Naivioji regresija – tai vienas iš būdų gauti regresijos lygtį. Jis geras, kai duomenų kiekis labai mažas, o dimensiškumas (požymių kiekis p) yra didelis.

Mokant perceptroną toliau, jis vis mažiau ir mažiau ignoruoja koreliacijas tarp vektoriaus $\boldsymbol{x}$ komponenčių $x_1, x_2, \ldots, x_p$. Nepamirškime, kad mūsų uždavinio formuluotėje kiekvieno požymio vidurkiai lygūs nuliui, o dispersijos – vienetui. Faktiškai mes operuojame su koreliacine matrica. Tad mokant toliau, t. y. didėjant iteracijų kiekiui t, gauname svorių vektorių, panašų į formule (2.10) aprašytąjį:

$$w_{(t)} = f_\eta(\eta)\left[\mathbf{S}_{\boldsymbol{xx}} + f_\lambda(\lambda)\begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 \end{bmatrix}\right]^{-1} \mathrm{S}_{yx},\ w_{0(t)}=0, \qquad (2.19)$$

kur skaliariniai koeficientai $f_\eta(\eta, t)$ ir $f_\lambda(\lambda, t$ priklauso nuo iteracijos numerio – t, η arba λ. Kai $t\to\infty$, $f_\eta(\eta, t)\to 1$, o $f_\lambda(\lambda, t)\to 0$. Išraiška (2.19) rodo, kad mokydami perceptroną toliau, gauname reguliarizuotą regresiją (žr. formulę (2.10)). Kai $t\to\infty$, priartėjame prie standartinės regresijos (2.4).

Tais atvejais, kai mokymo duomenų kiekis n mažesnis nei požymių kiekis p, standartinė regresija nebeveikia. Tokiais atvejais statistikos teorija siūlo vietoj paprasto, standartinio, matricos $\mathbf{S}_{\boldsymbol{xx}}$ apvertimo naudoti vadinamąją *pseudoinversiją*. Teoriškai parodyta, kad mokydami perceptroną nurodytu būdu taip pat artėjame prie standartinės regresijos su pseudoapverstąja kovariacine matrica. Visi šie keturi regresijos sudarymo būdai gaunami ne tik minimaliųjų kvadratų metodu, bet gali būti išvesti griežtai matematiškai, darant prielaidą, kad bendras y ir $\boldsymbol{x}$ pasiskirstymas yra daugiamatis normalusis. Norint šį faktą pabrėžti, šis skyrelis taip ir buvo pavadintas.

2.2.4. Robastinė ir atraminių vektorių regresijos

Nuostolių funkcijos (2.11), (2.12) ir (2.13), galima sakyti, specialiai pritaikytos gauti robastinę ir minimaksinę regresijas naudojant vienasluoksnį perceptroną. Norėdami VsP gauti prognozavimo lygtį, kuri „pasveria“ mokymo vektorius priklausomai nuo jų atstumo (nuokrypio, kitaip sakant, paklaidos) iki prognozės reikšmės, minimizuosime tokią suminių nuostolių funkciją

$$nuostoliai = \frac{1}{n}\sum_{j=1}^{n}\Psi(y_j - (x_{1j}w_{1j} + x_{2j}w_{2j} + ... + x_{pj}w_{pj} + w_0)) \quad =$$

$$= \frac{1}{n}\sum_{i=1}^{n}\Psi(y_i - \mathbf{z}_i\mathbf{v}^T), \qquad (2.20)$$

kur $\Psi(c) = \begin{cases} 1, & \text{jei } |c| > c_{\max} \\ 1-\cos(c)), & \text{jei } |c| < c_{\max} \end{cases}$,

o parametras $c_{\max}$ charakterizuoja funkcijos $\Psi(c)$ „platumą“. Kuo didesnis parametras $c_{\max}$, tuo tolimesni vektoriai (labiau nutolę nuo prognozavimo hiperplokštumos) leidžiami dalyvauti mokymo procese. Priešingai, kuo parametras $c_{\max}$ mažesnis, tuo labiau bus slopinami patys didžiausi nuokrypiai. Taikant *delta taisyklę* VsP mokyti, po *t*-tosios iteracijos (svorių pataisymo) daroma tokia perceptrono svorių vektoriaus pataisa:

$$\mathbf{v}_{t+1} = \mathbf{v}_t - \eta \frac{1}{n}\sum_{i=1}^{n}\left.\frac{\partial\Psi(y_j - \mathbf{v}^T\mathbf{z}_j)}{\partial\mathbf{v}}\right|_t, \qquad (2.21)$$

kur $\dfrac{\partial\Psi(y_j - \mathbf{v}^T\mathbf{z}_j)}{\partial\mathbf{v}} =$

$$= \begin{cases} 0, & \text{jei } |y_j - \mathbf{v}^T\mathbf{z}_j| \times (\pi \times c_{\max}^t) > \pi \\ \sin(y_j - \mathbf{v}^T\mathbf{z}_j), & \text{jei } |y_j - \mathbf{v}^T\mathbf{z}_j| \times (\pi \times c_{\max}^t) < \pi. \end{cases}$$

Kai reikia ignoruoti tolimus, netipinius stebėjimus, iš pradžių pasirenkame didelę $c^1_{\max}$ reikšmę. Augant mokymo iteracijų (epochų) skaičiui *t*, parametrą $c^t_{\max}$ mažiname. Visą laiką reikia sekti, kuri mokymo vektorių dalis pakliūva į „nejautrumo zoną“. Mokymo sunkumas tas, kad šiuo atveju reikia parinkti ir optimalų mokymo epochų skaičių, ir parametrus, charakterizuojančius $c^t_{\max}$ mažėjimo procesą. Deja, šis nelengvas uždavinys kol kas nėra visiškai išspręstas.

Norėdami gauti minimaksinę regresiją, minimizuosime:

$$nuostoliai = \frac{1}{n}\sum_{j=1}^{n}(y_j - (x_{1j}w_{1j} + x_{2j}w_{2j} + ... + x_{pj}w_{pj} + w_0))^S =$$

$$= \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{z}_i\mathbf{v}^T)^S , \qquad (2.\ 22)$$

kur laipsnio rodiklis *s* nusako nuostolių funkcijos „statumą“, apibrėžiantį, kurie mokymo vektoriai prognozuojami su mažiausiu tikslumu, būtent, kurie turi dalyvauti mokymo procese.

Mokydami totalinio gradiento metodu turėsime:

$$\mathbf{v}_{t+1} = \mathbf{v}_t - \eta\, S\, \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{v}^T\mathbf{z}_i)^{S-1}\,\mathbf{z}_i^T . \qquad (2.23)$$

Laipsnio rodiklis *s* turi būti sveikasis lyginis skaičius. Mokymo pradžioje reikėtų pasirinkti *s* = 2. Tada turėsime standartinę minimaliųjų kvadratų regresiją. Paskui skaičius *s* didinamas. Dėl to vis didesnis dėmesys (svarba) bus suteikiamas labiausiai nutolusiems vektoriams. Mokymo pabaigoje tik nedidelė netiksliausiai prognozuojančių vektorių dalis apibrėš hiperplokštumos padėtį. Šie mokymo vektoriai gali būti vadinami „atraminiais“. Duomenų analizėje šis metodas žinomas *atraminių vektorių* (angl. *support vector*) pavadinimu. VsP – tai dar vienas būdas gauti *atraminių vektorių regresiją*.

Atraminių vektorių regresija nėra labai populiari, nes pagrindinis dėmesys skiriamas toliausiai nuo prognozuojančios hiperplokštumos nukrypusiems vektoriams. Kalbant apie vienasluoksnio perceptrono gebėjimus realizuoti praktiškai visus tiesinės regresijos lygties sudarymo būdus, duomenų analizės srityje tyrinėtojams ir praktikams reikia atkreipti didesnį dėmesį į VsP. Vienas iš galimų VsP patobulinimų būtų duomenų transformavimas, atliekamas prieš jį mokant. Tai turėtų būti daroma tiek siekiant pagerinti skaičiavimo tikslumą (išvengti išsigimusių arba beveik išsigimusių kovariacinių matricų), tiek siekiant pagreitinti skaičiavimą. Bet tais atvejais, kai duomenų kiekis nėra didelis (o tai būtų bene svarbiausia), prognozuojama tiksliau.

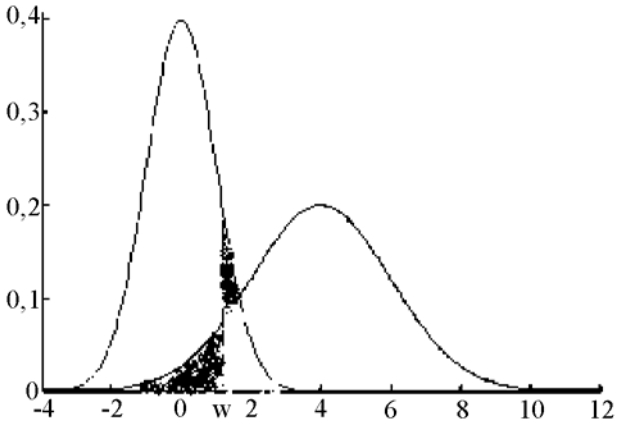
Visų pirma prieš mokant perceptroną iš visų turimų duomenų (mokymo, validavimo, testavimo) reikia atimti mokymo imties vidurkį. Antra, visų požymių reikšmes reikėtų padalyti iš mokymo imties vidutinių kvadratinių nuokrypių, tuo visų požymių dispersijas padarant lygias vienetui (mokymo imtyje). Trečia, prieš mokant perceptroną duomenis vertėtų pasukti taip, kad požymiai taptų nebekoreliuoti. Ketvirta, pasukus vėl reikėtų požymių dispersijas normalizuoti. Po perceptrono mokymo reikėtų grįžti į pradinę požymių erdvę.

TREČIAS SKYRIUS. TIESINIAI KLASIFIKAVIMO ALGORITMAI

3.1. Klasifikavimo uždavinys. Apriorinės klasių tikimybės. Klasifikavimo klaidų rūšys

3.1.1. Apriorinės klasių tikimybės. Pirmosios ir antrosios rūšies klasifikavimo klaidos

Norint sudaryti klasifikavimo taisyklę naudojantis statistinių sprendinių metodais, daroma prielaida, kad požymių pasiskirstymų tankiai $f(x \mid \Pi_1)$ ir $f(x \mid \Pi_2)$ pirmojoje ir antrojoje klasėse yra žinomi. 11 pav. pirmosios klasės tankis, nubrėžtas kaip raudona varpo formos kreivė, yra kairėje, antrosios – dešinėje. Abu tankiai kertasi. Todėl klasifikuoti be klaidų yra neįmanoma. Jei klasifikuotume pagal slenkstį, 11 pav. pažymėtą raide w, tai dalis antrosios klasės vektorių būtų priskiriami pirmajai klasei.



11 pav. Dviejų klasių tankiai ir pirmosios bei antrosios rūšies klasifikavimo klaidos

Normaliojo pasiskirstymo atveju tą klaidą (tikimybę, kad atsitiktinai parinktas antrosios klasės vektorius bus priskirtas pirmajai klasei) nesunku apskaičiuoti analitiškai:

$$P_{2/1} = \int_{-\infty}^{w} f(x \mid \Pi_2)dx = \Phi(-\frac{\mu_2 - w}{\sigma_2}) , \qquad (3.1)$$

kur $\Phi(c)$ – standartinio normaliojo tankio (jo vidurkis 0, o dispersija 1) kumuliatyvinė pasiskirstymo funkcija ($\Phi(c) \to 0$, kai $c \to -\infty$, $\Phi(c) \to 1$, kai $c \to +\infty$, $\Phi(c) = 0{,}5$, kai $c = 0$).

Analogiškai antrosios rūšies klaidų tikimybė išreiškiama tokiu būdu:

$$P_{1/2} = \int_{w}^{\infty} f(x \mid \Pi_1)dx = \Phi(\frac{\mu_1 - w}{\sigma_1}) . \qquad (3.2)$$

Klaidos tikimybė $P_{2/1}$ proporcinga 11 pav. *kairiajam* (violetine spalva pažymėtam) plotui. Klaidos tikimybė $P_{1/2}$ proporcinga *dviejų dešiniųjų trikampių* (pažymėtų geltona ir žalia spalvomis**)** plotų sumai. Jei klasių apriorinės tikimybės vienodos, t. y. $q_2 = q_1 = 0{,}5$, tai bendra klasifikavimo klaidos tikimybė

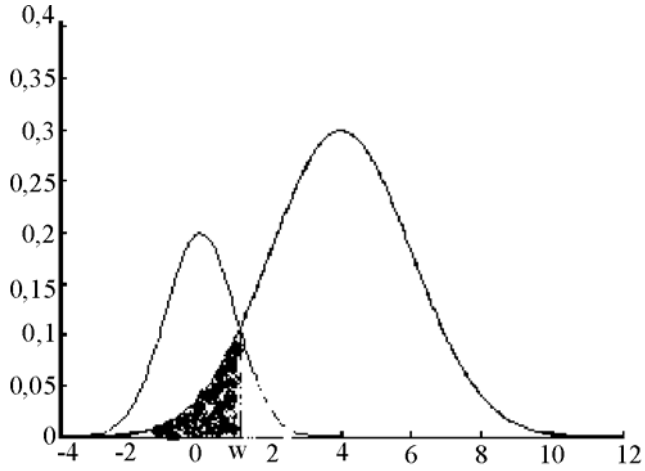
$$P_{\text{sum}} = \frac{1}{2}(P_{1/2} + P_{2/1}).$$

Jeigu slenkstį w pastumsime dešinėn, į tą vietą, kur tankiai $f(x \mid \Pi_1)$ ir $f(x \mid \Pi_2)$ susikerta, sumoje nebeliks *viršutinio* plotelio **(**pažymėto žalia spalva**)**. Toks klasifikatorius bus optimalus pagal suminės klasifikavimo klaidos tikimybės kriterijų.

Panagrinėkime atvejį, kai klasių apriorinės tikimybės nevienodos. Pavyzdžiui, tiriant įmonės bankroto prognozę iš 500 atsitiktinai parinktų įmonių „gerais laikais“ šansus bankrutuoti artimiausią pusmetį turės gal tik kelios įmonės. Tad bankrutuojančių įmonių klasės tikimybė bus gal tik $q_2 = 0{,}01$, o kitos (nebankrutuojančių įmonių) klasės apriorinė tikimybė bus $q_1 = 1 - q_2 = 0{,}99$. Tada suminė klasifikavimo klaidos tikimybė

$$P_{\text{sum}} = q_2\, P_{2/1} + q_1\, P_{1/2}\ . \qquad (3.3)$$

Dėl tos priežasties reikės žiūrėti, kurioje vietoje kertasi „pasvertieji tankiai“ $q_1 f(x \mid \Pi_1)$ ir $q_1 f(x \mid \Pi_2)$. 12 pav. yra gautos kreivės iš 11 pav. esančių kreivių tuo atveju, kai $q_1 = 0{,}25$, $q_2 = 0{,}75$. Matome, kad optimali slenksčio w reikšmė ženkliai pasislinko į kairę.



12 pav. Dviejų klasių tankiai, padauginti iš apriorinių tikimybių

Taigi *apriorinės klasių tikimybės yra svarbios* sprendžiant praktikos uždavinius. Kaip ir pavyzdyje su įmonių bankrotais, daugelyje kitų duomenų analizės uždavinių apriorinės klasių tikimybės yra nežinomos ir sunkiai nustatomos. Jos priklauso nuo aplinkybių. Pavyzdžiui, krizių ir didelių permainų laikotarpiu bankrutuojančių įmonių tikimybė gali daug kartų išaugti.

Kiekvienas netinkamas (neteisingas) sprendimas klasifikavimo uždavinyje „kainuoja“. Jeigu sveiką žmogų pripažinsime sergančiu, turėsime nuostolių. Ir atvirkščiai, jei sergantį pripažinsime sveiku ir jo negydysime arba gydysime netinkamai, vėl bus nuostolių. Tad dar viena problema klasifikavimo uždavinyje yra susijusi su netinkamo klasifikavimo „kaina“. Tarkime, $K_{1/2}$ – tai kaina, kurią mokėsime, jei

antrosios klasės objektą pripažinsime kaip pirmosios. Analogiškai $K_{2/1}$ – tai kaina, kurią mokėsime, jei pirmosios klasės objektą pripažinsime kaip antrosios. Tada bendra (suminė) netinkamo klasifikavimo kaina

$$K_{\text{sum}} = q_2 P_{2/1} K_{2/1} + q_1 P_{1/2} K_{1/2}.$$

Vertindami neteisingo klasifikavimo kainas, klasifikuoti turėtume pagal tai, kuri sandauga didesnė – $q_2 P_{2/1} K_{2/1}$ ar $q_1 P_{1/2} K_{1/2}$. Reikėtų pažymėti, kad praktikoje klasifikavimo kainos beveik visada nežinomos ir nustatomos labai subjektyviai. Akivaizdu, kad jos priklauso nuo to, kas moka už „nuostolius“. Pavyzdyje su pacientu kainos priklausys nuo to, kas moka – pacientas, gydymo įstaiga ar valstybė. Visais atvejais jos bus skirtingos. Taigi kainos yra subjektyvios.

3.1.2. Euklidinio atstumo, tiesinis Fišerio ir kvadratinis klasifikatoriai

1.2.1 poskyryje kalbėjome apie daugiamatį normalųjį pasiskirstymą, charakterizuojamą vidurkių vektoriumi $\boldsymbol{\mu}_i$ ir kovariacine matrica $\boldsymbol{\Sigma}_i$ (apatinis indeksas i čia simbolizuoja klasės numerį). Jeigu tartume, kad tikrasis duomenų pasiskirstymas yra normalusis, tai klasifikatoriui sukurti pirmiausia iš mokymo duomenų reikėtų gauti kiekvienos klasės vidurkių ir kovariacinių matricų įverčius – $\bar{\boldsymbol{x}}_1$, $\bar{\boldsymbol{x}}_2$ ir $\mathbf{S}_1$, $\mathbf{S}_2$. Tad dviejų klasių atveju klasifikavimo taisyklė būtų tokia:

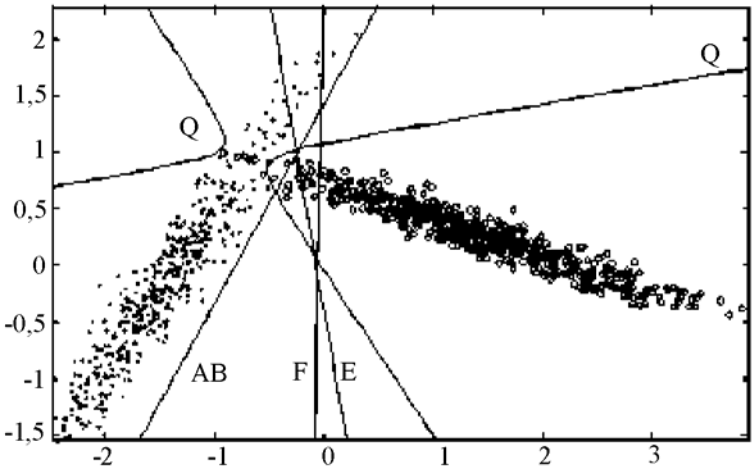
$\boldsymbol{x}$ priskiriame klasei Π_1, jei $g(\boldsymbol{x}) = q_1\varphi(\boldsymbol{x}, \bar{\boldsymbol{x}}_1, \mathbf{S}_1) - q_2\varphi(\boldsymbol{x}, \bar{\boldsymbol{x}}_2, \mathbf{S}_2) > 0$,

$\boldsymbol{x}$ priskiriame klasei Π_2, jei $g(x) = q_1\varphi(\boldsymbol{x}, \bar{\boldsymbol{x}}_1, \mathbf{S}_1) - q_2\varphi(\boldsymbol{x}, \bar{\boldsymbol{x}}_2, \mathbf{S}_2) \le 0$.

(3.4)

$\mathbf{S}_2 \neq \mathbf{S}_1$, taigi skiriamoji riba tarp pirmosios ir antrosios klasių nebus tiesė. 13 pav. dvimatėje erdvėje pavaizdavome dviejų klasių vektorius (pliusiukai ir skrituliukai) ir lygtimi (3.4) apibrėžtą skiriamąją ribą (kai q_2=q_1=0,5). Tai yra kvadratinis skiriamasis paviršius Q. Dvimačiu atveju – tai dvi hiperbolės.

Esminis supaprastinimas gaunamas, jei daroma prielaida, kad kovariacinės matricos lygios, t. y. $\mathbf{\Sigma}_2=\mathbf{\Sigma}_1$. Tada naudotume bendrą (suvidurkintą) kovariacinę matricą, $\mathbf{S} = ½ (\mathbf{S}_1+\mathbf{S}_2)$.



13 pav. Dviejų klasių mokymo vektoriai ir keturios skiriamosios ribos

Pastaruoju atveju skiriamasis paviršius yra tiesinis (tiesė – dvimatėje erdvėje, plokštuma – trimatėje erdvėje ir hiperplokštuma – aukštesnio matiškumo erdvėje). Vietoje skirtumo tarp tankių, (3.4) formulėje naudotų diskriminantinei (skiriamajai) funkcijai apibrėžti, dažnai naudojamas dviejų tankių santykio logaritmas:

$$g(\boldsymbol{x}) = \log \frac{q_1 f(\boldsymbol{x} \mid \bar{\mathbf{x}}_1, \mathbf{S})}{q_2 f(\boldsymbol{x} \mid \bar{\mathbf{x}}_2, \mathbf{S})}. \qquad (3.5)$$

Šiuo atveju gausime tiesinę, diskriminantinę taisyklę F, kuri yra iš esmės paprastesnė nei kvadratinė Q, be to, jai sudaryti reikia daug mažiau mokymo vektorių:

$$g(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{w}^{\mathrm{F}}+w_0, \qquad (3.6)$$

kur $\boldsymbol{w}^{\mathrm{F}} = \mathbf{S}^{-1}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)$,

$$w_0 = - ½ (\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2)\boldsymbol{w}^{\mathrm{F}} + \log\left(\frac{q_1}{q_2} \right).$$

Klasifikavimas vyksta pagal diskriminantinės funkcijos ženklą (teigiama, neigiama). Šią klasifikaciją pirmasis pasiūlė žinomas statistinių metodų kūrėjas Ronaldas Fišeris, todėl ji ir vadinama jo vardu. Kai kuriuose praktiniuose uždaviniuose, kur mokymo duomenų daug, kvadratinė diskriminantinė funkcija Q pagal teisingai klasifikuojamų vektorių skaičių gerokai lenkia Fišerio taisyklę. 13 pav. taip pat rodo, kad kvadratinė taisyklė Q daug geresnė nei tiesinė F. Šiuo atveju Andersonas ir Bahaduras yra pasiūlę modifikuoti Fišerio taisyklę:

$$\boldsymbol{w}^{\mathrm{AB}} = (\gamma_1 \mathbf{S}_1 + (1 - \gamma_1)\mathbf{S}_2)^{-1}(\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2),$$

$$w^{\mathrm{AB}}{}_0 = - \frac{1}{2} (\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2)\boldsymbol{w}^{\mathrm{F}} + \gamma_2 + \log\left(\frac{q_1}{q_2} \right). \qquad (3.7)$$

Formulėje (3.7) turime du nežinomus parametrus γ_1 ir γ_2. Juos paprasčiausia rasti skaitiniu būdu, minimizuojant analitiškai užrašytą klasifikavimo paklaidos tikimybę. Andersono ir Bahaduro (AB) tiesinę klasifikavimo taisyklę galime gauti ir mokydami vienasluoksnį perceptroną. Šis išradimas iš esmės supaprastina diskriminantinę funkciją, sumažina reikalingų jai sudaryti mokymo vektorių skaičių ir daugumoje praktinių uždavinių leidžia gauti mažą klasifikavimo paklaidos tikimybę.

Kai mokymo duomenų yra mažai, o požymių daug, ir jų kiekio negalime sumažinti neprarasdami požymių informatyvumo, tai klasifikavimo taisyklę turime dar labiau supaprastinti. Tiesiog formulėje (3.6) galime praleisti kovariacinę matricą, kas lygiavertiška prielaidai, kad požymiai statistiškai nepriklausomi ir jų dispersijos vienodos. Tokiu atveju faktiškai vektorių $\boldsymbol{x}$ klasifikuojame pagal jo euklidinį atstumą iki klasių vidurkių $\bar{\boldsymbol{x}}_1$, $\bar{\boldsymbol{x}}_2$ (jei $q_2 = q_1$). Todėl šis klasifikatorius vadinamas *euklidinio atstumo klasifikatoriumi*, o šioje knygoje žymimas raide E.

3.1.3. Bajeso, asimptotinė, sąlyginė ir laukiamoji klasifikavimo klaidos tikimybės

Sudarydami ir analizuodami klasifikavimo taisyklę pagal empirinius duomenis, susiduriame su keletu klasifikavimo paklaidų tipų. Jeigu skirtumus tarp jų ignoruosime, galime smarkiai apsirikti ir „paleisti vėjais“ lėšas, išleistas šiam darbui atlikti.

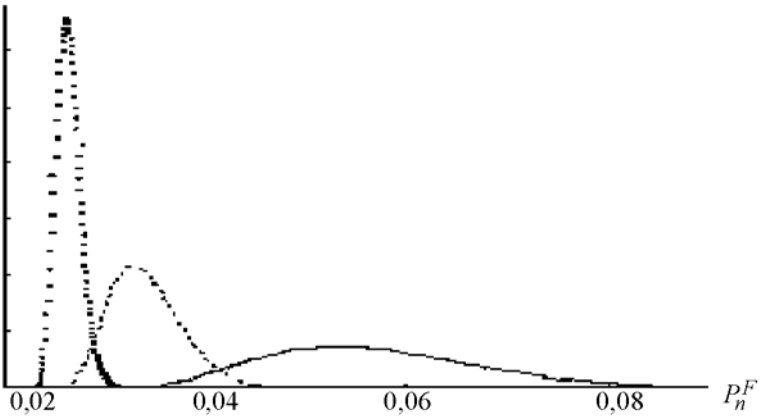
Jeigu žinome pasiskirstymo tankius ir apriorines klasių tikimybes, tai naudodamiesi statistinių sprendinių teorija galime sudaryti optimalią klasifikavimo taisyklę (3.5), kurios klaida bus aprašoma formule (3.3). Statistinių sprendinių teorija remiasi viena iš pagrindinių tikimybių teorijoje Bajeso teorema. Todėl klasifikavimo klaidą (3.3) vadinsime *Bajeso* klaida ir žymėsime P_{B}.

Ką tik pristatėme keletą statistinių klasifikatorių, skirtų daugiamačio normaliojo pasiskirstymo modeliui. Jeigu tikrasis duomenų modelis skirsis nuo postuluojamojo, bet to nežinant klasifikatorius vis tiek bus sudarytas pagal šį modelį, tai jo klasifikavimo klaidos tikimybė gali viršyti Bajeso klaidą. Pavyzdžiui, tikrasis požymių pasiskirstymas – normalusis – yra su parametrais $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, $\boldsymbol{\Sigma}_1$, ir $\boldsymbol{\Sigma}_2$. Be to, kovariacinės matricos skiriasi, t. y. $\boldsymbol{\Sigma}_2 \neq \boldsymbol{\Sigma}_1$. Jeigu klasifikatorių Q sudarysime pagal tikrąsias vidurkių ir kovariacinės matricos reikšmes, tai jo klasifikavimo klaidos tikimybė P_{∞}^{Q} sutaps su P_{B}. Tačiau jei sudarysime tiesinį Fišerio klasifikatorių F imdami $\boldsymbol{\Sigma} = ½\,(\boldsymbol{\Sigma}_1+\boldsymbol{\Sigma}_2)$, tai jo klasifikavimo klaida P_{∞}^{F} dažniausiai viršys P_{B}. Klasifikatoriaus, sudaryto pagal metodą, tarkime, A, tiksliai žinant reikiamus pasiskirstymų $f(x \mid \Pi_1)$ ir $f(x \mid \Pi_2)$ parametrus bei apriorines tikimybes q_1 ir q_2, klasifikavimo tikimybę vadinsime *asimptotine klasifikatoriaus* A *klaida* ir žymėsime P_{∞}^{A}. Begalybės ženklas čia primena, jog reikiamus parametrus mes tiksliai įvertinsime tik turėdami „begalinį mokymo duomenų kiekį“.

Bajeso ir asimptotinė klasifikavimo klaidos – tai tik abstrakcijos. Jos reikalingos norit suprasti klasifikatorių sudarymo uždavinio problematiką. Faktiškai klasifikatoriams sudaryti naudojami riboto dydžio „mokymo duomenys“. Statistikoje, jei daroma prielaida, kad

duomenys normalieji, klasifikatoriui sudaryti naudojami imties įverčiai $\bar{\boldsymbol{x}}_1$, $\bar{\boldsymbol{x}}_2$ arba $\bar{\boldsymbol{x}}_1$, $\bar{\boldsymbol{x}}_2$, $\mathbf{S}_1$, $\mathbf{S}_2$.

Sudarant euklidinio atstumo klasifikatorių E naudojami vidurkių vektorių įverčiai $\bar{\boldsymbol{x}}_1$ ir $\bar{\boldsymbol{x}}_2$, o sudarant Fišerio klasifikatorių F naudojami $\bar{\boldsymbol{x}}_1$, $\bar{\boldsymbol{x}}_2$ ir $\mathbf{S} = ½ (\mathbf{S}_1+\mathbf{S}_2)$. Minėtus įverčius lemia atsitiktinai surinkti ribotos apimties duomenys, taigi šiuo būdu sudaryto klasifikatoriaus klaidos tikimybė bus atsitiktinis dydis. Ją žymėsime P_n^{F} ir vadinsime *sąlygine klasifikavimo klaida*. Euklidinio atstumo klasifikatoriaus, sudaryto iš mokymo imties, klaidą žymėsime P_n^{E}. Čia apatinis indeksas simbolizuoja, kad klasifikatorius sudarytas iš *n* mokymo vektorių, o viršutinis – klasifikatoriaus sudarymo būdą. 14 pav. pateikiame iliustraciją.



14 pav. Fišerio klasifikatoriaus sąlyginės klaidos pasiskirstymo tankiai, kai mokymo vektorių kiekis: $n = 40$ (dešinysis, lėkštas pasiskirstymo tankis), 120 ir 360

Bene pagrindinė atsitiktinių dydžių charakteristika yra jų matematinė „viltis“ – vidurkis. Mes naudosime sąlyginės klasifikatorių klaidos vidurkius, vadindami juos *laukiama klasifikatoriaus klaida*

(arba laukiama generalizavimo klaida) ir žymėdami EP_n^{F}, EP_n^{E} ir t. t. Visada $P_\infty^{\mathrm{A}} \geq P_{\mathrm{B}}$, $EP_n^{\mathrm{A}} \geq P_{\mathrm{B}}$, $P_n^{\mathrm{A}} \geq P_{\mathrm{B}}$. Paprastai $EP_n^{\mathrm{A}} > P_\infty^{\mathrm{A}}$, tačiau atsitiktinai kartais gali pasitaikyti, kad $P_n^{\mathrm{A}} < P_\infty^{\mathrm{A}}$.

3.1.4. Reguliarizuotas, robastinis ir atraminių vektorių klasifikatoriai

Jei mokymo duomenų kiekis nedidelis, o požymių daug, tai sudėtingus klasifikatorius reikia bandyti supaprastinti. Vienas iš būdų supaprastinti kvadratinį, Andersono ir Bahaduro bei Fišerio klasifikatorius – naudoti reguliarizuotus kovariacinės matricos įverčius. Kalbėdami apie reguliarizuotą regresiją, prie kovariacinės matricos, gautos iš mokymo duomenų, diagonalinių elementų pridėdavome po teigiamą konstantą λ:

$$S_R=S+\lambda\begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{bmatrix}. \qquad (2.9)$$

Klasifikavimo uždavinyje galime elgtis taip pat. Tai padeda. Kaip ir regresijos uždavinyje, čia irgi reikia parinkti geriausią reguliarizavimo parametro reikšmę (žr. 2.1.3 poskyrio komentarus).

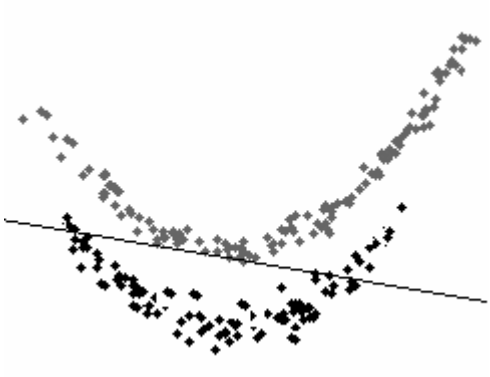
Jei mokymo duomenyse turime nemažai stebėjimų vektorių, pernelyg toli nutolusių nuo vidurkių, tai jie laikomi iš dalies „klaidingais“, todėl jų indėliai į klasifikavimo ar prognozavimo taisyklės sudarymą turėtų būti dirbinai sumažinti (žr. 2.2.4 poskyrį). Paprasčiausias būdas sumažinti tolimų stebėjimų įtaką skaičiuojant vidurkį – naudoti iteracinius robastinius įverčius, pvz.:

$$\overline{\boldsymbol{x}}_{\boldsymbol{i}}^{\text{robastinis}} = \frac{1}{N_i}\sum_{j=1}^{N_i} x_{\boldsymbol{i}j}\,\Psi\left((x_{ij} - \overline{\boldsymbol{x}}_i^{\text{robastinis}})(x_{ij} - \overline{\boldsymbol{x}}_i^{\text{robastinis}})^T\right), \qquad (3.8)$$

kur $\Psi(c) = \begin{cases} 1/(1+c-\alpha), & \text{jei } c > \alpha \\ 1, & \text{jei } c < \alpha \end{cases}$.

Vertinant kovariacinę matricą irgi elgiamasi panašiai: tolimi stebėjimai vertinami su mažesniu svoriu. Formulėje (3.8) pateiktos funkcijos $\Psi(c)$ forma ir parametrai (pvz., „pločio“ parametras α) gali kisti ir turi priklausyti nuo turimų duomenų.

Sudarant klasifikavimo taisyklę galutinis tikslas yra minimizuoti klasifikavimo klaidos tikimybę. Kai duomenų pasiskirstymai normalieji, ankstesniuose poskyriuose aprašyti metodai gali būti labai geri (jei duomenų kiekis pakankamas). Tačiau jei duomenų pasiskirstymai iš esmės skiriasi nuo normaliųjų, tai ir robastinis vidurkių ir kovariacinės matricos įvertinimas nebepadės. 15 pav. pateikiame dviejų klasių atskyrimo pavyzdį, kur nenormališkumo problema iškyla visu aštrumu. Šio tipo uždaviniui spręsti pasiūlyta nemažai euristikomis pagrįstų algoritmų, tačiau nė vienas iš jų negarantuoja absoliučiai geriausio sprendinio. Vienas iš daugelio tam taikomų metodų – tai vienasluoksnis perceptronas, apie kurį kalbėsime vėliau.

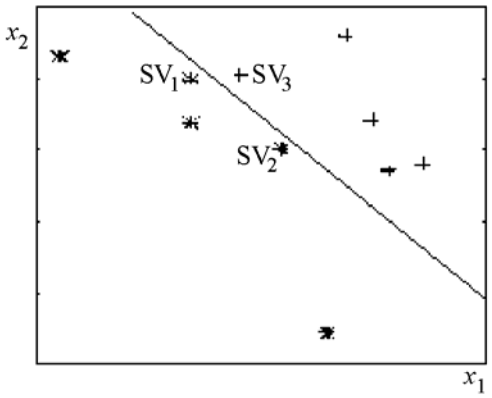


15 pav. Tiesinis klasifikatorius, minimizuojantis empirinę klasifikavimo klaidą

Pastarąjį dešimtmetį labai populiarus klasifikavimo algoritmas – tai maksimalaus tarpo, o kalbant plačiąja prasme, *atraminių vektorių klasifikatorius* (angl. *support vector*). Maksimalaus tarpo klasifikato-

rius gaunamas tada, kai naudojantis hiperplokštuma yra įmanoma mokymo duomenyse esančius dviejų klasių vektorius atskirti be klaidų. 16 paveiksle pateikiame dviejų klasių tiesinio atskyrimo pavyzdį, kur klases skiriančioji tiesė yra maksimaliai nutolusi nuo atraminių vektorių SV_1, SV_2, ir SV_3.

Minėto tipo klasifikavimo algoritmas vadinamas *maksimalaus tarpo klasifikatoriumi*. Jeigu klasės tiesiškai neatsiskiria, galime požymių kiekį specialiai padidinti įvesdami papildomus, iš turimų originaliųjų požymių padarytus naujus išvestinius požymius, pvz., $x_1 x_2$, $x_1 x_3$ ir t. t. Kai bendras senų ir naujų požymių kiekis pasieks mokymo duomenų kiekį n, klasės būtinai tiesiškai atsiskirs. Žinoma, jos gali atsiskirti ir atvejais, kai $p < n$. Tada naujoje (praplėstoje) požymių erdvėje galime sudaryti maksimalaus tarpo klasifikatorių. Tokiu atveju šis klasifikatoriaus sudarymo būdas jau vadinamas *atraminių vektorių* (angl. *support vector*) pavadinimu. Yra metodų, kaip padaryti atraminių vektorių klasifikatorių, kur leidžiama, kad dalis mokymo vektorių būtų klasifikuojama neteisingai. Tokiais atvejais klases skiriančioji hiperplokštuma vedama maksimaliai didžiausiu atstumu „nuo arčiausiai nuo jos esančių ir dar teisingai klasifikuojamų mokymo vektorių“. Tai vadinamasis *minkštasis* (angl. *soft*) *atraminių vektorių klasifikatorius.*



16 pav. Tiesinis klasifikatorius, kurio klases skiriančioji tiesė maksimaliai nutolusi nuo atraminių vektorių SV_1, SV_2, ir SV_3

Formaliai tam, kad rastume atraminius vektorius, turime spręsti kvadratinio programavimo uždavinį. Tai nelengvas uždavinys. Skaitytojui, norinčiam detaliau susipažinti su atraminių vektorių klasifikatoriais, rekomenduojama paskaityti apžvalginius, specialiai tam skirtus straipsnius arba įsidiegti ir išbandyti nemokamą programinę įrangą (http://www.csie.ntu.edu.tw/~cjlin/libsvm). Kitame poskyryje kalbėdami apie vienasluoksnio perceptrono mokymą, parodysime, kaip šio tipo klasifikatorius gauti specialiu būdu mokant VsP.

3.1.5. Klasifikatoriaus sudėtingumo ir mokymo duomenų kiekio ryšys

Kalbėdami apie standartinės prognozavimo minimaliųjų kvadratų regresijos lygties sudarymą matėme, kad kuo daugiau požymių, tuo daugiau reikia vektorių norimai mokymo kokybei pasiekti. Jei duomenų nedaug, vietoj kvadratinės regresijos galime naudoti reguliarizuotą arba netgi naiviąją regresijas. Kitas būdas – atmesti nereikalingus, mažai prognozuojamą rodiklį veikiančius požymius. Klasifikacijos uždavinyje susiduriame su analogiškomis problemomis. Šiame poskyryje nagrinėsime, kaip požymių ir mokymo duomenų kiekis veikia įvairaus sudėtingumo klasifikavimo taisyklių mokymo kokybę.

Jeigu požymių pasiskirstymas „sferiškai normalusis“ (duomenų kovariacinė matrica diagonalinė ir jos diagonaliniai elementai σ^2 tarpusavyje lygūs), tada euklidinio atstumo klasifikatoriaus asimptotinė klasifikavimo klaida gali būti išreikšta tokia formule:

$$P_{\infty}^{\mathrm{F}} = P_{\mathrm{B}} = \Phi(-\frac{1}{2}\delta), \tag{3.9}$$

kur $\Phi(c)$ – standartinio normaliojo tankio kumuliatyvinė pasiskirstymo funkcija, t. y. tikimybė, kad atsitiktinis dydis bus mažesnis už konstantą c; $\delta = ((\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\,(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T)^{-1/2}/\sigma$ yra atstumas tarp klasių, vadinamas *Mahalanobio atstumu*, vardu žmogaus, kuris pirmas jį pasiūlė naudoti klasių atskiriamumui apibūdinti.

Laukiama klasifikavimo klaida išreiškiama formule (3.9), kur vietoj tikrojo atstumo tarp klasių δ įstatytas „efektyvusis atstumas“

δ_n^{E}, įvertinantis tai, kad riboto dydžio mokymo duomenų atveju klasės skiriamoji hiperplokštuma nukrypsta nuo optimalios, ir tada atstumas tarp klasių (jų atskiriamumas) matuojamas jau kitaip:

$$\delta_n^{\mathrm{E}} = \delta T_n^{\boldsymbol{\mu}}, \tag{3.10}$$

kur $T_n^{\boldsymbol{\mu}} = 1/\sqrt{1+\frac{2p}{N\delta^2}}$. (3.11)

$T_n^{\boldsymbol{\mu}}$ – tai daugiklis, išreiškiantis, kiek „kainuoja“ vidurkių $\boldsymbol{\mu}_1$ ir $\boldsymbol{\mu}_2$ įvertinimas iš mokymo duomenų.

Formulė (3.11) rodo, jog klasifikavimo klaida padidėja dėl to, kad sudarant euklidinio atstumo klasifikatorių iš mokymo duomenų reikia įvertinti 2*p* charakteristikų, skirtingų kiekvienai klasei (*p*-mačius vektorius μ_1 ir μ_2). Šis padidėjimas priklauso nuo santykio $2p/N$.

Fišerio klasifikatorius yra sudėtingesnis nei euklidinio atstumo klasifikatorius. Čia reikia įvertinti ne tik 2*p* vidurkių vektorių $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$ charakteristikas, bet ir $p(p\text{-}1)/2$, kovariacinės matricos $\boldsymbol{\Sigma}$ komponentes, kurios yra „bendros abiem klasėms“ (charakterizuoja jas abi). Fišerio klasifikatoriaus atveju greta koeficiento $T_n^{\boldsymbol{\mu}}$ reikės naudoti dar ir papildomą narį – $T_n^{\boldsymbol{\Sigma}}$:

$$T_n^{\boldsymbol{\Sigma}} = 1/\sqrt{\frac{n}{n-p}} \ . \tag{3.12}$$

Narys $T_n^{\boldsymbol{\Sigma}}$ yra atsakingas už tai, kad kovariacinė matrica įvertinama netiksliai. Kai matrica $\boldsymbol{\Sigma}$ yra bendro pavidalo, Mahalanobio atstumas užrašomas truputį sudėtingiau, nei buvo nurodyta formulėje (3.9):

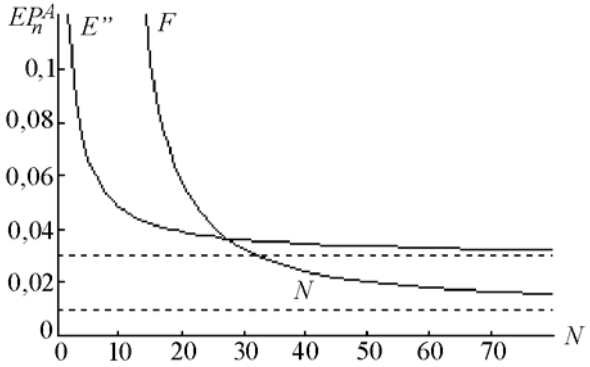
$$\delta = ((\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\, \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T)^{-1/2}. \tag{3.13}$$

Fišerio klasifikatoriaus laukiamoji klasifikavimo klaida bus išreiškiama formule (3.9), kur vietoj tikrojo atstumo tarp klasių δ bus įstatytas efektyvusis:

$$\delta_n^{\mathrm{F}} = \delta T_n^{\boldsymbol{\mu}} \, T_n^{\boldsymbol{\Sigma}}. \qquad (3.14)$$

Formulė (3.12) rodo, kad „kaina“, mokama už $p(p\text{-}1)/2$ matricos $\boldsymbol{\Sigma}$ komponenčių įvertinimą, nepaprastai išauga, kai požymių kiekis p darosi artimas mokymo duomenų kiekiui n. Tai matome iš grafiko F, pateikto 17 pav.

Tokiu atveju Fišerio klasifikatoriaus naudoti nebegalima. Reikia naudoti paprastesnį euklidinio atstumo klasifikatorių arba, pavyzdžiui, reguliarizuotą diskriminantinę analizę (žr. 3.1.4 poskyrio pradžią). Šis reiškinys, žinomas kaip *žirklių efektas*, iliustruotas dviem grafikais, pateiktais 17 pav.



17 pav. Euklidinio atstumo (E) ir Fišerio (F) klasifikatorių *žirklių efektas*

Iš 17 pav. matome, kad esant mažam duomenų kiekiui reikia naudoti paprastesnį – euklidinio atstumo – klasifikatorių. Prie tiesinio Fišerio klasifikatoriaus reikia pereiti tik esant didesniam mokymo duomenų kiekiui. *Žirklių efektas* galioja ne tik minėtiems dviem klasifikatoriams, bet ir apskritai, kai nagrinėjami du skirtingo sudėtingumo

klasifikatoriai ar statistiniais duomenimis paremtos prognozavimo taisyklės.

3.1.6. Vienasluoksniu perceptronu paremto klasifikavimo algoritmo evoliucija iteraciniame mokymo procese

Tiek regresijos, tiek ir klasifikavimo uždaviniuose, mokydami VsP, galime gauti kelis vienas po kito einančius klasifikavimo algoritmus. Klasifikavimo uždavinyje, rašydami nuostolių funkciją (žr. formulę (1.4)):

$$nuostoliai = \frac{1}{n}\sum_{j=1}^{n}(y_j - f(\boldsymbol{x}_j\boldsymbol{w} + w_0))^2 \ , \qquad (1.4)$$

netiesinės aktyvavimo funkcijos *f*(*suma*) (žr. 5 pav.) jau nebepraleisime, o vietoj trokštamų išėjimų y_j nuostolių funkcijos išraiškoje (1.4) naudosime dvi skirtingas konstantas, po vieną kiekvienai iš dviejų klasių. Jei aktyvavimo funkcija tokia, kaip parodyta 5 pav.:

$$f(suma) = \frac{1}{1+e^{-suma}}, \qquad (3.15)$$

tai trokštami išėjimai galėtų būti: $t_1 = 0$ (pirmosios klasės vektoriams), $t_2 = 1$ (antrosios klasės vektoriams). 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*. Jei stimuliavimas mažas, mokymas vyks naudojant tik labai trumpą aktyvavimo funkcijos *f*(*suma*) dalį, kur *suma* reikšmė mažai nutolsta nuo nulio. Toje trumpoje zonoje funkcija *f*(*suma*) yra beveik tiesinė.

Panagrinėkime 2.2.3 poskyryje jau minėtą transformaciją, kur kiekvienas mokymo duomenų požymis „normalizuojamas“ taip, kad jo vidurkis būtų lygus nuliui, o dispersija – vienetui (formulė (2.17)). Tarkime, kiekvienoje iš dviejų klasių turime po $N = n/2$ mokymo vektorių. Mokymą pradėkime nuo vektoriaus, kurio kiekviena komponentė lygi nuliui.

Esant išvardytoms sąlygoms, po pirmos totalinio gradiento iteracijos („parodomi“ visi N mokymo procese dalyvaujantys kiekvienos klasės vektoriai) gausime:

$$\boldsymbol{w}_{(1)} = k_{\mathrm{VSP}}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2),\ w_{0(1)} = 0, \tag{3.16}$$

kur skaliarinis koeficientas k_{VSP} neveikia klasifikavimo hiperplokštumos padėties, nes klasifikavimas vykdomas pagal sumos $\boldsymbol{x}_j \boldsymbol{w} + w_0$ ženklą.

Gautą svorio išraišką lygindami su svoriais, gautais tiesiniam Fišerio klasifikatoriui (formulė (3.6)), matome, kad formulėje (3.16) nebėra kovariacinės (šiuo atveju koreliacijų) matricos **S**. Taigi po pirmos iteracijos perceptronas „ignoruoja“ koreliacijas tarp vektoriaus $\boldsymbol{x}$ komponenčių skirtingose klasėse. Tai euklidinio atstumo klasifikatorius, kadangi atliekant klasifikavimą pasverta suma diskriminantinės funkcijos reikšmė $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{w}_{(1)} + w_{0(1)}$ lyginama su nuliu ir, kaip ką tik minėjome, teigiamas koeficientas k_{VSP} nebevaidina jokios reikšmės.

Mokant perceptroną toliau, kol svoriai dar maži ir dar „dirbame“ tiesinėje aktyvavimo funkcijos dalyje, po t-tosios iteracijos svorių vektorius gali būti išreikštas taip:

$$\boldsymbol{w}_{(t)} = k_{\mathrm{VSP}}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)\left[S + \frac{2}{(t-1)\eta}\begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 \end{bmatrix}\right]^{-1}. \tag{3.17}$$

Tai reguliarizuota klasifikacija, kurios reguliarizacijos parametras $\lambda = \dfrac{2}{(t-1)\eta}$, augant iteracijų skaičiui, mažėja. Riboje, kai λ labai

didelis, nario $\frac{2}{(t-1)\eta}\begin{bmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & 1 \end{bmatrix}$ įtaka visai sumažės, ir visiškai priartėsime prie standartinio Fišerio klasifikatoriaus, aptarto 3.1.2 poskyryje. Jeigu mokymui naudojamų p-mačių vektorių skaičius mažesnis nei p, perceptronas realizuos tiesinį Fišerio klasifikatorių su kovariacinės matricos pseudoinversija. Šiuo atveju klases skirianti hiperplokštuma bus vienodai nutolusi nuo pirmosios ir antrosios klasių mokymo vektorių. Kitaip sakant, gausime maksimalaus tarpo klasifikatorių. Tai įdomi perceptrono savybė.

Jei trokštami išėjimai t_1 ir t_2 yra arti nulio ar vieneto, tai toliau mokant perceptroną jo svoriai išauga. Naudojant netiesinę aktyvavimo funkciją, jos išsilenkimai „pradeda jaustis“: VsP išėjimai

$$(1+e^{-\mathbf{x}_{ij}\mathbf{w}+w_0})^{-1}$$

priartėja prie 1 arba 0. Tada dideli atsilenkimai, jeigu tokių mokymo duomenyse yra, mažai veikia mokymo procesą. Vadinasi, VsP tampa robastiškas, t. y. atsparus dideliems mokymo duomenų nuokrypiams (netikslumams). Tai labai puiki VsP savybė.

Jei mokant bus naudojamos ribinės trokštamų išėjimų reikšmės $t_1 = 0$ (pirmajai klasei), $t_2 = 1$ (antrajai klasei), tai esant labai dideliems svoriams netiesinio VsP išėjimai $(1+e^{-\mathbf{x}_{ij}\mathbf{w}+w_0})^{-1}$ visiškai priartės prie 1 arba 0. Tada nuostolių funkcija (1.4) išreikš neteisingai suklasifikuotų mokymo vektorių dalį (dažnį), t. y. empirinę klasifikavimo klaidos tikimybę. Tai bus ankstesniame poskyryje minėtas minimalios empirinės klaidos klasifikatorius.

Šiuo atveju susiduriame su problema, kai esant kad ir nedidelei neteisingai klasifikuojamų mokymo vektorių daliai svoriai ypač neišaugs, nes neteisingai klasifikuojami vektoriai „temps svorių vektorių atgal“. Kad mokant netiesinį VsP būtų priartėta prie minimalios empirinės klaidos klasifikatoriaus, reikia kokiu nors būdu priversti svorius augti. Vienas iš būdų – prie mokymui naudojamos nuostolių funkcijos pridėti specialų svorius augti skatinantį *antireguliarizacijos* narį:

$$nuostoliai = \frac{1}{n}\sum_{j=1}^{n}(y_j - f(\boldsymbol{x}_j\boldsymbol{w} + w_0))^2 + \lambda\,(h^2 - \boldsymbol{w}^T\boldsymbol{w})^2, \quad (3.18)$$

kur koeficientas h^2 nusako norimą perceptrono svorių dydį, o reguliarizavimo parametras λ nusako „optimalų santykį“ tarp abiejų minimizuojamų nuostolių funkcijos narių,

$$\frac{1}{n}\sum_{j=1}^{n}(y_j - f(\boldsymbol{x}_j\boldsymbol{w} + w_0))^2 \quad \text{ir} \quad (h^2 - \boldsymbol{w}^T\boldsymbol{w})^2.$$

Kaip ir visada, esant nežinomam uždaviniui nėra labai jau nelengva rasti tinkamas parametrų h^2 ir λ reikšmes. Jų tenka ieškoti „bandymų ir klaidų“ metodu.

Kiekvieno mokymo vektoriaus „indėlis“ į nuostolių funkciją (3.18) priklauso nuo jo atstumo nuo klases skiriančios hiperplokštumos. Jei VsP paremtas klasifikatorius mokymo duomenyse nebedaro klaidų, o tai būtų visada įmanoma, jeigu pridėtume papildomus, iš esamų požymių „išvestus“ požymius, tai esant ribinėms trokštamų išėjimų reikšmėms ($t_1 = 0$, $t_2 = 1$), svoriai galėtų labai smarkiai išaugti. Esant dideliems svoriams, tik vektoriai, esantys arčiausiai klases skiriančios plokštumos, dalyvaus mokymo procese. Kitų, bent truputėlį toliau esančių, vektorių indėlis bus ženkliai mažesnis. Todėl klases skiriančios hiperplokštumos padėtis pradės priklausyti tik nuo nedidelio kiekio arčiausiai jos esančių mokymo vektorių, kuriuos galėtume vadinti „atraminiais“. Taigi mokydami perceptroną ką tik minėtomis sąlygomis priartėjame prie atraminių vektorių (maksimalaus tarpo) klasifikatoriaus. Praktiškai tai nelengva padaryti, nes išaugus svoriams pasvertosios sumos $\boldsymbol{x}_j\boldsymbol{w} + w_0$ taip pat smarkiai padidėja. Dėl to gradientas, stipriai veikiantis VsP mokymo greitį, labai priartėja prie nulio. Todėl mokymo žingsnį η tenka didinti. Tai nėra lengva, nes jei η didinsime per lėtai, tai ir mokymas vyks labai lėtai. Jei η didinsime per greitai, algoritmas gali diverguoti. Pastaruoju atveju galima gauti visiškai neteisingas svorių vektoriaus $\boldsymbol{w}$ reikšmes ir didelį empirinės klasifikavimo klaidos dažnį. Tokiu atveju mokymo procesą reikia kartoti iš naujo parenkant mažesnį mokymo žingsnio augimo greitį. Prak-

tiškas patarimas: mokant perceptroną naudoti tokią parametro η augimo taisyklę: – $\eta_{t+1} = \eta_t \times \gamma$, kur $\gamma = 1{,}0001$ arba kiek didesnė konstanta. Konstantą γ taip pat reikia parinkti klaidų ir bandymų metodu. Papildomai galima naudoti svorių augimą skatinantį antireguliarizacijos narį $\lambda\ (h^2 - \boldsymbol{w}^T\boldsymbol{w})^2$.

Šiame poskyryje išdėstyta medžiaga dar kartą patvirtina, kad netiesinis vienasluoksnis perceptronas turi nemažai universalumo požymių. Tai geras metodas ieškant paslėptų dėsningumų daugiamačiuose duomenų masyvuose. Tačiau teisingai jį taikyti nėra lengva. Perceptroną naudojant reikia turėti ne tik nemaža teorinių žinių (didžioji dalis jo mokymo subtilybių išdėstyta šioje knygoje), bet ir didelę praktinio darbo patirtį.

3.2. Optimalus sustojimas mokant perceptroną. Pradinių svorių vektoriaus įtaka tikslumui

Vienasluoksnio perceptrono evoliucija mokymo metu parodė, kad jo nuostolių funkcija keičiasi mokymo metu. Ji tampa vis sudėtingesnė. Mokydami perceptroną, patenkinę aukščiau šioje knygoje išdėstytus reikalavimus, iš pradžių gauname paprasčiausią naiviąją regresiją, arba euklidinio atstumo klasifikatorių, jei sprendžiame atpažinimo (klasifikavimo) uždavinį. Vėliau pereiname prie reguliarizuotų regresijų arba klasifikatorių, kurių reguliarizavimo parametras λ mažėja didėjant mokymo iteracijų skaičiui. 3.1.5 poskyryje aiškinto *žirklių efekto* esmė – esant tam tikram mokymo duomenų kiekiui reikia parinkti jam optimalaus sudėtingumo klasifikatorių. Kadangi perceptronas besimokydamas darosi vis sudėtingesnis, tai esant fiksuotam mokymo duomenų kiekiui mokymo procesą reikia laiku sustabdyti, kad būtų galima gauti optimalaus sudėtingumo klasifikavimo, arba regresijos, algoritmą.

Teorijos išaiškinimo sumetimais mes darėme prielaidą, kad mokyti pradedama nuo svorių vektoriaus, kurio komponentės lygios nuliui. Kas bus, jei svorių vektorius bus kitoks? Kas būtų, jei remdamiesi kokia nors papildoma informacija mes sugebėtume jį nustatyti „beveik tiksliai“? Šiuos klausimus panagrinėsime detaliau. Pradėkime aiškintis

nuo regresijos uždavinio ir nuo standartinės, minimalių kvadratų metodu paremtos regresijos.

Tarkime, pradinis svorių vektorius yrs $\boldsymbol{v}_{(0)}$. Šio poskyrio analizėje laikysime jį atsitiktiniu, kad pradiniu prognozavimo momentu paklaidą $\boldsymbol{v}_{(0)}\boldsymbol{z}$-$y$ galėtume nagrinėti kaip atsitiktinį dydį, kurio vidurkis yra 0 ir dispersija – $(\sigma_{start})^2$. Pažymėkime svorių (koeficientų) vektorių, kurį gavome minimizavę kvadratų sumos nuostolių funkciją, raide $\boldsymbol{v}_n$. Prognozavimo lygties paklaida $\boldsymbol{v}_n$-y yra atsitiktinis dydis. Jei duomenys pasiskirstę pagal normalųjį dėsnį, tai paklaidos dispersija bus (žr. 2.1.2 poskyrį) tokia:

$$(\sigma_n)^2 \approx (\sigma_0)^2 \frac{n}{n-p}. \tag{3.19}$$

Imituodami perceptrono sustojimą, panagrinėkime kitą regresijos lygties gavimo būdą:

$$\boldsymbol{v}(\kappa) = (1-\kappa)\,\boldsymbol{v}_{(0)} + \kappa \boldsymbol{v}_n, \tag{3.20}$$

kur $0 < \kappa < 1$.

Šios, naujai sudarytos, regresijos lygties paklaida $\boldsymbol{v}(\kappa)$-y taip pat bus atsitiktinis dydis. Statistinė šios naujoviškos prognozavimo taisyklės analizė parodė, kad didžiausias tikslumas pasiekiamas, kai

$$\kappa \approx \frac{1}{1 + \dfrac{p}{n-p}\dfrac{(\sigma_0)^2}{(\sigma_{start})^2}}. \tag{3.21}$$

Remiantis lygtimi (3.21) išeina, kad kuo tikslesnis pradinis svorių vektorius (σ_{start} mažesnė), tuo parametras κ turėtų būti mažesnis, o kalbant apie perceptroną, – tuo anksčiau turėtų būti sustabdytas mokymas.

Ši analizė taip pat rodo, kad informaciją, sukauptą pradiniame svorių vektoriuje, galime išsaugoti, jei perceptrono mokymas baigiamas laiku. Tai ypač svarbu ieškant dėsningumų labai dideliuose duomenų

masyvuose, kai perceptroną mokyti tenka tik su dalimi turimų mokymo vektorių. Jei mokytume iki galo ir perceptrono mokymo proceso nestabdytume anksčiau, tai prarastume informaciją, glūdinčią pradiniame svorių vektoriuje $\boldsymbol{v}_{\text{start}}$.

Naudodami aprašytąją statistinių prognozavimų algoritmų savybę maksimaliam tikslumui gauti, susiduriame su poreikiu žinoti pradinių svorių tikslumą. Jei pradiniai svoriai gauti iš prieš tai apdorotų duomenų, tai pradinio vektoriaus tikslumas gali būti įvertintas pagal formules (2.6) ar (3.19). Daug problemiškiau ankstyvo stabdymo idėją taikyti klasifikavimo algoritmuose. Šiuo atveju tikslių klasifikavimo taisyklės paklaidos formulių dar nėra išvesta, todėl tenka naudotis empiriniais metodais (naudoti validavimo, t. y. papildomus, duomenis).

KETVIRTAS SKYRIUS. NETIESINIS PROGNOZAVIMAS IR KLASIFIKAVIMAS

4.1. Netiesinės požymių transformacijos

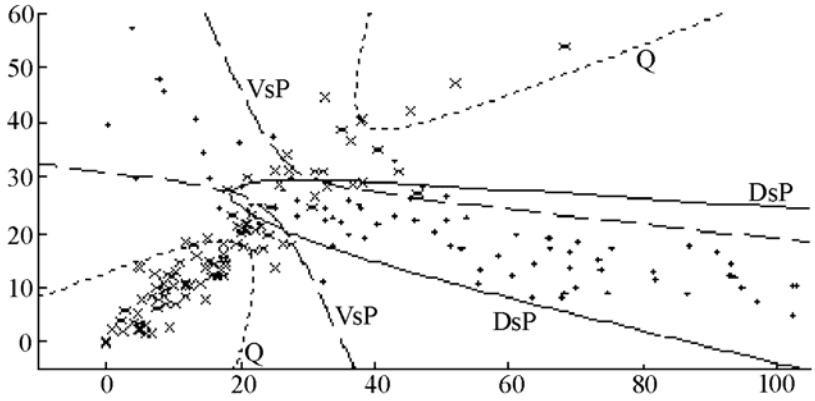
Ankstesniuose skyriuose apžvelgtieji klasifikavimo ir prognozavimo algoritmai leidžia rasti dėsningumus, jei prognozuojamas požymis tiesiškai priklauso nuo jį prognozuojančių požymių arba jei klasės atsiskiria hiperplokštuma. Kai kuriais atvejais tiesinių priklausomybių nėra, klasės hiperplokštuma neatsiskiria. Tada tenka kurti sudėtingesnius netiesinius algoritmus.

Bene paprasčiausias būdas sudaryti netiesinę taisyklę – vietoj originaliųjų požymių $x_1, x_2, \ldots, x_p$ naudoti išvestinius, padarytus naudojant netiesines transformacijas. Pavyzdžiui, prie turimų požymių pridėję 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 mokydami $p(p+1)/2$ matavimų erdvėje galime gauti kvadratinę diskriminantinę funkciją. 18 paveiksle pateikiamas pavyzdys, kuriame turime dvi beveik pagal normalųjį pasiskirstymą pasiskirsčiusias klases, dvi kvadratinės diskriminantinės funkcijos suformuotas skiriamąsias hiperboles ir kvadratinį skiriamąjį paviršių, gautą mokant perceptroną penkių požymių x_1, x_2, $(x_1)^2$, x_1x_2, $(x_2)^2$ erdvėje. Šiuo atveju klasifikatoriaus Q klaidos tikimybė – $P_n^{\mathrm{Q}} = 0{,}195$. Naudodami VsP gauname daug geresnį rezultatą – $P_n^{\mathrm{VSP}} = 0{,}095$. Tai pasiekėme dėl to, kad kvadratiniam klasifikatoriui Q sudaryti iš mokymo duomenų reikėjo įvertinti net 30 parametrų, o VsP buvo mokomas penkiamatėje erdvėje.

Praktiniame darbe dažnai pasitelkiami požymiai, imti panašumo į mokymo vektorius pagrindu. Tarkime, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ yra n mokymo vektorių, nepaisant, ar klasifikavimo, ar prognozavimo uždavinį sprendžiame. Tada kiekvieno mokymo (testinio taip pat) vektoriaus atžvilgiu galime suskaičiuoti n atstumų iki mokymo vektorių $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$

$$z_j = D(\boldsymbol{x}, \boldsymbol{x}_n) = (\boldsymbol{x}, \boldsymbol{x}_n)\,(\boldsymbol{x}, \boldsymbol{x}_n)^T \qquad j = 1, 2, \ldots, n. \qquad (4.1)$$

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, jis leistų gauti hiperplokštumą, vienodai nutolusią nuo visų n mokymo vektorių. Naudoti visus n požymius neracionalu. Praktiškai reikėtų sumažinti požymių kiekį, atrenkant tik pačius svarbiausius (žr. tolesnį skyrių). Autoriaus nuomone, bene racionaliausias variantas – mokyti VsP.



18 pav. Klasifikavimo ribos, gautos naudojant kvadratinę diskriminantinę funkciją Q VsP penkių naujų požymių erdvėje ir daugiasluoksnį perceptroną (su dviem paslėptais neuronais)

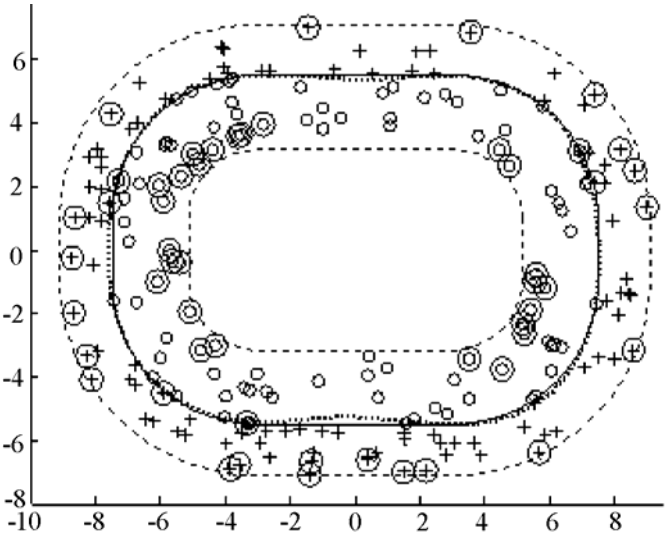
Tiesioginis panašumo požymių naudojimas ne visada pasiteisina. Dažniausiai naudojami netiesine branduolio funkcija papildomai apdoroti požymiai, pvz.:

$$z_j^* = \exp(-D(\mathbf{x}, \mathbf{x}_n)/\lambda), \quad j=1, 2, \ldots, n. \qquad (4.2)$$

Šiuo atveju gan svarbu parinkti tinkamą atstumo matavimo skalės parametrą λ. Transformacija (4.2) sumažina labai nutolusių vektorių įtaką. Bet kuriuo atveju klasifikatoriaus sudarymo uždavinys nėra lengvas, nes reikia ne tik sumažinti požymių kiekį, bet ir pasirinkti

požymių tipą ((4.1) ar (4.2)); nustatyti parametrą λ; rasti, kada optimaliai stabdyti perceptrono mokymą. 19 paveiksle pateikiamas pavyzdys, kuriame turime po šimtą mokymo vektorių kiekvienos iš dviejų klasių dvimatėje erdvėje.

Viena klasė yra antrosios „viduje“. Transformacijomis (4.1) ir (4.2) iš 200 mokymo vektorių buvo sudaryti 200 požymių ($\lambda = 20$) ir šioje erdvėje mokytas vienasluoksnis perceptronas. 19 paveikslas rodo, kad VsP sudaryta dvi klases skiriančioji kreivė (punktyrinė linija), labai artima optimaliai (ištisinė elipsė). Skrituliais pažymėti abiejų klasių mokymo vektoriai, esantys arčiausiai klases skiriančios hiperplokštumos 200 požymių erdvėje.



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

Anksčiau parodėme, kaip sudaryti klasifikatorių naujomis, netiesinėmis, transformacijomis gautų požymių erdvėje. Tie patys naujų požymių sudarymo metodai tinka ir sprendžiant prognozavimo uždavinius. Vienasluoksnio perceptrono naudojimas pasiteisina ir čia.

4.2. Neparametriniai *k*-artimiausių kaimynų ir Parzeno lango klasifikatoriai

Vienas ir populiariausių tradicinių klasifikavimo ir prognozavimo metodų, veikiančių pagal principą „pasakyk, kas tavo draugai, aš pasakysiu, kas esi tu“, yra *k*-artimiausių kaimynų taisyklė. Tarkime, $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, ... , $\boldsymbol{x}_n$ yra *n* mokymo vektorių sprendžiant prognozavimo uždavinį. Tarkime, y_1, y_2, ... , y_n yra juos atitinkančios prognozuojamo požymio reikšmės. Mūsų uždavinys – turimam vektoriui *x* prognozuoti *y* reikšmę. Naudodamiesi formule (4.1), raskime vektoriaus $\boldsymbol{x}$ panašumus (euklidinius atstumus) į visus *n* mokymo vektorius $D(\boldsymbol{x}, \boldsymbol{x}_1)$, $D(\boldsymbol{x}, \boldsymbol{x}_2)$, ... , $D(\boldsymbol{x}, \boldsymbol{x}_n)$. Tarkime, vektorius $\boldsymbol{x}$ panašiausias į vektorių $\boldsymbol{x}_s$. Tada vektoriui $\boldsymbol{x}$ bus priskiriama y_s reikšmė. Jeigu spręsime klasifikavimo uždavinį, vektoriui $\boldsymbol{x}$ bus priskiriamas vektoriaus $\boldsymbol{x}_s$ klasės numeris. Tai vieno artimiausio kaimyno taisyklė, žinoma ne tik atpažinimo, duomenų analizės, bet ir gretutinėse dirbtinio intelekto srityse.

Jei sprendimą apie požymio *y* reikšmę ar klasės numerį darysime ne pagal vieną vektorių, o pagal daugelį jų, sakykim, pagal *k* jų, tai turėsime *k*-artimiausių kaimynų taisyklę. Ši taisyklė nereikalauja mokymo. Jai reikia tik nemažos atminties ir daug skaičiavimo laiko, reikalingo atstumams (panašumams) suskaičiuoti. Daugelio autorių nuomone, tai labai geras atpažinimo ir prognozavimo metodas, ypač populiarus sprendžiant vaizdų atpažinimo uždavinius. Kompiuterio atminčiai, skaičiavimo apimčiai taupyti yra pasiūlyta daug būdų, kaip atrinkti nedidelį kiekį „netoli ribos“ esančių vektorių, kaip pagreitinti skaičiavimus aukšto dimensiškumo požymių erdvėje. Augant kompiuterių galingumui šis metodas vis labiau populiarėja.

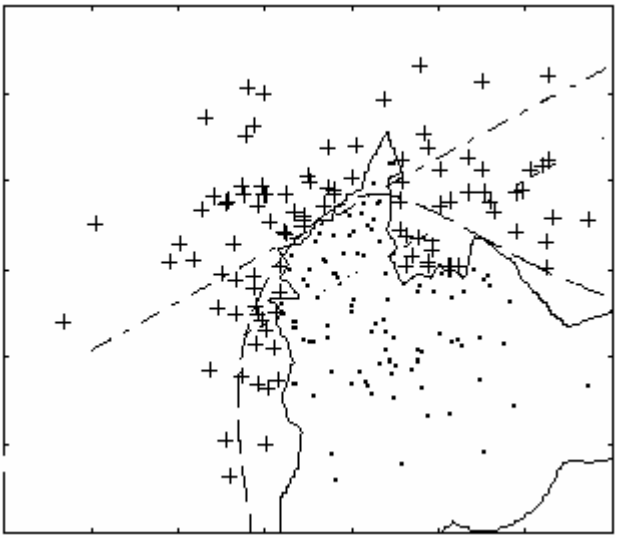
Neparametrinėje *k*-artimiausių kaimynų taisyklėje visi *k* mokymo vektorių lygiomis teisėmis dalyvauja nustatant prognozės reikšmę arba klasės numerį. Intuityviai kyla idėja, kad artimesni vektoriai turėtų būti vertinami „su didesniu svoriu“, o tolimesni – su mažesniu. Tam galime panaudoti netiesinę transformaciją (4.2) ir klasifikavimo (prognozavimo taip pat) uždavinį spręsti atsižvelgiant į pasiskirstymo tankių įverčius:

$$\hat{f}\ (\boldsymbol{x} \mid \Pi_i) = \frac{1}{N_i}\gamma(\lambda)\sum_{j=1}^{N_i}\exp\left(-(\boldsymbol{x} - \boldsymbol{x}_j)\ (\boldsymbol{x} - \boldsymbol{x}_j)^T/\lambda\right), \qquad (4.3)$$

kur $\gamma(\lambda)$ yra normalizuojantis rodiklis, o λ – *glotninimo parametras*.

Tankio įvertis formule (4.3) aprašo neparametrinį, *Parzeno langu* vadinamą, pasiskirstymo tankio įvertį, kurį paprastai naudojame klasifikavimo uždaviniams spręsti, kaip aprašyta 3.1.2 poskyryje, diskriminantinės funkcijos išraiška (3.4). Čia labai svarbus uždavinys yra parinkti tinkamą glotninimo parametro λ reikšmę.

20 paveiksle pateikiamas pavyzdys, kur turime šimtą dviejų klasių mokymo vektorių dvimatėje erdvėje. Viena klasė iš dalies „apgaubia“ antrąją. Piešinyje matome tris Parzeno lango klasifikatoriaus formuojamas skiriamąsias kreives, nubrėžtas atvejis, kai $\lambda = 0{,}001$ (kreiviausia linija), $\lambda = 2$ ir $\lambda = 1000$ (praktiškai tai jau tiesė).



20 pav. Trys Parzeno lango klasifikatoriaus formuojamos skiriamosios kreivės, nubrėžtos atvejais, kai $\lambda = 0{,}001$ (kreiviausia linija), $\lambda = 2$ ir $\lambda = 1000$

Kad būtų parinkta geriausia glotninimo parametro reikšmė, rekomenduojame iš anksto normuoti kiekvieną požymį taip, kad jo dispersija mokymo duomenyse būtų lygi vienetui. Tada reikėtų parinkti kokią dešimtį λ reikšmių, pvz.: 0,001, 0,003, 0,01, 0,03, 0,1, 0,3, 1, 3, 10, 100, 1000. Kad sutaupytume skaičiavimo laiko, atstumų matricą tarp mokymo vektorių skaičiuojame tik vieną kartą (kai požymių yra daug, o daugiausia laiko užima skaičiavimo etapas). Paskui cikle, kiekvieno iš n turimų mokymo vektorių kiekvienos λ reikšmės atveju, randame pasiskirstymo tankių reikšmes (klasifikuojamus vektorius praleisdami). Taip vienu ypu gauname visus „slenkančio egzamino“ klasifikavimo klaidos tikimybių įverčius, atitinkančius visą dešimtį tirtų parametro λ reikšmių. Tada pasirenkame λ reikšmę, atitinkančią mažiausią klasifikavimo klaidos dažnį. Parzeno lango tankio įverčiai (4.3) taip pat gali būti naudojami ir netiesinei neparametrinei prognozavimo lygčiai sudaryti. Problemos čia tos pačios. Tai parametro λ reikšmės parinkimas.

4.3. Daugiasluoksniai perceptronai

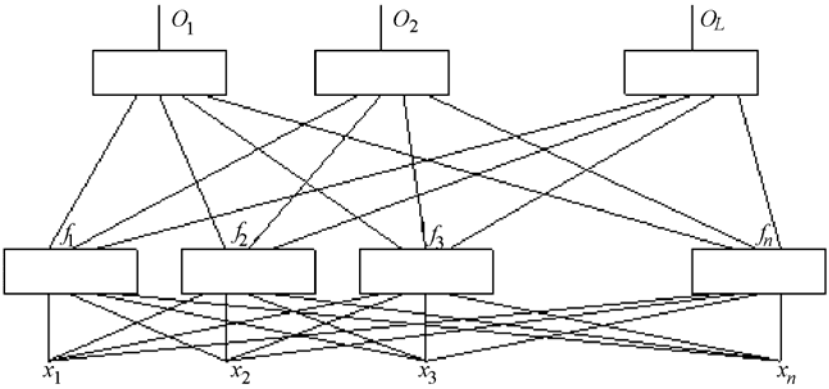
4.3.1. Daugiasluoksnio perceptrono architektūra ir mokymas

Kaip ir vienasluoksnis perceptronas, daugiasluoksnis jo variantas – tai taip pat gamtos inspiruotas, informaciją apdorojantis mazgas. Tikima, kad gyvų mąstančiųjų organizmų smegenyse sudėtingam informacijos apdorojimui ir sprendimų priėmimui taikomi daugiasluoksniai perceptronai (DsP). Atskiri neuronai yra surikiuoti į sluoksnius. Įėjimo signalai $x_1, x_2, \ldots, x_p$ patenka į vadinamojo „paslėptojo“ sluoksnio neuronus. Paslėptųjų sluoksnių gali būti keli. Žemesnio sluoksnio neuronų išėjimo signalai keliauja į aukštesnį sluoksnį, kol galų gale pasiekia išėjimo sluoksnį. Nuolat daugelį kartų sumuojami, netiesiškai transformuojami signalai gali suformuluoti sudėtingas sąvokas, reikalingas toliau priimant sprendimus pagal informaciją, esančią DNT įėjimuose. 21 paveiksle pateikiame DsP su vienu paslėptu sluoksniu schemą.

21 pav. pavaizduotasis DsP turi daug įėjimų, daug išėjimų (juose yra po vieną VsP) ir h paslėptų neuronų, vienasluoksnių perceptronų. Į perceptrono įėjimą pateikus p-matį vektorių $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)$, visuose paslėpto sluoksnio neuronuose suskaičiuojamos pasvertosios sumos:

$$s_v = \sum_{v=1}^{p} x_{v1v} + w_{j0} \ , \ j = 1 \ , 2, \ldots h, \qquad (4.4)$$

kurios tuoj pat nukreipiamos į atitinkamas aktyvacijos funkcijas (pvz., (3.15)), kurių išėjimai $f(s_v)$, $j = 1, 2, \ldots h$, tampa DsP išėjimo sluoksnio įėjimais.

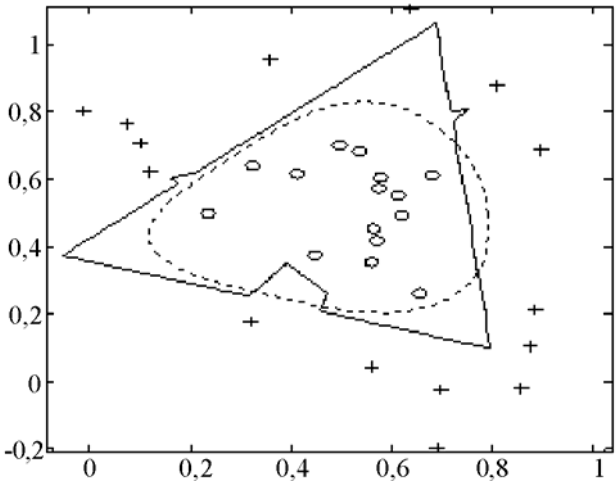


21 pav. DsP su vienu paslėptu sluoksniu schema

Kaip ir VsP, daugiasluoksnis perceptronas gali spręsti klasifikavimo ir prognozavimo uždavinius. Palyginti su VsP daugiasluoksniame perceptrone mokydami kiekvienam *p*-mačiam įėjimo vektoriui $\boldsymbol{x}$ turime pateikti *K*-matį trokštamų išėjimų reikšmių vektorių $\boldsymbol{t} = (t_1, t_2, \ldots, t_K)$. Jeigu sprendžiame prognozavimo uždavinį, tai išėjimo sluoksnyje paprastai aktyvavimo funkcija tiesinė, o vektoriaus $\boldsymbol{t}$ komponentės – realieji skaičiai.

Jei sprendžiame klasifikavimo uždavinį, tai išėjimo neuronų turi būti tiek, kiek turime klasių. Jei naudojama sigmoidinė aktyvavimo funkcija (3.15), tai trokštamų išėjimų vektoriaus komponentės būna 0 arba 1, kur vienetukas paprastai dedamas išėjime, atitinkančiame mokymo vektoriaus klasės numerį – $\boldsymbol{t} = (t_1, t_2, \ldots, t_K)$. 22 paveiksle pavaizduotas jau apmokyto daugiasluoksnio perceptrono, turinčio du

įėjimus, šešis paslėptus neuronus ir vieną išėjimą su netiesiniu VsP jame, skiriamasis paviršius (punktyrinė linija).



22 pav. Dvi duomenų klasės ir sėkmingai apmokyto DsP skiriamasis paviršius (punktyrinė linija) ir paviršius, gautas padauginus paslėptojo sluoksnio neuronų svorius iš 1000 (laužtinė linija)

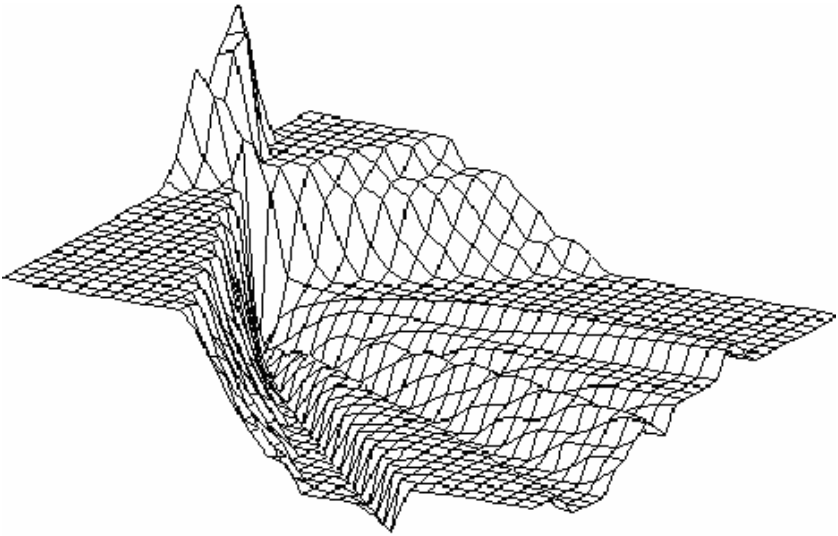
Paslėptajame sluoksnyje turime netiesinę, lėtai augančią aktyvavimo funkciją, taigi skiriamasis paviršius – glotnioji kreivė. Tačiau jeigu specialiai dirbtiniu būdu padauginsime paslėptojo sluoksnio svorius iš tūkstančio, pasvertųjų sumų s_v (formulė (4.4)) reikšmės padidės tūkstantį kartų. Tokiu atveju aktyvavimo funkcijų reikšmės priartės prie nulio arba vieneto. Dėl to klasių skiriamoji riba taps „gabalais tiesinė“ (žr. 22 paveikslą, laužtinė linija). Matome, kad paslėptojo sluoksnio netiesinė aktyvavimo funkcija vaidina labai svarbų vaidmenį.

4.3.2. Mokymosi ir lokalinių minimumų klausimai

Kaip minėjome, perceptrono mokymo procese minimizuojama daug sudėtingesnė nei (1.4) nuostolių funkcija. Sudėtingumas išauga dėl to, kad šiuo atveju vietoj vieno išėjimo jau turime K išėjimų. Be to, atsirado

dar vienas paslėptas sluoksnis. Todėl DsP su vienu paslėptu sluoksniu sumavimas išėjimo sluoksnyje 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. Paminėsime, kad pakankamai aiškus aprašas, kaip sudaryti ir mokyti perceptronus, yra pateiktas *Matlab'e* (*Neural network 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, su daugybe lokalinių (vietinių) minimumų nuostolių funkciją. Nuostolių funkcija yra sudaryta iš daugybės „griovių" plokščiu dugnu ir stačiomis sienomis, kaip pavaizduota 23 paveiksle.

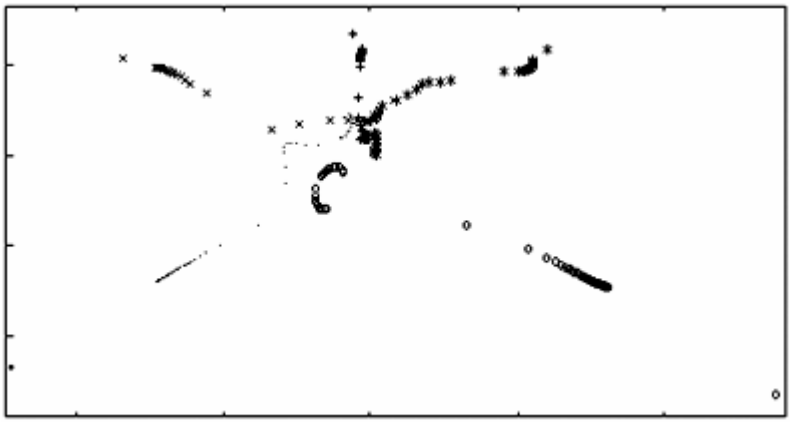


23 pav. Daugiasluoksnio perceptrono nuostolių funkcija

„Grioviai“ išeina iš vieno centro. Globalinis minimumas dažniausiai būna viename iš griovių, bet kadangi jų yra labai daug, tai problema atrasti globalinį griovį (globalinį minimumą) išties yra labai sunki.

Mokydami paprastai pradedame nuo mažų svorių. Mokyti nuo „nulinių“ svorių, kaip rekomendavome dirbant su vienasluoksniu perceptronu, čia nebegalima. Visi paslėpti neuronai taptų vienodi, nes mokydami kiekvienam neuronui naudotume vieną tą pačią taisyklę, kiekvienam paslėptajam neuronui taip pat pateiktume vis tuos pačius duomenis. Viena vertus, gerai, kad pradedame nuo mažų svorių, nes spėjame, kad minimumas dažniausiai yra nedidelių svorių zonoje. Kita vertus, pradėjus mokyti nuo mažų svorių, dalis neuronų pakliūva į tą patį lokalinį „griovį“ ir mokymo metu susivienodina su kitu, į jį panašiu, neuronu. Tą mes patyrėme mokydami su duomenimis, pateiktais 22 paveiksle. Kad neuronai nesupanašėtų, buvome priversti generuoti svorius labai plačiame jų reikšmių intervale.

Mokant daugiasluoksnį perceptroną, jo svoriai kurį laiką „pasisukinėja apie centrą“ ir po to, pakliuvę į „griovį“, jo jau laikosi – auga į visas puses. 24 paveiksle ne dviejų, o santykinai kelių požymių erdvėje pavaizduota, kaip auga penki jo svoriai.



24 pav. Daugiasluoksnio perceptrono svorių augimas mokymo metu

Paveiksle aiškiai matome, kad po tam tikro „pasisukinėjimo apie centrą“ dalies svorių komponentės auga paprasčiausiai tik tiesiškai. Augimo greitis laipsniškai mažėja. Priežastis ta, kad augant svoriams auga ir pasvertosios sumos. Dėl to gradientas mažėja, ir svorių „pataisymai“ tampa vis mažesni.

Kad būtų sušvelninta lokalinių minimumų problema, mokymas atliekamas ne vieną, o keletą kartų, kiekvieną sykį pradedant nuo skirtingų pradinių svorių. Tai vadinamasis „*multistart*“ (daugiastartis) režimas. Kadangi mokant svoriai auga, tai ir gradientas mažėja. Dėl šios priežasties mokymo žingsnis mokymo metu turėtų būti didinamas. Bet čia gali „išlįsti“ kiti pavojai, pvz., galime peršokti į kitą minimumą. Dažnai vietoj gradientą minimizuojančio mokymo algoritmo naudojami sudėtingesni, antros eilės, metodai (žr. *Matlab'o* neuroninių tinklų paketą). Tada mokymas pagreitėja, bet atsiranda dar didesnė galimybė pakliūti į lokalinį minimumą. Dėl tos priežasties greitesniems, antros eilės, metodams *multistart* režimas reikalingas labiau.

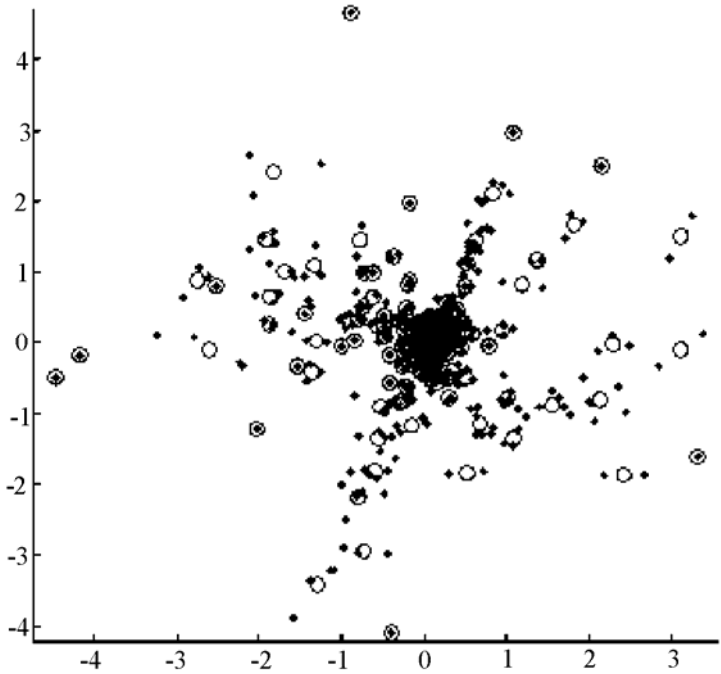
4.4. Spindulinių bazinių funkcijų ir mokymo vektorių kvantavimo neuroniniai tinklai

4.4.1. Mokymo vektorių kvantavimo neuroniniai tinklai

Vieno artimiausio kaimyno klasifikavimo taisyklėje sprendimas priimamas pagal klasifikuojamo vektoriaus $\boldsymbol{x}$ atstumą iki artimiausio mokymo vektoriaus $\boldsymbol{x}_{\text{artimiausias}}$. Vektorių $\boldsymbol{x}$ priskiriame tai klasei, kuriai priklauso vektorius $\boldsymbol{x}_{\text{artimiausias}}$. Kai mokymo vektorių n labai daug, atpažinimo procesas yra lėtas: reikia suskaičiuoti n atstumų. Vienas iš būdų – tai mokymo vektorius sugrupuoti kompaktiškomis grupėmis, kuriose vektoriai išsidėstę arti vienas kito. Tarkime, grupių kiekis visose klasėse bus r ($r << n$). Jų centrus pažymėsime $\boldsymbol{C}_1$, $\boldsymbol{C}_2$, ... , $\boldsymbol{C}_r$. Kiekvienai klasei priklausančius centrus galime rasti taikydami *blokinių analizės* metodus. Vektoriaus $\boldsymbol{x}$ klasifikaciją (prognozavimą taip pat) atliksime pagal artimiausio jam centro klasės numerį. Paprasta.

Gretimus skirtingoms klasėms priklausančius centrus skirsime hiperplokštumomis (faktiškai euklidinio atstumo klasifikatoriais). Tad atlikdami sprendimo priėmimo procesą jau minėtu būdu turėsime ga-

balais tiesinį klasifikatorių. 25 paveiksle dvimatėje specialiai sudarytų požymių erdvėje pavaizdavome 200+200 gerų ir blogų elektros variklių vibracijas atspindinčių vektorių ir blokinių analizės būdu surastus centrus, pažymėtus skrituliukais. Įpareigojame skaitytoją pažymėti klases skiriančią, iš tiesių sudarytą laužtinę liniją.



25 pav. Dvi elektros variklių klasės ir blokinių analizės metu rasti centrai klasėse

Pateiktame pavyzdyje ieškant centrų nebuvo kreipiamas dėmesys į klasių atskiriamumą. Mokymo vektorių kvantavimo neuroniniai tinklai tai daro. Naudodami specialią nuostolių funkciją jie tiksliau randa centrus ir leidžia gauti geresnį klasifikatorių. Minėti centrai suskirsto daugiamatę požymių erdvę į atskiras, hiperplokštumomis aprėžtas sritis – kvantus. Todėl jie ir vadinami šiuo vardu. *Matlab'o* neuroninių

tinklų pakete yra programos, leidžiančios sudaryti mokymo vektorių kvantavimo neuroninius tinklus.

4.4.2. Spindulinių bazinių funkcijų neuroniniai tinklai

25 paveiksle matyti, kad kai kurie centrai iš blogųjų elektros motorų (raudoni taškai) patenka tarp tankiai išsidėsčiusių gerųjų motorų (mėlyni taškai). Mokymo vektorių neuroninis tinklas nekreipia dėmesio į vektorių kiekius kiekvienoje grupėje, jų spindulius (tūrius). Spindulinių bazinių funkcijų (angl. *Radial Basic Functions*) neuroninis tinklas tai daro. Čia kiekvieno testinio vektoriaus indėlis į klasifikavimo procesą priklauso ne tik nuo grupės centro padėties, bet ir nuo to, kiek vektorių yra grupėje, koks vektoriaus $\boldsymbol{x}$ atstumas iki grupės centro $\boldsymbol{C}_r$.

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

$$išėjimas_i\,(\boldsymbol{x}) = \frac{1}{N_i}\sum_{j=1}^{N_i} v_{ij}\ \exp(-\,(\boldsymbol{x}-\boldsymbol{C}_{ij})\,(\boldsymbol{x}-\boldsymbol{C}_{ij})^T/\lambda_{ij}), \qquad (4.5)$$

kur N_i yra centrų skaičius i-tosios klasės j-tojoje grupėje, o parametrai $\boldsymbol{C}_{ij}$, λ_{ij} ir v_{ij} su indeksas „*ij*“ atitinkamai charakterizuoja *ij*-tosios grupės „centrą“, „spindulį“ ir „apriorinę tikimybę“.

Norint rasti nežinomas parametrų $\boldsymbol{C}_{ij}$, λ_{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 nežinomų parametrų reikšmės randamos blokinių analizės būdu. Blokinių centrai charakterizuoja pradines $\boldsymbol{C}_{ij}$ reikšmes, vektorių kiekis grupėje – parametrą v_{ij}, o mokymo vektorių grupėje vidutinis nuokrypis nuo „centro“ $\boldsymbol{C}_{ij}$ – parametrą λ_{ij}. *Matlab'o* neuroninių tinklų paketas turi tam reikalingą programinę įrangą.

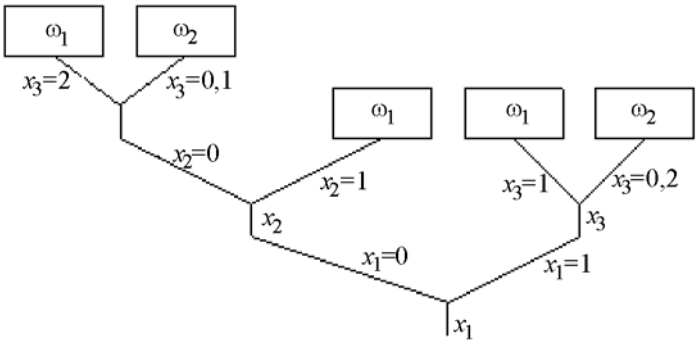
Viena vertus, spindulinių bazinių funkcijų neuroninis tinklas panašus į Parzeno lango klasifikatorių, kadangi skaičiuoja „branduolio funkciją“ *exp*(-*atstumas*($\boldsymbol{x}$, $\boldsymbol{C}_{ij}$). Kadangi centrų mažiau nei mokymo vektorių, tai atpažinimo fazėje SBF daug greitesnis nei Parzeno lango klasifikatorius, tačiau SBF reikalauja daug daugiau kompiuterinių išteklių parametrams išmokti. Už viską reikia mokėti. Mašinų moky-

mo (*machine learning*) ir dirbtinių neuroninių tinklų mokslinėse bendruomenėse yra netgi toks sparnuotas posakis „*there is no free lunch*“. 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 jie lėtesni.

4.5. Sprendimo medžiai

Sprendimo medžiai yra vieni iš pagrindinių instrumentų analizuojant didelius empirinių duomenų masyvus. Jie gali būti taikomi klasifikavimo ir prognozavimo uždaviniams spręsti. Jų veikimo principas paprastas, todėl juos labai mėgsta taikomųjų sričių mokslininkai: biologai, medikai, ekonomistai ir netgi filologai.

26 paveiksle pavaizdavome labai paprastą binarinį – iš binarinių šakų sudarytą – sprendimų medį, kur kiekviename mazge sprendimas šakojasi dviem kryptimis.



26 pav. Binarinis sprendimų medis

Medis sudarytas, kai požymiai binariniai – arba „nulis“, arba „vienetas“. Apačioje, medžio kamiene, pirmajame išsišakojime (maz-

ge), sprendimą priima „galingiausias“, reikšmingiausias požymis x_1. Jeigu $x_1 = 1$, tai sprendimas perduodamas dešinėje esančioms medžio *šakoms*, šiuo konkrečiu atveju – požymiui x_3. Jeigu $x_3 = 1$, tai daromas galutinis sprendimas: $\boldsymbol{x}$ priklauso klasei Π_1 (tai pirmasis medžio *lapas*). Jeigu $x_3 = 0$ arba 2, tai daromas galutinis sprendimas: $\boldsymbol{x}$ priklauso klasei Π_2.

Jeigu $x_1 = 0$, sprendimas perduodamas kairėje esančioms medžio šakoms. Pradžioje sprendimas atliekamas pagal požymį x_2. Jeigu $x_2 = 1$, tai daromas galutinis sprendimas: $\boldsymbol{x}$ priklauso klasei Π_1. Jeigu $x_2 = 0$, sprendimas perduodamas dar kairiau esančioms medžio šakoms, būtent požymiui x_3. Lipant medžiu dar aukščiau, vėl daromi galutiniai sprendimai: $\boldsymbol{x}$ priklauso klasei Π_1, jeigu $x_3 = 2$, ir $\boldsymbol{x}$ priklauso klasei Π_2, jeigu $x_3 = 0$ arba 1.

Pavaizduotame sprendimų medyje požymiai x_1, x_2, x_3 priima diskretines reikšmes 0, 1, 2. Sprendimų medžiai naudojami ir tais atvejais, kai požymiai priima tolydines reikšmes. Pvz., j-tajam požymiui priskiriama reikšmė „nulis“, jei $x_j < a_j$. Tam pačiam požymiui priskiriama reikšmė „vienetas“, jei $a_j \le x_j < b_j$. Reikšmė „dvejetas“ priskiriama, jei $x_j \ge b_j$. Sprendimo medžio architektūra, slenksčių a_j, b_j, $j = 1, 2, \ldots$, reikšmės turi būti nustatomos medžio sudarymo metu. Paminėtina, kad tie patys požymiai gali dalyvauti skirtinguose medžio mazguose. Slenksčiai gali skirtis. Atskiruose mazguose sprendimus priiminėti gali net atskiros požymių grupės. Pavyzdžiui, kažkuriame iš svarbių mazgų galime panaudoti vienasluoksnį perceptroną, paremtą septyniais požymiais.

Sudarant sprendimų medį labai svarbus klausimas – jo dydis. Esant fiksuotam mokymo duomenų kiekiui visada galime sukurti tokį sprendimų medį, kuris suskirstys visus mokymo vektorius be klaidų. Labiausiai tikėtina, kad toks medis bus blogas jį testuojant kitais, mokymo procese nenaudotais, duomenimis. Todėl „aukso vidurio“ problema čia neišvengiama. Kaip ir kituose duomenų analizės uždaviniuose, kad būtų nustatyta optimali medžio architektūra (medžio aukštis, šakų kiekis, lapų kiekis ir t. t.), tenka naudoti nepriklausomus, ankstesniame mokymo procese nenaudotus, validavimo duomenis (žr. 5.1 poskyrį).

Sprendimų medis gali būti naudojamas ne tik klasifikavimo (atpažinimo) uždaviniams spręsti, bet ir prognozavimo taisyklėms sudaryti. Daugelio autorių tyrimai parodė, kad tinkamai sudaryti sprendimų medžiai dažniausiai pralenkia kitus klasifikavimo ar prognozavimo metodus. Jie labai aiškūs tiek savo sudarymo koncepcija, tiek gauto rezultato interpretacija. *Matlab'o* pakete irgi yra programinė įranga sprendimo medžiams sudaryti.

4.6. Bendrieji požymių kiekio, imties tūrio ir tikslumo klausimai

2.1.2 ir 3.1.5 poskyriuose kalbėjome apie požymių, mokymo duomenų kiekio ir prognozavimo arba klasifikavimo taisyklių tikslumo sąryšį. Vienas nuo kito priklauso: augant mokymo imties tūriui laukiama klasifikavimo (arba prognozavimo) klaida mažėja, o augant požymių kiekiui – auga ir klaida (žinoma, jei požymių sistemos informatyvumas nesikeičia). Naudojant netiesinius algoritmus – tas pats. Pavyzdžiui, naudojant daugiamačiu normaliuoju tankiu paremtą kvadratinę diskriminacinę funkciją Q, vertinamų kiekvienos klasės parametrų kiekis yra proporcingas ne požymių skaičiui p, o jų kvadratui p^2, o laukiamos klasifikavimo klaidos augimas yra susijęs su p^2. Todėl kvadratinis klasifikatorius yra labai jautrus požymių kiekiui. Esant dideliam požymių kiekiui jo naudoti nereikėtų. Panaši situacija ir naudojant neparametrinį, Parzeno tankiu paremtą, klasifikatorių. Teoriniai išvedžiojimai rodo, kad tada laukiama klasifikavimo klaida auga eksponentiškai su požymių kiekiu p. Tai bloga šio klasifikatoriaus savybė. Laimė, teoriniai išvedžiojimai gauti tuo atveju, kai duomenys visomis kryptimis turi vienodą išsibarstymą. Iš tikrųjų duomenų išsibarstymas dažniausiai būna didelis tik keliose kryptyse. Kitose kryptyse išsibarstymas mažas, tad faktiškai tikrasis šio tipo klasifikatorių dimensiškumas yra ženkliai mažesnis nei požymių kiekis p. Bet vis tiek, esat ypač dideliam požymių kiekiui, šio tipo klasifikatorių naudingumas yra problemiškas.

Šiuo požiūriu spindulinių bazinių funkcijų ir mokymo vektorių kvantavimo neuroniniai tinklai turi privalumų. Klasifikavimo metu kiekvieni du gretimi priešingų klasių centrai formuoja euklidinio at-

stumo klasifikatorius. Jei juos modifikuotume taip, kad tie centrai nebūtų sudaromi iš daugelio kiekvienos klasės mokymo vektorių, tai mažo mokymo vektorių problema sušvelnėtų, nereikėtų jos taip bijoti. Panašių problemų kyla ir dėl daugiasluoksnio perceptrono. Dažnai DsP veikia gerai nepaisant labai pesimistinių teorinių prognozių, teigiančių, kad mokymo duomenų kiekis turi būti panašus ar net viršyti perceptrono svorių kiekį. Kaip sakoma, „ne toks velnias baisus, kaip jis piešiamas“. Aktyvacijų funkcijų netiesiškumai, esant dideliems svoriams, stabdo perceptrono sudėtingumo augimą, dėl to tik DsP išėjimo sluoksnio neuronų kiekis daugiausia ir nusako perceptrono sudėtingumą. Daugelis šiame poskyryje minėtų problemų nėra pakankamai gerai teoriškai išnagrinėtos ir laukia savo sprendėjų.

PENKTAS SKYRIUS. POŽYMIŲ SISTEMOS FORMAVIMAS

5.1. Klasifikavimo ir prognozavimo tikslumo vertinimo metodai

5.1.1. Tikslumo vertinimo problema

Parenkant tinkamiausią nagrinėjamam uždaviniui spręsti klasifikavimo ar prognozavimo metodą, nustatant optimalų perceptrono mokymo iteracijų kiekį, išrenkant požymius būtina įvertinti gauto duomenų analizės algoritmo darbo tikslumą. Praktiškai duomenys retai būna normalieji. Todėl pasinaudoti teoriniais rezultatais ne visada galima. Teorija reikalingiausia bendram problemos supratimui. Konkretiems tikslumo įvertinimams atlikti reikia praktikos. Kalbant konkrečiau, turint jau sudarytą klasifikavimo ar prognozavimo taisyklę reikia ją išbandyti praktiškai. O tai įmanoma tiktai turint realius duomenis.

2.1.2 poskyryje kalbėdami apie parametrus, naudojamus prognozavimo tikslumui įvertinti, parodėme, kad tikslumo įverčiai, gauti antrąkart naudojant mokymo vektorius prognozavimo paklaidai įvertinti, būna smarkiai pasislinkę (optimistiniai, rodantys geriau nei iš tikrųjų), jeigu požymių kiekis yra artimas mokymui panaudotų vektorių kiekiui. Tas pats prisiderinimo prie mokymo duomenų efektas stebimas ir klasifikavimo uždavinyje. Todėl šis tikslumo vertinimo būdas laikomas nepatikimu ir šiuo metu taikomas retai. Išimtis galėtų būti uždaviniai, kuriuose mokymo duomenų iš tikrųjų yra labai daug.

Kad būtų patikimai įvertintos klasifikavimo arba prognozavimo klaidos, reikia naujų, iki tol nė karto nenaudotų, duomenų. Jie vadinami *validavimo (tikrinimo) duomenimis*. Čia jokių problemų neiškyla, jeigu duomenų validavimui turime pakankamai daug. Problemų atsiranda, kai duomenų nėra daug, ir tyrinėtojas turi suskaidyti juos į dvi dalis: mokymo ir validavimo. Tikslumo vertinimo metodas, kai juos skaidome į mokymo ir validavimo, angliškai vadinamas *cross validation*. Kai kur jis dar tebevadinamas *hodl-out* metodu. Turimus duomenis suskaidžius į dvi dalis, mokymo duomenų kiekis sumažėja.

Jų pradeda nebeužtekti sudėtingam algoritmui sudaryti. Tada kyla būtinybė dalį charakteristikų (požymių) atmesti. O tam vėl reikia duomenų. Užburtas ratas! Todėl šiame skyriuje pakalbėsime apie įvairias „gudrybes“, pasiūlytas išvengti minėto skaidymo.

5.1.2. Klasifikavimo klaidos vertinimo tikslumas

Tarkime, validavimo imtyje yra n_{val} vektorių, o tikroji jau sukurto klasifikatoriaus klasifikavimo klaidos tikimybė yra P_n^{A}. Tarkime, atlikus n_{val} vektorių klasifikavimą, $n_{\text{val}}^{\text{klaid}}$ iš jų atpažinti neteisingai. Tokiu atveju klasifikavimo klaidos įvertis bus toks:

$$\hat{P}_n^{\text{A}} = \frac{n_{\text{val}}^{\text{klaid}}}{n_{\text{val}}}, \tag{5.1}$$

kur stogelis virš vertinamo parametro išraiškos pabrėžia, kad tai ne tikslus dydis, o iš eksperimentinių duomenų gautas įvertis.

Įvertis $\hat{P}_n^{\text{A}}$ yra gautas iš atsitiktinai surinktų validavimo (tikrinimo) duomenų. Todėl jis laikytinas atsitiktiniu dydžiu. Jei mokymo ir validavimo duomenys statistiškai nepriklausomi, tai $\hat{P}_n^{\text{A}}$ yra nepaslinktas P_n^{A} įvertis: $\hat{P}_n^{\text{A}}$ matematinė viltis lygi P_n^{A}. Kadangi $n_{\text{val}}^{\text{klaid}}$ yra taip pat atsitiktinis dydis, pasiskirstęs pagal binominį pasiskirstymą, tai įverčio $\hat{P}_n^{\text{A}}$ dispersija tokia:

$$\text{Var}\,(\hat{P}_n^{\text{A}}) = \frac{P_n^{\text{A}}(1-P_n^{\text{A}})}{n_{\text{val}}}. \tag{5.2}$$

Išraiška (5.2) rodo, kad kuo didesnė validavimo imtis, tuo tikslesnis P_n^{A} įvertinimas. Praktiškai P_n^{A} nežinome, todėl norėdami įvertinti įverčio $\hat{P}_n^{\text{A}}$ tikslumą vietoj P_n^{A} esame priversti naudotis atsitiktiniu

dydžiu – pačiu įverčiu $\hat{P}_n^{\mathrm{A}}$. Dėl tos priežasties klasifikavimo klaidos tikimybės įverčio tikslumas gaunamas faktiškai žemesnis, nei nurodyta formule (5.2). Tikslumas sumažėja dar ir todėl, kad nemažoje dalyje praktinių atvejų validavimo ir mokymo imtys nebūna statistiškai nepriklausomos.

Grįžkime prie prognozavimo uždavinio. Tarkime, tikroji prognozavimo lygties $\boldsymbol{z}^T\boldsymbol{v}$ paklaida yra σ_n^2. Panagrinėkime atvejį, kuriame validavimo duomenys naudojami jau iš mokymo duomenų sudarytos prognozavimo lygties kvadratinei paklaidai įvertinti:

$$(\hat{\sigma}_n^{\mathrm{valid}})^2 = \frac{1}{n_{\mathrm{valid}}} \sum_{j=1}^{n_{\mathrm{valid}}} (y_j - \boldsymbol{z}_j^{\boldsymbol{T}}\boldsymbol{v})^2 \quad . \tag{5.3}$$

Tada įverčio (5.3) dispersija tokia:

$$\mathrm{Var}\,((\hat{\sigma}_n^{\mathrm{valid}})^2) = \frac{2(\sigma_n^2)^2}{n_{\mathrm{val}}}. \tag{5.4}$$

5.1.3. Duomenų masyvo skaidymas į mokymo ir validavimo imtis

Turimus duomenis suskaidžius į dvi dalis, sumažėja tiek mokymo, tiek ir validavimo imtys. Kai duomenų nedaug, jų skaidymas tampa skausminga problema. Turint galingus skaičiavimo išteklius duomenų skaidymo problemą galima iš dalies „apeiti“ taikant kryžminio skaidymo metodą (angl. *k-fold crosss validation*). Vadovaujantis šiuo metodu, *n* turimų vektorių atsitiktinai „sumaišomi“ ir padalijami į *k* daugmaž lygių dalių. Klasifikavimo ar prognozavimo algoritmo mokymas atliekamas sujungus (*k*-1) dalių į laikinąją mokymo imtį. Validavimas (klasifikavimo ar prognozavimo klaidos vertinimas) atliekamas naudojant likusią (nepanaudotą mokymui) duomenų dalį. Toks eksperimentas atliekamas *k* kartų, kaskart keičiant validavimui naudotų duomenų dalį. Taigi mokymui naudojami kaskart beveik visi turimi vektoriai (jei *k* didelis), o įvertinant tikslumą taip pat dalyvauja visi *n*

turimų vektorių. Ribiniu atveju, kai norima kiek įmanoma padidinti mokymo imties tūrį, $k = n$. Tada šis metodas vadinamas *slenkančiuoju egzaminu*.

Reikėtų paminėti, kad klasifikavimo arba prognozavimo tikslumo vertinimas labai priklauso nuo duomenų sumaišymo pačioje eksperimento pradžioje. Tai ypač pastebima, kai $k = 2$. Tad prognozavimo ar klasifikavimo algoritmo *tikslumo* vertinimo *tikslumui* pakelti rekomenduotina maišymą atlikti daug (dešimtis ar net šimtus) kartų. Atkreipiame skaitytojų dėmesį į tai, kad šiame sakinyje žodis *tikslumas* pavartotas du kartus: mes vertiname *tikslumo rodiklio tikslumą.*

5.1.4. Asimptotinių klasifikavimo ir prognozavimo klaidų vertinimas

Išrenkant svarbiausius, tiriamąjį reiškinį veikiančius, požymius ar parenkant klasifikavimo arba prognozavimo algoritmo tipą, kyla klausimas, ar galima įvertinti ne tik laukiamą, bet ir asimptotinę algoritmo klaidą. Atsakymas yra teigiamas: taip, galima. Tam reikia dar kartą žvilgtelėti į 7 paveikslą ir formules (2.6) bei (2.7). Matome, kad esant gan dideliems mokymo imties tūriams (kai $\sqrt{\frac{n}{n-p}} < 1{,}5$) laukiamos ir įsivaizduojamos prognozavimo klaidos grafikai darosi simetriški linijos $\sigma_0 = constant$ atžvilgiu. Panašus simetriškumas stebimas ir klasifikavimo uždavinyje. Pavyzdžiui, tiesinio Fišerio klasifikatoriaus įsivaizduojamos (empirinės) klasifikavimo klaidos

$$\hat{P}_n^{\text{F isivaizd}} = \frac{n_{\text{mok}}^{\text{klaid}}}{n_{\text{mok}}}, \tag{5.5}$$

gautos iš n_{mok} mokymo imties vektorių, matematinė viltis gali būti rasta pagal formulę

$$\text{E}\,\hat{P}_n^{\text{F isivaizd}} = \Phi(-\frac{1}{2}\delta/(T_n^{\boldsymbol{\mu}}\, T_n^{\boldsymbol{\Sigma}}), \tag{5.6}$$

kur koeficientai $T_n^{\boldsymbol{\mu}}$ ir $T_n^{\boldsymbol{\Sigma}}$ apibrėžti (3.11) ir (3.12) formulėmis.

Kitas būdas, kaip turint tik empirinę klasifikavimo klaidą $\hat{P}_n^{\text{F isivaizd}}$ analitiniais skaičiavimais įvertinti Fišerio klasifikatoriaus generalizavimo paklaidą, yra aprašytas 7.2 poskyrio pabaigoje.

Vadovaujantis simetriškumo prielaida, Fišerio klasifikatoriaus asimptotinės klasifikavimo klaidos įvertinimu galėtų būti toks vidurkis:

$$\hat{P}_{\infty}^{\text{F}} = \frac{1}{2}\left(\hat{P}_n^{\text{F}} + \hat{P}_n^{\text{F isivaizd}}\right), \qquad (5.7)$$

kur $\hat{P}_n^{\text{F}}$ ir $\hat{P}_n^{\text{F įsivaizd}}$ yra laukiamos ir įsivaizduojamos klasifikavimo klaidos įverčiai, gauti taikant kryžminio skaidymo metodą (*k-fold crosss validation*), kai $k = 2$.

Vietoj kryžminio skaidymo metodo su $k = 2$, $\hat{P}_n^{\text{F}}$ įvertinimui galima naudoti *slenkantįjį egzaminą* ($k = 2$), o įsivaizduojamą klaidą $\hat{P}_n^{\text{F įsivaizd}}$ vertinti eksperimentu, kai visi duomenys buvo naudojami ir mokant, ir testuojant. Variantų daug. Asimptotinės klaidos vertinimai, analogiški ką tik aprašytajam, gali būti naudojami ir taikant kitus klasifikavimo metodus, prognozavimo lygties sudarymo būdus.

5.2. Dimensiškumo mažinimas. Požymių išrinkimo metodai

Kaip minėta, atliekant dėsningumų paiešką empiriniuose duomenyse labai svarbus duomenų analizės algoritmo sudėtingumo ir mokymo duomenų kiekio ryšys. Algoritmo sudėtingumas iš esmės priklauso nuo nagrinėjamų požymių (rodiklių, charakteristikų) kiekio. Kuo daugiau požymių nagrinėsime, tuo sudėtingesnis statistine prasme bus duomenų analizės algoritmas. Kai kuriuose praktikoje pasitaikančiuose uždaviniuose (pavyzdžiui, genų išraiškos (angl. *gene expression*), didelės rezoliucijos spektrų) požymių yra tūkstančiai ar net dešimtys tūkstančių. Šio tipo uždaviniuose stebėjimo vektorių kartais būna netgi visai mažai. Tad kyla kausimas, kaip požymių kiekį sumažinti neprarandant vertingos informacijos. Šis uždavinys yra la-

bai sunkus. Ypač kai reikia įrodyti, kad duomenų analizės metu gautos išvados yra patikimos.

Kalbėdami apie dimensiškumo mažinimą atkreipsime dėmesį į skirtumą tarp požymių išrinkimo (*feature selection*) ir požymių išskyrimo (*feature extraction*). Pirmuoju atveju iš p požymių atrenkamas mažesnis kiekis, sakykim, r ($r < p$ arba $r << p$) „labiausiai informatyvių“ požymių. Šiuo atveju realizuojant atpažinimo algoritmą išmatuoti reikia tik r požymių. Antruoju atveju sukonstruojama r skirtingų funkcijų $x_{\text{naujas } j} = f_j(\boldsymbol{x})$ ($j = 1, 2, \ldots, r$). Pastaruoju atveju realizuojant atpažinimo algoritmą išmatuoti reikia visus p požymių.

Matuojamos charakteristikos (požymiai) paprastai būna statistiškai priklausomos, taigi idealiai išrinkti požymius įmanoma tik peržiūrėjus visas įmanomas kombinacijas, sudarytas iš r požymių, o jų daug: $C_p^r = \dfrac{p!}{(p-r)!r!}$. Pavyzdžiui, norint iš 30 požymių išrinkti 10 geriausių, reikėtų peržiūrėti 30 045 015 kombinacijų. Tai neįmanoma. Todėl tai atliekama paprastinimo būdu.

Nuoseklaus pridėjimo metodo (angl. *forward selection*) pirmame etape peržiūrima p požymių individualiai ir išrenkamas geriausias. Paskui peržiūrima p-1 požymių porų, kur vienas požymis jau išrinktas. Išrenkama „geriausia pora“. Trečiame etape peržiūrima p-2 požymių trejetukų. Procesas tęsiamas iki r požymių.

Nuoseklaus atmetimo metodo pradžioje perrenkama p požymių rinkinių, kuriuose trūksta vieno iš p požymių. Išrenkamas ir fiksuojamas geriausias p-1 požymių rinkinys. Kitame etape iš šio (geriausio) rinkinio p-1 kartų atimama po vieną požymį ir vėl išrenkamas geriausias rinkinys. Procesas tęsiamas tol, kol suformuojamas r požymių rinkinys. Akivaizdu, kad šis metodas netinka, jei p ypač didelis. Galimi ir „kombinuoti variantai“, kai paeiliui po vieną du požymius pridedama arba atimama. Tokių, kombinatorika paremtų, algoritmų yra daug.

Be kombinatorika pagrįstų požymių išrinkimo algoritmų, galime taikyti „atsitiktinę paiešką“, kur daug kartų atsitiktinai generuojama r požymių ir išrenkamas geriausias variantas. Nepaisant paprastumo, šis metodas yra efektyvus. Šio metodo efektyvumas išauga, jei kokiu nors būdu pasiseka nustatyti keletą ar keliolika „tikrai gerų požymių“, kurie įtraukiami į visus atsitiktinai generuojamus požymių rinkinius.

5.3. Duomenų projektavimas į žemo dimensiškumo erdvę taikant statistinius metodus ir naudojant neuroninius tinklus

5.3.1. Dimensiškumo mažinimo problema

Jei atskiri požymiai tarpusavyje statistiškai priklausomi, kai kada neįmanoma atrinkti mažą informatyvių požymių kiekį. Tad be požymių išrinkimo, požymių išskyrimas vaidina svarbų vaidmenį sprendžiant duomenų analizės ir dėsningumų paieškos uždavinius. Išskiriant požymius specialiai sudaromas mažas kiekis naujų požymių, kurių erdvėje klasifikavimo, prognozavimo ar klasterizavimo uždaviniai tampa daug paprastesni. Ypač plačiai požymių išskyrimas taikomas „tiriamojoje duomenų analizėje“ (angl. *exploratory data analysis*), kai norima į duomenis „pažvelgti iš šalies“, projektuojant juos į dvimatę arba trimatę erdves, kad būtų galima įvairiais rakursais juos pamatyti kompiuterio ekrane. Formalūs kriterijai, realizuoti naudojant duomenų analizės programinę įrangą, ne visada išsprendžia duomenų analizatoriaus keliamus reikalavimus, todėl pastarasis ir norėtų, žvelgdamas į kompiuterio ekraną, pamatyti, ar klasės skiriasi, o jei skiriasi, tai ar tiesiškai, ar ne; nustatyti, ar ryšys tarp požymių tiesinis ar ne; kaip smarkiai klasės kertasi. Pavieniais atvejais vizuali, preliminari duomenų analizė padeda suskirstyti duomenis į klases, konstatuoti netipinių stebėjimų buvimą ir dar daugelį iš anksto nenumatytų dalykų.

Požymių išskyrimo ir jų duomenų projektavimo į dvimatę ar trimatę erdves metodų yra labai daug. Panagrinėsime keletą iš jų.

5.3.2. Pagrindinių komponenčių metodas

Tai pats seniausias ir populiariausias metodas. Pagal šį metodą duomenų vektoriai yra „pasukami“ (padauginami iš ortogonalios matricos **G**)

$$\boldsymbol{x}_{\text{nauji}} = \boldsymbol{x}\,\mathbf{G} \tag{5.8}$$

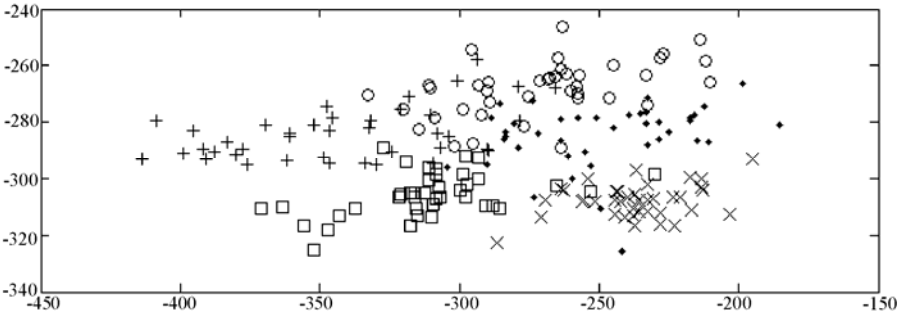
tam, kad viename naujame požymyje duomenų dispersija (išsibarstymas) σ_1^2 būtų didžiausia. Priminsime, kad ortogonali dydžio $p \times p$ matrica **G** turi tokią savybę:

$$\mathbf{G}\mathbf{G}^T = \mathbf{G}^T\mathbf{G} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{bmatrix}. \qquad (5.9)$$

Pagal šį metodą antrasis požymis (nauja kryptis) turi būti statmenas (ortogonalus) pirmajam ir turėti maksimalią dispersiją σ_2^2 iš to duomenų variabiliškumo, kuris dar liko atmetus pirmąją kryptį. Taip einama tolyn dispersijos mažėjimo kryptimi. Ortogonaliai duomenų pasukimo matricai (naujoms duomenų kryptims, naujiems požymiams) rasti taikomas pagrindinių komponenčių metodas. Randama duomenų kovariacinė matrica **S**, kuri išskaidoma į ortogonaliąją pasukimo matricą **G** ir diagonalinę duomenų dispersijų matricą:

$$\mathbf{\Delta} = \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \sigma_p^2 \end{bmatrix}: \qquad \mathbf{S} = \mathbf{G}\,\mathbf{\Delta}\,\mathbf{G}^T. \qquad (5.10)$$

Šiam metodui pailiustruoti 27 paveiksle pateikiame penkių rūšių kviečių duomenis, pagrindinių komponenčių metodu suprojektuotus iš 12-matės erdvės į dvimatę. Kiekviena kviečių rūšis pavaizduota skirtingais ženklais ir spalvomis. Paveikslėlis rodo, kurios klasės skiriasi labiau, kurios mažiau. Klasės šiek tiek skyrėsi ir originalių požymių erdvėje, tačiau duomenis pasukus jų atsiskyrimas tapo geriau pastebimas.



27 pav. Penkių klasių kviečių duomenys, suprojektuoti į dvimatę erdvę

5.3.3. Požymių išskyrimas naudojant daugiasluoksnius perceptronus

Gan populiarus būdas sumažinti požymių erdvę yra vadinamasis *asociatyvinis dirbtinis neuroninis tinklas* (ADNT), sudarytas naudojant daugiasluoksnį perceptroną. Paprasčiausiame variante ADNT – tai prognozavimui naudojamas DsP (jo įėjimuose esančiuose VsP aktyvacijos funkcijos yra tiesinės) su vienu paslėptuoju sluoksniu, turintis p įėjimų, p išėjimų ir r paslėptųjų neuronų. Primename, kad r – tai naujųjų, išskirtųjų, požymių skaičius. Kad suprojektuotume duomenis į mažesnio dimensiškumo erdvę, perceptroną mokome su turimais p-mačiais duomenimis kaip trokštamų išėjimų vektorius naudodami tuos pačius įėjimo vektorius. Tad šiuo atveju: $\boldsymbol{y}_j = \boldsymbol{x}_j, j = 1, 2, \ldots, n$.

Paslėptojo sluoksnio neuronai suskaičiuoja pasvertąsias sumas, aprašomas lygtimis (4.4). Jų išėjimai, kaip minėta 4.3 poskyryje patenka į aktyvacijos funkcijų įėjimus (žr. formulę (3.15)), kur jie netiesiškai transformuojami ir tampa DsP išėjimo sluoksnio įėjimais. Apmokius perceptroną paslėptajame sluoksnyje gauti išėjimai

$$s_v = \sum_{v=1}^{p} x_{v1v} + w_0 \ , \qquad j = 1, 2, \ldots r$$

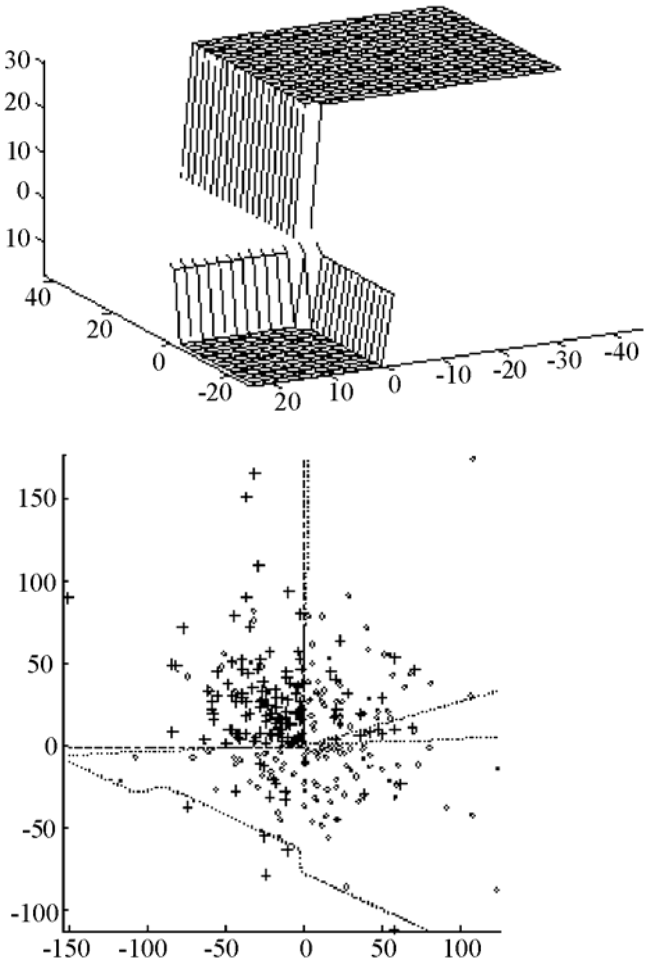
ir yra naujieji, ką tik išskirtieji, požymiai.

Tai tiesinis požymių išskyrimas, tačiau kriterijus, pagal kurį skaičiuojama požymių išskyrimo kokybės funkcija, yra netiesinis (paslėp-

tajame sluoksnyje skaičiuojant mokymo kokybės kriterijaus reikšmes naudojamos netiesinės aktyvavimo funkcijos). Šiuo metodu p originaliųjų požymių yra „suspaudžiami“ iki r jų, o paskui išėjimo sluoksnis bando juos atstatyti iki p požymių. Jau buvome aprašę perceptroną, turintį vieną paslėptąjį sluoksnį. Be šio varianto, kai kada dar naudojamas ir sudėtingesnis, kur turime tris paslėptuosius sluoksnius. Kiekvieno iš jų neuronai turi netiesinę aktyvavimo funkciją. Tuomet ir požymių išskyrimas tampa netiesinis. Reikia paminėti, kad šiuo atveju perceptrono mokymas tampa daug sunkesnis, reikalaujantis gerokai daugiau mokymo iteracijų. Yra daug didesnis pavojus pakliūti į minimizuojamos nuostolių funkcijos „blogą“ lokalinį minimumą. Todėl mokymo procesas kartojamas daugelį kartų, kiekvieną kartą pradedant nuo skirtingų pradinių svorių vektorių. Tokio tipo požymiams išskirti reikia naudoti gana daug mokymo vektorių. Jei jų bus per mažai, galime pernelyg smarkiai prisiderinti prie mokymo duomenų, dėl to toks požymių išskyrimas gali tapti visiškai nepatikimas. Apie šį prisiderinimo prie mokymo duomenų efektą dar kalbėsime šio poskyrio pabaigoje.

Aprašytuose požymių iškyrimo metoduose klasių indeksai nebuvo naudojami. Šiuos metodus buvo galima taikyti tiek klasifikavimo, tiek prognozavimo uždaviniams. Dabar paminėsime metodą, panašų į asociatyvinį DNT, kur klasių indeksai vaidina esminį vaidmenį – jie valdo požymių išskyrimo procesą. Požymiams, reikalingiems sprendžiant K klasių atpažinimo uždavinį, išskirti naudokime daugiasluoksnį perceptroną su r paslėptų netiesinių VsP. Skirtingai nuo asociatyvinio DNT, perceptrono išėjimuose esantys neuronai turės netiesinę aktyvacijos funkciją. Skirtingai nuo asociatyvinio DNT, mokymo signalas bus nebe vektorius $\boldsymbol{y}_j = \boldsymbol{x}_j$, o vektorius, sudarytas iš vienetukų ir nuliukų, kaip priimta tradiciniame klasifikavimui naudojamame DsP. Vienetukai bus tik tose iš K pozicijų, kurios atitinka į perceptroną patenkančio mokymo vektoriaus $\boldsymbol{x}$ klasės numerį. Kaip ir asociatyviniame DNT, šiuo atveju požymių išskyrimas bus tiesinis (kaip nauji požymiai naudojamos sumos $s_v = \sum_{v=1}^{p} x_{v1v} + w_0$, $j = 1, 2, \ldots r$), o minimizuojamas kriterijus – netiesinis.

28 paveikslo kairėje pavaizdavome daugiasluoksnio perceptrono su trimis paslėptais neuronais skiriamąjį paviršių trimatėje naujų požymių erdvėje. Matome, kad paviršius yra sudėtingos formos, netiesinis.



28 pav. Klases skiriančios ribos trimatėje ir dvimatėje erdvėse su DsP gautų požymių

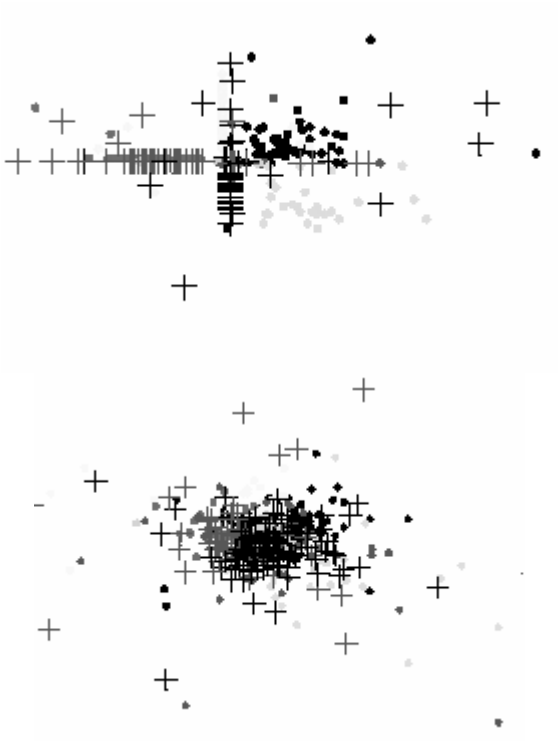
Statiniu režimu trimatėje erdvėje sunku pavaizduoti duomenis, kad būtų lengva juos analizuoti, tyrinėti įvairiausiais rakursais. Todėl šio paveikslo dešinėje, naudodami VsP su dviem paslėptais neuronais, suprojektavome duomenis į dvimatę erdvę. Matome, kad dvi klasės (sąlygos, kai naftos kaina augs, – pirmoji klasė, ir sąlygos, kai naftos kainos mažės, – antroji klasė) skiriasi gerai. Paprastesnė skiriamoji riba (brūkšneliais) gauta mokant perceptroną trumpiau, sudėtingesnė riba (taškais) gaunama mokant perceptroną ilgiau. Matome, kad ir požymių išskyrimo uždavinyje permokymo efekto neišvengiame. Tad atsargiems reikia būti ir čia.

Požymius išskiriant į naują požymių erdvę projektuojantis algoritmas yra paremtas informacija, esančia mokymo duomenyse. Todėl vertinant požymių išskyrimo procedūros efektyvumą, reikia stebėti klasių vektorių išsidėstymą ne tik mokymo, bet ir validavimo duomenyse. Taip išvengsime pernelyg optimistinių išvadų, nes DsP vis dėlto yra gan sudėtingas, daugeliu parametrų (svorių) aprašomas modelis.

Toliau pateikiame dešimties klasių mokymą ir testinių vektorių išsidėstymą dviejų naujų, gautų pagal ką tik aprašytąją metodiką, požymių erdvėje. Šiame uždavinyje buvo nagrinėjami 10 klasių 1500 spektrinių požymių duomenys, charakterizuojantys dešimčia skirtingų ligų užkrėstų mielių bandinius. Nepaisant labai paprastos daugiasluoksnio perceptrono architektūros (tik du paslėpti neuronai), pradinių požymių kiekis (1500) palyginti su 500 vektorių, naudojamų perceptrono mokymui, vis dėlto buvo per didelis. Tuo ir paaiškinamas faktas, kad testiniuose (validavimo) duomenyse klasių struktūros nebematome.

Šis pavyzdys pateiktas specialiai, kad parodytų, jog *išvadomis, paremtomis naudojant tik mokymo duomenis, ne visada galime tikėti.*

Šis pavyzdys pateikiamas ir dėl to, kad duomenų analitikas visada turėtų galvoje, jog egzistuoja realus pavojus rasti ne tik duomenyse esančius dėsningumus, bet ir juose nesančius dėsningumus. Štai 29 paveikslo viršutinis piešinys akivaizdžiai rodo ryškią duomenų struktūrą. Patikrinus su validavimo duomenimis, jos jau nebematome. Su šiais duomenimis dar kartą susidursime 8 skyriuje, skirtame teorinei medžiagai įsisavinti praktiškai.



29 pav. 500 mokymo ir 501 testinių vektorių projekcijos į dvimatę daugiasluoksniu perceptronu gautų požymių erdvę

ŠEŠTAS SKYRIUS. DĖSNINGUMŲ PAIEŠKA BESIKEIČIANČIOJE APLINKOJE

6.1. Genetiniai mokymo algoritmai

Tradiciniais pirmosios ar antrosios eilės optimizavimo metodais paremtų daugiasluoksnių perceptronų, spindulinių bazinių funkcijų, mokymo vektorių kvantavimo neuroninių tinklų mokymas labai jautrus lokaliniams minimumams. Genetiniai optimizavimo algoritmai – tai patobulinti atsitiktinės paieškos algoritmai. Savo veikimo principą jie „pasiskolino“ iš gamtos. Palyginti su pirmosios ar antrosios eilės optimizavimo metodais, genetiniai algoritmai daug lėtesni, tačiau jie sėkmingiau sprendžia lokalinių minimumų problemas. Augant kompiuterių skaičiavimo greičiams, jie tampa vis populiaresni. Jie taip pat naudojami ir duomenų analizės uždaviniuose. Šiame poskyryje išnagrinėsime jų veikimo principą.

Mokant dirbtinį neuroninį tinklą reikia nustatyti jo svorių w_1, w_2, ..., w_P reikšmes. Akivaizdu, kad šių svorių reikšmes galime užrašyti naudodami ne tik dešimtainę, bet ir dvejetainę sistemas. Pastaruoju atveju svoriai w_1, w_2, ..., w_P gali būti užkoduoti ilga vienetukų ir nuliukų seka. Pavyzdys:

1 1 0 1 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 1 1 1 0 0 1 1 0 1 0 0 0 1 1 1 1 0 0 0
1 0 1 1 1 0 1 0

Šią ilgą seką suskirstykime į dalis:

1 1 0 1 0 0 0 1 0 1 1 0 1 0 1 0 0 0 0 1 0 1 1 1 0 0 1 1 0 1 0 0 0
1 1 1 1 0 0 0 1 0 1 1 1 0 1 0

Remdamiesi genetikos terminais, visą seką pavadinkime genetiniu kodu, o atskiras to kodo dalis – genomomis. Viena ar dvi genomos galėtų aprašyti vieną dirbtinio neuroninio tinklo svorį. Kaip ir paprasčiausiu atsitiktinės paieškos optimizavimo metodu iš pradžių sugene-

ruokim aibę, sakykim, n_{gen} genetinių kodų. Turėdami mokymo duomenis, apskaičiuokime nuostolių funkciją, kuri galėtų būti viena iš nagrinėtųjų šioje knygoje. Genetinių algoritmų teorijoje ši minimizuojamoji (arba maksimizuojamoji) funkcija vadinama tvirtumo (*fitness*) vardu. Surikiuokime tvirtumo funkcijos reikšmes pagal dydį ir išrinkime n_{ger} geriausių. Tai grupė genetinių kodų, kuriems bus leista generuoti naujus genetinius kodus – „vaikus“. Iš dviejų geriausių grupių parinktų kodų genomų, atsitiktinai parinkę jų numerius, sudarome naują genetinį kodą. Toliau pateiktas pavyzdys – du skirtingi genetiniai kodai.

1 1 0 1 0 0 0 1 0 1 1 0 1 0 0 1 0 0 0 1 1 1 0 1 1 1 0 1 0 0 0
1 1 0 0 0 1 0 1 0 1 0 0 0 1 0

1 1 0 1 0 1 1 1 0 0 0 0 0 0 0 1 0 1 0 1 1 1 1 1 1 1 0 1 0 0 0
1 1 1 1 0 1 0 1 0 1 0 0 1 0 1

Štai du iš ankstesniųjų atsitiktinai surinktų kodų:

1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 1 0 1 0 0 0
1 1 0 0 0 1 0 1 0 1 0 0 1 0 1

1 1 0 1 0 1 1 1 0 0 0 0 0 0 0 1 0 1 0 1 1 1 0 1 1 1 0 1 0 0 0
1 1 1 1 0 1 0 1 0 1 0 0 0 1 0

Ši operacija vadinama „kryžminimu“ (angl. *cross-over*). Gamtos pavyzdžiu dar įvedama mutacija, kuri palengvina išbristi iš blogų lokalinių ekstremumų (minimumų – mūsų nagrinėjamu atveju). Toliau pateiktas pavyzdys, kuriame mutacijos pažymėtos juodu šriftu.

1 1 0 1 0 0 **1** 1 0 0 0 0 0 0 0 1 0 0 0 1 1 **0** 1 1 1 1 0 1 0 0 0
1 1 0 **1** 0 1 0 1 0 1 0 0 1 0 1

1 1 0 1 0 1 1 1 0 0 1 **1** 0 0 0 0 1 0 1 0 1 1 1 0 1 **1** 0

1 1 1 1 0 1 0 1 0 1 0 0 0 1 0

Taip sudaroma nauja genetinių kodų grupė. Ją nagrinėjant kitame (antrame) mokymosi proceso cikle vėl įvertinamos tvirtumo funkcijos reikšmės, atrenkami geriausi genetiniai kodai, vykdomos kryžminimo operacijos, vėl atsitiktinai generuojamos mutacijos. Taip mokymosi procesas tęsiasi. Jo metu tvirtumo funkcijos reikšmė auga, svoriai vis tikslinami. Jei mokymosi algoritmo parametrai yra n_{gen}, n_{ger}, mutacijų dažnis ir kiti nustatomi neteisingai, procesas gali sustoti ir anksčiau.

Mes aprašėme paprasčiausią variantą. Šį algoritmą tobulindami galėtume į parametrus, kuriuos reikėtų išmokti, įtraukti ir n_{gen}, n_{ger} bei mutacijų dažnį ir 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 irgi gamtos evoliucijos pasiūlyta idėja. Kai kuriose genetinių algoritmų modifikacijose vietoj vieno tvirtumo kriterijaus naudojama keletas. Kad būtų patekta į geriausiųjų grupę, genetinis kodas turi patenkinti ne vieną, o keletą ar net visus kriterijus. Pavyzdžiui, sprendžiant atpažinimo uždavinį kriterijai galėtų būti:

- pirmosios rūšies klaidos tikimybė (žr. 3.1.1 poskyrį),
- antrosios rūšies klaidos tikimybė,
- mokymosi laikas,
- atpažinimo laikas,
- kompiuterio atmintis, reikalinga tarpiniams rezultatams, duomenims saugoti.

Per pastarąjį dešimtmetį atsirado kombinuotieji mokymo metodai (angl. *memetic algorithms*), kur mokymo procese kartu dalyvauja ir genetiniai, ir gradientiniai, arba antrosios eilės, optimizavimo algoritmai. Po kiekvieno genetinio mokymo ciklo įvyksta adaptacinis mokymosi ciklas. Tvirtumo funkcija(-os) skaičiuojama tik po įvykusio adaptacinio mokymosi. Paveldimos tik genetinio mokymosi metu įgytos svorių reikšmės. Tai gamtos evoliucijos inspiruota idėja.

6.2. Daugiaagentės sistemos. Kintančių dėsningumų radimas

Veikiamas netiesinių, chaotiškų, gamtos ir žmogaus ūkinės veiklos sukeltų procesų, pasaulis nuolat, kasdien, kas metai keičiasi. Tai tapo pastebima naudojant naujas technologijas, juolab kad jos minėtą kitimą gerokai ir lemia. Tapo būtina tuos kitimus analizuoti, prognozuoti jų tendencijas. Kintant aplinkai, kinta ir matuojamos charakteristikos, jas aprašantys požymiai. Dėl šios priežasties duomenys tampa nebehomogeniški, ir jų kiekio didinimas, paremtas ilgesniu stebėjimo laiku, lemia tai, kad stebimi dėsningumai gali būti prarasti. Pats kitimo procesas vyksta netolygiai, nevienodu greičiu. Pažymėtina, kad skirtingos charakteristikos gali kisti įvairiu greičiu. Tad tampa neįmanoma nustatyti optimalaus daugiamačių stebėjimų surinkimo laiko intervalo.

Tokiomis aplinkybėmis, vienu metu galime kurti keletą duomenų analizės modelių, 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 analizės 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 poskyryje aprašysime vieną iš galimų pačios gamtos evoliucijos proceso inspiruotą daugiaagentę duomenų analizės ir dėsningumų radimo duomenyse sistemą. Kadangi aplinka keičiasi, tai nebegalime naudoti optimalumo kriterijų, kuriuos naudojome aukščiau aprašytose statinėse klasifikavimo ir prognozavimo sistemose. Gamtoje veikia tik išlikimo kriterijus. Mes jį ir naudosime. Šiame poskyryje apsiribosime klasifikavimo uždavinio analize.

Tarkime minėto tipo daugiaagentę sistemą (DAS) sudaro maksimaliai L vieno tipo adaptyvių intelektualių agentų. Jie turi mokytis spręsti klasifikavimo į dvi kategorijas (klases) uždavinius. Iš daugelio žinomų klasifikavimo uždaviniui spręsti skirtų metodų pasirinkome vienasluoksnį perceptroną. Pasirinkimo argumentai tokie:

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

Skaičius L apribotas „iš viršaus“. Tai simbolizuoja, kad pasąmonėje, kurdami šią daugiaagentę atpažinimo sistemą, mes galvojame, jog gamtos ištekliai yra riboti. Manoma, kad kas tam tikrą laiko tarpą (tam tikrą momentų skaičių) aplinka keičiasi. Modelyje klasifikavimo uždavinys pakeičiamas nauju (žr. 30 paveikslą).

30 pav. Atpažinimo uždavinių seka. Uždaviniui pasikeitus, kurį laiką perceptronas mokomas naudojant tiek naujus, tiek ir senus mokymo duomenis („Abu uždaviniai“, dešinėje)

Kad išliktų, agentas privalo per trumpą laiką, sakykim, per $iter_{\max}$ iteracijų, išmokti spręsti klasifikavimo uždavinį, darydamas mažiau nei $P_{\text{leistinas}}$ klasifikavimo klaidų. Kiekvienas agentas sprendžia praktiškai beveik tą patį *išmokimo ir išlikimo* uždavinį. Skirtingi agentai sprendžia tik šiek tiek skirtingus uždavinius. Be to, kiekvienas iš agentų mokymui naudoja savo individualius mokymosi algoritmo parametrus. Kad populiacija (daugiaagentė sistema) nežūtų, ilgą laiką vykstančio aplinkos keitimosi (atpažinimo uždavinių kaitaliojimo) metu žuvus agentui, jis pakeičiamas nauju. Naujas agentas pradedamas mokyti nuo „nulinių“ svorių, tačiau iš atsitiktinai parinkto „labai sėkmingo“ agento jis paveldi pastarojo „mokymosi stilių“ (mokymosi algo-

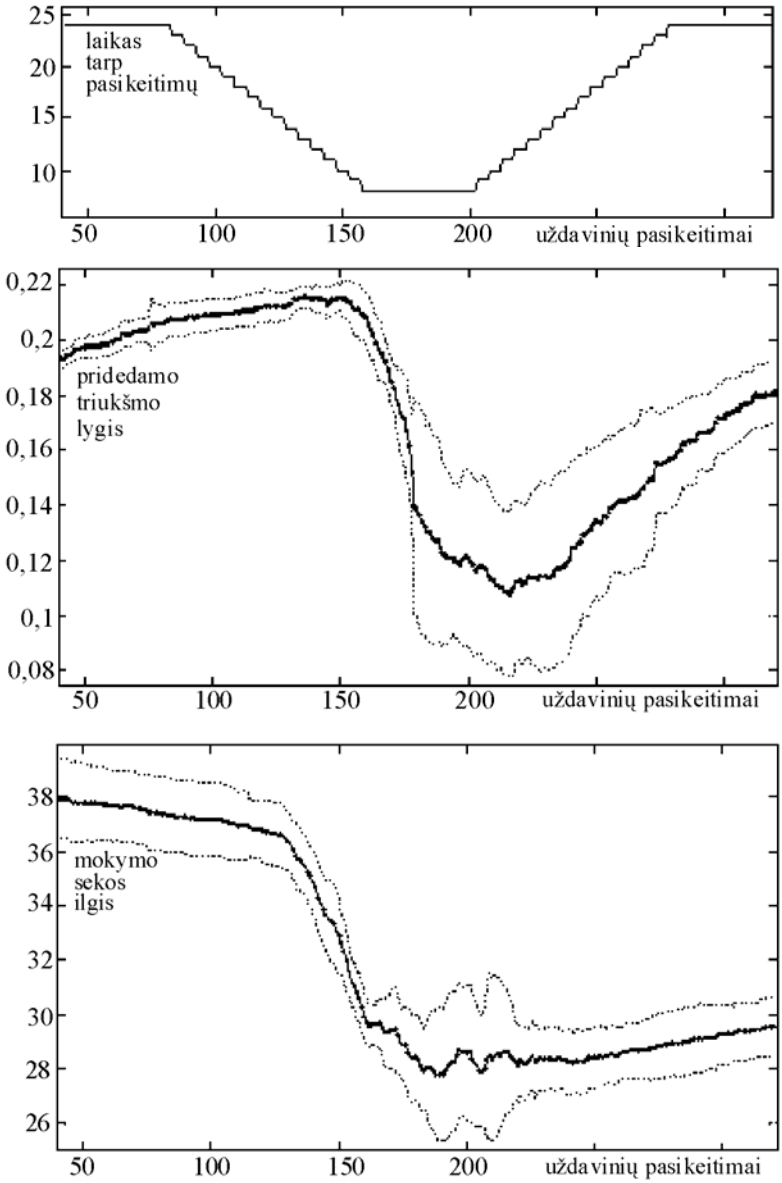
ritmo parametrus). Be to, naujo agento (vaiko) leistinas mokymosi laikas pailginamas iki $\beta_{vaik} \times iter_{max}$ iteracijų ($\beta_{vaik} > 1$).

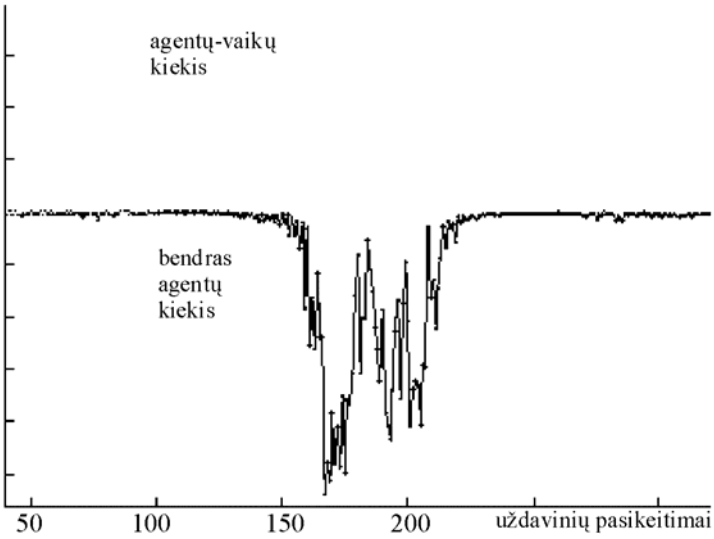
Mokymosi stilius ir yra tai, kuo adaptyvūs intelektualūs agentai tarpusavyje skiriasi:

a) mokymo duomenų sekos ilgiu (*s*-tasis agentas mokomas n_s ilgio, laiko atžvilgiu iš paties paskutinio laikotarpio, daugiamačiais stebėjimo vektoriais),
b) požymiais, naudojamais klasifikuojant (tiek jų dimensiškumu, tiek jų sudėtimi),
c) reguliarizavimo parametru (kad agentas greičiau mokytųsi ir jo svoriai neaugtų pernelyg greitai; tokiu atveju agentas galėtų sparčiau prisitaikyti prie pasikeitusių aplinkos sąlygų. Dalies α_s atsitiktinai parinktų mokymo vektorių trokštamų išėjimų reikšmės turėtų būti sukeistos),
d) skirtumu tarp trokštamų išėjimų $\Delta t = t_2 - t_1$, kuris taip pat lemia perceptrono svorių augimą ir jo persimokymo greitį.

Daliai agentų nespėjus laiku išmokti savo uždavinio, jie žūva. Ilgos aplinkos pasikeitimų serijos metu vieni agentai vis pakeičiami naujais. Kolektyviai besimokydama daugiaagentė sistema sprendžia savo išlikimo klausimą. Tam ji turi išmokti klasifikuoti atpažinimo uždavinius su mažesne nei $P_{leistinas}$ klasifikavimo klaidų tikimybe. Tad išliekančių agentų mokymosi stiliaus charakteristikos laikui bėgant prisitaiko prie pokyčių dinamikos. Tokios sistemos veikimo rezultatą pateikiame 31 paveiksle:

a) laikotarpių tarp atpažinimo uždavinių keitimosi grafikas;
b) neteisingų mokymo direktyvų dalies, parametro α_s, vidutinis kitimas (standartinių nuokrypių, pažymėta mėlynai) 330-ties atpažinimo uždavinių keitimosi laikotarpiu;
c) vidutinis kiekvieno agento mokymosi laiko kitimas;
d) bendras „gyvų“ agentų daugiaagentėje sistemoje (juoda) ir naujų, dar „vaikais“ esančių, agentų (žalia) kiekio kitimas.


laikas
tarp
pasikeitimų
uždavinių pasikeitimai
pridedamo
triukšmo
lygis
uždavinių pasikeitimai
mokymo
sekos
ilgis
uždavinių pasikeitimai



31 pav. Daugiaagentės atpažinimo sistemos dinamika 330-ties aplinkos pasikeitimų metu

Kad galėtume aiškiau atskleisti paveldimų mokymosi stiliaus charakteristikų priklausomybę nuo aplinkos pokyčių stiprumo, aprašomame eksperimente laikas t_{pasikeit} tarp dviejų vienas po kito einančių atpažinimo uždavinių pasikeitimų keitėsi pagal iš anksto nustatytą dėsningumą, kaip pavaizduota viršutiniame 31 paveikslo grafike. Matome, kad paveldimos mokymosi stiliaus charakteristikos kinta priklausomai nuo laiko t_{pasikeit}. Taip pat matome, kad esant didesniems aplinkos pasikeitimams daugiau agentų žūva, populiacija tampa mažesnė, daugiau turime agentų-vaikų.

Daugiaagenčių sistemų modeliavimas parodė, kad agentų populiacijų atsparumas didesniems aplinkos pokyčiams nemažai priklauso nuo agentų kiekio jų populiacijoje. Kuo didesnė populiacija, tuo daugiau galimybių ji turi išgyventi pačius stipriausius aplinkos (atpažinimo uždavinio) pokyčius. Todėl pagal gamtos evoliucijos inspiruotas idėjas „savaime susiformuoja“ optimizavimo kriterijus. Tai populiacijos dydis didžiausių aplinkos pokyčių laikotarpiu.

Aprašėme vieną iš paprasčiausių grupinio mokymosi stiliumi paremtos daugiaagentės sistemos modelių. DAS tampa atsparesnė didesniems aplinkos pokyčiams, jei jos agentai pasiskirsto į nevienodo dydžio grupes, kuriose jie vienas kitam padeda išlikti. Bendravimas tarp grupių labai ribotas: tik esant kritinei situacijai, pati stipriausia grupė padeda jau bežūstančiai, ypač negausiai, grupei tuo, kad vienas ar keli iš jos labai sėkmingų agentų perduoda savo mokymosi stiliaus parametrus nesėkmingoje grupėje daromiems agentams-vaikams. Todėl agentų grupės populiacijoje nebežūva.

Aprašytoji mokymosi ir duomenų analizės metodologija taikytina nestacionarių dinaminių sekų analizėje, kur statiniai duomenų analizės metodai tampa nebeveiksmingi.

SEPTINTAS SKYRIUS. DUOMENŲ ANALIZĖS TIKSLUMAS IR NAUDINGUMAS

Vykdant duomenų analizę daugiamačiuose duomenų masyvuose, be daugelio metodų gero teorinio išmanymo, reikia mokėti atpažinti galimus „paklydimus“, kurie gali niekais paversti visas įdėtas pastangas. Apie daugumą šio tipo problemų jau kalbėjome. Tai buvo:

1) klasifikavimo arba prognozavimo klaidos augimas, jeigu mokymui buvo naudojamas nepakankamas duomenų kiekis (2.1.2, 3.1.5 posk.),
2) klasifikavimo arba prognozavimo algoritmų prisiderinimas prie mokymo duomenų ir neteisingas klasifikavimo ar prognozavimo klaidos įvertinimas, jeigu tam buvo naudojami mokymo duomenys (2.1.2, 5.1.2 posk.),
3) poreikis naudoti nepriklausomus, mokymo procese nedalyvavusius, validavimo duomenis, siekiant teisingai įvertinti klasifikavimo, prognozavimo arba požymių išskyrimo algoritmų darbo tikslumą (5.1.3 posk.),
4) poreikis naudoti paprastesnius klasifikavimo, prognozavimo ar požymių išskyrimo algoritmus, kai turime nedaug duomenų (2.1.3, 2.2.3, 3.1.5, 3.1.6, 5.3.3 posk.),
5) perceptrono persimokymo reiškinys ir dėl to kylanti mokymo proceso stabdymo problema (3.2 posk.).

Šiame, baigiamajame, teorijos skyriuje pakalbėsime apie generalizavimo klaidos mažėjimą ir augimą didėjant požymių kiekiui, kai mokymui naudojamų duomenų kiekis yra ribotas. Taip pat aptarsime rezultatų patikimumo problemas, kylančias vieną ir tą pačią validavimo imtį naudojant daugelį kartų ir dėl to prie jos (validavimo imties) prisiderinant, o šiems duomenims faktiškai tampant mokymo duomenimis.

7.1. Optimalus požymių kiekis

Kalbėdami apie standartinės minimalių kvadratų regresijos panaudojimą prognozavimo uždaviniams spręsti, formule (2.7) pateikėme prognozavimo tikslumo lygtį, kurią čia dar kartą pakartosime:

$$\sigma_{\text{generalizavimo}} \approx \sigma_0 \sqrt{\frac{n}{n-p}}, \qquad (7.1)$$

kur

σ_0 – asimptotinis (idealus) tikslumas (vidutinė kvadratinė paklaida) tuo atveju, kai mokymo vektorių kiekis n yra „begalinis“, o požymių kiekis p yra fiksuotas;

$\sigma_{\text{generalizavimo}}$ – generalizavimo tikslumas (vidutinė kvadratinė paklaida), jei iš n mokymo vektorių sudarytą prognozavimo lygtį naudosime naujiems, mokymo procese dar nenaudotiems, duomenis prognozuoti.

Anksčiau pateikta formulė rodo, kad tais atvejais, kai požymių daug, o mokymo duomenų kiekis nėra didelis, generalizavimo klaida gali tiek smarkiai išaugti, kad ir prognozavimo lygtį sudaryti taptų nebetikslinga. Kad išvengtume šios problemos, požymių kiekį galime mažinti, naudoti reguliarizuotą regresiją arba perceptroną, mokytą specialiu būdu, bet trumpiau. 5.2 poskyryje kalbėjome apie požymių išrinkimo metodus, reikalingus požymių kiekiui sumažinti, kai mokymo duomenų nėra pakankamai daug.

Dėl to, kas buvo ką tik paminėta, kyla poreikis panagrinėti, kas darosi, kai požymių kiekis kinta, o mokymo duomenų kiekis lieka tas pats. Spręsdami prognozavimo uždavinį, šiai analizei galime taikyti (7.1) formulę, kurioje, priklausomai nuo požymių kiekio p, parametras σ_0 kinta. Beveik akivaizdu, kad kuo daugiau požymių imsime, tuo labiau didės idealus tikslumas, t. y. parametras σ_0 mažės. Jeigu požymius parinkti sektųsi neblogai, tai pirmiausia parinktume pačius geriausius, o tik vėliau pridėtume vis blogesnius požymius. Tai būtų idealus variantas. Panagrinėkime jį.

Darykime prielaidą, kad ryšys tarp idealaus prognozavimo tikslumo σ_0 (p) ir požymių kiekio p gali būti aprašomas tokia lygtimi:

$$\sigma_0 (p) = \sigma_0^* + \beta / (1 + \alpha \times (p-1)) , \qquad (7.2)$$

kur koeficientas σ_0^* nusako ribinį idealų tikslumą, kurį būtų galima teoriškai gauti, kai požymių ir mokymo duomenų kiekiai artėja prie begalybės. Parametrų β ir σ_0^* suma nusako pirmojo požymio indėlį į prognozavimo paklaidos sumažėjimą, t. y. vieno (pirmojo, geriausio) požymio kokybę (prognozavimo tikslumą, gaunamą, jei regresijos lygčiai sudaryti naudosime tik šį vieną požymį). Koeficientas α nusako, kaip ženkliai ideali prognozavimo paklaida mažėja pridedant po vieną naują požymį.

Pavyzdys. Tarkime $\sigma_0^* = 4$, $\beta = 2$, $\alpha = 0{,}5$.

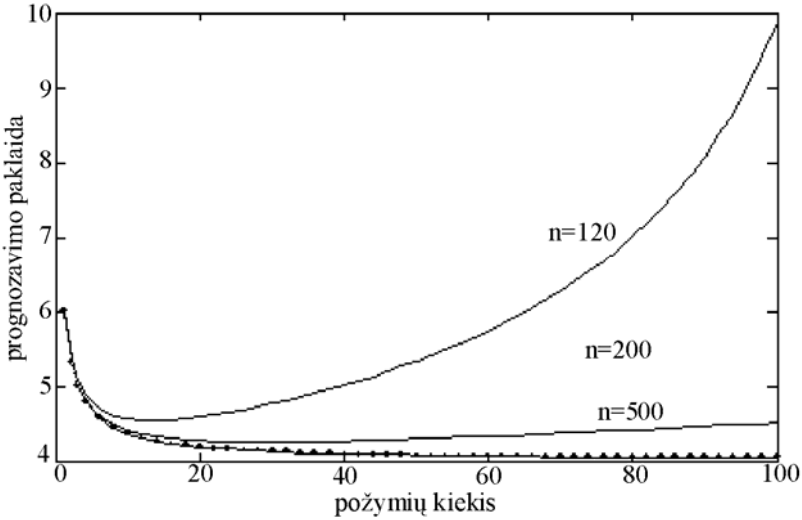
Tada $\sigma_0 (p) = 4 + 2 / (1 + 0{,}5 \times (p-1))$.

32 paveiksle pateikiame idealaus tikslumo σ_0 priklausomybės nuo požymių kiekio grafiką (juoda su taškais kreivė), kuris rodo, kad iš pat pradžių ideali prognozavimo paklaida mažėja sparčiai, tačiau vėliau šis mažėjimas sulėtėja ir tampa vos pastebimas.

Visiškai kitoks vaizdas gaunamas, kai mokymo duomenų kiekis yra ribotas. Skaičiavimai pagal formulę (7.1) rodo, kad situacijoje, kai mokymo vektorių *n* yra 500, prognozavimo paklaida pradžioje mažėja, vėliau nusistovi ir maždaug požymių kiekiui viršijus 30 pradeda augti (mėlyna kreivė). Kai *n* = 200, šis augimas prasideda nuo 20-ojo požymio (žalia kreivė), o kai *n* = 120, dar anksčiau – net nuo 15-ojo požymio (raudona kreivė). Vadinasi, turint 500 mokymui skirtų vektorių, reikėtų imti 20–40 požymių. Tačiau jei mokymo vektorių yra mažiau, pavyzdžiui, 200, reikėtų apsiriboti tik geriausiais 10–20 požymių. Čia mes sąmoningai nenurodome tikslaus optimalaus požymio kiekio, norėdami pabrėžti tokius požymio kiekio nustatymo aspektus:

1. Kreivių *paklaida* = *funkcija*^{A} (*požymių kiekis*) „dugnas“ labai lėkštas, tad ir tikslus požymių kiekis nėra labai svarbus. Svarbiausia, kad tyrinėtojas atkreipia dėmesį į minimą reiškinį ir nenaudoja per mažai ar per daug požymių.

2. Optimalus požymių kiekis labai priklauso nuo kreivių „*ideali paklaida = funkcija*^{B} (*požymių kiekis*) pobūdžio (tuoj tai parodysime). Faktiškai tos priklausomybės nežinome. Jos nėra fiksuotos, o priklauso nuo būdo, kaip požymiai buvo parenkami (žr. 5.2 poskyrį), taip pat nuo atskirų požymių ar jų rinkinių kokybės įvertinimo tikslumo. Paprastai požymių kokybė vertinama naudojant eksperimentinius duomenis (žr. 5.1.4 poskyrį). Tad ką tik minėta *funkcija*^{B} iš principo negali būti tiksliai įvertinta.

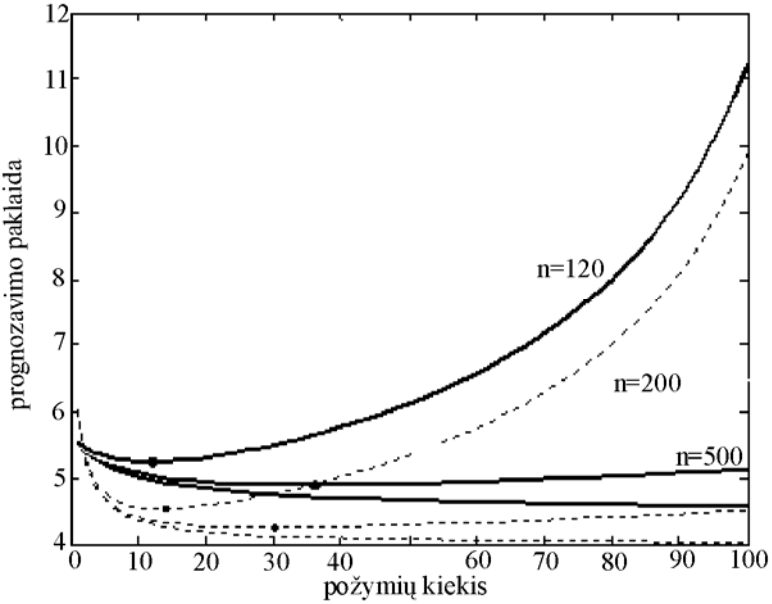


32 pav. Prognozavimo paklaidos priklausomybė nuo požymių ir mokymo duomenų kiekio

Kad parodytume *funkcijos*^{B} įvertinimo tikslumo įtaką optimalaus požymio kiekio įvertinimo rezultatui, tarkime, kad anksčiau minėtame pavyzdyje požymių surikiavimo tikslumas mažesnis, t. y. β =1,0, α = 0,1. Be to, „nežinodami“ vietoj $\sigma_0^* = 4$, mes naudosime $\sigma_0^* = 4{,}5$. Tokiu atveju mūsų spėjama priklausomybė bus:

$$\sigma_0 (p) = 4{,}5 + 1{,}0 \ / (1 + 0{,}1 \times (p\text{-}1)).$$

Pagal formulę (7.1) gauname naują kreivių sistemą, kur ką tik nubrėžtų kreivių spalva atitinka ankstesniąsias, tačiau jos pavaizduotos storesnėmis linijomis (žr. 33 paveikslą). Palyginimui pateikiame 32 paveiksle matytas punktyrinėmis linijomis nužymėtas kreives. Aiškumo dėlei, kiekvienos kreivės minimumas pažymėtas tašku.

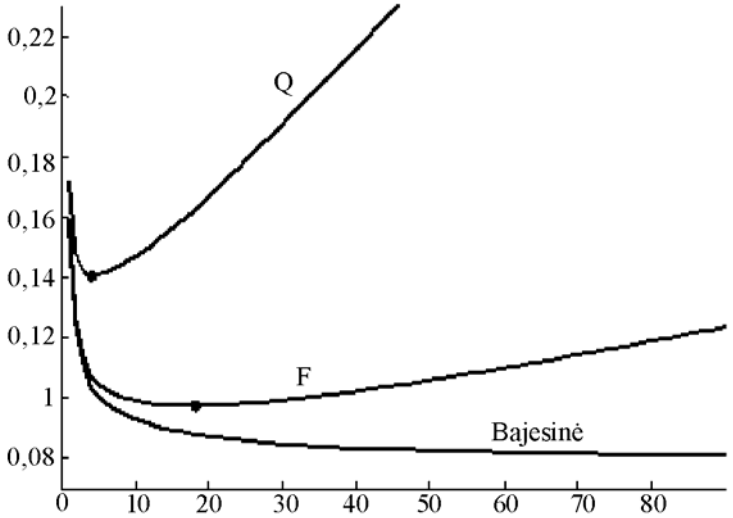


33 pav. Prognozavimo paklaidos priklausomybė nuo požymių ir mokymo duomenų kiekio: požymių rikiavimo įtaka optimalaus požymių kiekio nustatymo tikslumui

33 paveiksle storesnėmis linijomis pavaizduotos kreivės rodo, kad optimalaus požymio kiekio p_{opt} įvertinimai pasislinko. Nepaisant gan didelio parametrų σ_0^*, β, α pasikeitimo, p_{opt} reikšmės pakito labai nedaug. Tai patvirtina ankstesnę mintį: *svarbiausia, kad tyrinėtojas atkreipia dėmesį į minimą reiškinį, į tai, kad mokymo duomenų kiekis ribotas, ir nenaudoja per mažai ar per daug požymių*.

Panašių problemų kyla ir sprendžiant klasifikavimo uždavinius. Toliau pateikiamuose grafikuose (žr. 34 paveikslą) turime Bajeso kla-

sifikavimo klaidos tikimybės kitimo priklausomai nuo naudojamų požymių kreivę. Tai apatinė (juoda) kreivė.



34 pav. Klasifikavimo klaidos priklausomybė nuo požymių kiekio tiesiniame ir kvadratiniame klasifikatoriuose

Duomenys: du daugiamačiai normalieji (Gauso) pasiskirstymai faktiškai su vienodomis kovariacinėmis matricomis. Augant požymių kiekiui Bajeso klasifikavimo klaidos tikimybė vis mažėja. Naudojant tiesinį Fišerio klasifikatorių, mokytą su ribotos apimties duomenimis, po šimtą vektorių iš kiekvienos klasės, klasifikavimo klaidos grafikas (vidurinis, mėlyna kreivė) „išsilenkia“: iš pat pradžių klasifikavimo klaidos tikimybė mažėja, pasiekia minimumą ir pradeda vėl augti. Priežastis ta pati, kaip ir ką tik nagrinėtame prognozavimo uždavinyje: kai požymių kiekis viršija 18, kiti požymiai nebesuteikia naujos naudingos, klasėms atskirti reikalingos informacijos daugiau, nei gaunama „triukšmo“, t. y. paklaidų, susikaupusių vertinant klasifikavimo taisyklės parametrus pagal gana nedidelį mokymo duomenų kiekį. Kai naudojame kvadratinį klasifikatorių, koeficientų, vertinamų pagal mo-

kymo duomenis, yra beveik dvigubai daugiau. Dėl to šiame uždavinyje klasifikavimo klaidos tikimybė yra 1,5 karto didesnė, o optimalus požymių kiekis yra ženkliai mažesnis – $p_{opt} = 4$. Tai viršutinė (raudona) kreivė.

Prognozavimo ir klasifikavimo algoritmų sudėtingumo, naudotų duomenų kiekio ir gaunamo tikslumo ryšys yra vienas iš svarbiausių klausimų praktiniame žinių išgavimo iš empirinių duomenų darbe. Optimalaus požymio kiekio nustatymo problemą paprastai sprendžiame klaidų ir bandymų metodu. Išnagrinėtų klasifikatorių (Fišerio, kvadratinio) asimptotinės klasifikavimo klaidos tikimybės priklausomybę galima įvertinti empiriškai pagal požymių kiekį, o vėliau bandyti skaičiuoti tikėtiną klasifikavimo klaidą pagal teoriškai išvestas formules.

7.2. Problemos, kylančios daugelį kartų naudojant validavimo duomenis

Spręsdami duomenų analizės uždavinius, tikrojo duomenų modelio nežinome. Todėl reikia rinktis iš daugelio galimų alternatyvų. Tokiais atvejais tikslumo įvertinimo uždavinį tenka spręsti daugelį kartų. Kaip minėjome, kad patikimiau įvertintume klasifikavimo, prognozavimo arba požymių išskyrimo/išrinkimo uždavinių tikslumą, reikia naudoti papildomus, anksčiau mokymo procese nedalyvavusius, validavimo (tikrinimo) duomenis. Praktikoje lyginant daugelio potencialiai gerų modelių tikslumą paprastai naudojama tik viena validavimo (tikrinimo) imtis. Štai čia ir kyla modelio parinkimo tikslumo problema, kurios esmę ir paaiškinsime pavyzdžiu.

Nagrinėjamas pavyzdys yra susijęs su svarbiu praktikai uždaviniu. Tai informatyviausių požymių parinkimas sprendžiant susirgimo leukoze prognozavimo uždavinį. Turint 72 dviem klasėms (sveiki ir sergantys leukoze žmonės) priklausančius 7129-mačius genų išraiškos vektorius reikėjo sudaryti klasifikavimo taisyklę, naudojančią labai nedidelį genų išraiškos požymių kiekį. Iš 72 vektorių renkant atsitiktinai buvo sudaryti mokymo ir testavimo (validavimo) duomenys. Kaip minėta, pradinių požymių yra ypač daug. Jų net 7129. Kad dimen-

siškumas būtų mažinamas efektyviau, požymių sistemų formavimas buvo atliktas dviem etapais.

Pirmame etape buvo sudaryta šimtas tūkstančių atsitiktinių genų (požymių) rinkinių. Kiekvienas iš rinkinių buvo sudarytas iš aštuonių pradinių genų. Iš šių rinkinių buvo atrinkta tūkstantis geriausių. Išnagrinėjus genų, patekusių tarp 1000-čio geriausiųjų, pasirodymo dažnius, požymių kiekis buvo dar kartą sumažintas. Vėl buvo atrinkta 200 dažniausiai juose pasitaikančių genų (požymių). Iš tų 200 požymių atsitiktiniu būdu buvo sudaryti trys milijonai aštuonmačių požymių rinkinių. 7.1 lentelės antroje eilutėje pateikiame devynių tiesinių Fišerio klasifikatorių, sudarytų naudojant skirtingus požymių rinkinius, klasifikavimo klaidos įvertinimus, gautus naudojant tik validavimui skirtus 36 vektorius. Žemutinėje eilutėje yra klasifikavo klaidų kiekis, gautas požymių rinkinių kokybę vertinant pagal dar kitus, tęstiniais laikomus, duomenis. Požymių parinkimas buvo atliekamas naudojant validavimo duomenis. Išrinkto (geriausio, darančio mažiausiai klaidų) požymių rinkinio tikslumas vertinamas naudojant tęstinius duomenis. Klasifikavimo dažniai, atitinkantys devynis nagrinėtus požymių rinkinius, pateikti kaip 2-matis devynių taškų pasiskirstymas 35 paveiksle.

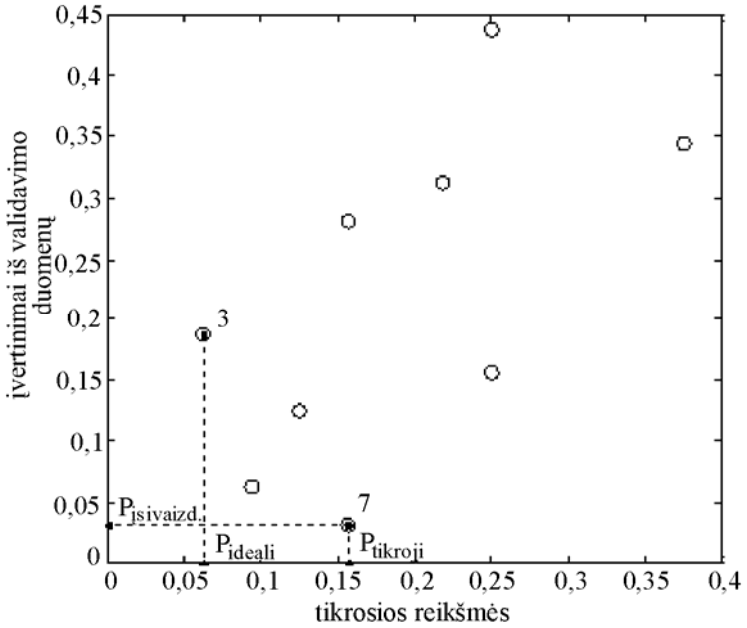
7.1 lentelė

Klasifikavimo klaidų kiekis, gautas naudojant validavimo ir testinius duomenis

Rinkinio numeris	1	2	3	4	5	6	7	8	9
Validavimo klaida	2	4	6	5	9	11	1	4	10
Testinė/tikroji klaida	3	8	2	8	5	12	5	4	7

Sprendžiant pagal validavimo imties įvertinimus, geriausias požymių rinkinys yra septintas, jo klasifikavimo klaidos įvertis yra 1/32, t. y. 3,13% klaidų. Šį klasifikavimo klaidos įvertį vadinsime *įsivaizduojama klasifikavimo klaida parenkant požymius* ir žymėsime $P_{\text{įsivaizd}}$. Faktinė septintojo požymių rinkinio klaidos tikimybė yra 5/32, t. y. 15,63% klaidų. Šį klasifikavimo klaidos įvertį vadinsime *tikraja klasifikavimo klaida parenkat požymius* ir žymėsime P_{tikroji}.

Iš tikrųjų geriausias požymių rinkinys yra trečiasis. Jo tikroji klasifikavimo klaida yra 2/32, t. y. 6,25% klaidų. Šį klasifikavimo klaidos įvertį vadinsime *idealia klasifikavimo klaida parenkant požymius* ir žymėsime P_{ideali} (žr. 35 paveikslą).



35 pav. Devynių požymių rinkinių klasifikavimo klaidos dažnių pasiskirstymas dvimatėje erdvėje

35 paveikslas rodo, kad įsivaizduojama klasifikavimo klaidos tikimybė gali būti gerokai mažesnė nei tikroji. Čia nieko keista. Parinkdami požymius, klasifikatoriaus ar prognozavimo algoritmo modelį, prisideriname prie validavimo duomenų – šie faktiškai tampa antraisiais mokymo duomenimis. Tik mokymo metodas čia kitoks – geriausio varianto parinkimas. Jei geriausią modelį rinksime ne iš devynių, o, sakykim, iš kelių milijonų variantų, tai ir prisiderinimo prie validavimo duomenų efektas bus dar labiau pastebimas.

Dabar parodysime, kaip prisiderinimo prie validavimo duomenų efektą įvertinti sprendžiant „genų raiškos duomenų (angl. *microarray*

gene expression data) analizės uždavinį. Kadangi duomenų labai mažai (tik 72 dviejų klasių vektoriai), pusė buvo naudoti mokymui, kita pusė – validavimui. Testavimui turimų duomenų nebeliko. Kadangi mokymo metu klasifikavimo taisyklė prie duomenų prisiderina, tai analizuojant duomenis paprastai reikalaujama, kad testiniai duomenys būtų nenaudojami mokymui. Šį reikalavimą „apeisime“ analitiškai „pataisydami“ empirinius, kiekvieną požymių rinkinį charakterizuojančius klasifikavimo klaidos įverčius.

Štai **analitiniais skaičiavimais paremta generalizavimo klaidos įvertinimo procedūra.** Šiuo tikslu buvo panaudotos laukiamos generalizavimo, EP_n ir mokymo (empirinės), EP_R, Fišerio klasifikatoriaus klasifikavimo klaidų išraiškos, anksčiau minėtos 3.1.5 ir 5.1.4 poskyriuose:

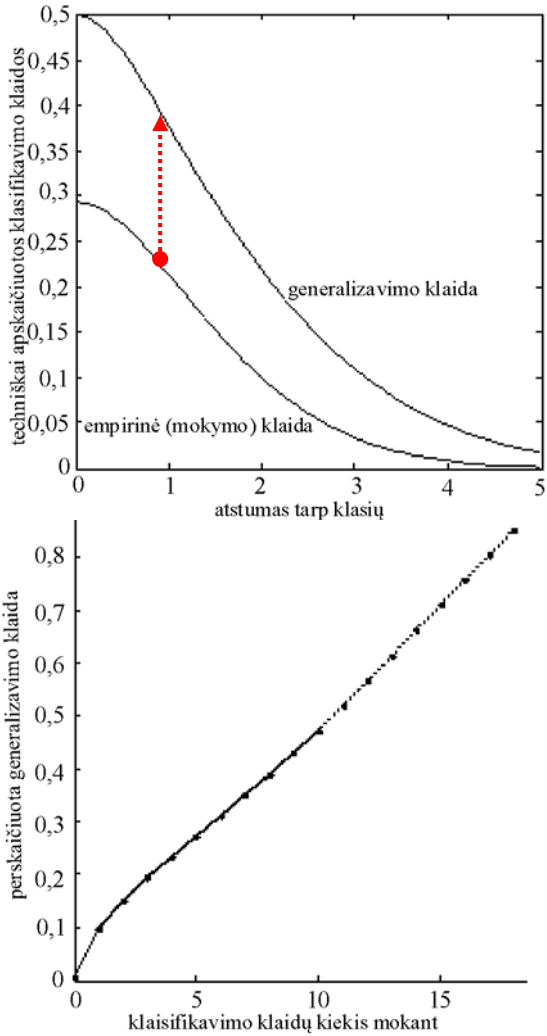
$$EP_n = \Phi(- ½\, \delta / T_{\mu\Sigma}), \tag{7.3}$$

$$EP_R = \Phi(- ½\, \delta \times T_{\mu\Sigma}), \tag{7.4}$$

kur $T_{\mu\Sigma} = \sqrt{(1 + \frac{1}{N}(1 + \frac{2p}{\delta^2}) + \frac{p}{N^2\delta^2})\frac{2N}{2N - p}}$,

δ – Mahalanob'io atstumas tarp klasių,
p – požymių kiekis (šiame konkrečiame uždavinyje $p = 8$),
N – mokymo vektorių kiekis vienoje klasėje (šiame konkrečiame uždavinyje $N = 18$).

Siekdami empirinius, iš mokymo duomenų gautus, klasifikavimo klaidos įverčius „pataisyti“ ir gauti nepasislinkusius generalizavimo klaidos įverčius, pirmiausia parinkome tokį atstumo tarp klasių δ kitimo intervalą [0,001: 0,01: 5]. Paskui kiekvienos šio intervalo reikšmės atžvilgiu pagal formules (7.3 ir 7.4) apskaičiavome EP_n ir EP_R reikšmes (žr. viršutinį 36 paveikslo brėžinį), kurias vėliau naudojome generalizavimo kreivei „atstatyti“.



36 pav. Viršuje – mokymo ir generalizavimo klasifikavimo klaidų priklausomybė nuo atstumo δ; apačioje – atstatyta (pataisyta) generalizavimo klaida kaip empirinės klasifikavimo klaidos funkcija

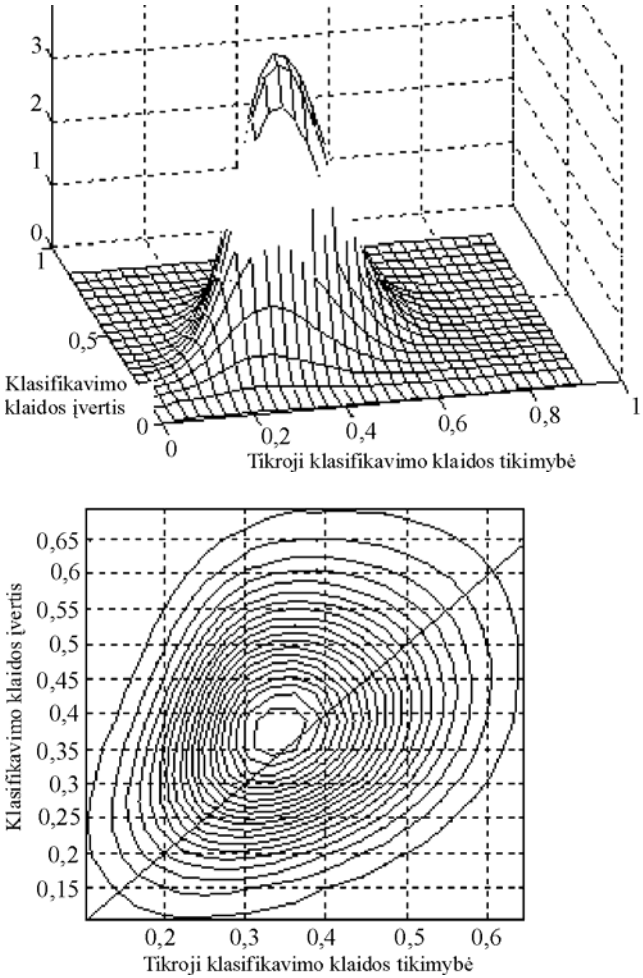
Žinant empirinių klasifikavimo klaidų reikšmes (apatinė violetinė kreivė), nesunku rasti ją atitinkančias generalizavimo klaidos reikšmes (viršutinė juoda kreivė), kaip parodyta raudona rodykle viršutiniame 36 paveikslo brėžinyje (punktyrinė rodyklė rodo „pataisymo“ dydį). Praktiškai iš šių dviejų grafikų sudarėme vieną, kur abscisių ašyje vietoj empirinės klasifikavimo klaidos buvo naudojamas iš validavimo imties įvertintas neteisingai klasifikuotų mokymo vektorių skaičius, o ordinačių ašyje – pagal mokymo duomenis teoriškai atstatytos generalizavimo klaidos reikšmės (žr. apatinį 36 paveikslo brėžinį).

Devynių taškų, pavaizduotų 35 paveiksle, ordinates anksčiau taip pat buvome apskaičiavę šiuo būdu. Pavaizduoti trijų milijonų taškų išsibarstymą dvimatėje erdvėje praktiškai neįmanoma, nes jie „užlipa“ vienas ant kito. Todėl 37 paveiksle turime 3 000 000 taškų (P_{test}, $P_{\text{įverčiai}}$) pasiskirstymo dvimatėje erdvėje histogramą. Kad geriau matytume statistinę priklausomybę tarp įverčių P_{test} ir $P_{\text{įverčiai}}$, 37 paveikslo apačioje pateikėme „kontūrinę diagramą“. Tai uždaros elipsės formos kreivės, kur dvimačio pasiskirstymo tankis vienodas. Iš diagramos matome, kad augant tikrajai klasifikavimo klaidai, taip pat auga ir klasifikavimo klaidos įverčio vidurkis bei jo dispersija. Koreliacija (statistinis ryšys) tarp įverčių P_{test} ir $P_{\text{įverčiai}}$ nėra didelė. Tai dėl to, kad tiek P_{test}, tiek $P_{\text{įverčiai}}$ buvo įvertinti naudojant mažus duomenų kiekius. Jei klasifikavimo klaidos būtų vertinamos naudojant didelius statistinių duomenų kiekius, koreliacija būtų artima vienetui. Kontūrinė diagrama rodo, kad mokymo duomenų transformavimas į validavimo (poslinkio, atsirandančio dėl mažo mokymo duomenų kiekio, eliminavimas) atliktas daugmaž teisingai: tiek vidurkiai, tiek dispersijos kryptimis P_{test} ir $P_{\text{įverčiai}}$ apytiksliai atitinka vienas kitą.

Turėdami gan didelį vektorių P_{test}, $P_{\text{įverčiai}}$ masyvą, galime ištirti, kaip

a) *įsivaizduojama klasifikavimo klaida parenkant požymius* $P_{\text{įsivaizd}}$,
b) *tikroji klasifikavimo klaida parenkant požymius* P_{tikroji} ir
c) *ideali klasifikavimo klaida parenkant požymius* P_{ideali}

priklauso nuo požymių rinkinių, iš kurių išrenkamas geriausias požymių rinkinys, kiekio m.



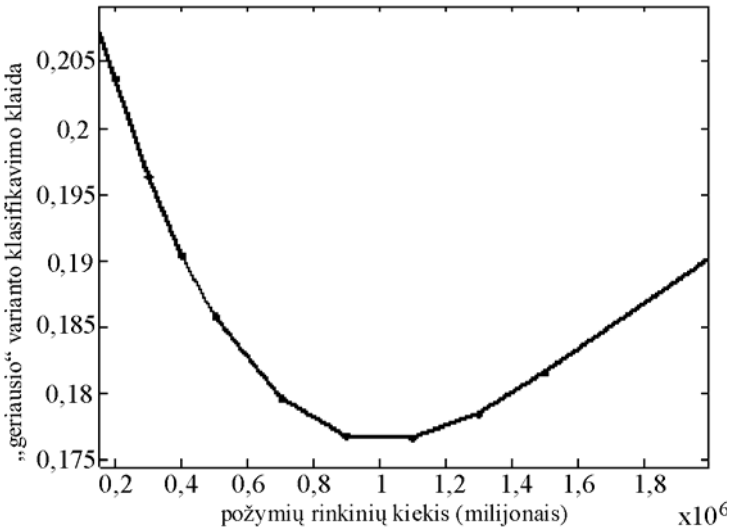
37 pav. Trijų milijonų P_{test}, $P_{\text{įverčiai}}$ reikšmių, atitinkančių tris milijonus aštuonmačių požymių rinkinių, pasiskirstymas dvimatėje erdvėje

35 paveiksle nagrinėjome devynis požymių rinkinius. Dabar nagrinėjame $M = 3\ 000\ 000$ požymių rinkinių. Iš $M = 3\ 000\ 000$ požymių rinkinių galime sudaryti

$$C_p^r = \frac{m!}{(M - m)!m!} \qquad (7.5)$$

kombinacijų po m požymių rinkinių, iš kurių išrenkamas geriausias rinkinys. Vadovaudamiesi šia kombinatorikos formule, nesunkiai galime sudaryti greitą skaičiavimo algoritmą vidutinėms P_{tikroji}, P_{ideali} ir $P_{\text{įsivaizd}}$ reikšmėms skaičiuoti (šis algoritmas pateikiamas autoriaus knygos *Statistical and Neural Classifiers. London: Springer-Verlag, 2001* pabaigoje).

38 paveiksle pateikiame grafiką, rodantį, kaip tikroji klasifikavimo klaida, parenkant požymius P_{tikroji}, priklauso nuo nagrinėjamų požymių rinkinių kiekio m. Matome, kad kol nagrinėjamų požymių rinkinių kiekis m nėra didelis, tikroji klasifikavimo klaida parenkant požymius mažėja, tačiau vėliau mažėjimas sulėtėja. Dar vėliau tikroji klaida pradeda augti.



38 pav. Tikrosios klasifikavimo klaidos, kylančios dėl požymių P_{tikroji} pasirinkimo, priklausomybė nuo nagrinėjamų požymių rinkinių kiekio m

Vėl gauname persimokymo efektą, stebėtą mokant perceptronus, klasifikatorius, nustatant optimalų požymių kiekį (žr. 7.1 poskyrį). Čia nieko keista. Parinkdami požymius prisiderinome prie validavimo duomenų, ir šie tapo mokymo duomenimis, kur, kaip minėta, stebimas „persimokymo efektas“. Čia šį efektą „aptikome“ dar kartą.

Panašus pernelyg stipraus prisiderinimo prie validavimo duomenų efektas gaunamas ir parenkant prognozavimo požymius, klasifikavimo ar prognozavimo algoritmo tipą, jų architektūrą (pvz., daugiasluoksnio perceptrono paslėptų neuronų kiekį), sprendžiant požymių išskyrimo uždavinį. Prisiderinimo prie validavimo duomenų efekto ignoruoti negalima, nes tai lemia neteisingą duomenų analizės rezultatų įvertinimą, kas gali būti susiję su dideliais laiko ir materialiniais nuostoliais vėliau.

AŠTUNTAS SKYRIUS. PRAKTINIŲ UŽDAVINIŲ SPRENDIMAS

Duomenų analizė – tai praktinė disciplina, kurios esmė – pažvelgti į duomenis nauju rakursu, siekiant išryškinti juose esančius dėsningumus. Todėl vien tik teoriškai išmanyti duomenų analizės metodus neužtenka. Būtina gerai įvaldyti aukščiau aprašytus ir kitus duomenų analizės metodus, įgyti nemažą praktinio darbo patirtį. Autoriaus praktika tokia: vos ne kas trečiame uždavinyje susidurdavome su nauja, iki tol neišspręsta duomenų analizės problema. Taigi duomenų analizė – ne tik praktinis, bet ir kūrybinis darbas, kur reikia įsigilinti į sprendžiamą uždavinį, bandyti atspėti galimus dėsningumus ir paskui juos rasti. Labai svarbu vėliau įrodyti, kad rasti dėsningumai tikrai tuose duomenyse yra. Šiame poskyryje pateiksime keletą praktinių uždavinių sprendimo pavyzdžių. Pradėsime nuo vienos iš vartotojui patogių programinių priemonių – *Matlab'o* programinės sistemos. Vėliau atliksime prognozavimo ir klasifikavimo uždavinius, tikrinsime duomenų analizės „mokslines problemas".

8.1. *Matlab'o* pradžiamokslis

Ankstesniuose poskyriuose buvo minėta, kad *k*-artimiausių kaimynų taisyklę ir Parzeno lango klasifikatorių arba prognozavimo algoritmą galime taikyti ir situacijose, kai mokymo vektorių kiekis labai mažas. *Parzeno lango algoritmai ir k-artimiausių kaimynų taisyklė – tai mokymas pagal pavyzdžius.* Jokie parametriniai metodai, tokie kaip Fišerio ar kvadratinis klasifikatoriai, standartinė minimaliųjų kvadratų regresija, neveiks, jei mokymo duomenų kiekis pernelyg mažas.

Remiantis nuostata, kad praktika, sekimas pavyzdžiu yra geriausias mokytojas pradinėje stadijoje, bandydami išaiškinti *Matlab'o* naudojimo principus, iškart pateiksime keletą užduočių, kurios padės studijuojančiajam, žiūrint į labai trumpas programėles ir naudojantis *Matlab'o* komandomis „*help*", pajusti *Matlab'o* kalbos struktūrą. Kol kas studijuojančiajam prireiks tik *Matlab Simulink* – tai pagrindinė *Matlab'o* programa, kuri „daro viską". Vėliau reikės ir *Neural Net-*

works Toolbox, kai spręsdami duomenų analizės uždavinius pradėsime naudoti sudėtingesnius dirbtinius neuroninius tinklus.

Pirmas žingsnis

a) savo kompiuteryje turime instaliuoti *Matlab* programą;

b) pasidaryti savo būsimų programų aplanką (angl. *folder*), nors galima dirbti ir standartiniame *Matlab'o* aplanke *Work*.

Savo aplanko pavadinimą ir vietą, kur jis įdėtas, reikia nurodyti:

File → Set path,

kitaip jūsų programa nebus randama. Į minėtą aplanką įsirašykite ir pateikite dvi programėles: *gausas.m* ir *fidensity.m*. Vėliau čia laikysite ir jūsų pačių pasidarytas programėles.

```
% x=fi(t);   tai standartinio normalojo N(0,1) tankio reikšmės radimas
function x=fidensity(t)
x=exp(-t.*t/2)/sqrt(2*pi);
return

function X = gausas(n,C,M)     %    n      daugiamačių,
%    nomaliai pasiskirsčiusių  vektorių generavimas.
[V D] = eig(C) ; % C yra kovariacine matrica, M yra vidurkių vektorius.
V = real(V);D = real(D);p=size(C,1);
X = randn(n,p)*sqrt(D)*V' + ones(n,1) * reshape(M,1,p);
return
```

Antras žingsnis

Štai paprasčiausia programėlė, kurią galite leisti tiesiai iš *Matlab Command window*:

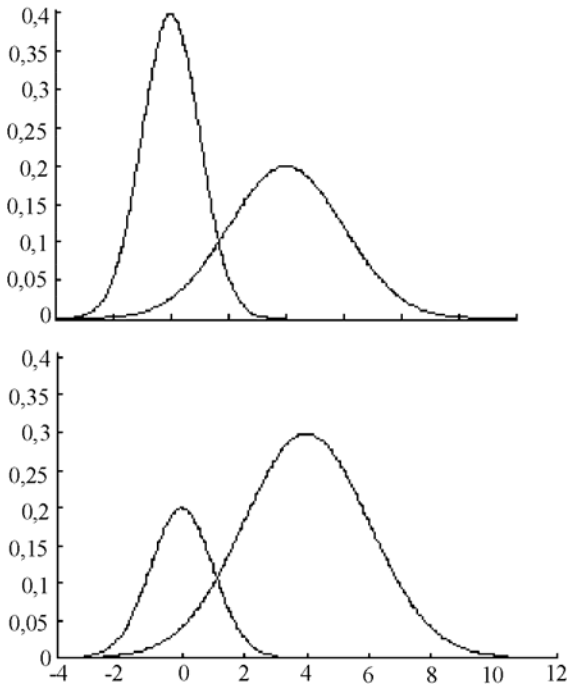
```
>> t=[-4:0.01:4]; x=fidensity(t);
figure(1);clf; axis([-4 12 0 0.405]);
hold on; plot(t,x,'r-',2*t+4,0.5*x,'b-')
t=[-4:0.01:4];x=fidensity(t);
figure(2);clf;axis([-4 12 0 0.405]);hold on;
plot(t,0.5*x,'r-',2*t+4,0.75*x,'b-')
```

Šioje užduotyje:

```
t=[-4:0.01:4];          % suformuojam vienmatį masyvą t.
x=fidensity(t);
```

% *suskaičiavom "standartinį: (vidurkis* 0*, dispersija* 1*) normalųjį tankį intervale* t.
figure(1);clf;
% *nurodėm piešinio numerį, išvalėm ankstesnius piešinius.*
axis([-4 12 0 0.405]) % *nurodėm x ir y ašių rėžius, kur bus piešiama*
hold on; % *pasakėm, kad jau nubrėžtą grafiką reikės atsiminti. Šis nurodymas % reikalingas, jei ant viršaus, naudodami kitą komandą, piešime dar ką nors.*
plot(t,x,'r-',2*t+4,0.5*x,'b-')
% *nupiešia vieną tankį raudonai* (‘r-‘) *, o kitą – mėlynai* (’b-’).

Jei kas neaišku, žiūrėkit >> help plot. Antra programėlės eilutė analogiška. 39 paveiksle matysite dvi normaliųjų tankių poras, į kurias žiūrėdami prisiminsite, kas yra vidurkis ir dispersija.



39 pav. Normalieji tankiai, besiskiriantys vidurkiais ir dispersijomis bei apriorinėmis klasių tikimybėmis

Rekomenduotume patiems pakaitalioti vidurkį, dispersiją ir išsiaiškinti, ką kiekviena eilutė atlieka, kaip, pakeitus vidurkį ar/ir dispersiją, pasiskirstymo tankio funkcijos atrodo vizualiai. Šiame vadovėlyje dėstomo kurso apimtyje tankiai ir jų supratimas bus reikalingi, kai kalbėsime apie klasifikavimo uždavinį, aiškinsimės optimalias Bajeso klasifikavimo taisykles.

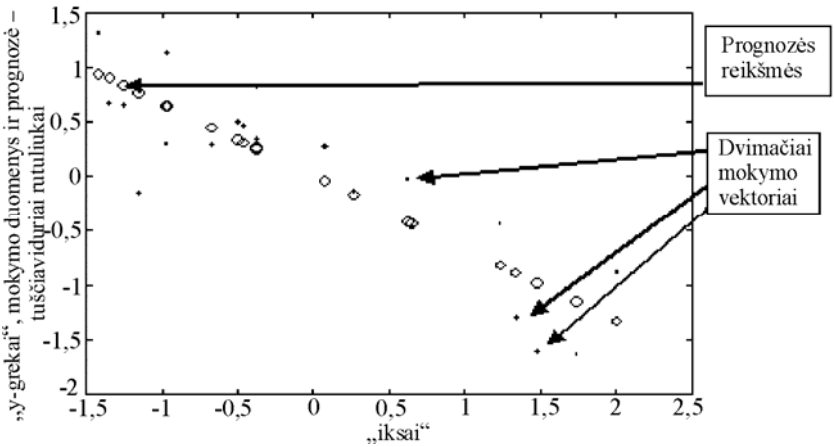
Dar viena užduotėlė:

```
N=20;D=gausas(N,[1,-0.8;-0.8,1], [4,2]);
Dnorm=D-ones(N,1)*mean(D);
% iš sugeneruotų duomenų atimam jų vidurkių vektorių
S=cov(Dnorm); % suskaičiuojam kovariacinę matricą
Sxx=S(1,1);Syx=S(2,1);
% išskiriam kovariacinės matricos elementus (reikės vėliau)
w=inv(Sxx)*Syx;
% apverčiam kovariacinę matricą ir padauginam ją  iš kovariacijų
Syx
yprognoze=Dnorm(:,1)*w; % suskaičiavom prognozę
figu-
re(3);plot(Dnorm(:,1),Dnorm(:,2),'r.',Dnorm(:,1),yprognoze,'bo')
N=20;D=gausas(N,[1,-0.8;-0.8,1], [4,2]);
% N – vektorių kiekis, D – sugeneruoti duomenys, kur pirmas stulpelis –
„iksas“,
% o antras stulpelis – „y-grekas“, – tai kintamasis, kurį ir reikia
prognozuoti
% [1,0.8;0.8,1] – kovariacinė matrica, kur dispersijos 1, o koreliacijos – 0,8,  % [4,2]   yra duomenų vidurkis
Dnorm=D-ones(N,1)*mean(D);
% iš sugeneruotų duomenų atimam jų vidurkį,
S=cov(Dnorm); % suskaičiuojam kovariacinę matricą
Sxx=S(1,1);Syx=S(2,1);    w=inv(Sxx)*Syx;
% išskiriam kovariacinės matricos elementus (reikės ateityje)
% suskaičiuojam prognozavimo lygties y=w*x svorių vektorių w
%  kadangi duomenų vidurkį  padarėm nulį, tai ir w0=0
yprognoze=Dnorm(:,1)*w;
```

% *suskaičiavom visų duomenų prognozę,* Dnorm(:,1)
figu-
re(3);plot(Dnorm(:,1),Dnorm(:,2),'r.',Dnorm(:,1),yprognoze,'bo');
% *nupiešėm ir duomenis, ir prognozę (mėlyni rutuliukai)*
xlabel('iksai');ylabel ('y-grikai, mokymo duomenys ir prognoze - melyni rutuliukai')
% *trečiame piešinyje* (figure(3)) *pažymėjom, kas yra ašyse.*

Tolesnis mokymosi žingsnis: bandom sudaryti tiesinę regresijos lygtį atveju, kai $\boldsymbol{x}$ yra p-matis vektorius. Štai minėtą užduotį realizuojanti programa.

```
N=30;p=6;D=gausas(N, 0.4*eye(p+1)+ones(p+1,p+1),
3+ones(1,p+1));
% N – vektorių kiekis, D – sugeneruoti duomenys, kur pirmi p stulpelių „iksas“ %, o (p+1)- mas – „y-grekas“, prognozuojamasis kintamasis
% 0.4 eye(p+1)+ones(p+1,p+1), – mūsų pačių nusistatyta kovariacinė
% matrica, kurioje dispersijos 1.4, o kovariacijos 1.0;
% 3+ ones(1,6) – tai lygu [3 3 3 3 3 3] – mūsų generuojamų
% duomenų vidurkis.
Dnorm=D-ones(N,1)*mean(D);
% iš sugeneruotų duomenų atėmėm jų imties vidurkį.
S=cov(Dnorm); % suskaičiavom kovariacinę matricą.
Sxx=S(1:p,1:p);Syx=S(1:p,p+1);
% išskyrėm kovariacinės matricos elementus
w=inv(Sxx)*Syx; % suskaičiavom prognozavimo lygties y=w*x svorių
% vektorių w. Kadangi duomenų vidurkį padarėm nulį, tai slenkstis
w0=0
```

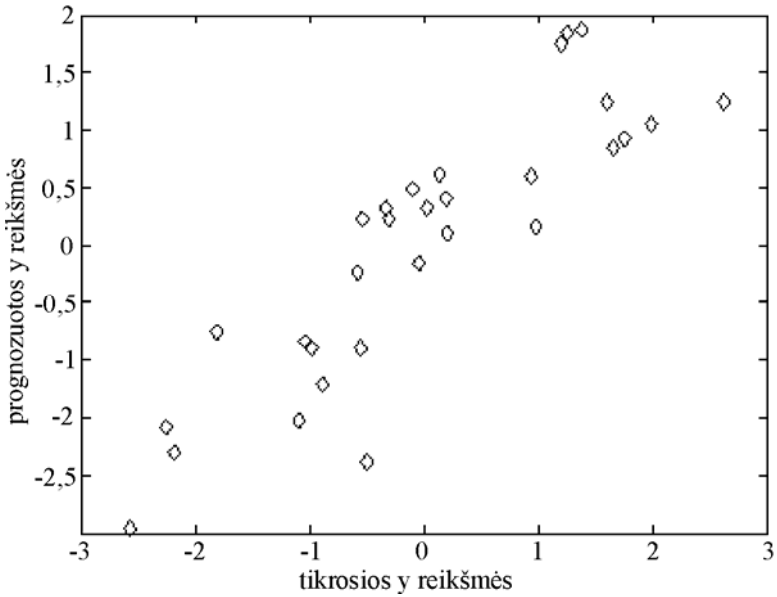


40 pav. Prognozuojančio (iksai) ir prognozuojamo (y-grekai – taškai (raudoni)) rodiklių pasiskirstymas. Prognozuotosios reikšmės – rutuliukai (mėlyni)

```
yprognoze=Dnorm(:,1:p)*w; % suskaičiavom prognozę
figure(4);plot(Dnorm(:,p+1),yprognoze,'rd')
% daugiamačiu atveju pavaizdavome:  y ašyje – prognozę
% x ašyje – tikrąsias prognozuojamas reikšmes, t. y. (p+1)-jį stulpelį,
xlabel('tikrosios y reiksmes');ylabel('prognozuotos y reiksmes')
disp(corrcoef([Dnorm(:,p+1),yprognoze]));disp('cia suskaiciavo
koreliacijos koeficienta tarp y ir y-prognozes')
```

Štai suskaičiuotasis koreliacijos koeficientas

```
1.0000    0.8734
0.8734    1.0000
```



41 pav. Tikrųjų ir prognozės reikšmių pasiskirstymas (išsibarstymas)

Tai, kas išdėstyta 8.1 poskyryje, *būtinai* kiekvienam reikia padaryti, išvargti pačiam.

Be „vargo“, savarankiško „apmąstymo“
gyvenime nieko nepasieksime.
Tai gyvenimo credo.

Atlikus pateiktas užduotis ir jas įsisavinus, bus daug lengviau suprasti teoriją. Bus nebesunku „skaityti“ pateikiamas programas, iškils mažiau neaiškumų. Žmogaus mokymasis panašus į dirbtinių neuroninių tinklų mokymą – padidinus mokymo žingsnį peršokama gerokai į priekį, bet vėliau pakliūvama į blogą lokalinį minimumą, iš kurio sunku „išlipti“.

Atlikus paprastesnes užduotis, galima pereiti prie sudėtingesnių uždavinių: regresijos lygties, klasifikavimo taisyklės sudarymo, vie-

nasluoksnio perceptrono ir daugiasluoksnių neuroninių tinklų. Tolesniuose poskyriuose aprašant minėtų uždavinių sprendimą bus daroma prielaida, kad studijuojantysis tikrai atliko ką tik minėtus praktinius darbus ir *Matlab'o* užduotis skaityti jau moka.

8.2. Darbas Nr. 1. Tiesinė prognozavimo lygtis

Darbo tikslas – išsiaiškinti prognozavimo uždavinio esmę, išmokti naudotis standartinėmis minimaliųjų kvadratų regresijos sudarymo programomis, išbandyti jas sprendžiant dirbtinai sugeneruotus ir realaus pasaulio uždavinio duomenis, išmokti įvertinti prognozavimo tikslumą. Trumpai tariant, – įsisavinti programinę įrangą ir ja naudojantis analizuoti duomenis, pritaikyti ankstesniuose skyriuose pateiktą teorinę medžiagą. Naudojant dirbtinius ir realaus pasaulio *(*angl. *real world*) duomenis bus sudaroma prognozavimo lygtis, taikant statistinius metodus ir vienasluoksnį perceptroną. Ypatingas dėmesys skiriamas prognozavimo tikslumui įvertinti, duomenims skirstyti į mokymo, testavimo ir tikrinimo (validavimo) duomenis, kur nustatomas optimalus neuroninio tinklo mokymo iteracijų kiekis. Studijuojantysis turėtų atlikti nemažą modeliavimo eksperimentų skaičių, kur bus tiriama požymių kiekio, mokymo sekos ilgio įtaka prognozavimo tikslumui.

Dalis medžiagos jau buvo aptarta 8.1 poskyryje, kur buvo pateiktas būdas ir programa, kaip sudaryti tiesinę regresijos lygtį pagal turimą „iksų" ir „y-grekų" duomenų masyvą. Dabar visas regresijos lygčiai gauti reikalingas eilutes, 8.1 poskyryje leistas tiesiog iš *Matlab'o Command window*, įforminsime kaip funkciją:

```
function [w,w0]=kv_regresija(D)
M=mean(D); % randam vidurkį
[N, r]=size(D); % randam duomenų masyvo dydžius.
p=r-1; % p – tai prognozuojančio vektoriaus, x, elementų (požymių) kiekis
Dnorm=D-ones(N,1)*M; % iš sugeneruotų duomenų atimam jų vidurkį
S=cov(Dnorm); % suskaičiuojam kovariacinę matricą
```

```
Sxx=S(1:p,1:p);Syx=S(1:p,p+1); % išskiriam kov. matricos elementus
w=(inv(Sxx)*Syx)'; w0=M(end)-M(1:end-1)*w';
% suskaičiuojam prognozavimo lygties svorius
return
```

Sudarytą programą dedame į aplanką (ang. *folder*) ir iš *Command window* leidžiame:

```
>>[W,w0]=kv_regresija(D);figure(5);
plot(D(:,end),D(:,1:end-1)*W'+w0,'kd')
```

Gauname jau anksčiau pateiktą generuotų ir prognozuotų parametro *y* reikšmių pasiskirstymą. Žemiau pateiksime programėlę, skirtą tiesinei prognozavimo lygčiai sudaryti su VsP pagalba.

Robastinės regresijos programos tekstas:

```
% su VsP pagalba rasti dideliems nukrypimams atsparią
% tiesinę prognozavimo lygtį
% autorius Šarūnas Raudys <raudys@ktl.mii.lt>
% A – mokymo duomenų (iksų) Nxp masyvas (N yra eilučių skaičius,
% p – stulpelių)
% Y – mokymo duomenų (y-grikų) Nx1 masyvas
% (N yra eilučių skaičius, 1 – stulpelių)
% At – testinių (validavimo) duomenų Ntxp masyvas
% Yt – testinių duomenų (y-grekų) Ntx1 masyvas
% iter – „batch“ iteracijų skaičius
% step – mokymo žingsnis
% Wstart – 1×(p+1) startinis svorių vektorius, w1, w2, …. , wp, w0 .
% Tai vektorius-eilutė
% Cmax – „apversto varpo“ plotis pačioje mokymo pradžioje
% Cmin – „apversto varpo“ plotis mokymo pabaigoje
% gama – parametras, nustatantis „apversto varpo“ siaurėjimo greitį
% augant iteracijų skaičiui
% W - 1*(p+1) – dimensijų galutinis svorių vektorius
% W - 1×(p+1) galutinis svorių vektorius (statistinėje regresijoje jį
% žymėjome [W,w0]), kur prognozuojant pirmos p komponenčių turi būti
% dauginamos iš p įėjimo vektoriaus x komponenčių,
% o paskutinis (p+1)-mas elementas – tai slenkstis w0.
```

```
% et – paklaida, išmatuota naudojant test duomenis, At, Yt.
% prieš mokant rekomenduojame iš A, At ir Y,Yt
%  atimti  masyvų A, Y vidurkius
% – pradėti mokyti nuo Wstart = zeros(1,p+1);
% BETA – robastiškumą valdančio parametro beta kitimas mokymo metu
% ROm, Rot – koreliacijos koeficientų kitimas
%  mokymo ir testavimo (validavimo) procese
% KIEK – faktiškai mokymo procese dalyvavusių vektorių kitimas mokymo
metu
function  [W,Wbest,er,et,BETA,ROm,ROt,KIEK]=
RobSLPCminmax(A,Y,At,Yt,iter,step,Wstart,Cmin,Cmax,gama)
[N , p ] = size(A ); Nt = size(At,1);
% randam mokymo ir testinių vektorių bei požymių kiekį
W=Wstart;Wbest=Wstart;  emin=99;
D=(Cmin*iter^gama-Cmax)/(Cmax-Cmin);C=Cmax*(D+1);
% emin – tai minimali prognozavimo paklaida.
%  pirmosios mokymo epochos atveju ji buvo nustatyta labai didelė
(emin=99).
AA=[A, ones(N,1)];AAt=[At, ones(Nt,1)];
% pridedam po stulpelį, sudarytą iš vienetukų
for i=1:iter   % ciklas iki iteracijų skaičiaus
dist=Y- AA * W';  % skirtumas tarp prognozės, AA * W' ir tikrojo „y-greko“
beta=C/(i^gama+D);if  i<10 beta=Cmax;end;
% koeficientas, nurodantis „robastiškumą“.
%  pirmąsias 9 epochas  „beta“ yra pastovus, o paskui vis mažėja
PIbeta=pi*beta; BETA(i)=PIbeta;
% varpo plotis – maksimali paklaida, kai patys tolimiausi stebėjimai
%  dar nebuvo ignoruojami
ind=find(abs(dist)<PIbeta); KIEK(i)=size(ind,1);
% randame eilės numerius tų mokymo vektorių, kurių paklaida dist
% telpa į „varpo“ plotį
W=W+step*sin(dist(ind))'*AA(ind',:); % svorių pataisymas
Pt=AAt*W'; dt=Pt-Yt;z=corrcoef([Pt,Yt]);ROt(i)=z(1,2);
% suskaičiuojamos ir įvertinamos
% prognozavimo reikšmės ir paklaidos validavimo (testiniuose) duomenyse
et(i)=sum(abs(dt))/Nt;if et(i)<emin emin=et(i);Wbest=W;end
% atsimenam geriausią rezultatą
Pm=AA*W';dr=Pm-
Y;z=corrcoef([Pm,Y]);ROm(i)=z(1,2);er(i)=sum(abs(dr))/N;
% suskaičiuojamos ir įvertinamos prognozavimo reikšmės ir paklaidos
```

```
%  mokymo duomenyse
end % ciklo pagal iteracijų skaičių pabaiga
return
```

Komentarai. Kad programa būtų trumpesnė ir galėtume dar pagreitinti skaičiavimą (*Matlab* skirtas dirbti su matricomis), prie mokymo ir testinių duomenų vektorių pridedame po stulpelį, sudarytą iš vienetų, ones(N,1).

Prieš mokant perceptroną rekomenduotina:

a) iš mokymo ir testinių duomenų A, Y ir At, Yt (tiek „iksų“, tiek „y-grekų“) atimti mokymo duomenų vidurkius;
b) pradinį svorių vektorių sudaryti iš „nuliukų“:

```
Wstart = zeros(1,p+1).
```

Londono biržos uždarymo indekso prognozavimas

Tikslas – panaudoti standartinę minimaliųjų kvadratų regresiją Londono biržos uždarymo kitos dienos pokyčiui prognozuoti. Šiam darbui turime 1383 dienų 11 požymių duomenų masyvą, kur pirmasis požymis ir yra tas indeksas, kurio pokytį vienos dienos laikotarpiu ir reikėtų prognozuoti.

Štai pirmas programos, kuri bus tam naudojama, etapas.

```
1)  load Mbirzele   % pakrauname duomenis
>> whos  %. Čia „whos“ yra  patikrinimas, „ar įvykdė“.
```

Paklausėme, kokius turime duomenis. *Matlab* parodė:

```
Name          Size            Bytes  Class     Attributes
Mbirzele      1383x11        121704            double
```

T. y. duomenys Mbirzele, 1383 vektoriai-eilutės, po 11 elementų kiekvienas. Tolesni programos darbo etapai nurodyti programos tekste kaip komentarai.

```
% ML_Birza_PROGNOZE  autorius:
%  Šarūnas Raudys raudys@ktl.mii.lt,
% Vienasluoksnio perceptrono taikymas prognozavimo uždaviniui
```

```
% LEISTI: Cmin=0.012;Cmax=2;gama=0.8;iter=1000;
%  ML_Birza_PROGNOZE(351,450,451,1370, [1 11],iter,0.001,
                                                    Cmin,Cmax,gama);
% nm1, nm2 – mokymo vektorių pradžia ir pabaiga
% nt1, nt2 – testinių  vektorių pradžia ir pabaiga
% fea – požymiai, kurie naudojami prognozavimui (iksai)
% iter – kiek iteracijų bus mokomas vienasluoksnis perceptronas (SLP, angl.)
% eta – VsP mokymo žingsnis
% Cmax – „apversto varpo“ plotis pačioje mokymo pradžioje
% Cmin – leistinas „apversto varpo“ plotis mokymo pabaigoje
% gama – parametras, nustatantis „apversto varpo“ siaurėjimo greitį
%  augant iteracijų skaičiui

function
L_Birza_PROGNOZE(nm1,nm2,nt1,nt2,fea,iter,eta,Cmin,Cmax,gama)
load Mbirzele   % įkraunu duomenis
B=Mbirzele(1:end,fea); clear birza
%  kad būtų trumpiau, duomenis pavadinu B, t. y. tik viena raide.
y=B(4:end-1,1)-B(3:end-2,1);
% prognozuojamas požymis, skirtumas B(t,1)-B(t-1,1),
%  čia bandom prognozuoti VIENĄ dieną į priekį
% b=[B(1:end-4,:),B(2:end-3,:),B(3:end-2,:)];
% taip būtų, jei prognozuodami naudotume 3 dienų istoriją;
% galite pabandyti prognozuoti dvi ar tris dienas į priekį. Gal jums pasiseks?
b=[B(2:end-3,:),B(3:end-2,:)];
%  čia suformuojam „iksus“ (prognozuojančių požymių vektorių)
%  iš 2 dienų istorijos (vakar ir šiandien). Tad jei naudoju požymius x1 ir
%  x11, tai prognozuojančių požymių kiekis p=4.
[nm,p]=size(b);b=[b,y];
% suformuoti išeities duomenys pirmiausia eina „iksai“ (masyvas b),
%  o pats dešinysis – „y-grekas“ – tai prognozuojamas parametras
c=(b-ones(nm,1)*mean(b(nm1:nm2,:)))./(ones(nm,1)*std(b(nm1:nm2,:)));
clear b
% išmetam b, kad kompiuterio atmintyje  neužimtų vietos;
% atėmėm mokymo duomenų vidurkį ir padalijom iš
% mokymo duomenų standartinio nuokrypio,
%  tai darom ir „iksams“, ir „y-grekams“.
%  Be to,  dar ir dėl skaičiavimų tikslumo
% surandam „iksų“ kovariacinę matricą, S1, randam
%  jos nuosavas reikšmes D ir vektorius T:
```

```
S1=cov(c(nm1:nm2,1:p));[T,D,V]=svd(S1);
Tr=T*inv(sqrtm(D+0.01*eye(p)));
% ortogonali matrica T duomenis pasuka taip,
%  kad naujos komponentės būtų nekoreliuotos,
%  matrica D^-1/2 normalizuoja naujas kryptis taip,
% kad visų naujų krypčių dispersija būtų VIENETAS;
% Dėl visa ko prie diagonalinių elementų pridedam po 0,01 tam,
% kad jei kuria nors, naujai suformuota, kryptimi duomenų  išsibarstymas
%  (dispersija naujoje kryptyje, t. y.
 % nuosava reikšmė = diagonalinės matricos „D“ elementas) būtų nulis arba
% artimas nuliui, tai  normalizuodami nebegalėtume bandyti dalyti iš „nulio“,
%  ir tos krypties  per daug nesureikšmintume
C=[c(:,1:p)*Tr,c(:,p+1)];
% pasukam prognozuojančius požymius taip, kad naujų (pasuktų) požymių
% erdvėje jie būtų nekoreliuoti ir turėtų (maždaug) vienetines dispersijas;
% knygelėje anksčiau kalbėjome, kad tada perceptronas greičiau mokosi
% ir kad po pirmosios iteracijos jau gali išeiti visiškai nebloga prognozė,
% ypač, jei „iksai“ tarpusavyje mažai koreliuoti.
am=C(nm1:nm2,1:p);ym=C(nm1:nm2,p+1);
at=C(nt1:nt2,1:p);yt=C(nt1:nt2,p+1);
% suformuoti mokymo ir test duomenys
[Wslp,Wbest,er1,et1,BETA,ROm,ROt,KIEK]=
RobSLPCminmaxNEW(am,ym,at,yt,iter,eta,zeros(1,p+1),Cmin,Cmax,gama);
% mokome vienasluoksnį perceptroną,
% skirtą  tiesinės prognozavimo lygties svoriams rasti
figure(1);clf;hold off;plot([1:iter],er1,'g:',[1:iter],et1,'k-');
% mokymo ir testavimo prognozavimo paklaidų kitimas
ym_SLPprognoze = am*Wbest(1:p)'+Wbest(p+1);
% suskaičiuojam prognozę mokymo duomenų
figure(2);plot([nm1:nm2],ym,'g-',[nm1:nm2],ym_SLPprognoze,'b-');
% mokymo duomenų tikros (žalios) ir prognozuotos (mėlynos) reikšmės
yt_SLPprognoze = at*Wbest(1:p)'+Wbest(p+1);
% suskaičiuojam testinių duomenų prognozę
figure(3);plot([nt1:nt2],yt,'g-',[nt1:nt2],yt_SLPprognoze,'b-') ;
% testinių duomenų tikros (žalios) ir prognozuotos (mėlynos) reikšmės
disp(corrcoef([yt(1:end-1,1),yt(2:end,1)]))
% suskaičiuojam ir atspausdinam koreliacijos koeficientus
% tarp šios ir rytdienos „y-grekų“
figure(4);plot(yt,yt_SLPprognoze,'b.')
% tikros versus prognozuotų SLP reikšmės testinių duomenų
```

```
figure(5);plot([nt1:nt2],yt-yt_SLPprognoze,'r-') ;
% paklaida įvertinta testiniams duomenims;
figure(6);plot([nm1:nm2],ym-ym_SLPprognoze,'r-') ;
% mokymo duomenų paklaida
figure(7);clf;hold off;plot([1:iter],ROt,'k-',[1:iter],ROm,'g--','linewidth',3);
% koreliacijų mokymo ir validavimo procese kitimas per iter mokymo epo-
chų
figure(8);clf;hold off;plot([1:iter],KIEK,'k-','linewidth',1.6);
% mokant dalyvavusių vektorių skaičiaus kitimas per iter mokymo epochų
[W,w0]=kv_regresija([am,ym]);ymP=am*W'+w0;
%  sudarom standartinę minimalių kvadratų regresiją,
% skirtą tiesinės prognozavimo lygties svoriams rasti
figure(9);plot(ym,ymP,'kd')
% grafiškai parodom tikslumo, įvertinto su  mokymo duomenimis, kitimą
ytP=at*W'+w0;figure(10);plot(yt,ytP,'kd')
% čia prognozuojami testiniai, mokymo procese nedalyvavę vektoriai
figure(11);plot(yt(2:end,1),yt(1:end-1,1),'ro')
% „bukas“ POKYČIO prognozavimas: atseit ryt bus taip kaip šiandien.
disp([corrcoef([yt(1:end-1,1),yt(2:end,1)]),
corrcoef([yt,ytP]), corrcoef([yt,yt_SLPprognoze]),
[min(et1);min(er1)],[sqrt(sum((yt-ytP).^2)/(nt2-nt1+1));
sqrt(sum((ym-ymP).^2)/(nm2-nm1+1))],
[sum(abs(yt-ytP))/(nt2-nt1+1);sum(abs(yt-yt_SLPprognoze ))/(nt2-nt1+1)]]);
%  šios penkios eilutės programoje turi būti pateikiamos kaip viena
%  spausdinu TEST koreliacijas:
%  a) koreliacijos koeficientas tarp šios ir rytdienos „y-grekų“
%  b) su standartine regresija,
    c) su VsPc;
% taip pat spausdinu su VsP gautas
% absoliutinių (7-ame ir 8-ame stulpeliuose) kvadratinių paklaidų reikšmių
% vidurkius,  apskaičiuotus pagal  testinius (viršuje) ir mokymo (apačioje)
% duomenis,  o  9-ame stulpelyje – su standartine regresija gauti  kvadratinių
% paklaidų  reikšmių vidurkiai  teste (viršuje) ir mokymo procese (apačioje)
v1=min(yt);v2=max(yt);
figure(12);plot(yt,ytP,'k.',yt,yt_SLPprognoze,'b.',[v1 v2],[v1 v2],'r:')
% čia parodyta  testinių vektorių prognozė su standartine regresija (juodai)
% ir su perceptronu (mėlynai)  raudona punktyrinė linija – taip būtų,
% jei prognozė būtų idealiai tiksli
return
% programos ML_Birza_PROGNOZE  PABAIGA
```

Programa *ML_Birza_PROGNOZE* turėtų būti tik „pavyzdys“, kur užrašyta keliasdešimt eilučių, t. y. „užduotėlių“, ir pakomentuota, ką kiekviena iš jų atlieka. Minėta programa įkrovė duomenis, suformavo dviejų dienų iš dviejų „svarbiausių“ biržos indeksų sudarytų duomenų masyvą, kad būtų galima prognozuoti biržos indekso pokytį vieną dieną į priekį. Prognozavimas atliekamas pagal dviejų dienų (vakar ir šiandien) dviejų požymių, pirmo (pats biržos pokytis) ir vienuolikto, informaciją. Tad prognozuojant naudojami tik keturi požymiai. Beje, tai yra geriausias požymių rinkinys, kurį pasisekė rasti. Buvo bandoma prognozuoti naudojant ir trijų–septynių dienų istoriją, naudoti daugiau rodiklių, bet geresnio tikslumo prognozuojant gauti nepavyko. Gali būti, kad studijuojantysis gaus ir geresnį rezultatą. Gal būtų verta pabandyti?

Kad perceptronas mokytųsi greičiau ir jau iš pat pradžių pasiektų gerą rezultatą, suformuotieji duomenys buvo modifikuoti. Iš kiekvieno požymio buvo atimtas mokymo duomenų vidurkis, kiekvienas požymis normalizuotas taip, kad jo vidutinis kvadratinis nuokrypis taptų artimas vienetui. Be to, prognozuojantys požymiai (iksai) buvo „dekoreliuoti“ (pasukti taip, kad koreliacijos tarp jų taptų artimos nuliui) ir jų vidutiniai kvadratiniai nuokrypiai dar kartą buvo normalizuoti, kad taptų artimi vienetui. Prognozuojamas požymis („y-grekas“) taip pat buvo normalizuotas, bet nebuvo „pasuktas“.

Kitame etape perceptronas yra mokomas, pradedant iš nuliukų sudarytų svorių vektoriumi `zeros(1,p+1)`. Kad būtų galima ignoruoti nestandartinius, netipiškus mokymo vektorius, buvo naudojama robastinė, vienasluoksniu perceptronu paremta regresija (žr. 2.1.4 ir 2.2.4 poskyrius). Kad perceptronui būtų „pasakyta“, kas yra „netipinis stebėjimas“, turi būti nustatyti net trys valdantieji parametrai: `Cmin`, `Cmax` ir `gama`. Pirmasis iš jų, `Cmax`, – tai maksimali prognozavimo paklaida (toliausiai nuo prognozavimo hiperplokštumos esančio mokymo imties vektoriaus) pačioje mokymo pradžioje, `Cmin` – tai maksimali „leidžiama“ prognozavimo paklaida. Tai toliausio, dar neignoruoto vektoriaus atstumas nuo hiperplokštumos mokymo pabaigoje. Parametras `gama` valdo kaip „leistinas atstumas“, `C` mažėja pradedant nuo `Cmax` iki galimos ribos `Cmin`. Kaip atstumas `C` turėtų būti

valdomas, kol kas dar nėra ištirta, ir tai galėtų būti būsimų tyrimų (ir tyrėjų) tikslas.

Kaip minėta, formuluojant sau keliamus darbo su tiesinėmis prognozavimo lygtimis uždavinius, pirmiausia reikia gerai įsisavinti *Matlab'o* kalbą, iš anksčiau pateiktų komentarų suprasti, ką kiekviena programos eilutė daro ir kam joje nurodyti skaičiavimai yra reikalingi. Tik įsisavinę paprastesnes programėles, naudodami „komandą" help papildomai informacijai rasti ir perskaitę teoriją, galime žengti pirmyn – prie sudėtingesnių uždavinių. Bandymas „peršokti" kai kuriuos darbo etapus sukels nesupratimą, nereikalingą sutrikimą ir netgi nepasitikėjimą savimi.

Programa *ML_Birza_PROGNOZE* leidžiama iš *Matlab Command window*

```
>>Cmin=0.012;Cmax=2;gama=0.8;iter=1000;
  ML_Birza_PROGNOZE
                (351,450,451,1370,[1 11],iter,0.001,Cmin,Cmax,gama);
```

Rezultatų spausdinimo komanda

```
disp([corrcoef([yt(1:end-1,1),yt(2:end,1)]),corrcoef([yt,ytP]),
corrcoef([yt,yt_SLPprog]), [min(et1')',min(er1')']',
[sqrt(sum((yt-ytP).^2)/(nt2-nt1+1));
                        sqrt(sum((ym-ymP).^2)/(nm2-nm1+1))]]);
```

pateikia:

a) koreliacijos koeficientą tarp šios ir rytdienos „y-grekų". Tai „buka" prognozė. Iš tikrųjų matome, kad šiuo atveju koreliacijos koeficientas netgi neigiamas (-0,495). O tai reiškia, kad jei šiandien biržos indeksas pakilo, tai rytoj tikriausiai nukris. Ir atvirkščiai.

```
 1.0000   -0.4951
-0.4951    1.0000
```

b) koreliacijos koeficientą tarp tikrosios ir prognozuotų „y-grekų" reikšmių, jei prognozuojant naudojama standartinė regresija, koreliacijos koeficientas ρ=0,9188:

```
1.0000    0.9188
0.9188    1.0000
```

c) koreliacijos koeficientą tarp tikrosios ir prognozuotų „y-grekų“ reikšmių, kai prognozuojant buvo panaudota robastinė VsP paremta regresija. Šiuo atveju koreliacijos koeficientas turbūt aukštesnis – ρ=0,9765:

1.0000 0.9765
0.9765 1.0000

d) tolesniuose dviejuose stulpeliuose pateikiami su VsP gauti absoliutinių (septintame) ir (aštuntame) kvadratinių paklaidų reikšmių vidurkiai, apskaičiuoti pagal testinius (viršuje) ir mokymo (apačioje) duomenis:

0.0622 0.0835
0.1839 0.4138

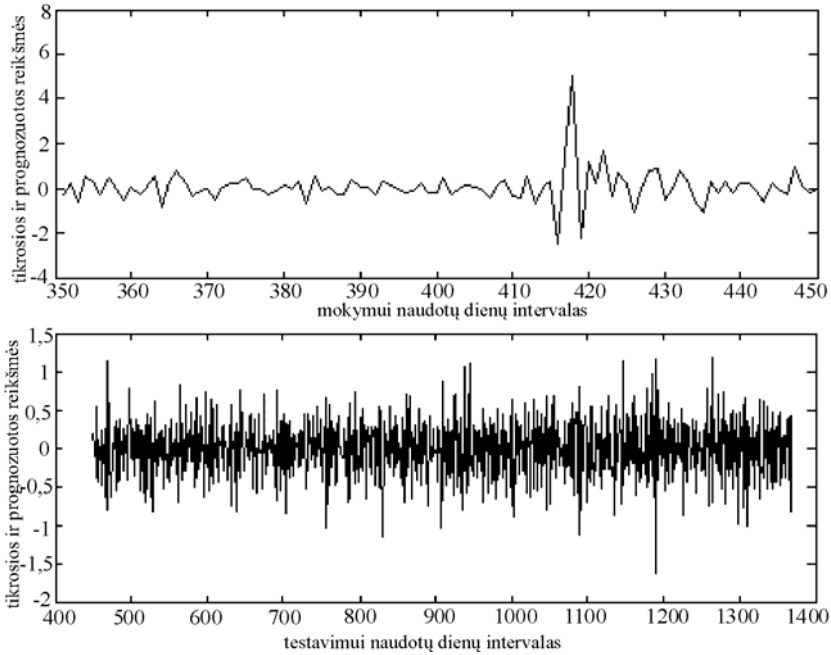
e) devintame stulpelyje pateikiami su standartine regresija gauti kvadratinių paklaidų reikšmių vidurkiai, apskaičiuoti pagal testinius (viršuje) ir mokymo (apačioje) duomenis:

0.2445
0.2880

Ką parodė minėta programa atlikti eksperimentai?

Reikia pažymėti, kad šį kartą parodėme ypač sėkmingą robastinės regresijos testavimo rezultatą: 42 paveikslo viršuje pateiktas biržos indekso tikrųjų reikšmių pokyčio grafikas rodo, kad 415–440-ties dienų laikotarpiu birža ypač (nestandartiškai) svyruoja.

Mokymo duomenys buvo specialiai parinkti taip, kad į juos būtų įtraukti netipiniai stebėjimai. Testavimo laikotarpiu (apačioje) tokių didelių svyravimų beveik nėra. Tad visais metodais gautas prognozavimo tikslumas testiniuose duomenyse yra aukštesnis. Apskritai prognozavimo tikslumas gana aukštas, todėl 42 paveiksle prognozuotosios ir prognozuojamosios reikšmės beveik sutampa. Vėl primename, kad geriau pamatyti skirtumus padėtų spalvotų grafikų analizavimas kompiuterio ekrane. Juos galima rasti minėtame KU tinklalapyje.



42 pav. Biržos indekso tikrųjų reikšmių pokyčio (matyti tik patys didžiausi svyravimai) ir VsP apskaičiuotos prognozės (mažesni svyravimai) kitimas mokymui ir testavimui parinktų dienų laikotarpiais

Kartojant eksperimentus keliolika kartų, robastinės regresijos mokymo procesą valdantis *parametrų* Cmin, Cmax ir gama *„komplektas“ taip pat buvo labai sėkmingai parinktas*, dėl to pasisekė beveik idealiai „ignoruoti“ netipinius stebėjimus ir gauti palyginti labai aukštą tikslumą. Tas ypač matoma lyginant VsP ir standartines minimalių kvadratų būdu gautas regresijas: VsP, matyt, gaunamos aukštesnės koreliacijos ir mažesnės prognozavimo paklaidos (tiek absoliutinė, tiek kvadratinė). Pateiktoje programoje įdėta keletas gautų rezultatų tikslumą charakterizuojančių rodiklių. Jie pavaizduoti 2D išsibarstymo diagramose. Pateiktą programą galima, o galbūt ir reikėtų, modifikuoti, įtraukiant naujus spausdinimus (kaip tai daryti – programoje yra

daug pavyzdžių), braižant papildomus grafikus, charakterizuojančius ne tik VsP darbą, bet ir standartine regresija skaičiuojamas prognozes. Reikėtų bandyti prognozuoti ne vieną, o daugiau dienų į priekį. Reikėtų naudoti daugiau rodiklių, galbūt ne dviejų, o trijų ar net keturių dienų biržos indeksų kitimo istorijas.

Analizuojant eksperimentinius duomenis ir ieškant juose dėsningumų, svarbiausia ne tiek patys skaičiavimai, kiek jų interpretavimas. Štai keletas iš galimų išvadų:

1. Yra prasmė prognozuoti ne patį biržos indeksą, o jo pokytį.

2. Palyginti su „buka“ prognoze „ryt bus taip, kaip ir šiandien“, prognozuodami perceptronu ar standartine regresija gavome ženkliai aukštesnį tikslumą. Šiame konkrečiame uždavinyje gautas tikslumas visiškai neblogas.

3. Koreliacijos koeficientas ir minimali vidutinė kvadratinė paklaida yra du skirtingi prognozavimo tikslumą nusakantys rodikliai.

4. Jei turime testinius (ar validavimo) duomenis, prasminga kaip galutinį rezultatą rinktis tą perceptrono svorių vektorių, kuris gaunamas „minimumo“ (vertinant pagal testinius duomenis, įvertintus daugelio mokymo iteracijų metu) taške.

Norėdami paskatinti studijuojančiuosius pačius analizuoti gautus rezultatus, paliekame tolesnes išvadas bandyti formuluoti patiems.

Per laboratorinius ir namų darbus studentas ar savarankiškai studijuojantysis pirmiausia turi išbandyti dėstytojo pateiktas programėles ir išsiaiškinti, ką kiekviena iš programos eilučių atlieka, ką reiškia kiekvienas atspausdintas rezultatas, nupieštas grafikas. Išanalizavęs programėlę, studijuojantysis turi pabandyti ką nors savarankiškai pakeisti, ir svarbiausia, – suprasti, ką jis keičia (ar požymių rinkinį, ar mokymo duomenų kiekį, ar mokymo algoritmo parametrus ir t. t.). Dėstytojo (ir studijuojančiojo taip pat) tikslas, kad studijuojantysis išmoktų *savarankiškai formuluoti uždavinius*, numatyti jų sprendimo būdus, nuspėti gautus tyrimo rezultatus ir atlikti tiriamąjį darbą. *Išmokimas savarankiškai formuluoti sau keliamus uždavinius, juos spręsti ir įvertinti gautą rezultatą yra BŪTINAS praktiniame darbe*.

Pirmoje pirmojo darbo dalyje dirbama su mažais, pagal normalųjį pasiskirstymą dirbtinai generuotais duomenimis. Šiuo atveju lengviau išsiaiškinti pagrindinius tiesinės regresijos sudarymo ir gautų

rezultatų įvertinimo principus. Sprendžiant šį uždavinį reikia susigeneruoti mokymo bei testinius duomenis ir atlikti su jais nemaža eksperimentų, iliustruojančių įvairius paties studijuojančiojo pasirinktus prognozavimo uždavinio aspektus.

Štai keletas galimų tyrimo krypčių užduočių:

1. Nagrinėti prognozavimo tikslumą, įvertinamą naudojant mokymo ir testinius duomenis.

2. Nagrinėti, kaip mokymo duomenų kiekis veikia gaunamą prognozavimo tikslumą.

3. Nagrinėti, kaip požymių kiekis veikia prognozavimo tikslumą ir kokią įtaką tai daro mokymo duomenų kiekiui.

4. Bandyti keisti programa *gausas.m* generuojamų duomenų kovariacinę matricą, priklausomybes tarp prognozuojamojo ir prognozuojančiųjų požymių. Nagrinėti jų įtaką prognozavimo tikslumui.

5. Šioje darbo dalyje galima bandyti savarankiškai keisti vieno ar keleto mokymo vektorių kai kurių požymių reikšmes (dirbtinai, rankiniu būdu), į generuotus duomenis įvedant triukšmą, t. y. didelius nuokrypius, pvz., po `D=gausas(N,[1,-0.8;-0.8,1], [4,2])`; pridedame: `D(13,2)=6;`). Taip šis tryliktasis stebėjimas tampa „tolimu nuokrypiu“. Tai atlikę, galime nagrinėti, kaip robastinė regresija juos (tuos tolimus nuokrypius) ignoruoja. Bandyti analizuoti, kaip robastiškumą valdantys parametrai gali būti panaudoti robastiškumui stiprinti.

Antra pirmojo darbo dalis turėtų būti atlikta su **realiais duomenimis**. Aukščiau buvo pateikta programėlė

ML_Birza_ PROGNOZE.m,

kur nurodyta nemaža komandinių eilučių, galinčių būti naudingų tiriant realius duomenis. Studijuojantysis gali pasirinkti ir kitus duomenis.

Antros namų darbo dalies *tikslas* – išmokti analizuoti realius duomenis, gauti kuo geresnę prognozavimo lygtį, rasti svarbiausius, tiriamam reiškiniui įtakos turinčius požymius, sugebėti kritiškai įvertinti darbo rezultatą. Darbo su generuotais normaliniais duomenimis dalyje buvo akcentuojami prognozuojančios lygties gavimo būdai, problemos, susijusios su mokymu, prognozavimo tikslumo įvertinimu. Antroje dalyje, atliekamoje su realiais duomenimis, akcentuojame praktinį darbo aspektą – naudingos informacijos išgavimą iš duomenų

masyvo ir jos panaudojimą. Čia vėl nėra išvardyti iš anksto nustatyto darbo etapai. Keičiant mokymo duomenis (imties tūrį, požymių rinkinį, prognozuojamų dienų skaičių, perceptrono parametrus) reikia pabandyti pagerinti prognozavimo su realiais duomenimis tikslumą.

Uždavinių formuluočių gali būti ir daugiau. Daugelis jų buvo minėtos aprašant duomenų analizės metodus. Dalis bus išvardyta 8.5 poskyrio pabaigoje. Siūlome studijuojančiajam išplėsti jų grupę.

8.3. Darbas Nr. 2. Tiesinė klasifikavimo lygtis

Darbo tikslas – išanalizuoti klasifikavimo uždavinio esmę, išmokti naudotis standartine tiesine diskriminantine Fišerio funkcija ir vienasluoksniu perceptronu, išbandyti šiuos du metodus sprendžiant dirbtinai generuotus ir realaus pasaulio duomenų analizės uždavinių duomenis, išmokti įvertinti gautą klasifikavimo tikslumą. Kaip ir regresijos uždavinio atveju, užduotis – įsisavinti programinę įrangą, išmokti ją modifikuoti ir ją naudojant analizuoti išmoktas teorines žinias.

Tiesiniai statistiniai klasifikatoriai. Programų tekstai:

Pirmiausia pateiksime programėlę, skirtą tiesinei klasifikavimo lygčiai sudaryti naudojant vienasluoksnį perceptroną. Programa gauna mokymo duomenų masyvus, klases – **A** ir **B**. Kiekvienas iš masyvų sudarytas ir *ma* eilučių, *k* pozicijų (elementų, komponenčių, požymių) kiekviena.

Pagalbinės programos (jų smulkiai aiškintis nebūtina, bet galima)

Standartinės tiesinės diskriminantinės Fišerio funkcijos gavimas (viena komandinė eilutė)

```
M1=mean(A);M2=mean(B);WFisher=[(M1-M2)*inv(cov(A)+cov(B)),0];
```

Suskaičiavome dviejų klasių vidurkius:

M1=mean(A); M2=mean(B),

kovariacines matricas cov(A)+cov(B) ir svorių vektorių WFisher. Slenkstį w0 laikome lygiu NULIUI, kadangi manome, kad mokymo duomenų vidurkis yra „nustumtas“ į nulinį tašką, t. y. M1+M2=0.

Tiesinės diskriminantinės funkcijos gavimas vienasluoksniu perceptronu slperceptron_as.m

```
%  Tiesinės diskriminantinės funkcijos gavimas su VsP pagalba
%  Autorius: Šarūnas Raudys  raudys@ktl.mii.lt
%  A, B – mokymo duomenų masyvai
%  n – mokymo iteracijų (epochų) skaičius
%  step – mokymo žingsnis
%  at, bt – testiniai (arba validavimo) duomenys
%  gama = 1,001 ar 1,03 – valdo mokymo žingsnio augimą mokymo metu;
%  po kiekvienos epochos mokymo žingsnis step  padauginamas iš gama,
%  tad, jei gama=1,001 ir mokymo iteracijų – tūkstantis, tai
%  mokymo metu mokymo žingsnis išauga 1.001^1000 =  2,7 karto.
%  LL, R2 – antireguliarizacijos parametrai, naudotini valdant svorių dydžius
function [WMIN,etest,etrain,swtest,cost]=
                slperceptron_as(A,B,n,step,target,Wstart,at,bt,gama,LL,R2)
swtest=[[1:min(n,10)],[12:2:min(n,30)],[35:5:min(n,100)],
[110:10:min(n,300)],[320:20:min(n,500)],[550:50:min(n,1000000)]];
% swtest – iteracijų numeriai, kada bus skaičiuojamos klasifikavimo klaidos,
% t. y. siekiant taupyti vietą ir laiką nurodomi epochų numeriai,
%  kada bus įvertintas klasifikavimo tikslumas
SWIP=1;minerror=1.0;
% minerror – nusistatoma minimalios klaidos pradinė reikšmė
[ma k] = size(A);  [mb k] = size(B);k1=k+1;mtab=size(at,1)+size(bt,1);
W=Wstart;WMIN=W;cost=[]; stepz=step/(ma+mb);
ta = (1-target) * ones(ma,1);
tb = target * ones(mb,1);
oa = ones(ma,1);AA=[A,oa];  ob = ones(mb,1);BB=[B,ob];
for i=1:n  %  ciklas iki n iteracijų
   da = AA * W';db = BB * W';fa = oa./(oa+exp(-da));fb = ob./(ob+exp(-db));
   za = ((ta-fa).* (fa - fa.*fa))'*AA;
   zb = ((tb-fb).* (fb - fb.*fb))'*BB;
W  = W + stepz * (za + zb)-LL*(W*W'-R2)*W;
          if  i==swtest(SWIP)
errt=size([find(W(1:k)*at'+W(k1)<0),find(W(1:k)*bt'+W(k1)>0)],2)/mtab;
etest(SWIP)=errt;
errm=size([find(W(1:k)*A'+W(k1)<0),
                                        find(W(1:k)*B'+W(k1)>0)],2)/(ma+mb);
        etrain(SWIP)=errm;
```

```
        da = AA * W';db = BB * W';
        fa = oa./(oa+exp(-da));fb = ob./(ob+exp(-db));
        cost(SWIP)=sqrt(((ta-fa)'*(ta-fa)+(tb-fb)'*(tb-fb))/(ma+mb));
         SWIP=SWIP+1;   if errt < minerror minerror = errt;WMIN=W;
        end;
    end;    stepz=stepz*gama;
end  %  ciklo iki n iteracijų pabaiga
swtest=swtest(1:SWIP-1);
return
```

Dvi klases skiriančios tiesės piešimas dvimatėje erdvėje plot_LDF(V,w,s).m

```
% plot_LDF([V(1),V(2),V(3),V(4)],w,'k-')
% sritytje, apibrėžiamoje x1 ir x2-uoju koordinatėmis V,
% nupiešia skiriamąjį paviršių (tiesę)
% w(1:2)*(x1,x2)'+w(3)=0, čia tiesės lygtis
% s – nurodo, kokia spalva ir kokia linija piešti dvi klases skiriančią tiesę

function plot_LDF(V,w,s)
    w1 = w(1:2); w0 = w(3);
    x = sort([V(1),V(2),(-w1(2)*V(3)-w0)/w1(1),(-w1(2)*V(4)-w0)/w1(1)]);
    x = x(2:3) ;
    y = (-w1(1)*x-w0)/w1(2);
    plot(x,y,s);
return
```

Pseudovalidavimo duomenų masyvo padarymas iš mokymo duomenų ColoredGAUSnoise.m

Programa iš mokymo duomenų padaro pseudovalidavimo duomenis, reikalingus siekiant nustatyti, kada būtų reikėję sustabdyti perceptrono mokymą. Kiekvienam mokymo vektoriui ji suranda KN (siūlau du arba tris) artimiausius kaimynus ir jų kryptimi sugeneruoja Nnoise naujus vektorius. Taip duomenų kiekis „išauga“ Nnoise kartų. „Triukšmo“ dispersija valdoma parametru NoiseSD. Nauji duomenys gali būti naudojami ne tik testuojant (nustatant optimalų mokymo iteracijų skaičių, parenkant geriausią požymių rinkinį ir pan.), bet ir mo-

kant vienasluoksnį arba daugiasluoksnį perceptronus. Pastaruoju atveju mokymo procese naudojame naujus, su triukšmu sugeneruotus duomenis, o testuojame su originaliaisiais, t. y. „mokymo“, duomenimis. Kai kas teigia, kad taip išeina netgi geriau, bet vadovėlio autorius tuo nėra įsitikinęs. Šio klausimo tyrimas galėtų būti savarankiško tyrimo darbo Nr. 4 objektas. Tai galėtų tapti ir rimto mokslinio tyrimo tema.

```
% k-NN DIRECTED (COLORED) NOISE INJECTION
% Autorius: Aušra Saudargienė, o
% Šarūnas Raudys <raudys@ktl.mii.lt> programą modifikavo.
% Iš kiekvieno mokymo vektoriaus X padaroma “Nnoise” naujų vektorių,
%  vektoriaus X komponentes atsitiktinai pastumiant (ten-atgal) KN jo
% artimiausių kaimynų kryptimis. Atsitiktinis POSTŪMIS pasiskirstęs pagal
%  normalųjį dėsnį, kurio vidurkis 0, ir standartinis nuokrypis “NoiseSD”,
% padaugintas iš atstumo iki artimiausio kaimyno, ka kryptimi.
% Teorija – paskaitų konspekte, straipsnyje ir knygoje:
% Skurichina M., Raudys S., Duin R. P. W. K-nearest neighbors directed
%  noise injection in  multilayer perceptron training, IEEE Trans.  on Neural
% Networks, 11:504-511, 2000, taip pat knygoje Raudys S. Statistical and
% Neural Classifiers: An integrated approach to design. Springer, 2001.
%  Ši triukšmo generavimo idėja  vėliau panaudota ir  “Method and system
%  for classifying a halftone pixel based on noise injected halftone frequency
% estimation. United States Patent 6185336. Issued on Feb. 6, 2001”
% Programos įėjimai:
% A – duomenų masyvas Na vektorių-eilučių po p požymių.
% KN – artimiausių kaimynų skaičius: rekomenduotina du arba trys.
% Nnoise – naujų vektorių, dirbtinai sugeneruotų iš vieno mokymo
%  vektoriaus, kiekis. Jei duomenų nedaug, galima imti Nnoise=10;
% NoiseSD – uždedamo triukšmo standartinis nuokrypis
% Programos išėjimai:
% NOISDATA – „užtriukšmintas“ ir “Nnoise” kartų padidintas mokymo
% duomenų masyvas

function NOISDATA=ColoredGAUSnoise(A,KN,Nnoise,NoiseSD);
[Na,p]=size(A);AT=A';%  standartinis triukšmo nuokrypis
NOISDATA=[];ONESV=ones(KN,1);ONESH=ones(1,p);
ONESDATA=ones(1,Na);
for i=1:Na
   dd=A(i,:)'*ONESDATA-AT;d=sum(dd.*dd);
% atstumų tarp vektoriaus A(i,:) ir visų tos klasės mokymo vektorių,
```

```
% masyvas AT, skaičiavimas
[ds,ind]=sort(d); % surikiuoja atstumus pagal dydį
neg=A([ind(2:KN+1)],:);% randa KN artimiausius kaimynus
DISTANCES=NoiseSD*(neg-ONESV*A(i,:));
% atstumai iki KN artimiausių kaimynų
% dabar bus ciklas iki generuojamų vektorių kiekio, kur prideda triukšmą
for nv=1:Nnoise deltam=(randn(KN,1)*ONESH).*DISTANCES;
% atsitiktinai sugeneruoti atstumai noism(nv,:)=A(i,:)+(sum(deltam));
% prie vektoriaus A(i,:) pridedame KN artimiausių vektorių vidurkius
end
NOISDATA=[NOISDATA;noism];
end
return
```

Štai trys eilutės, skirtos paties metodo ir triukšmo pridėjimo programai testuoti ir suprasti. Šias eilutes paleidę, lengviau suprasite programos veikimo principus.

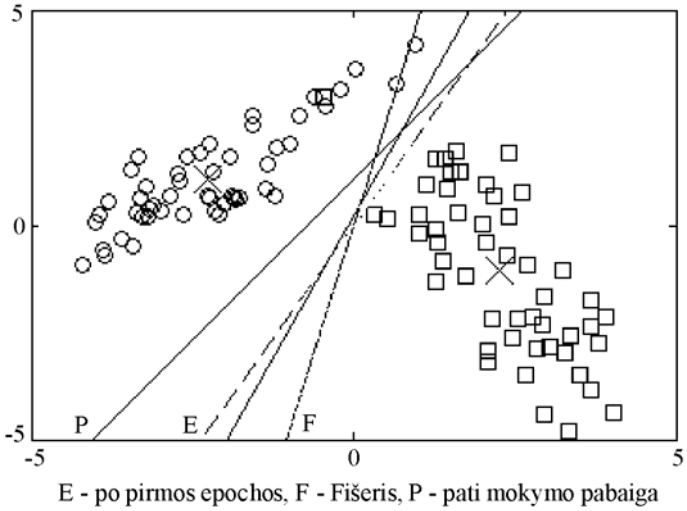
```
a=randn(20,2);A=[a(:,1),0.1*a(:,2)+a(:,1).^2];
BN=ColoredGAUSnoise(A,2,3,1);
figure(1);plot(A(:,1),A(:,2),'ro',BN(:,1),BN(:,2),'g.').
```

Pagrindinės programos (jas analizuoti būtina)

Tiesinių diskriminantinių funkcijų, gautų su perceptrono pagalba, tyrimas naudojant dvimačius normalinius (Gauso) duomenis

KeliKlasifikatoriai.m Programa generuoja dvimačius, pagal normalųjį (Gauso) pasiskirstymą pasiskirsčiusius mokymo ir testinius duomenis, moko perceptroną, parodo, kaip kinta klases skirianti tiesė mokymo metu, įvertina klasifikavimo tikslumą. Kad perceptronas mokytųsi greičiau, iš pat pradžių randamas mokymo duomenų vidurkis (vektorius **MM**), jis atimamas kiekvienoje klasėje tiek iš mokymo, tiek ir iš testinių duomenų. Tada, pradėję perceptroną mokyti nuo „nulinių“ svorių, po pirmos iteracijos gauname euklidinio atstumo klasifikatorius, o tai yra labai gera pradžia. Tai rodo ir *juoda brūkšnine linija* pažymėta tiesė, visiškai neblogai skirianti mokymo duomenis (Figure 2, *Matlab'o* programoje arba 43 pav.). Paskui vėl kartojame perceptrono mokymą pradėdami nuo nulinių svorių dvidešimt, du šimtus,

tūkstantį iteracijų. Po kiekvieno mokymo su programa *plot_LDF* kaskart skirtingomis spalvomis piešiame klases skiriančias tieses (žr. Figure 2, *Matlab'o* programoje arba 43 paveikslą).



43 pav. Dviejų klasių duomenys ir jas skiriančios tiesės, gautos įvairiais būdais

Vienasluoksnis perceptronas gali realizuoti standartinę tiesinę diskriminantinę Fišerio funkciją, taigi ją ir bandome gauti mokydami perceptroną. Dvi eilutės

```
M1=mean(am);M2=mean(bm);
WFisher=[(M1-M2)*inv(cov(am)+cov(bm)),0];
```

suskaičiuoja svorių vektorių WFisher. Paskui nubrėžiame Fišerio klasifikatoriaus rastą skiriamąją tiesę:

```
plot_LDF([-5 5 -5 5],WFisher,'k-');  % juoda linija 43 paveiksle
```

Mūsų pavyzdyje klasės atsiskiria palyginti gerai, todėl klasifikavimo klaidos tikimybė yra maža. Jei trokštami išėjimai yra NULIS arba VIENETAS, tai perceptrono svoriai greitai išauga, o tai apsunki-

na gauti Fišerio klasifikatorių. Kad svoriai neišaugtų ir galėtume parodyti, jog Fišerio klasifikatorius iš tikrųjų gali būti gautas, perceptrono mokymą specialiai modifikuojame. Tam naudojame mažai besiskiriančius trokštamus išėjimus: 0,4 ir 0,6:

```
[WMINf,etestf,etrain,SW,cost]=
                    slperceptron_as(am,bm,16000,1,0.4,Wstart,at,bt,1.0001,0,0).
```

Rezultatas – 43 paveiksle pateikiama žydra punktyrinė linija, kuri praktiškai sutampa su juodąja.

KAIP PIRMĄ KARTĄ KREIPTIS Į ŠIĄ PROGRAMĄ:
>> KeliKlasifikatoriai

```
% autorius: Šarūnas Raudys  raudys@ktl.mii.lt
% tikslas: 2-mačių 2 klasių atveju pamatyti, kaip mokosi perceptronas,
% kaip kinta dvi klases skirianti tiesė
% pradedama nuo euklidinio atstumo klasifikatoriaus (po 1-os iteracijos),
% einama link Fišerio, o galiausiai priartėjama prie atraminių vektorių
% (support vector) klasifikatoriaus
% kreipimasis: iš Matlab'o Command window: >> KeliKlasifikatoriai
Nmok=50;Ntest=250;iteraciju=1;step=0.1;target=0;Wstart=[0 0 0]; ga-
ma=1;LL=0;R2=40;randn('seed',12345);
% nustatome atsitiktinių skaičių generatorių
m1=[2.6 -1];m2=[-2.2 1];;K1=[1 -1.3;-1.3 4];K2=[1 0.8;0.8,1];
% nusistatomi duomenų vidurkiai ir kovariacinės matricos
am=gausas(Nmok,K1,m1);bm=gausas(Nmok,K2,m2);MM=mean([am;bm]);
% susigeneruoju mokymo duomenis,
% randu abiejų klasių bendrą vidurkį: 2-matį vektorių MM
am=am-ones(Nmok,1)*MM;bm=bm-ones(Nmok,1)*MM;
at=gausas(Ntest,K1,m1);bt=gausas(Ntest,K2,m2);
% sugeneruoju kiekvienoje klasėje Ntest normaliųjų 2-mačių vektorių
at=at-ones(Ntest,1)*MM;bt=bt-ones(Ntest,1)*MM;
% iš mokymo ir test duomenų atimu vidurki MM.
[WMIN,etest1,etrain,SW,cost]=slperceptron_as(am,bm,iteraciju,step,target,
                                                    Wstart,at,bt,gama,LL,R2);
% mokau perceptroną
M1=mean(am);M2=mean(bm);
figure(2);clf;plot(am(:,1),am(:,2),'bo',bm(:,1),bm(:,2),'ro');hold on;
plot_LDF([-5 5 -5 5],WMIN,'k--');
plot(M1(1),M1(2),'bx',M2(1),M2(2),'rx','MarkerSize',15);0,pause(4)
```

```
% nupiešiu mokymo duomenis, parodau jų vidurkius, nupiešiu skiriamąją
%  tiesę po 1-os iteracijos. Tai euklidinio atstumo klasifikatorius
[WMIN,etest2,etrain,SW,cost]=slperceptron_as(am,bm,20,step,target,Wstart,
                at,bt,gama,LL,R2);plot_LDF([-5 5 -5 5],WMIN,'r:') ;1,pause(4)
% po 20 iteracijų skiriamoji tiesė (raudona) pajudėjo vos vos
[WMIN,etest3,etrain,SW,cost]=slperceptron_as(am,bm,200,step,target,
        Wstart,at,bt,1.0001,LL,R2);plot_LDF([-5 5 -5 5],WMIN,'g-'); pause(.4)
% po 200 iteracijų skiriamoji tiesė (žalia) taip pat vos vos tepajudėjo, tačiau
 % ji  faktiškai jau „užlipo“ ant raudonos
[WMIN,etest4,etrain,SW,cost]=slperceptron_as(am,bm,1000,step,target,
                  Wstart,at,bt,1.001,LL,R2);plot_LDF([-5 5 -5 5],WMIN,'b-');
% po 1000 iteracijų skiriamoji tiesė (mėlyna) jau matomai pajudėjo,
% bet visgi dar labai nedaug
[WMIN,etest5,etrain5,SW,cost]=slperceptron_as(am,bm,16000,step,target,
                 Wstart,at,bt,1.002,LL,R2);plot_LDF([-5 5 -5 5],WMIN,'m--');
% mokau perceptroną 16 000 iteracijų, smarkiau didinu mokymo žingsnį,
%  kad gaučiau atraminių vektorių klasifikatorių – brūkšninę violetinę liniją
xlabel('violetine – mokymo pabaiga, juoda – Fiseris')
figure(1);clf;plot(at(:,1),at(:,2),'c.',bt(:,1),bt(:,2),'m.',
    am(:,1),am(:,2),'bo',bm(:,1),bm(:,2),'ro',m1(1),m1(2),'bx',m2(1),m2(2),'rx')
% piešia  mokymo ir testinius duomenis, pažymi tikruosius  klasių centrus
WFisher=[(M1-M2)*inv(cov(am)+cov(bm)),0];
figure(2);plot_LDF([-5 5 -5 5],WFisher,'k-');
% brėžiu Fišerio klasifikatoriaus skiriamąją tiesę
[WMINf,etestf,etrain,SW,cost]=slperceptron_as(am,bm,16000,1,0.4,
                                                Wstart,at,bt,1.0001,0,0);
plot_LDF([-5 5 -5 5],WMINf,'c:');
disp('taip mokant išėjo Fišerio klasifikatoriaus žydra spalva')
% Teko pavargti, siekiant rasti sąlygas (net 16 000 iteracijų,
% ir mokymo žingsnis po % kiekvienos iteracijos vis buvo
% po 0,1 PROCENTO didinamas), kad skiriamoji
% tiesė (žydra) faktiškai sutaptų su statistinio Fišerio klasifikatoriaus tiese,
%  žr. apačioje:
figure(1);hold on;plot_LDF([-5 5 -5 5],WMIN,'m--');
plot_LDF([-5 5 -5 5],WFisher,'k-');
% brėžiu mokymo ir test duomenis bei Fišerio (juoda)
%  ir perceptrono (violetinę) galutinę skiriamąsias tieses
figure(3);clf;plot(SW,etest5,'m-',SW,etestf,'c:',SW,etrain5,'m:');
% braižau klasifikavimo klaidų mažėjimą augant mokymo iteracijų skaičiui
```

Programėlės KeliKlasifikatoriai.m pagrindinis piešinys – tai mokymo duomenų išsibarstymas dvimatėje erdvėje, Fišerio klasifikatoriaus skiriamoji tiesė (juoda) bei kelios abi klases skiriančios tiesės, gutos mokant VsP įvairų iteracijų skaičių. Viena jų (žydra) praktiškai sutampa su Fišerio sprendiniu (juoda linija). Violetinė tiesė nutolusi maždaug vienodais atstumais nuo artimiausių priešingų klasių mokymo vektorių ir apytiksliai realizuoja atraminių vektorių (*support vector*) klasifikatorių.

Atpažinimo algorimo, skirto atskirti gerus elektros variklius nuo blogų, sudarymas naudojant vienasluoksnį perceptroną. Klasifikavimas_suSLP_U.m

```
% KAIP PIRMĄ KARTĄ KREIPTIS Į ŠIĄ PROGRAMĄ
% Q=1;N=100;kiek=10;sigma=0.8;iter=500; step=0.1;LL=0.8;
% Klasifikavimas_suSLP_U    %  arba
% Q=2;N=100;kiek=10;sigma=0.8;iter=500; step=0.1;LL=0.8;
% Klasifikavimas_suSLP_U
%  Q=1 VECTORS: 1:2:end - for training, and
%  VECTORS: 2:2:end - for testing,              Q=2 - vice versa (priešingai)
% kiek – naujų dirbtinių vektorių kiekis „triukšmo“ pridėjimo programoje
% sigma – standartinis triukšmo nuokrypis
% generuojant pseudovalidavimo duomenis
% iter – batch iteracijų (epochų) skaičius
% autorius: Šarūnas Raudys  raudys@ktl.mii.lt
% Tai tik „standartinė“ programa; ją supratę mokėtume ją
% modifikuoti ir parodyti VsP pritaikymą gan nelengvam uždaviniui
load vibration;
% nuskaitom elektros motorų vibravimo spektrų duomenis
% A01 ir B01 (pirma–antra klasės)
[NT, p]=size(A01); % duomenų dydis
L1=1-LL;
% čia LL – reguliarizavimo parametras, nustatantis,
%  kaip pasukę pradėsim mokyti perceptroną
A=A01(Q:2:NT,1:p);B=B01(Q:2:NT,1:p);
% Q nurodo, kuri duomenų dalis bus naudojama mokymui,
%  o kuri – testavimui. Kas antrą imam mokymui,
%  o kitus kas antrą – testavimui
VID=mean([A;B]);SS=inv(sqrt(0.5*diag(diag(cov(A)+cov(B))+0.02)));
A=(A-ones(N,1)*VID)*SS;B=(B-ones(N,1)*VID)*SS;
```

```
% atimam vidurkį, padalijam iš standartinių nuokrypių
A1=A01(3-Q:2:NT,1:p);B1=B01(3-Q:2:NT,1:p);A1=(A1-
ones(N,1)*VID)*SS;B1=(B1-ones(N,1)*VID)*SS;
% tas pats test duomenims
AN=ColoredGAUSnoise(A,3,kiek,sigma);
BN=KUcoloredGAUSnoise(B,3,kiek,sigma);
% padarom pseudovalidavimo duomenis,
%  prie kiekvieno mokymo vektoriaus pridedami  kiek „užtriukšmintų“
%  vektorių.  Pvz., jei kiek=5, padarom penkis naujus vektorius
K1=cov(A);K2=cov(B);K=K1*0.5+K2*0.5;[T,D,V]=svd(K);
Tr=T*inv(sqrtm(D*L1+LL*eye(p)));
% randam duomenų pasukimo matricą „Tr“.
% Tai darom todėl, kad požymiai taptų nebekoreliuoti ir
% kiekvieno naujo požymio dispersijos būtų artimos VIENETUI
As=A*Tr;Bs=B*Tr;A1s=A1*Tr;B1s=B1*Tr;AsN=AN*Tr;BsN=BN*Tr;
% mokymo, testinių ir pseudovalidavimo duomenų pasukimas
figure(3);plot(As(:,1),As(:,2),'r.',Bs(:,1),Bs(:,2),'b.')
% pažiūrim, kaip atrodo klasės dviejų pirmųjų pasuktų požymių erdvėje
Ps=3;Pk=2;figure(4);plot(As(:,Ps),As(:,Pk),'r.',Bs(:,Ps),Bs(:,Pk),'b.')
% žiūrim, kaip tos klasės atrodo kitų laisvai pasirinktų „Ps“ ir „Pk“ požymių
% erdvėje. Studijuojantysis gali pažiūrėti, kaip perceptronas mokysis,
%  jei duomenų nepasuksim. Tikėtina, kad  mokymas vyks lėčiau. Nors
% visko būna. Jei pasukimas „blogas“ (pvz., jį konstruojame iš pernelyg mažo
%  duomenų kiekio), tai pasukimas gali ir pabloginti.  FILOSOFIJA:
% Kiekvienas geras metodas, jį blogai naudojant, gali ir pakenkti.

Wstart=zeros(1,p+1); % startinis svorių vektorius (nuliai) –
% rekomenduotina pabandyti ir kitokius startus, pvz.,
% Wstart=zeros(1,p+1)+0.2;
[W1t,etest1,etrain,swtest,ct]=
                        slperceptron_as(As,Bs,iter,step,0,Wstart,A1s,B1s,1,0,10);
% mokom su As,Bs+ testuojam su testiniais duomenimis, A1s,B1s
P1=size(find(W1t(1:p)*A1s'+W1t(p+1)<0),2);
P2=size(find(W1t(1:p)*B1s'+W1t(p+1)>0),2);
% suskaičiuojam, kiek neteisingai suklasifikavo pirmos (P1) ir
%  antros (P2) klasių testinių vektorių
disp([size(A1s,1)-P1,P1;P2,size(B1s,1)-P2]);
% klasifikavimo klaidų kiekį atspausdinam kaip lentelę
[W1,etestV,etrain,swtest,ct]=
                        slperceptron_as(As,Bs,iter,step,0,Wstart,AsN,BsN,1,0,10);
```

```
% mokom su mokymo duomenimis,
% bet testuojam su pseudo-validation („užtriukšmintais“)
figure(1);clf;plot(swtest,etest1,'k-',swtest,etestV,'r-',swtest,etrain,'g:');
xlabel('iterations');ylabel('red-validation,
                                        black-test, green - training error rates')
disp('etest1end min(etest1) etestVend min(etestV)
                                    etrainend min(etrain)     Startas nuliai');
% spausdinant ant ekrano, rezultatas nurodo, kas ką reiškia
disp([etest1(end),min(etest1),etestV(end),min(etestV),
                                                etrain(end),min(etrain)])
% spausdina rezultatus

Wstart=zeros(1,p+1)+0.09;
% pakeičiame startinį svorių vektorių,
% kad parodytume, jog išeis blogiau, mokymą kartojame  pradėdami nebe
nuo
% nulinių svorių. Matome, kad iš tikrųjų išeina blogiau.
[W1T,etest1,etrain,swtest,ct]=
                    slperceptron_as(As,Bs,iter,step,0,Wstart,A1s,B1s,1,0,10);
% kad pamatytume skirtumą tarp pseudovalidavimo ir testinių duomenų,
% mokom, kaip ir anksčiau, bet testuojam ne pseudovalidavimo
%  duomenimis, bet testiniais
[W1V,etestV,etrain,swtest,ct]=
                    slperceptron_as(As,Bs,iter,step,0,Wstart,AsN,BsN,1,0,1);
% mokom As,Bs bet testuojam su pseudo-validation.
% To reikėtų, siekiant įvertinti optimalų
% mokymo iteracijų skaičių iterValidOpt. Tada imtume perceptroną, mokytą
%  “iterValidOpt” epochų. Studijuojančiajam uždavinį paleisti DU kartus
%  (validuojant su testiniais ir % pseudovalidavimo duomenimis)
%  reikia tam,  kad suprastų (pajaustų) persimokymo efektą,
% ir tam, kad naudodamas  pseudovalidavimo duomenis, galėtų šį efektą
%  įvertinti, nors ir nevisiškai tiksliai.
%  Tačiau suprasti šį efektą yra labai svarbu.
figure(2);clf;plot(swtest,etest1,'k-',swtest,etestV,'r-',swtest,etrain,'g:');
disp('etest1end min(etest1) etestVend minEtestV) etrainend
                                        minEtrain  Startas jau nebe nuliai');
%  spausdinam tekstą, nurodantį, ką žemiau spausdinsime.
 disp([etest1(end),min(etest1),etestV(end),min(etestV),etrain(end),
                                                        min(etrain)])
%  spausdinam klasifikavimo tikslumą mokant ir testuojant (validuojant).
```

Aprašant darbą Nr. 2 reikėtų pateikti:

A. Darbo tikslą, t. y. išvardyti, kokie konkretūs teoriniai teiginiai apie vienasluoksnių perceptronų sudarymo principus, inicializavimą, mokymo parametrus, išdėstyti paskaitose ar rasti savarankiškai studijuojant literatūrą, bus šiame, studijuojančiojo atliktame, darbe patvirtinti/tirti.

B. Darbo eigą, kur bus aprašyti konkretūs eksperimentai, jų atlikimo sąlygos, pateikti rezultatai, nurodyta, kas nauja, lyginant su šioje knygoje pateiktomis programėlėmis, buvo padaryta.

C. Konkrečias išvadas, teiginius. Nurodyti, kokie įvertinimai, grafikai ir kaip patvirtina namų darbo autoriaus formuluojamus teiginius.

Šio tipo „reikalavimai“ galioja ne tik šiam, bet ir kitiems namų darbams, taip pat ir kursiniams, magistriniams darbams.

8.4. Darbas Nr. 3. Daugiasluoksnis perceptronas

Darbo tikslas – išnagrinėti daugiasluoksnio perceptrono veikimo esmę, susipažinti su jo mokymo ypatybėmis, praktiškai susidurti su mokymo sunkumais, daugiasluoksnį perceptroną pritaikyti sprendžiant realaus pasaulio duomenų vaizdavimo, klasifikavimo ir prognozavimo uždavinius. Šį metodą bandysime naudodami realius duomenų analizės uždavinius. Mokysimės įvertinti gautą klasifikavimo, požymių išskyrimo tikslumą. Užduotis – įsisavinti programinę įrangą ir ja naudojantis analizuoti anksčiau šiame vadovėlyje išdėstytą teorinę medžiagą. Naudosime standartines *Matlab'o Neural Networks toolbox* esančias programas. Iš daugelio neuroninių tinklų tipų ir jų mokymo metodų naudosime populiariausią. Tai daugiasluoksnis perceptronas, mokomas antros eilės Levenberg-Marquardt metodu. Šis algorimas yra greitas, tačiau dažnai pakliūva į lokalinius minimumus. Todėl naudosime „*multistart*“ režimą, kai perceptroną mokome ne vieną, o keletą kartų ir iš jų pasirenkame sėkmingiausią. Norint įvykdyti keliamas užduotis, reikia įsidiegti *Matlab Neural Networks toolbox*, kur yra daugiasluoksnį perceptroną realizuojančios programos.

Toliau pateiksime pavyzdinį programos, parašytos *Matlab'o* kalba, tekstą, kur nurodyta nemaža komandų (programos eilučių), kurias galima būtų panaudoti ką tik minėtiems uždaviniams spręsti.

Duomenys. Tiek vienasluoksniame, tiek daugiasluoksniame perceptrone ir mokydami, ir testuodami (arba validuodami) privalome turėti įėjimo vektorių masyvą ir trokštamų (išėjimo) vektorių masyvą. Skirtingai nuo vienasluoksnių perceptronų, *p*-mačių įėjimo vektorių masyvas (pavadinkime jį **P**), – tai $p \times n$ lentelė, sudaryta iš *n* stulpelių, *p* komponenčių (elementų) kiekviename iš jų. Kitaip apibūdinant, masyvas **P** – tai *p* eilučių, kurios visos sudarytos iš *n* elementų. Pažymėtina, kad programos VsP ir DsP parengtos ne vieno žmogaus, todėl duomenys jose apibrėžiami skirtingai (eilutės ir stulpeliai sukeisti). Trokštami išėjimai, masyvas **T** – tai *n* vektorių stulpelių, sudarytų iš *K* reikšmių, kur *K* – išėjimo neuronų kiekis. Jei sprendžiame dviejų klasių atpažinimo uždavinį, tai išėjimas vienas, t. y. $K = 1$. Jei klasių daugiau nei dvi, tai išėjimų skaičius lygus klasių skaičiui. Iš *K* elementų sudarytame stulpelyje *K* iš jų lygūs nuliui, o likęs vienas (tikroji klasė) – vienetui (jei aktyvavimo funkcija sigmoidinė, būtent „`logsig`“). Prognozavimo uždavinyje išėjimų skaičius lygus prognozuojamų parametrų kiekiui. Mokant daugiasluoksnį perceptroną paprastai pateikiami mokymo duomenys (masyvai **P** ir **T**) ir testiniai (arba validavimo) duomenys.

Užkrėstų skirtingo tipo virusais mielių ekspresatpažinimas – tai svarbus ir labai aktualus šiuolaikinis uždavinys. Analizuojami duomenys, masyvas **MIEL1501,** – tai 1001 1501-mačių vektorių masyvas. Šiame masyve pirmosios 1500 įėjimo vektorių komponenčių – tai 1500 spektro atskaitymų. Kiekvienas iš 1001 mielių mėginio užkrėstas kažkuria iš 10-ties ligų. Dešimtyje klasių turime 113, 84, 116, 83, 120, 56, 90, 97, 113 arba 129 vektorius. Paskutinis, 1501-asis, požymis – tai klasės numeris. Tyrimo tikslas: sudaryti atpažinimo (klasifikavimo) taisyklę, leidžiančią pagal 1500 įėjimo vektoriaus (spektro atskaitymų) matavimų patikimai nustatyti, kuriai iš dešimties klasių šis vektorius priklauso.

Toliau pateikiame *Matlab'o* programėlę, skirtą duomenims vizualizuoti naudojant daugiasluoksnį perceptroną. Štai keletas komentarų. Jei programėlę naudosime klasifikavimo ar prognozavimo uždaviniams spręsti, teks ją šiek tiek modifikuoti.

Įėjimo požymių yra daug – net 1500. Pusę turimų vektorių, t. y. 500, naudosime mokymui, t. y. daugiasluoksnio perceptrono svoriams rasti. Likusią duomenų pusę (501 vektorių) naudosime testavimui (tikrinimui), t. y. klasifikavimo klaidos dažniui įvertinti, momentui, kada DsP-no mokymo procesą reikia stabdyti, rasti. Įėjimo požymių yra labai daug, taigi DsP svoriams rasti 500 vektorių yra aiškiai per maža. Todėl įėjimo požymių kiekį reikia mažinti. Pirmas būdas sumažinti požymių kiekį buvo imti tik kas kelintą požymį, pvz.:

```
Pmok=MIEL(1:2:1001),1:rr:1500)'.
```

Šiame pavyzdyje imame kas *rr*-tąjį požymį. Jeigu, pavyzdžiui, *rr* = 20, tai atpažinimo taisyklę sudarysime tik iš 75 požymių. Visi bandymai imti tik kas kelintą požymį sėkmingi nebuvo.

Kitas požymių kiekio sumažinimo variantas – tai suvidurkinti *rr* paeiliui einančių požymių. Šis būdas taip pat nebuvo sėkmingas.

Trečias būdas sumažinti klasifikavimo taisyklei naudojamų požymių kiekį – tai euklidinio atstumo klasifikatoriaus panaudojimas pasuktoje požymių erdvėje. Turėdami dešimt klasių, suskaičiuojame dešimt pasvertų sumų

$$y_j(\boldsymbol{x}) = \boldsymbol{x}^T \, \bar{\boldsymbol{x}}_j - \tfrac{1}{2} \, \bar{\boldsymbol{x}}_j^T \, \bar{\boldsymbol{x}}_j \;, j = 1, 2, \ldots, 10,$$

kur $\bar{\boldsymbol{x}}_j$ – tai j-tosios klasės vidurkių vektorius, įvertintas iš mokymo imties.

Požymiai $y_1(\boldsymbol{x})$, $y_2(\boldsymbol{x})$, ... , $y_{10}(\boldsymbol{x})$ ir bus naudojami klasifikavimo taisyklei sudaryti.

Programa sudaryta iš dviejų dalių. Pirmoje dalyje formuojama požymių sistema ir tolesniame tyrimo procese naudojamas klasių rinkinys, mokymui ir testavimui naudojamų duomenų rinkiniai. Antroje dalyje formuojamas ir mokomas daugiasluoksnis perceptronas, pateikiami rezultatai.

Štai kreipimasis į programą

```
>> clear;STST=24642;rand('state',STST);KIEKiter=[8 32]; FIGU=5; Hid-
                    den=2; klases=[1 3 5 10]; Yeast_MLP_FeatureExtraction;
```

Kreipiantis į programą nurodoma:

```
STST=24642;
%  tai atsitiktinių skaičių generatoriaus pradinis nustatymas,
%  davęs neblogą rezultatą.
KIEKiter=[8 32];  %  tai mokymo epochų skaičius (32), skaičius 8 reiškia, kad
%   rezultatai bus spausdinami kas  8  mokymo epochas, kurių kiekvienoje yra
%   tiek iteracijų, kiek yra mokymo vektorių.
Hidden=2; %  tai paslėptų neuronų kiekis
FIGU=5;
%  tai pirmo piešinio, kur pradedame pateikti duomenis
%  dvimatėse erdvėse numeris. Kartojant eksperimentą galima
%  nurodyti kitą skaičių. Tada grafikus galėsime lyginti.
%  klases=[1 3 5 10]; nurodomas tiriamų klasių rinkinys. Šiame pavyzdyje
%  grafikų piešimo aiškumui pagerinti nurodytos tik keturios pačios
%  gausiausios klasės.
```

Pateikiamame eksperimente antroji dalis vykdoma daugelį kartų (*multistart* režimu), kur kiekvieną kartą perceptronas mokomas pradedant skirtingais svorių vektoriais. Procedūra buvo kartojama daugelį kartų tol, kol pasisekė gan neblogai atskirti keturias klases: 2.5% klaidų mokymo ir 11.7% klaidų testavimo metu.

PIRMA užduoties dalis: duomenų dimensiškumo mažinimas. Klasių bei mokymo ir testavimo duomenų sudarymas

```
load MIEL;%  nuskaitome duomenis,
nntwarn off;
% anuliuojame komentarus, jei buvo naudota senesnė Matlab'o versija.

%  PIRMAS POŽYMIŲ KIEKIO MAŽINIMO IR
%  NORMALIZAVIMO ETAPAS
T=[];KlasNr=[];Nklasiu=size(klases,2); p=1500;
```

```
KlasN=round(MIEL(:,end)./100)';
%  anksčiau, spausdinant duomenis, mums pateiktuose duomenyse klasių
%   numeriai buvo padauginti  iš 100, todėl dabar padalijome
M=mean(MIEL(1:2:end,1:p));sdev=std(MIEL(1:2:end,1:p));
% randame pusės duomenų vidurkį ir standartinius nuokrypius
mstD1=mean(sdev)/10;
MIEL=(MIEL(:,1:p)-ones(1001,1)*M)./(mstD1+ones(1001,1)*sdev);
% duomenis „normalizavome“ atimdami vidurkį ir suvienodindami dispersi-
jas
Nkkk=[113 84 116 83 120 56 90 97 113 129];
% vektorių kiekis klasėse
ko=zeros(p,p); for j=1:10 F=find(KlasN==j);
ko=[ko+cov(MIEL(F(1:2:end)))*Nkkk(j)]; end;
% suradom bendrą visoms klasės kovariacinę matricą „ko“
[T,D,V]=svd(ko/1001);Tt=T*sqrtm(inv(D+0.01*eye(p)));
% radom matricos „ko“ nuosavus vektorius ir nuosavas reikšmes
%  bei duomenų pasukimo matricą Tt
MIEL=MIEL*Tt(:,1:490);
% duomenis pasukame palikdami tik pirmuosius 490 naujų požymių. Tai
% darom todėl, kad likusiose 1010-yje krypčių duomenų dispersija NULIS,
% o požymių net 1500, tuo tarpu mokymo vektorių tik 500.

%  ANTRAS POŽYMIŲ KIEKIO MAŽINIMO IR NORMALIZAVIMO
%  ETAPAS
M=[];TR=[];T=[];
for j=1:10 F=find(KlasN==j);M(j,1:490)=mean(MIEL(F,:));end
%   suradom klasių vidurkių vektorius
for j1=1:10 W=[M,-0.5*sum((M.*M),2)];end;
% suskaičiavom dešimt naujų požymių, t. y.
% (dešimt euklidinio atstumo klasifikatorių) darančią matricą W
MIEL=[MIEL,ones(1001,1)]*W'; % padarėm 10 naujų požymių
for j=1:Nklasiu
F=find(KlasN==klases(j));T=[T,F];KlasNr=[KlasNr,j*ones(1,size(F,2))];end;
MIEL=MIEL(T,:);Nviso=size(MIEL,1);
% suformavom užduotyje nurodytas Nklasiu
M=mean(MIEL(1:2:end,1:10));sdev=std(MIEL(1:2:end,1:10));
mstD1=mean(sdev)/10;
```

```
MIEL=(MIEL-ones(Nviso,1)*M)./(mstD1+ones(Nviso,1)*sdev);
% vėl normalizavom duomenis
raN1=randperm(Nviso);% sumaišėm skaičius nuo 1 iki Nviso
Pmok=MIEL(raN1(1:2:Nviso),:)';Pval=MIEL(raN1(2:2:Nviso),:)';
% sudarėm mokymo ir validavimo (testavimo) duomenis
KlasNr=KlasNr(raN1);
clear MIEL M T TR KlasN sdev  mstD1  % clear  ištrina tai, ko nebereiks
Kmok=KlasNr(1:2:end);Kval=KlasNr(2:2:end);Nm=size(Pmok,2);
Nv=size(Pval,2);T=zeros(Nklasiu,size(KlasNr,2));
for j=1:Nklasiu T(j,find(KlasNr==j))=1;end;clear KlasNr
Tmok=T(:,1:2:end);Tval=T(:,2:2:end);mima=minmax(Pmok);clear  T;pack
% suformavom mokymo ir testavimo (arba validavimo) duomenis ir
%  trokštamus išėjimus
% mima – kiekvieno duomenų požymio minimumai ir maksimumai
%  ANTRA užduoties dalis:  perceptrono mokymas,
%  rezultatų pateikimas ir įvertinimas
%  DABAR MOKYSIME DAUGIASLUOKSNĮ PERCEPTRONĄ su
%  DVIEM paslėptais neuronais tam, kad duomenis projektuotume į
%  DVIMATĘ erdvę pasižiūrėti,
%   ir klasifikavimo uždavinio sunkumui įvertinti
[w11, b11, w21, b21] = initff(Pmok,Hidden,'logsig',Nklasiu, 'logsig');
%  sudarom MLP (daugiasluoksnį perceptroną), turintį "Hidden" paslėptų
%  neuronų ir Nklasiu išėjimų
[w11,b11,w21,b21,te,tr]=
            trainlm(w11,b11,'logsig',w21,b21,'logsig',Pmok,Tmok,KIEKiter);
 %  mokom perceptroną  KIEKiter  epochų
% ČIA BUS KLASIFIKAVIMO KLAIDŲ  MATRICOS SKAIČIAVIMAS
% Nklasiu klasių atveju
net0=nnt2ff(mima,{w11,w21},{b11,b21},{'logsig','logsig'});
% naudodami ankstesnio neuroninio tinklo mokymo rezultatus duomenims
%  pavaizduoti 2D erdvėje suformuojam daugiasluoksnį perceptroną  net0
OUTmok=sim(net0,Pmok);OUTval=sim(net0,Pval);
% randam perceptrono net0 išėjimus mokymo ir test duomenims
EV=zeros(Nklasiu,Nklasiu);ER=EV;
% ER ir EV – tai KLAIDŲ KIEKIO MATRICOS
[mx,IND]=max(OUTmok);PerrR=sum(abs(sign(-Kmok+IND)))/Nm;
for i1=1:Nm ER(Kmok(i1),IND(i1))=ER(Kmok(i1),IND(i1))+1; end;
```

```
% surašo klasifikavimo rezultatus į masyvą ER
[mx,IND]=max(OUTval);PerrV=sum(abs(sign(-Kval+IND)))/Nv;
for i1=1:Nv EV(Kval(i1),IND(i1))=EV(Kval(i1),IND(i1))+1; end;
% surašo EV
disp([ER,EV]);disp([PerrR,PerrV]) % kiek kuriai klasei priskyrė vektorių
% čia bus duomenų projektavimas į 2D erdvę DVIEJŲ daugiasluoksnio
% perceptrono paslėptų neuronų erdvėje  (dar prieš sigmoidinę funkciją)
net1 = nnt2ff(mima,{w11},{b11},{'purelin'});
%  net1 – tai vieno sluoksnio perceptronas su tiesine aktyvavimo funkcija
Ymok=sim(net1,Pmok);Yval=sim(net1,Pval); % DU nauji požymiai
PL=[' k.b.y.m.r.c.g.k+r+b+'];
% nurodomos spalvos ir žymekliai klasėms piešti
figure(FIGU);clf;hold on;
for j=1:Nklasiu z=find(Kmok==j); % z – tai j-tosios klasės mokymo vektoriai
plot(Ymok(1,z),Ymok(2,z),PL(1,2*j:2*j+1),'MarkerSize',8);end
% pavaizduojame mokymo duomenis dvimatėje erdvėje
figure(FIGU+1);clf;hold on;
for j=1:Nklasiu  z=find(Kval==j); % z – tai j-tosios klasės validavimo vekto-
riai plot(Yval(1,z),Yval(2,z),PL(1,2*j:2*j+1),'MarkerSize',8); end
% pavaizduojame testinius (validavimo) duomenis dvimatėje erdvėje
%  Dabar duomenis projektuosime į dvimatę (2D) erdvę
 % PAGRINDINIŲ KOMPONENČIŲ METODU
[u,d,v]=svd(cov(Pmok'));
% randame nuosavus vektorius ir nuosavas reikšmes.
tra=u*inv(sqrtm(d));
% suskaičiuojame duomenų pasukimo matricą „tra“
Ymok=(Pmok'*tra)';Yval=(Pval'*tra)';
% pasukame mokymo ir testavimo duomenis
figure(FIGU+2);clf;hold on;for j=1:Nklasiu  z=find(Kmok==j);
plot(Ymok(1,z),Ymok(2,z),PL(1,2*j:2*j+1),'MarkerSize',8);end;
% pavaizduojame mokymo duomenis dvimatėje erdvėje
figure(FIGU+3);clf;hold on;for j=1:Nklasiu  z=find(Kval==j);
plot(Yval(1,z),Yval(2,z),PL(1,2*j:2*j+1),'MarkerSize',8);end;
%  pavaizduojame validavimo (arba testinius) duomenis dvimatėje erdvėje.
```

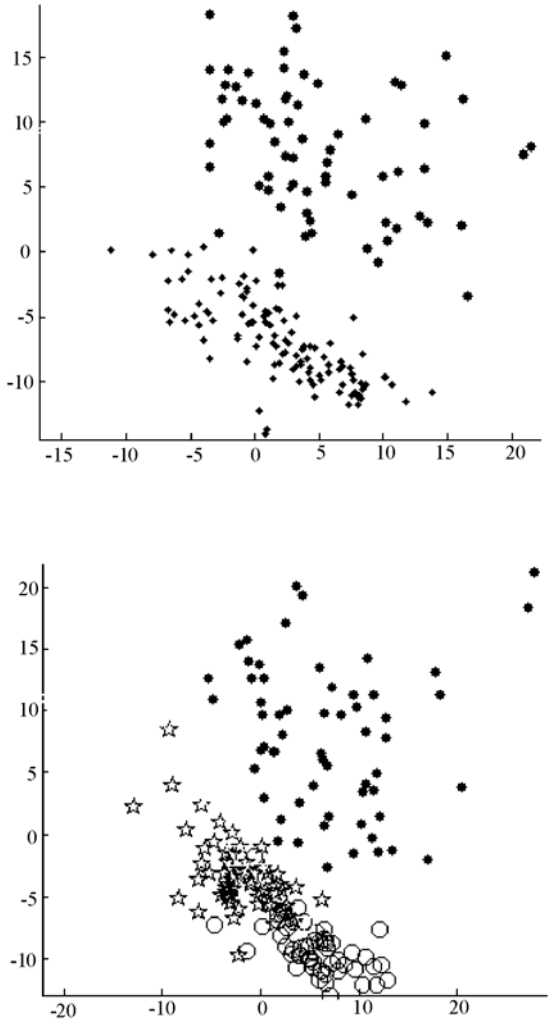
Štai sėkmingo eksperimento rezultatas (jis priklauso nuo atsitiktinių skaičių generatoriaus startinio skaičiaus).

```
>>clear;SS=24642;rand('state',SS);KIEKiter=[8 32];FIGU=5;Hidden=2;
                                    klases=[1 3 5 10];  Yeast_MLP_FEAkovas;
  TRAINLM: 0/64 epochs, mu = 0.001, SSE = 515.868.
  TRAINLM: 8/64 epochs, mu = 1e-005, SSE = 85.1044.
  TRAINLM: 16/64 epochs, mu = 1e-005, SSE = 67.7451.
  TRAINLM: 24/64 epochs, mu = 0.01, SSE = 66.6637.
  TRAINLM: 32/64 epochs, mu = 1e-006, SSE = 65.9845.
  TRAINLM: 40/64 epochs, mu = 0.1, SSE = 51.9992.
  TRAINLM: 48/64 epochs, mu = 0.001, SSE = 47.8448.
  TRAINLM: 56/64 epochs, mu = 0.0001, SSE = 46.9517.
  TRAINLM: 64/64 epochs, mu = 0.1, SSE = 44.7633.
  TRAINLM: Network error did not reach the error goal.
```

Štai mokymo ir testavimo klasifikavimo klaidų matricos, kur diagonalėse nurodyti **teisingai** klasifikuotų vektorių kiekiai, o nediagonalėse – neteisingai klasifikuotų vektorių kiekiai.

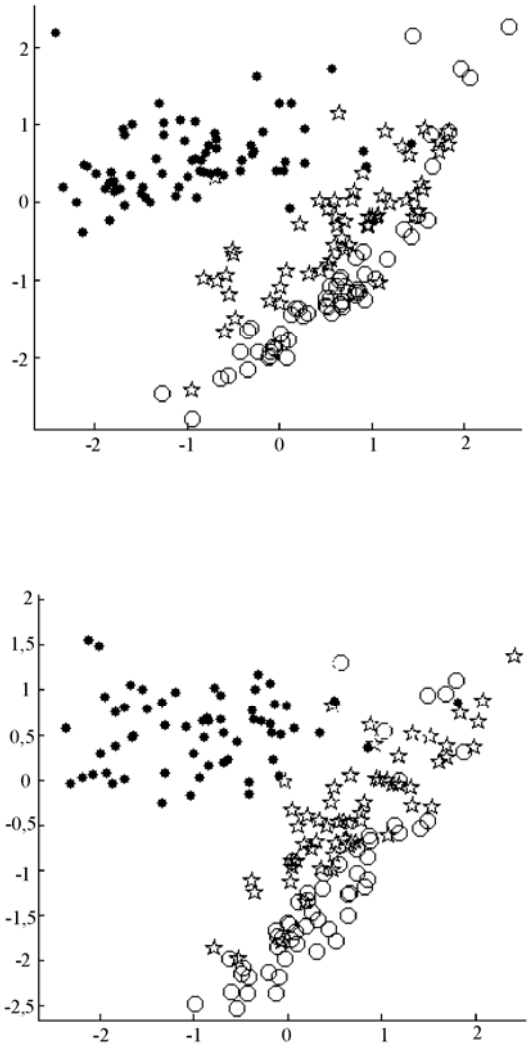
Mokymas				Testavimas			
54	0	0	0	**50**	7	2	0
2	**56**	2	1	4	**46**	5	0
0	1	**60**	0	1	3	**54**	1
0	0	0	**63**	0	5	0	**61**
		0,0251				0,1172	

44 paveiksle pateikiame mokymo ir testinių duomenų išsibarstymo dvimatėse erdvėse diagramas.



44 pav. Mokymo ir validavimo duomenų išsibarstymas dviem paslėptais neuronais suformuotoje erdvėje

Siekiant geriau įžiūrėti skirtumus, reikėtų kompiuterio ekrane paanalizuoti spalvotus grafikus, pateiktus minėtame KU tinklalapyje



45 pav. Mokymo ir validavimo duomenų išsibarstymas dviem pagrindinėmis komponentėmis suformuotoje erdvėje

Pateiktos diagramos rodo, kad naujai suformuotoje požymių erdvėje klasės mokymo duomenyse aiškiai skiriasi. Testiniuose duomenyse klasės skiriasi daug prasčiau. Tai normalu, nes mokymo vektorių kiekis nėra didelis, o požymių vis dėlto nėra labai mažai, – net dešimt. Jei jų būtų buvę 1500, testinių duomenų klasės beveik nesiskirtų. Užtat jei DsP nepakliūtų į nevykusį lokalinį minimumą, būtų nesunku pasiekti, kad mokymo duomenų klasės skirtųsi idealiai. Vadovėlio autorius siūlo tai pabandyti.

Darbas susideda iš dviejų–trijų dalių:

Pirma dalis. Požymių išskyrimas ir duomenų pavaizdavimas dvimatėje erdvėje (žr. anksčiau nurodytą programą).

Antra dalis. Klasifikavimo (atpažinimo) uždavinio sprendimas naudojant daugiasluoksnį perceptroną. Užduoties pavyzdys anksčiau jau buvo pateiktas. Šiame namų darbo etape reikia siekti geriau atskirti klases testuojant. Galima, o ir reikėtų, **didinti paslėptų neuronų kiekį**, tirti, kaip jų kiekis veikia rezultatą. Neturime pamiršti, kad egzistuoja lokalinių minimumų problema, todėl tyrimus reikia atlikti *multistart* režimu. Vienas iš tyrimo aspektų – gauti kuo mažesnį testinių klaidų dažnį. Sprendžiant šį uždavinį, naudojama ColoredGAUSnoise.m programėlė – reikėtų pasidaryti 10-mačius pseudovalidavimo duomenis, kurie leistų nustatyti, kiek mokymo epochų mokyti ir kurį iš daugelio *multistart* režimu atliktų mokymų pasirinkti. Paskui gautą rezultatą reikėtų klasifikuoti su testiniais duomenimis naudojant *Matlab'o* komandas, anksčiau surašytas po komentaro: % ČIA BUS KLASIFIKAVIMO KLAIDOS SKAIČIAVIMAS.

Reikia paminėti, kad dalijant duomenis į mokymo ir testinius sukuriamas nemenkas atsitiktinumų elementas. Todėl praktinėje duomenų analizėje eksperimentus su kiekvienu paslėptųjų neuronų kiekiu reikėtų pakartoti po keletą (sakykim, dešimtį) kartų, kiekvieną iš atsitiktinai turimų duomenų „sumaišius". Tai nemažai skaičiavimų reikalaujantis darbas.

Trečia dalis. Pritaikyti daugiasluoksnį perceptroną prognozavimo uždaviniui spręsti. Pavyzdys: pakartoti biržos indekso prognozavimą naudojant nebe vienasluoksnį, o daugiasluoksnį perceptroną. Prognozavimo uždavinyje išėjimo sluoksnyje esanti aktyvavimo funkcija turėtų būti tiesinė.

Darbo tikslus ir apimtį pasirenka studijuojantysis. Čia svarbu: paties studijuojančiojo apsisprendimas dėl namų darbo apimties ir tikslų neapibrėžtoje aplinkoje. Realioje duomenų analizėje taip pat niekas iš anksto, ką ir kaip daryti, nepasakys. Apsispręsti reikia pačiam. Duomenų analizė – tai kuriamasis darbas. Turime duomenis, ir dažnai net nežinom, ko ieškom. Nežinom netgi, ar koks nors dėsningumas šiuose duomenyse egzistuoja apskritai. Nežinom, ar pakankamas informatyvių požymių kiekis, ar pakankamai reprezentatyvūs duomenys.

8.5. Darbas Nr. 4. Savarankiškas duomenų tyrimas

Tikslas – studijuojančiojo iniciatyvos ir savarankiškumo ugdymas formuluojant sau keliamus uždavinius, šių uždavinių sprendimas, ataskaitos, rodančios gautų rezultatų naudą, parengimas. Šiame darbe studijuojantysis **pats pasirenka tiriamąjį uždavinį**, susiranda duomenis, suformuluoja tyrimo tikslus, bando argumentuoti, kodėl šiuos uždavinius reikia spręsti, darbo pradžioje bando prognozuoti galimas hipotezes apie duomenis, išvadas, nurodyti, kam šios išvados galėtų būti naudingos. Darbe galima (o galbūt netgi ir reikia) naudoti įgūdžius ir programas, įsisavintas atliekant ankstesnius namų darbus (Nr. 1, 2 ir 3).

Savarankiškame darbe spręstini uždaviniai

Aukščiau minėjome nemaža spręstinų klausimų. Tai galėtų būti:

a) klasifikavimo/prognozavimo klaidos mažėjimas augant mokymo duomenų kiekiui,

b) perceptrono persimokymo reiškinys,

c) pradinių svorių įtaka mokymosi greičiui klasifikavimo ir prognozavimo uždaviniuose,

d) optimalaus sustojimo nustatymas naudojant validavimo arba pseudovalidavimo duomenis,

e) klasifikavimo/prognozavimo tikslumo įvertinimas,

f) skaidymo į mokymo ir testinę imtis įtaka klasifikavimo/prognozavimo algoritmo kokybės įvertinimo tikslumui, duomenų randomizavimo (sumaišymo vienos klasės viduje) įtakos tyrimas,

g) laipsnio rodiklio nuostolio nuostolių funkcijoje (2.1) įtaka prognozavimo lygties tikslumui,

h) klasifikavimo/prognozavimo taisyklių tikslumo įvertinimų, paremtų testine arba/ir mokymo imtimi, dinamika augant mokymo imties tūriui,

i) reguliarizavimo nario įtaka regresijos lygties tikslumui,

j) reguliarizuoto Fišerio klasifikatoriaus tyrimas, kai mokymo duomenų yra nedaug,

k) kelių klasifikavimo taisyklių, panaudotų sprendžiant vieną ar kelis uždavinius, palyginimas,

l) kelių prognozavimo taisyklių, panaudotų sprendžiant vieną ar kelis realaus pasaulio uždavinius, palyginimas,

m) didelių nuokrypių įtaka klasifikavimo/prognozavimo lygties tikslumui,

n) daugiasluoksnio perceptrono inicializavimo ir paslėptų neuronų kiekio įtaka klasifikavimo/prognozavimo rezultatui,

o) vienasluoksniu ir daugiasluoksniu perceptronais paremtų klasifikatorių svorių augimo mokymo metu vaizdavimas, trokštamų išėjimų įtaka šiam reiškiniui,

p) *žirklių efekto* vaizdavimas daugiasluoksnių perceptronų atveju,

q) klasifikavimo taisyklės, paremtos sprendimų medžiais ir daugiasluoksniu perceptronu, palyginimas sprendžiant konkretų atpažinimo uždavinį (*Matlab'e* yra sprendimų medžių sudarymo taisyklės),

r) duomenų vaizdavimas dvimatėse erdvėse pagrindinių komponenčių metodu ir naudojant daugiasluoksnius perceptronus,

s) klasifikavimo/prognozavimo taisyklių, gautų dviem būdais: naudojant genetinius algoritmus arba daugiasluoksnius perceptronus, palyginimas,

t) klasifikavimo/prognozavimo taisyklių, gautų naudojant vienasluoksnį ir daugiasluoksnį perceptronus, palyginimas,

u) atraminiais vektoriais (http://www.csie.ntu.edu.tw/~cjlin/libsvm) ir vienasluoksniais perceptronais paremtų klasifikavimo taisyklių, palyginimas,

v) kelių informatyvių požymių sistemų išrinkimo algoritmų palyginimas,

w) klasifikavimo/prognozavimo lygties tikslumo priklausomybės nuo požymių kiekio nustatymas,

x) klasifikavimo/prognozavimo lygties prisiderinimo prie validavimo imties vaizdavimas,

y) Parzeno lango ir RBF neuroninių tinklų darbo tikslumo palyginimas,

z) genetinio algoritmo realizavimas mokant nesudėtingą DsP.

Realūs duomenys

Kalifornijos universiteto tinklalapyje "UCI Machine Learning Repository"

http://archive.ics.uci.edu/ml/
http://mlearn.ics.uci.edu/MLRepository.html

galima rasti daugybę eksperimentams tinkamų duomenų, kurie plačiai naudojami tiriant naujai kuriamus duomenų analinės algoritmus. Mokslinėje literatūroje nurodoma dar daugelis interneto adresų, kur galima rasti tyrimams reikalingų duomenų. Žemiau pateikiame tik dalį iš jų.

http://www.cs.cmu.edu/TextLearning/datasets.html.
http://www.uk.research.att.com/facedatabase.html
http://www.daviddlewis.com/resources/testcollections/reuters21578/
http://www.niaad.liacc.up.pt/statlog/datasets.html.
www.research.all.com./~vann/ocr/moist.
http://www.cs.toronto.edu
http://www.yeastgenome.org/
http://www.genome.wi.mit. edu/MPR/data set ALL AML.html.
http://www.nist.gov/srd/nistsd4.htm
http://peipa.essex.ac.uk/ipa/pix/faces/manchester/test-hard/
http://rvl1.ecn.purdue.edu/_aleix/aleix face DB.html
http://www.stjuderesearch.org/data/ALL1/.
www.cs.umn.edu/~jieping/Research.html.
http://www.research.att.com/lewis,
http://www.molbio.princeton.edu/colondata.
http://www-genome.wi.mit.edu/mpr/table_AML_ALL_samples.rtf.
http://microarray.princeton.edu/oncology/affydata/index.html.
http://yann.lecun.com/exdb/mnist/index.html
http://trec.nist.gov
http://rvl1.ecn.purdue.edu/~aleix/aleix face DB.html
http://peipa.essex.ac.uk/ipa/pix/faces/manchester/test-hard/
http://asi.insa-rouen.fr/~gloosli/coreVSsimple.html
http://www.stat.uni-muenchen.de/service/datenarchiv/welcome e.html.
http://www.cise.u..edu/~cjermain/RiversStocks.html.
http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/ai-repository/ai/areas/n% eural/bench/cmu
ftp://ftp.cs.columbia.edu/pub/CAVE/SLAM coil-20 coil-100/coil-100/coil-100.zip.
http://www-2.cs.cmu.edu/afs/cs/project/theo-4/text-learning/www/datasets.html.

BAIGIAMOSIOS PASTABOS

Knygoje aprašyta nemaža klasifikavimo, prognozavimo, požymių išskyrimo, požymių išrinkimo, duomenų blokinių sudarymo (klasterizavimo) metodų, paremtų statistiniais metodais ir dirbtiniais neuroniniais tinklais. Didžiausią dėmesį skyrėme ne tiek detalių nagrinėjimui (jos gana gerai pateiktos laisvai prieinamoje arba perkamoje standartinėje duomenų analizės programinėje įrangoje), kiek konceptų, glūdinčių šių metodų sudarymo procese, analizavimui, gautų rezultatų interpretavimui ir rezultatų patikimumo įvertinimui.

Norint tinkamai taikyti duomenų analizės metodus, reikia nemažo patyrimo, praktikos kalbantis su duomenis pateikusiais asmenimis, užsakovo sprendžiamų problemų supratimo, nuoširdaus įsigilinimo į užsakovo keliamas mokslines / praktines hipotezes, būdų, naudotų renkant bei registruojant empirinius duomenis, išmanymo. Ypatingą dėmesį reikia atkreipti į mokymo, validavimo, testavimo duomenų formavimą, papildomus (neregistruojamus) veiksnius, galėjusius atsirasti duomenų rinkimo metu.

Vienas iš būdų patikrinti, ar duomenys yra gerai surinkti, ar jie gerai suskirstyti į mokymo, validavimo ir testavimo duomenis, yra duomenų sumaišymas dar kartą ir minėto skirstymo pakartojimas. Jei išvados pasikartoja, tai rezultatais galima pasitikėti labiau. Tačiau jei testavimo duomenys kokiu nors būdu buvo naudoti daugiau kartų, tai duomenų analizės išvadų patikimumas mažėja. Vienas iš žinomų buvusios Sovietų Sąjungos mokslininkų prof. I. S. Pinskeris klasifikavimo uždaviniams spręsti pasiūlė metodą dar kartą panaudoti duomenis, klasių numerius sąmoningai parenkant atsitiktinai. Jei ir vėl rezultatai „bus geri“, vadinasi, atlikta duomenų analizė nekokybiška ir ja pasitikėti negalima. Tai paprastas, bet efektyvus būdas nesantiems duomenyse dėsningumams rasti.

LITERATŪRA

Peter Cabena, Pablo Hadjnian, Rolf Stadler, Jaap Verhees, Allesandro Zanasi, Discovering Data Mining: From Concept to Implementation, Prentice Hall, 1997, ISBN 0137439806.

Ronen Feldman, James Sanger, The Text Mining Handbook, Cambridge University Press, ISBN 9780521836579.

Pang-Ning Tan, Michael Steinbach and Vipin Kumar, Introduction to Data Mining, 2005, ISBN 0-321-32136-7.

Galit Shmueli, Nitin R. Patel and Peter C. Bruce, Data Mining for Business Intelligence, 2006, ISBN 0-470-08485-5.

Richard O. Duda, Peter E. Hart, David G. Stork, Pattern Classification, Wiley Interscience, ISBN 0-471-05669-3.

Phiroz Bhagat, Pattern Recognition in Industry, Elsevier, ISBN 0-08-044538-1.

Ian Witten and Eibe Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, 2000, ISBN 1-55860-552-5.

Mark F. Hornick, Erik Marcade, Sunil Venkayala, Java Data Mining: Strategy, Standard, And Practice: A Practical Guide for Architecture, Design, And Implementation (Broché):

Yike Guo and Robert Grossman (editors), High Performance Data Mining: Scaling Algorithms, Applications and Systems, Kluwer Academic Publishers, 1999.

Trevor Hastie, Robert Tibshirani and Jerome Friedman, The Elements of Statistical Learning, Springer, 2001, ISBN 0387952845.

D. Hand, H. Mannila, P. Smyth, Principles of Data Mining. MIT Press, Cambridge, MA., 2001.

Kantardzic Mehmed, Data Mining: Concepts, Models, Methods, and Algorithms. John Wiley & Sons, 2003.

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

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

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.

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

A. Sakalauskas, Statistika su Statistika, Vilnius, 1999.

A. Budrevičius, Prognozavimo metodai vadyboje: nuotolinis kursas, parengta 2002 m., prieiga internetu: http://distance.nsc.vu.lt.

V. Čekanavičius, G. Murauskas. Statistika ir jos taikymai, I, Vilnius, TEV, 2000.

G. Murauskas, Statistika ir jos taikymai, II, Vilnius, TEV, 2002.

V. Rudzkienė, Socialinė statistika, Vilnius: MRU leidybos centras, 2005.

V. Dikčius, Marketingo tyrimai. Teorija ir praktika, Vilniaus vadybos kolegija, Vilnius, 2003.

Statistika ir duomenų analizės programinė įranga, Distancinio mokymo kursas. Projekto vadovas V. Janilionis, Kaunas: KTU, 1999, http://fmf.ktu.lt/janil/stat1.htm.

DALYKINĖ RODYKLĖ

lokalinis minimumas
Mahalanobio atstumas
maksimalaus tarpo klasifikatorius
minimaksinė regresija
minimalių kvadratų metodas
mokymo duomenys
mokymo vektorių kvantavimo
neuroniniai tinklai
nuostolių funkcija
optimalus sustojimas
pagrindinių komponenčių metodas
Parzeno lango klasifikatorius
požymių transformacijos
požymių išrinkimas
požymių išskyrimas
požymių kiekis
prognozavimas
prognozavimo paklaida
prognozavimo tikslumas
regresijos lygtis
reguliarizuotas klasifikatorius
reguliarizuota regresija
robastinė regresija
skiriamosios kreivės
skiriamasis paviršius
skiriančioji plokštuma

Klaipėdos universiteto leidykla

Šarūnas Raudys. ŽINIŲ IŠGAVIMAS IŠ DUOMENŲ

Klaipėda, 2008

SL 1335. 2008 09 22. Apimtis 10,5 sąl. sp. l. Tiražas 200 egz.
Klaipėdos universiteto leidykla, Herkaus Manto g. 84, LT-92294 Klaipėda
Tel. (8~46) 398 891, el. paštas: leidykla@ku.lt
Spausdino spaustuvė „Druka“, Šilutės pl. 79, LT-94101 Klaipėda