VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Magistro baigiamasis darbas

Klaidų kainoms jautrūs klasifikavimo algoritmai

Cost-sensitive classification algorithms

Atliko: 2 kurso 1 grupės studentai
Albertas Gimbutas ..... (parašas)
Gražina Laurinavičiūtė ..... (parašas)

Darbo vadovas:
prof. hab. dr. Šarūnas Raudys ..... (parašas)

Recenzentas:
dr. Linas Būtėnas ..... (parašas)

Vilnius
2013

Santrauka

Praktiniuose taikymuose, kaip kad medicinos diagnostikoje ir veidų atpažinime, pripažintas santykinis klasifikavimo klaidų kainų, priklausančių nuo tikrosios ir priskirtosios klasių, skirtumas. Šis darbas apima klaidų kainoms jautraus hibridinio klasifikatoriaus, sudaryto iš sprendimo medžio ir daugiasluoksnio perceptrono, kūrimą ir analizę. Hibridiniam klasifikatoriui konstruoti buvo realizuoti kainoms jautrių *C4.5* sprendimo medžių variantai bei klaidų kainoms jautrus daugiasluoksnis perceptronas, jiems kombinuoti panaudota *Banerjee* metodika. Atlikti eksperimentai su realaus pasaulio ir sintetiniais duomenimis leidžia teigti, kad hibridinis klasifikatorius gali būti naudingas, t. y. sumažinti jį inicializavusio sprendimo medžio klasifikavimo kainą ir pasiekti šį rezultatą greičiau nei atsitiktiniais svoriais inicializuotas daugiasluoksnis perceptronas.

Raktiniai žodžiai: nesubalansuoti duomenys, jautrus kainoms, klasifikavimas, dirbtiniai neuroniniai tinklai, sprendimų medis, hibridinis, daugiasluoknis perceptronas.

Summary

In practical applications including medical diagnosis and face detection it has been admitted that classification errors might differ in relative cost depending on the real and predicted classes. The work comprises implementation and analysis of a cost-sensitive hybrid classifier, consisting of a decision tree and a multi-layer perceptron. The hybrid classifier was constructed using several varieties of a cost-sensitive *C4.5* decision tree and a cost-sensitive multi-layer perceptron, which were combined using the *Banerjee* method. Conducted experiments with real world and synthetic data allow to conclude that the hybrid method might be useful, namely, decrease the misclassification error cost of the initializing tree and achieve this result faster than a randomly initialized multi layer perceptron.

Keywords: imbalanced dataset, cost-sensitive, classification, artificial neural network, decision tree, hybrid, multilayer perceptron.

Turinys

Terminai

1. Aktyvavimo funkcija (angl. activation function)
2. ANT - klasikinis atgalinio perdavimo neuroninis tinklas (angl. back-propagation neural network)
3. Apriorinė tikimybė (angl. a priori probability)
4. Baudos funkcija (angl. penalty function)
5. Bendri klasifikavimo nuostoliai (angl. cumulative cost)
6. DNT - dirbtinis neuroninis tinklas (angl. artificial neural network)
7. DSP - daugiasluoksnis perceptronas (angl. multilayer perceptron)
8. Gaubiamasis metodas (angl. wrapper method)
9. Kainoms jautrus (angl. cost-sensitive)
10. Klaidos skidimo atgal metodas (angl. error-backpropagation)
11. Klasifikatorius - klasifikavimo metodas
12. Klasifikavimo klaidos kaina (angl. classification error cost)
13. Klasifikavimo skirstinys (plokštuma) - požymių erdvės taškų priklausymo skirtingoms klasėms vizualizacija
14. Nuosekli kainų matrica (angl. consistent cost matrix)
15. Nuostolių funkcija (angl. cost or loss function)
16. Paporinės klasifikavimo kainos (angl. pair-wise costs)
17. Pasirinkto laiko algoritmai (angl. anytime algorithms)
18. SM - sprendimo medis (angl. decision tree)
19. Tiesioginis perdavimas (angl. feed-forward)

Įvadas

Klasifikavimo uždavinį naudojant induktyvaus pobūdžio mokymąsi plačiai mėginta spręsti orientuojantis į klasifikavimo klaidų skaičiaus minimizavimą. Panaudojant aibę mokymosi duomenų - vektorių, kuriems įvardyta priklausomybė tam tikrai klasei, - konstruojamas algoritmas, besistengiantis kuo didesnį skaičių elementų priskirti teisingai klasei.

Kita vertus, realiuose taikymuose klasifikatoriaus daromos klaidos gali turėti skirtingą kainą, atsižvelgiant į tai, kurios klasės elementas kuriai kitai klasei priskirtas. Pavyzdžiui, diagnozuojant pacientui cukrinį diabetą labai svarbu ligonio nepalaikyti sveiku, kad būtų kuo greičiau pradėtas atitinkamas gydymas ir nerizikuojama būklės komplikacijomis. Dėl to klasifikatoriui, pagal paciento duomenis priimančiam sprendimą, galėtų būti žinoma, jog yra santykinai gerokai brangiau suklysti suklasifikuojant ligonį kaip sveiką nei atvirkščiai. Kitokio pobūdžio taikymas būtų telekomunikacijų įmonės, turinčios atlikti savo tinklo priežiūrą, trikio vietos nustatymas. Jeigu gedimai galimi skirtingos kainos elementuose, tai juos neteisingai nustačius gausis, kad už pakaitinį elementą teks mokėti daugiau arba mažiau nei reikia iš tikrųjų, be to, atliktas keitimas gali neišspręsti problemos ir teks patirti papildomų nuostolių atliekant diagnostiką ir pakartotinį keitimo bandymą. Dėl to klaidų kainos šiame kontekste turėtų atspindėti objektyvias finansines pasekmes. Apskritai ir kitų taikymo sričių reikėtų ieškoti ten, kur galimas įvarių anomalijų aptikimo poreikis: įsilaužimo į tinklą aptikimas, modernių augalų gamyba, programinės įrangos kokybės įvertinimas, teksto klasifikavimas, sukčiavimų aptikimas, apgavikų skambučių nustatymas, naftos išsiliejimo dėmių aptikimas palydovo paveikslėliuose ir kt. [SWKK09].

Tokiais atvejais prasminga konstruoti klasifikatorių, kurio tikslas - minimizuoti bendrą klaidų kainą, pvz., persidengiančių klasių atveju padarant daugiau pigesnių klaidų.

Įvertinus skirtingų klasifikatorių savybes, šiame darbe pasirinkta nagrinėti dviejų tipų klasifikatorius: dirbtinius neuroninius tinklus ir sprendimo medžius. Sprendimo medis yra greitai apmokomas klasifikatorius, suteikiantis dalykinės srities ekspertui lengvai suprantamą simbolinę sprendimo priėmimo reprezentaciją. Kita vertus, jo klasifikavimo tikslumas nėra labai aukštas. Dirbtinių neuroninių tinklų klasifikavimo tikslumas aukštesnis netgi esant triukšmui duomenyse, tačiau jų apmokymas lėtas ir nėra taisyklių, kurios nusakytų geriausią būdą daugiasluoksnio perceptrono architektūrai apibrėžti.

Kaip matyti, abiejų tipų klasifikatoriai turi savo stipriųjų ir silpnųjų pusių. Ga-

lima pastebėti, kad vieno iš jų privalumai kompensuoja kito trūkumus. Ši idėja buvo panaudota literatūroje siekiant sukurti hibridinį klasifikatorių, kuriame sprendimo medis naudojamas neuroniniam tinklui inicializuoti [Rou06], [Ban97].

Dėl aukščiau išvardytų realaus gyvenimo situacijų, pirmiausia susipažinta su galimybėmis įvesti jautrumą klasifikavimo klaidų kainoms kiekviename iš šių klasifikatorių pavieniui. Išsiaiškinta, kad sprendimo medžių atveju išskiriami pavienio arba daugelio kombinuojamų medžių metodai. Vieno medžio metodai apmokomi greičiau, tačiau jie pralaimi daugelio medžių metodams pagal klaidų kainos įverčius. Neuroninių tinklų srityje kainos įvedamos iš esmės keičiant daugiasluoksnio perceptrono nuostolių funkciją.

Šiame darbe taip pat siekiama sukombinuoti sprendimo medžius ir dirbtinius neuroninius tinklus su tikslu panaudoti jų stipriąsias savybes, tačiau skirtingai nei ankstesni metodai, norima užtikrinti, kad gautoji kombinacija būtų jautri klasifikavimo klaidų kainoms.

Keliama tokia **hipotezė**:

Įmanoma panaudoti pavienį sprendimo medį, kurio apmokymas greitas, tačiau jautrumas kainoms silpnas, inicializuoti tinklui, kurio architektūra ir parametrai nežinomi, be to, apmokymas lėtas, tačiau jautrumas kainoms geras, kad būtų gautas greitai apmokomas gero jautrumo kainoms klasifikatorius.

Kaip pagrindas kombinacijai pasirinktas metodas [Ban97]. Jo aukšto lygmens schema, modifikuota pridedant jautrumą klaidų kainoms, būtų tokia:

1. Apmokyti kainoms jautrų sprendimo medį;
2. Inicializuoti tinklo architektūrą panaudojant gautą sprendimo medį;
3. Papildomai apmokyti gautąjį neuroninį tinklą, panaudojant kainoms jautrią nuostolių funkciją.

Tikimasi, kad papildomas neuroninio tinklo mokymas truks trumpiau nei jo mokymas be pasiūlytos inicializacijos. Taip pat, kad rezultatas bus geresnis už pavienio sprendimo medžio, naudojamo tinklui inicializuoti, pagal vidutinę klasifikavimo klaidų kainą.

Siekiant patikrinti iškeltą hipotezę, reikia atlikti šiuos uždavinius:

1. Išanalizuoti kainų įvedimo į sprendimo medžius ir dirbtinius neuroninius tinklus pavieniui alternatyvas, remiantis literatūra;
2. Realizuoti tinkamiausias klaidų kainoms jautrių sprendimo medžių ir dirbtinių neuroninių tinklų alterantyvas ir palyginti jas savarankiškais eksperimentais;

3. Eksperimentiškai palyginti įvairių sprendimo medžių alternatyvų kombinacijas su dirbtinių neuroninių tinklų alternatyvomis.

Šiame darbe atlikta:

1. Susipažinta su kainų įvedimo alternatyvomis;

2. Dėl didelės jų įvairovės atliktas aukšto lygmens palyginimas remiantis literatūra;

3. Python aplinkoje realizuoti kainoms jautrūs sprendimo medžių ir dirbtinių neuroninių tinklų variantai bei jų kombinavimo būdas;

4. Atlikti eksperimentai, kuriais pademonstruotas realizuotų klasifikatorių veikimas su sintetiniais ir realaus pasaulio duomenimis ir patvirtinta, kad hibridinis klasifikatorius gali pasiteisinti pagerindamas tiek jį inicializavusį sprendimo medį pagal klaidų kainą, tiek atsitiktiniais svoriais inicializuotą DSP pagal iteracijų skaičių.

Darbas remiasi šiomis prielaidomis:

1. Mokymosi ir testiniuose duomenyse nėra trūkstamų reikšmių. Kitaip sakant, duomenys surinkti kruopščiai, tiksliai, todėl matavimų paklaidos bei triukšmas buvo nereikšmingi. Trūkstamų reikšmių problema šiame darbe nenagrinėjama, nes skirtingiems klasifikavimo metodams jos sukelia skirtingų problemų, o tai reikalauja plačios analizės.

2. Duomenų vektoriai turi saugoti informaciją, kuri būtų susieta su atitinkamo vektoriaus klase. Priešingu atveju klasės nustatymas būtų tiesiog spėliojimas.

3. Mokymo duomenų aibė turi talpinti bent po vieną kiekvienos klasės atstovą, kad būtų įmanoma atpažinti visas klases.

4. Eksperimentams naudojamų realaus pasaulio duomenų klasių dažniai nekeičiami. Papildomų pastangų surinkti daugiau rečiau pasitaikančių klasių duomenų nededama.

Šis darbas organizuotas taip. Pradedama nuo klasifikavimo uždavinio pristatymo pirmame skyriuje. Toliau pateikiamos kainoms jautrių dirbtinių neuroninių tinklų ir sprendimo medžių apžvalgos antrame ir trečiame skyriuose. Ketvirtame skyriuje pateikiama šių klasifikatorių apjungimo būdų apžvalga. Algoritmų realizacijos detalės aptariamos penktame skyriuje. Atlikti eksperimentai pateikti šeštame skyriuje. Paskutiniame skyriuje pateikiamos išvados.

Šio darbo veiklas autoriai pasidalijo taip: A. Gimbutas buvo atsakingas už daugiasluoksnio perceptrono jautrumo kainoms apžvalgą ir metodo įgyvendinimą, o G. Laurinavičiūtė - analogiškai už sprendimo medžių dalį. A. Gimbutas realizavo hibridizacijos metodą, o G. Laurinavičiūtė atliko egzistuojančių hibridų apžvalgą. Eksperimentus abu autoriai formulavo ir išvadas darė kartu. A. Gimbutas realizavo eksperimentinę infrastruktūrą MIF superkompiuteriui.

1. Klasifikavimo uždavinys

Klasifikavimo problema - kaip pagal turimus duomenis[1] $\boldsymbol{D}$, kurie susideda iš n duomenų taškų (vektorių[2]) $\boldsymbol{x}_i \in \mathbb{R}^p$, $i = 1, ..., n$, bei žinomas jų klases $class(\boldsymbol{x_i}) \in \{1, 2, ..., m\}$, $i = 1, ..., n$, sudaryti metodą, kuris galėtų nustatyti vektoriaus $\boldsymbol{x'} \in \mathbb{R}^p$ klasę.

Nagrinėjant realaus pasaulio duomenis iškyla dvi problemos:

- klasių pasitaikymo dažniai skiriasi. Pavyzdžiui, geologiniuose matavimuose vienos uolienos aptinkamos dažniau nei kitos.
- klasių klasifikavimo kainos skiriasi. Pavyzdžiui, medicinos diagnostikoje palaikyti sergantį pacientą sveiku *"kainuoja"* kur kas brangiau, nei sveiką pacientą - sergančiu.

Dauguma standartinių klasifikavimo metodų nekorektiškai traktuoja šias problemas, nes daro prielaidą, kad $\boldsymbol{D}$ klasės turi vienodą pasitaikymo dažnį (duomenys subalansuoti) bei kad klasifikavimo klaidų kainos yra lygios [SKW07]. Nesubalansuotų duomenų problema kyla tiek DNT, tiek SM klasifikavimo metodams [Mar00]. Šių problemų sprendimai nagrinėjami 2 bei 3 poskyriuose.

Klasifikavimo klaidų kainų matrica (arba tiesiog kainų matrica) talpina **paporines** klasifikavimo kainas, kurios nurodo, kaip brangu turimos klasės i vektorių priskirti prognozuojamai klasei j. Kainų matricą žymėsime C, o jos elementus $C[i, j]$. Kai klasifikatorius teisingai nustato duomenų vektoriaus klasę, tada klasifikavimo klaidos kaina lygi nuliui. Taigi laikydami, kad skirtingų klasių skaičius yra m, o kainų įverčiai c_{ij}, gauname $m \times m$ dydžio kainų matricą:

$$C[i,j] = \begin{cases} c_{ij}, & i \neq j \\ 0, & i = j, \end{cases} \quad i, j = 1, 2, ..., m \; . \tag{1}$$

Kai kainų matricos visi elementai, išskyrus nulius įstrižainėje, yra vienetai, turime įprastą klaidų *skaičių* minimizuoti siekiantį klasifikavimą.

Praktiniai tyrimai rodo, kad kai kurie nesubalansuotų klasių įvertinimo metodai korektiškai veikia tik nagrinėjant dviejų klasių duomenis [ZLL06], todėl šiame darbe naudojamų klasifikavimo metodų savybės testuojamos ir su trijų bei daugiau klasių duomenimis.

Nors atskyrėme *duomenų subalansavimo* bei *klasifikavimo kainų įvedimo* problemas, jos yra labai tampriai susijusios. Galima įžvelgti tokį šių problemų sąryšį:

[1] Pusjuodės didžiosios raidės žymi matricas.
[2] Pusjuodės mažosios raidės žymi vektorius.

duomenų subalansavimo problema yra *klasifikavimo kainų įvedimo* problemos poaibis, nes duomenų subalansavimui tereikia kiekvienos klasės kainą padauginti iš jos apriorinės tikimybės. Kita vertus, *duomenų subalansavimo* problema yra nepriklausoma, nes ji apima metodus, kaip galima tiksliai įvertinti klasės apriorinę tikimybę. Nagrinėdami sprendimų medžius, šias problemas apjungsime ir nagrinėsime tik *klasifikavimo kainų įvedimo* problemą. O daugiasluoksnio perceptrono atveju, šias problemas atskirsime, nes literatūroje yra aptinkamos skirtingos klasių apriorinių tikimybių įvertinimo metodikos.

2. Dirbtinių neuroninių tinklų apžvalga

2.1. Bendrieji dirbtinių neuroninių tinklų principai

2.1.1. Perceptronas

Perceptronas[3] yra iteraciškai apmokomas tiesinis klasifikatorius. Įvedami žymėjimai: duomenų vektorių $\boldsymbol{x} = (x_1, x_2, ..., x_p)$ praplečiame vienetu, $\boldsymbol{z} = (1, x_1, x_2, ..., x_p)$, perceptrono įėjimų svorių vektorių $\boldsymbol{w} = (w_1, w_2, ..., w_p)$ praplečiame w_0, $\boldsymbol{v} = (w_0, w_1, w_2, ..., w_p)$. Naudosime sigmoidinę glodninimo funkciją:

$$f(x) = \frac{1}{1 + e^{-x}} \ . \tag{2}$$

Tada i-tajam duomenų vektoriui perceptrono išėjimas apskaičiuojamas taip:

$$o_i = f(\boldsymbol{x}_i \boldsymbol{w}^T + w_0) = f(\boldsymbol{z}_i \boldsymbol{v}^T) \ . \tag{3}$$

i-tajam duomenų vektoriui nuostoliai, kurie minimizuojami pagal svorius $\boldsymbol{v}$:

$$nuostoliai_i = (t_i - o_i)^2 \ . \tag{4}$$

Čia: t_i - i-tojo vektoriaus trokštamas išėjimas, atitinkantis jo klasę. Dviejų klasių atveju vienos klasės trokštamas išėjimas būna 1, o kitos 0. Nors keičiant trokštamų išėjimų santykį atsiranda įdomūs efektai, pavyzdžiui, kinta mokymosi greitis, jie šiame darbe nenagrinėjami.

Nuostolių funkcija n ilgio duomenų aibei:

$$nuostoliai = \frac{1}{n} \sum_{i=1}^{n} (t_i - o_i)^2 \ . \tag{5}$$

Nuostolių minimizavimas atliekamas taikant greičiausio nusileidimo (angl. steepset descent) metodą. Kitaip sakant, modifikuojant svorių vektorių priešinga *nuostoliai* gradientui pagal $\boldsymbol{v}$ kryptimi:

$$\boldsymbol{v}_{new} = \boldsymbol{v}_{old} - \eta \frac{\delta nuostoliai}{\delta \boldsymbol{v}} \ . \tag{6}$$

Čia: η - mokymo žingsnis, lemiantis $|\boldsymbol{v}|$ augimo greitį, pvz.: $\eta = 0{,}25$.

[3]Perceptronu vadinsime McCulloch-Pitts neuroną su sigmoidine aktyvavimo funkcija.

Sigmoidinės funkcijos (2) išvestinė:

$$f'(x) = f(x)(1 - f(x)) \ . \tag{7}$$

Nuostolių funkcijos (5) išvestinė pagal $\boldsymbol{v}$:

$$\frac{\delta nuostoliai}{\delta \boldsymbol{v}} = \frac{1}{n}\sum_{i=1}^{n} \frac{\delta (t_i - o_i)^2}{\delta (t_i - o_i)} \frac{\delta (t_i - o_i)}{\delta o_i} \frac{\delta o_i}{\delta \boldsymbol{z}_i \boldsymbol{v}^T} \frac{\delta \boldsymbol{z}_i \boldsymbol{v}^T}{\delta \boldsymbol{v}} \ . \tag{8}$$

Išskleidus gaunama:

$$\frac{\delta nuostoliai}{\delta \boldsymbol{v}} = \frac{1}{n}\sum_{i=1}^{n} 2(t_i - o_i)(-1) f'(\boldsymbol{z}_i \boldsymbol{v}^T)\boldsymbol{z}_i = -\frac{2}{n}\sum_{i=1}^{n} (t_i - o_i) o_i (1 - o_i)\boldsymbol{z}_i \ . \tag{9}$$

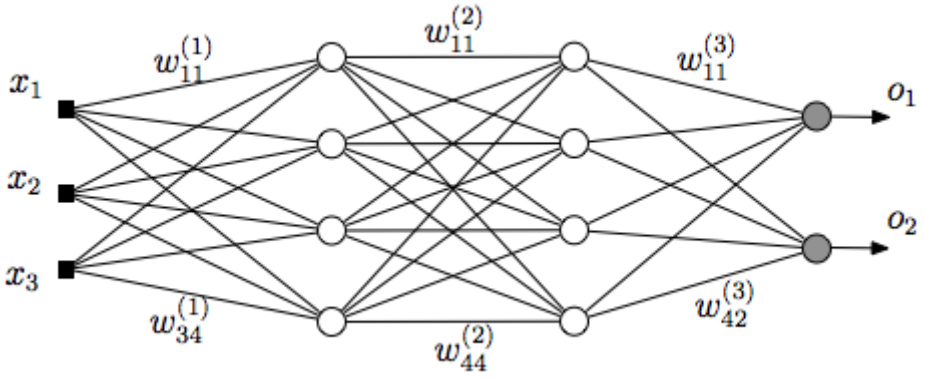
Perceptronas pradedamas mokyti nuo nulinių svorių $\boldsymbol{v}^{(0)} = (0, ..., 0)$. Iteraciškai mokomas pagal (6) minimizuoja nuostolių funkciją (5), taip išmoksta nustatyti pateiktojo duomenų vektoriaus $\boldsymbol{x'} \in \mathbb{R}^p$ klasę. Klasės nustatymas vyksta apskaičiuojant perceptrono išėjimą (3), panaudojant mokymo metu gautus svorius $\boldsymbol{v}_{min}$ bei pateiktąjį duomenų vektorių $\boldsymbol{x'}$. Jeigu gautasis išėjimas didesnis nei 0,5, tada $\boldsymbol{x'}$ priskiriama klasė, kurios trokštamas išėjimas buvo 1, priešingu atveju priskiriama klasė, kurios trokštamas išėjimas buvo 0.

2.1.2. Dirbtiniai neuroniniai tinklai

Dirbtinis neuroninis tinklas - tinklas, sudarytas iš apjungtų perceptronų. Perceptronus galima apjungti įvairiais būdais, pavyzdžiui, dirbtinis neuronas gali turėti rekurentinę jungtį į save. Tačiau kuo sudėtingesnė tinklo sandara, tuo komplikuočiau jį apmokyti. Praktiniuose taikymuose dėl savo paprastumo dažniausiai taikomas daugiasluoksnis perceptronas (angl. multilayer perceptron). Tai yra **tiesioginio perdavimo** (angl. feed-forward) neuroninis tinklas, apmokomas **klaidos sklidimo atgal** (angl. error back-propagation) metodu.

Daugiasluoksnis perceptronas susideda iš sluoksnių, kuriuose gali būti skirtingas perceptronų skaičius. Visi gretimų sluoksnių neuronai yra sujungiami. Daugiasluoksnis perceptronas turi vieną įėjimo sluoksnį (žr. 1 pav. juodai užtušuoti elementai), vieną arba daugiau paslėptų sluoksnių (žr. 1 pav. baltai užtušuoti elementai) bei vieną išėjimo sluoksnį (žr. 1 pav. pilkai užtušuoti elementai). Įėjimo sluoksnis teoriškai nėra sluoksnis, nes susideda ne iš perceptronų, o iš duomenų vektoriaus $\boldsymbol{x}$ komponenčių (praktikoje priimta vadinti sluoksniu). **Tiesioginio perdavimo** metodas nusako, kad paslėptų sluoksnių perceptronai apskaičiuoja išėjimus (žr. (3)

formulę) bei gautas reikšmes perduoda sekančiam sluoksniui. Taip reikšmės sklinda tik viena kryptimi, kol patenka į išėjimo sluoksnį, kuriame yra apskaičiuojamas daugiasluoksnio perceptrono išėjimas.



1 pav.: Daugiasluoksnis perceptronas, kurio sandara [3, 4, 4, 2]

Išėjimo sluoksnis turi tiek perceptronų, kiek $\boldsymbol{D}$ turi klasių. Kiekvienam išėjimo sluoksnio perceptronui yra priskiriama skirtinga klasė. Jeigu duomenų vektoriaus $\boldsymbol{x}$ klasė sutampa su išėjimo sluoksnio perceptronui priskirta klase, jo trokštamas išėjimas būna 1, priešingu atveju 0. Taip išgaunamas efektas, kad kiekvienas išėjimo sluoksnio perceptronas stengiasi atpažinti jam priskirtą klasę. Apmokytas daugiasluoksnis perceptronas pateiktajam duomenų vektoriui $\boldsymbol{x'}$ grąžina m reikšmių iš $[0, 1]$ intervalo (kiekvienai $\boldsymbol{D}$ klasei po vieną reikšmę); kuri reikšmė būna didžiausia, ta klasė ir priskiriama duomenų vektoriui $\boldsymbol{x'}$.

Išėjimo sluoksnio j-tojo perceptrono trokštamas išėjimas randamas pagal i-tojo mokymo vektoriaus klasę:

$$t_{ij} = \begin{cases} 1, & class(\boldsymbol{x}_i) = j \\ 0, & class(\boldsymbol{x}_i) \neq j \end{cases} . \tag{10}$$

Išėjimo sluoksnio j-tojo perceptrono nuostolių funkcija atitinka (5) išraišką. Daugiasluoksnio perceptrono, išėjimo sluoksnyje turinčio m perceptronų, nuostolių funkcija:

$$nuostoliai = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} (t_{ij} - o_{ij})^2 . \tag{11}$$

Čia: o_{ij} - j-tojo išėjimo sluoksnio perceptrono išėjimo reikšmė i-tajam duomenų vektoriui.

Paslėpto sluoksnio perceptronų nuostolių funkcija nėra apibrėžta, nes nežinomas trokštamas išėjimas. Tačiau kaip rasti paslėpto sluoksnio perceptrono nuosto-

lių funkcijos gradientą, nusako **klaidos sklidimo atgal** metodas. Pirmiausia išskleiskime nuostolių funkciją išėjimo sluoksnio j-tajam perceptronui, bei įsiveskime perceptrono lokalaus gradiento pasižymėjimą $\xi^{(j)}$:

$$\frac{\delta nuostoliai}{\boldsymbol{v}_j} = -\frac{2}{n}\sum_{i=1}^{n}(t_{ij} - o_{ij})o'_{ij}\boldsymbol{z}_i = -\frac{2}{n}\sum_{i=1}^{n}\xi_i^{(j)}\boldsymbol{z}_i \ . \tag{12}$$

Klaidos sklidimo atgal metodas nusako, kad paslėpto sluoksnio k-ajam perceptronui lokalaus gradiento $\xi^{(k)}$ forma yra [Hay04]:

$$\xi^{(k)} = \sum_{i=1}^{n} o'_{ik}\sum_{j\in P_k}\xi_i^{(j)}w_{kj} \ . \tag{13}$$

Čia:

- P_k - aibė perceptronų, kurie yra prijungti prie k-tojo perceptrono išėjimo,
- w_{kj} - k-tojo bei j-tojo perceptronų jungties svoris.

Taigi k-tojo paslėpto sluoksnio perceptrono nuostolių pataisa apskaičiuojama:

$$\frac{\delta nuostoliai}{\delta \boldsymbol{v}_k} = -\frac{2}{n}\xi_k\boldsymbol{z}_k = -\frac{2}{n}\boldsymbol{z}_k\sum_{i=1}^{n} o'_{ik}\sum_{j\in P_k}\xi_i^{(j)}w_{kj} \ . \tag{14}$$

Daugiasluoksnis perceptronas mokomas epochomis (viena epocha atitinka perceptrono mokymo vieną iteraciją). Epochos metu atliekamų veiksmų seka:

1. **Tiesioginio perdavimo** metodu yra apskaičiuojamos tinklo išėjimų reikšmės. Kitaip sakant, mokymo duomenų vektorius patenka į pirmąjį paslėptą sluoksnį, jame kiekvienas perceptronas atlieka (3) duomenų transformaciją ir gautą rezultatą perduoda sekančiam sluoksniui. Ši veiksmų seka atliekama kiekvienam sluoksniui, kol yra apskaičiuojamos tinklo išėjimų reikšmės (išėjimo sluoksnio perceptronų išėjimai).

2. **Klaidos sklidimo atgal** metodu yra apskaičiuojamos svorių korekcijos. Kitaip sakant, išėjimo sluoksnyje kiekvienam perceptronui yra apskaičiuojamos lokalių išvestinių reikšmės, atliekamos svorių korekcijos. Lokalių išvestinių reikšmės perduodamos paskutiniam paslėptam sluoksniui ir atliekamos svorių korekcijos. Ši veiksmų seka atliekama visiems paslėptiems sluoksniams. Taip pakoreguojami visi tinklo svoriai.

Mokymas gali būti **nuoseklus** (angl. sequential) arba **totalinis** (angl. batch). Nuoseklaus mokymo atveju paeiliui su kiekvienu mokymo duomenų vektoriumi

atliekami visi epochos žingsniai. Totalinio mokymo atveju - kiekvienas epochos etapas atliekamas su visais mokymo duomenų vektoriais iš karto ir sukaupta rezultatų matrica perduodama sekančiam epochos etapui. Šio darbo autoriai pasirinko **totalinį** mokymo būdą, nes programuojant masyvų programavimo paradigma, algoritmo veiksmų seka yra lakoniškesnė.

Yra įvairių metodų, kaip galima inicializuoti daugiasluoksnį perceptroną, bet nei vienas metodas nėra pagrįstai geriausias [Rou06]. Todėl tinkami metodai, kurie inicializuoja perceptronus su nedideliais, bet skirtingais pradiniais svoriais. Svoriai turi būti nedideli, kad perceptronai gebėtų efektyviai mokytis. Svoriai turi būti skirtingi, nes sutapus/supanašėjus dviejų perceptronų svoriams, šie perceptronai atliktų analogišką klasifikavimą (veiktų kaip vienas perceptronas). Šiame darbe naudojama daugiasluoksnio perceptrono inicializavimo strategija yra pradinius svorius parinkti atsitiktinai iš intervalo $[-1, 1]$.

Optimalaus mokymo sustojimo problema dažniausiai sprendžiama panaudojant validavimo duomenis. Validavimo duomenys - duomenys, turintys tokį patį skirstinį kaip ir mokymo duomenys, bet susidedantys iš kitokių vektorių. Mokant klasifikatorių yra matuojama klasifikavimo klaida tiek mokymo, tiek validavimo duomenims. Klasifikavimo klaida su mokymo duomenimis visada mažėja, nes klasifikatorius prisitaiko prie jų. Klasifikavimo klaida su validavimo duomenimis iš pradžių mažėja, nes klasifikatorius geriau atpažįsta duomenų skirstinį, bet po kurio laiko pradeda didėti, nes per daug prisitaiko prie mokymo duomenų ir pradeda modeliuoti vien jiems būdingus ypatumus.

Klasifikatorių patikimumui įvertinti naudojami testavimo duomenys. Testavimo duomenys - duomenys, turintys tokį patį skirstinį kaip ir mokymo duomenys, bet niekaip nenaudoti mokymui ar optimaliam sustojimui nustatyti. Klasifikavimo klaida testavimo duomenims parodo tikėtiną tikrąją klasifikavimo klaidą, kuri būna didesnė nei klasifikavimo klaida su mokymo ar validavimo duomenimis.

Testavimo duomenims sudaryti yra taikomi įvairūs metodai, pavyzdžiui, mokymo duomenų praplėtimas triukšmu [SRD00]. Šiame darbe panaudotos dvi strategijos:

1. Turimą duomenų vektorių aibę atsitiktinai išmaišyti ir padalinti į tris dalis: mokymo, validavimo bei testavimo. Daugiasluoksnio perceptrono optimaliam sustojimui nustatyti naudoti validavimo duomenis.

2. Mokyti daugiasluoksnį perceptroną fiksuotą iteracijų skaičių ir kiekvienos iteracijos metu apskaičiuoti klaidą su testiniais duomenimis. Iteracija, su kuria buvo gauta mažiausia klaida su testiniais duomenimis žymi optimalią kainą,

kurią įmanoma išgauti, pritaikius idealų optimalaus sustojimo metodą.

Kad klasifikatoriaus mokymas vyktų efektyviau, prieš mokymą duomenys yra normalizuojami. Kitaip sakant, su mokymo, validavimo ir testavimo duomenimis atliekamos transformacijos: atimamas duomenų vidurkis bei padalinama iš duomenų standartinio nukrypimo.

Daugiasluoksnio perceptrono visų perceptronų forma yra tokia pati, todėl tinklo struktūrą vienareikšmiškai nusako sąrašas iš perceptronų kiekių kiekviename sluoksnyje. Pavyzdžiui, 1 pav. tinklo struktūra yra [3, 4, 4, 2]. Įėjimo sluoksnio elementų skaičių nusako duomenų vektoriaus komponenčių skaičius, todėl šio sluoksnio dydis gali būti išskaičiuojamas iš $\boldsymbol{D}$ duomenų matricos. Taigi daugiasluoksnio perceptrono architektūrą dar galima nusakyti sąrašu, kuriame pateikiami tik paslėptų sluoksnių bei išėjimo sluoksnio perceptronų skaičiai.

Tinklo perceptronų struktūra sutampa, todėl tinklo būseną galima vienareikšmiškai nusakyti vien sluoksnių jungčių ((15) formulėje $w_{ij}, i = 1,...,g, j = 1,...,p$) bei perceptronų glodninimo funkcijų poslinkių (angl. bias) ((15) formulėje $w_{i0}, i = 1,...,g$) reikšmėmis. Toliau, tinklo būseną vaizduosime sąrašu, sudarytu iš svorių matricų. Svorių matrica nusako dviejų gretimų sluoksnių jungčių vertes. Pavyzdžiui, sluoksnių, turinčių p ir g perceptronų, jungčių svorių matricos pavidalą apibrėšime taip:

$$\boldsymbol{V_{pg}} = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1p} & w_{10} \\ w_{21} & w_{22} & \cdots & w_{2p} & w_{20} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ w_{g1} & w_{g2} & \cdots & w_{gp} & w_{g0} \end{pmatrix}. \tag{15}$$

Pastebėsime, kad kiekvienas perceptronas turi nulinį svorį, nors 1 pav. jie nėra vaizduojami. Nuliniai svoriai vizualizuojami kaip paskutinis (15) svorių matricos stulpelis.

2.2. Kainoms jautrūs dirbtiniai neuroniniai tinklai

Klasifikavimo klaidų kainas galima įvertinti įvairiai modifikuojant daugiasluoksnio perceptrono realizaciją. Literatūroje aptikti šie metodai:

1. Modifikuoti jau apmokytą daugiasluoksnį perceptroną, perkeliant klasifikavimo ribą (angl. Threshold-Moving):

   (a) Modifikuojant išėjimo sluoksnio perceptronų reikšmių interpretaciją [ZLL06], [Cie09], [KK98]. Kiekvieno daugiasluoksnio perceptrono iš-

ėjimo sluoksnio perceptrono grąžinama reikšmė perskaičiuojama taip:

$$\forall i \in Output : o_{new}^{(i)} = \eta \sum_{c \in classes} o_{old}^{(i)} C[i,c] \ . \tag{16}$$

Čia:

- η - normalizacijos narys, kuris gali būti lygus didžiausiam iš atnaujintų svorių. Šio nario reikia, kad atnaujintos išėjimų reikšmės būtų iš $[0, 1]$ intervalo;
- $o^{(i)}$ - i-tojo išėjimo sluoksnio perceptrono reikšmė;
- $classes$ - aibė, sudaryta iš visų skirtingų duomenų aibės D klasių;
- $Output$ - išėjimo sluoksnio perceptronų indeksų aibė;
- $C[i,c]$ - c-tosios klasės elemento priskyrimo i-tajai klasei klaidos kaina.

(b) Keičiant išėjimo sluoksnio perceptronų nulinius svorius [PWK12]. Algoritmo esmė yra išėjimo sluoksnio nulinių svorių modifikacija:

$$\forall i \in Output : w_{new}^{(i0)} = w_{old}^{(i0)} + \Delta w^{(i)} \ . \tag{17}$$

Rekomenduojamas iteracinis algoritmas, kurio kiekvienos iteracijos metu yra įvertinamas gautasis klasifikavimo skirstinys, tada išėjimo sluoksnio svoriai modifikuojami taip, kad skirstinys panašėtų į trokštamą skirstinį. Iteracinis procesas stabdomas, kai gaunamas pakankamai panašus į trokštamą skirstinys.

2. Mokymosi žingsnio modifikavimas [KK98]. Tiesa, autorius formulę apibrėžė **nuosekliai** klasifikavimo klaidų matricai, bet šią formulę galima nesunkiai perdaryti ir **paporinei** klasifikavimo klaidų matricai:

$$\forall i \in Output : \eta_{new}^{(i)} = \eta_{old}^{(i)} C[i] \ . \tag{18}$$

3. Modifikuoti daugiasluoksnio perceptrono nuostolių funkciją (vidutinės kvadratinės klaidos (angl. mean square error) apskaičiavimo algoritmą), į jį įtraukiant klasifikavimo kainas.

$$nuostoliai = \frac{1}{n} \sum_{i=1}^{n} \sum_{j \in Output} (C[class(x_i),j](t_j - o_j))^2 \ . \tag{19}$$

Čia $class(x_i)$ - i-tojo mokymo duomenų vektoriaus klasė.

[KK98] parodė, kad šis metodas yra pranašesnis už kitus tada žinomus klasifikavimo klaidų įvertinimo metodus. Tai patvirtina ir kiti autoriai, kurie šiam metodui sudarinėjo skirtingus subalansavimo bei klasifikavimo klaidų matricų radimo būdus:

(a) [ASC08], [ASGV11] pasiūlė skirtingų metodų išbalansuotų duomenų korektiškam klasifikavimui, iš kurių geriausias yra $\frac{n}{n_i}$ nario įtraukimas į nuostolių funkciją (n - bendras mokymo vektorių skaičius, n_i - i-tosios klasės mokymo vektorių skaičius):

$$nuostoliai = \frac{1}{n} \sum_{i \in classes} \frac{n}{n_i} nuostoliai_i \ . \tag{20}$$

(b) [MSH+12] apibrėžė sudėtinį klasifikavimo klaidų matricos sudarymo metodą.

4. Įvedant paporinę nuostolių funkciją [RR10], kuri, pasak autoriaus, reikalauja mažesnio mokymo epochų skaičiaus nei vidutinės kvadratinės klaidos metodas.

$$nuostoliai = \sum_{h \in classes} \frac{1}{n_h} \sum_{j \in classes} C[h,j] \sum_{i=1}^{n_h} (f(\boldsymbol{w}_j - \boldsymbol{w}_h)^T \boldsymbol{x}_i^{(h)}))^2 \ . \tag{21}$$

Čia: $\boldsymbol{x}_i^{(h)}$ žymi i-tąjį h-tosios klasės duomenų vektorių.

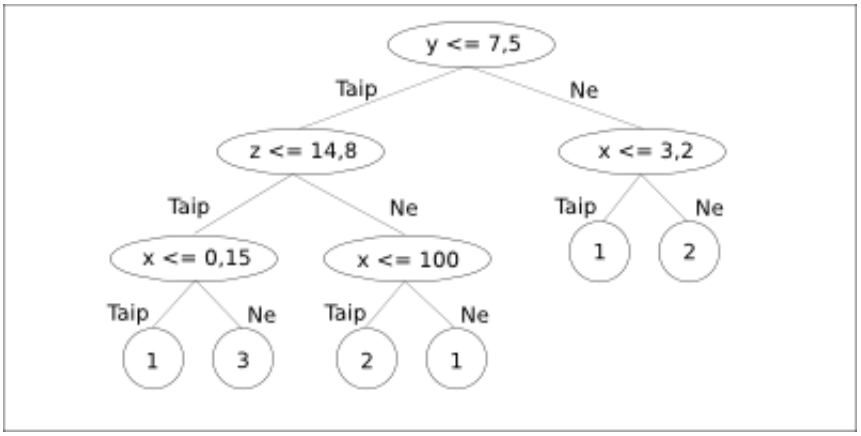
5. Keičiant išėjimo sluoksnio perceptronų trokštamų išėjimų vertes [Rau08]. Šis metodas kritikuojamas, nes modifikuojant trokštamus išėjimus atsiranda netiesiškumai, todėl sunku sudaryti tikslią kainų matricą.

3. Sprendimo medžių apžvalga

3.1. Bendrieji sprendimo medžių principai

Sprendimo medžiu vadinamas medžio pavidalo klasifikatorius, priskiriantis klasėms daugiamačius vektorius, kurių požymiai gali būti tiek kategoriniai, tiek tolydieji kintamieji.

Medį sudaro arba lapas, pažymėtas klasės etikete, arba struktūra, apimanti su dviem ar daugiau pomedžių sujungtą sprendimo priėmimo mazgą [Qui96]. Pastarosios rūšies mazgai apibrėžiami testo pavidalu, o jų pomedžiai atitinka visas įmanomas šio testo baigtis (pvz., žr. 2).



2 pav.: Sprendimo medžio pavyzdys

Duotas vektorius **klasifikuojamas** vykdant rekursyvų procesą, pradedamą nuo medžio šaknies. Jei dabartinis nagrinėjamas mazgas yra lapas, vektoriui priskiriama su juo siejama klasės etiketė. Kita vertus, jeigu tai sprendimo priėmimo mazgas, vektoriui pritaikomas šio mazgo testas ir leidžiamasi vienu lygiu gilyn - į pomedį, atitinkantį gautąją testo baigtį. Procesas baigiamas pasiekus lapo mazgą, kitaip sakant, kai nustatoma vektoriaus priklausomybė klasei.

Vadovaujantis plačiai žinomu godžiu (angl. greedy) iš viršaus žemyn medžio **konstravimo** (mokymosi) procesu, pradedama nuo pilnos mokymosi duomenų aibės, sudarytos iš daugiamačių vektorių su žinoma klasine priklausomybe. Mokymosi tikslas - nustatyti klasifikavimo taisyklę, tinkamą klasifikuoti tiek mokymui naudotus, tiek dar nematytus, naujus tų pačių klasių vektorius. Jeigu aibėje jau yra mažai vektorių, sukuriamas lapas, kuriams priskiriama dažniausiai aibėje aptinkama klasė. Priešingu atveju, sukuriamas sprendimo priėmimo mazgas, kuris susiejamas su pagal tam tikrą kriterijų parenkamu geriausiu testu, duotai aibei turinčiu ne mažiau kaip dvi skirtingas baigtis. Pagal jas pradinė aibė padalijama į poaibius ir kiekvienam

iš jų konstruojamas naujas pomedis. Kiekviename gilesniame lygmenyje mokymo vektorių poaibis vis mažesnis, dėl to dalijimo procesas būtinai baigsis lapo sukūrimu.

Sprendimo priėmimo mazgų testai dalija požymių erdvę hiperpaviršiais, kurių savybes lemia naudojamų požymių skaičius ir kombinavimo būdai. Pvz., atsižvelgiant į vieno požymio arba kelių požymių tiesinės kombinacijos reikšmes, bus atitinkamai gaunamos požymių ašims statmenos arba pasvirosios hiperploštumos. Dėl reikalavimo testams turėti bent dvi skirtingas baigtis duotai duomenų aibei bet kokie šią sąlygą tenkinantys testai lems baigtinį mokymo aibę atitinkantį sprendimo medį. Dėl to potencialūs testai **prioritetizuojami** remiantis specialios paskirties euristikomis, pvz., siekiant kuo mažesnio medžio mazgų skaičiaus, ir pasirenkami geriausieji.

Sukonstruotas sprendimo medis būna pernelyg prisitaikęs prie mokymosi duomenims būdingų ypatumų, pvz., triukšmo, ir taip sumažėja jo gebėjimas apibendrinti - priimti teisingą sprendimą naujiems duomenims. Kita vertus, tiksliai duomenų ypatumus modeliuojantis medis turi daug mazgų ir jo taisykles (atliekamų testų visumą) tampa sudėtinga suvokti dalykinės srities ekspertui. Dėl šių priežasčių prieš naudojimą atliekamas medžio genėjimas. Toks genėjimo būdas dar vadinamas po-genėjimu (angl. post-pruning), jam alternatyvus yra prieš-genėjimas (angl. pre-pruning), kurio esmė - dar auginant medį prieš pridedant naują mazgą patikrinti, ar tai pasitarnaus pasirinktam klasifikavimo kriterijui pagerinti.

Medžio **genėjimas** yra procesas, kurio metu tam tikri jo pomedžiai pakeičiami lapais, jeigu keitimas pagerina pasirinktą kriterijų, pvz., tikėtiną klaidų skaičių. Rezultatas yra medis, kurio klasifikavimo tikslumas naujiems duomenims didesnis, o paties medžio sudėtingumas mazgų skaičiaus prasme - mažesnis.

3.1.1. Kertiniai sprendimo medžių algoritmai

Šiame skyrelyje aptariami du kertiniai klaidų skaičių minimizuojantys sprendimo medžių konstravimo algoritmai.

ID3 algoritmas. Konkretus algoritmas, kuriuo remiasi dauguma sprendimo medžių konstravimo metodų, taip pat auginamas ką tik aprašytu principu iš viršaus žemyn, yra **ID3**, pasiūlytas Quinlan'o [Qui86]. Algoritmas tinkamas tik kategoriniams duomenims klasifikuoti ir į klaidų kainas nėra atsižvelgiama. Testai mazguose parenkami remiantis informacijos teorijos sąvokomis. Medžio genėjimas nėra atliekamas.

Tarkime, turime mokymo duomenų aibę D, joje yra vektorių, priklausančių k skirtingų klasių. Kai visi vektorių požymiai yra kategoriniai kintamieji, sprendimo

medžio kiekvieno mazgo testai turi tokį pavidalą: $X = A_i$, kur X - požymis, A_i - viena iš m galimų jo reikšmių: $i = 1, ..., m$.

Kriterijus, kuriuo remiantis ID3 mazgo testu pasirenkamas požymis, yra *informacijos išlošis* (angl. information gain). Kuo jis didesnis, tuo vienalytiškesni klasių prasme yra gautieji poaibiai ir tuo mažesnio vėlesnių dalijimų skaičiaus prireiks, t. y. tuo tinkamesnis požymis. Informacijos išlošiui skaičiuoti apibrėžiami dydžiai:

1. Apriorinis klasės tikimybės įvertis:

$$p_c = \frac{\left|D_{class(x)=c}\right|}{|D|} \tag{22}$$

   Čia $D_{class(x)=c}$ yra D vektoriai, priklausantys klasei c, $c \in classes$.

2. Vektorių *aibės informacija* (angl. set information) - informacijos kiekis, reikalingas aibės vektoriaus klasei nustatyti:

$$Info(D) = -\sum_{c \in classes} p_c log_2 p_c \tag{23}$$

3. *Testo informacija* (angl. test information) - informacijos kiekis, vis dar reikalingas vektoriaus klasei nustatyti, atlikus sprendimo priėmimo mazgo testą:

$$Info_{test}(D, T) = \sum_{i=1}^{m} \frac{|D_i|}{|D|} Info(D_i) \tag{24}$$

   Čia D_i yra D vektoriai, kuriems testas T turėjo baigtį i, m - galimų testo baigčių skaičius. Diskretiesiems požymiams tai reiškia, kad D_i sudaro vektoriai su vienodomis testuojamo atributo reikšmėmis.

4. Informacijos išlošis - informacijos kiekio sumažėjimas po to, kai vektorių aibei pritaikytas testas T:

$$Gain(D, T) = Info(D) - Info_{test}(D, T)\ . \tag{25}$$

C4.5 algoritmas. Vėlesnis ID3 išplėtimas **C4.5** [Qui93] papildytas sugebėjimu apdoroti tolydžiuosius požymius ir trūkstamas reikšmes. C4.5 naudotas daugelyje vėlesnių algoritmų, kurie jį arba modifikavo, arba panaudojo kaip dalį kompoziciniame metode. C4.5 algoritmą sudaro auginimo ir genėjimo fazės.

Auginimo fazėje C4.5 algoritme mazgo testo parinkimo kriterijus išplečiamas, įvedant *santykinio informacijos išlošio* (angl. gain ratio) sąvoką. Šis dydis išreiškia

naudingos klasifikavimui informacijos dalį, kuri sugeneruojama pritaikius pasirinktą testą vektorių aibei. Pasirenkamas požymis, kuriam jis didžiausias. Papildomai ID3 skaičiuojamiems dydžiams, apibrėžiama:

1. *Dalijimo informacija* (angl. split information) - informacijos kiekis, reikalingas testo baigčiai nustatyti :

$$Info_{split}(D,T) = -\sum_{i=1}^{m} \frac{|D_i|}{|D|} log_2(\frac{|D_i|}{|D|}) \ , \tag{26}$$

   dydžiai apibrėžti kaip ir testo (24) informacijos formulėje.

2. Santykinis informacijos išlošis - informacijos išlošio ir dalijimo informacijos santykis:

$$Gain\ ratio(D,T) = \frac{Gain(D,T)}{Info_{split}(D,T)} \tag{27}$$

Tolydžiuosius požymius naudojantys testai turi pavidalą $X \leq X^{thres}$. Tokiu būdu testo baigčių skaičius $m = 2$, t. y. turime dvejetainį medį. Siekiant parinkti X^{thres}, požymio X skirtingos reikšmės, aptinkamos aibėje D, išrikiuojamos didėjimo tvarka: $X_{i_1} \leq X_{i_2} \leq ... \leq X_{i_k}$. Tokiu atveju X^{thres} bus lygi vienos iš gretimų reikšmių porų aritmetiniam vidurkiui:

$$X_j^{thres} = \frac{X_{i_j} + X_{i_{j+1}}}{2} \ , \ j = 1, ..., m-1 \ . \tag{28}$$

Siekiant kad visos ribinės reikšmės būtų aptinkamos pačiuose duomenyse, X^{thres} galima parinkti tiesiogiai iš $X_{i_1} \leq X_{i_2} \leq ... \leq X_{i_k}$.

C4.5 medis genėjamas *klaidomis grįsto genėjimo* (angl. error-based pruning) metodu, siekiant supaprastinti medžio struktūrą, pašalinti triukšmo įtaką ir per didelio prisitaikymo prie mokymosi duomenų efektą.

Procedūra apima klaidų skaičiaus įverčių skaičiavimą sukonstruoto medžio lapuose ir vidinuose mazguose.

Lapo klaidų skaičiaus įvertis randamas atsižvelgiant į šiuos dydžius:

- n - lapo padengiamų mokymosi vektorių skaičius;
- E - lapo neteisingai suklasifikuotų mokymosi vektorių skaičius.

Naudojamasi euristika, teigiančia, kad neteisingai suklasifikuotų vektorių skaičius gali būti modeliuojamas kaip binominis atsitiktinis dydis: $E \sim B(n,p)$, kur p - klaidų skaičiaus šiame lape tikimybė.

Parametro p pesimistiniu įverčiu laikoma pagal mokymosi duomenis nustatyta pasikliautinio intervalo su pasikliovimo lygmeniu Q viršutinė riba $\bar{p}$. Tuomet dydis $\bar{E} = n \times \bar{p}$ yra laikomas lapo klaidų skaičiaus įverčiu.

Kiekvienam pomedžiui randami du klaidų skaičiaus įverčiai:

1. $\bar{E}_l$ - Klaida, kuri būtų gauta pomedį pakeitus lapu su klaidų skaičių minimizuojančia klase;

2. $\bar{E}_v$ - Klaida, lygi visų pomedžio lapų klaidų įverčių sumai.

Kai pirmasis įvertis mažesnis už antrąjį, pomedis pakeičiamas geriausią klasę atitinkančiu lapu.

3.2. Kainoms jautrūs sprendimo medžiai

Iki šiol pasiūlyta nemažai įvairių sprendimo medžius naudojančių algoritmų, atsižvelgiančių į skirtingo tipo kainas klasifikavimo procese, kuriuos susistemino neseniai atlikto tyrimo autoriai [LV13].

Kainos sprendimo medžių kontekste būna skirtingų rūšių: klaidų, sprendimo priėmimo mazgų testų, laukimo ir kitos kainos. Testo kainą tenka padengti, siekiant išsiaiškinti vektoriaus trūkstamo požymio reikšmę, pvz., medicinos diagnostikos srityje - atliekant specifinį tyrimą. Laukimo kaina įvertina laiką, reikalingą testo rezultatui sužinoti. Klaidų kainas tenka padengti, kai vektorius priskiriamas neteisingai klasei, pvz., nustačius tinklo gedimą brangesniame elemente nei yra iš tikrųjų klaidos kaina bus atitikamų elementų kainų skirtumas. Esama metodų, užtikrinančių jautrumą įvairioms šių kainų kombinacijoms.

Kita vertus, esama skirtingų jautrumo kainoms užtikrinimo sprendimo medžiuose būdų, tarp kurių išryškėja dvi grupės: godūs algoritmai, sukuriantys vieną medį, ir negodūs algoritmai, sukuriantys ir kombinuojantys daug medžių.

Dalis pirmųjų adaptuoja mazgo testo pasirinkimo kriterijų medžio konstravimo etape pagal kainas, kita dalis naudoja kainų matricą kombinacijoje su nepriklausomai nuo jos sugeneruotu medžiu.

Antroji metodų grupė apima sprendimus, paremtus genetiniais algoritmais, stochastiniais ir gaubiamaisiais medodais (*bagging*, *boosting* metodais ir daugybinėmis struktūromis) [LV13].

3.2.1. Vieno medžio metodai

Metodai šioje grupėje pasižymi palyginus nedideliu skaičiavimo resursų poreikiu, tačiau įmanoma, kad skirtingų medžių alternatyvų aibė, kurią jie sugeba ištirti,

nėra pakankamai didelė, kad būtų sukonstruotas tinkamiausias klasifikatorius, t.y. tikėtina patekti į lokalųjį minimumą įmanomų medžių erdvėje.

Algoritmai, naudojantys kainas medžio auginimo procese

Statistinių įverčių išplėtimas. Algoritmai šioje kategorijoje atlieka medžio auginimą iš viršaus žemyn ir tam tikru būdu naudoja ar minimizuoja informacijos išlošį (25). Dalyje algoritmų, siekiant parinkti mazgo testą, apibrėžiamas ICF (angl. Informaction Cost Function) matas, kuriuo įforminamas prioriteto teikimas pigesniems, tačiau aukštą informacijos išlošį suteikiantiems požymiams. Pvz., EG2 algoritme [Núñ91] turime išraišką:

$$ICF_X(D) = \frac{2^{Info(D)} - 1}{(C_X + 1)^w} \ . \tag{29}$$

Čia C_X - požymio X kaina, D - duomenų aibė, w - parametras, reguliuojantis požymio kainos įtakos stiprį.

Kita dalis algoritmų pagal klaidų kainas modifikuoja informacijos išlošio skaičiavime naudojamus apriorinių tikimybių įverčius, t. y. vietoj tikimybės įverčio (22) [BFOS84] naudojamas

$$\tilde{p}_i = \frac{C[i]}{\sum_{j \in classes} C[j] p_j} p_i \ , \tag{30}$$

C yra kainų matrica, o $C[i]$ - kainų vektorius:

$$C[i] = \sum_{j \in classes} C[i,j] \ , i \in classes. \tag{31}$$

Be to, C4.5 algoritmas leidžia naudoti pasvertus mokymo vektorius, kad mokymo procese jiems būtų skirtas didesnis dėmesys. Tuomet apriorinių klasių tikimybių (22), testo informacijos (24) ir dalijimo informacijos (26) formulėse vertinant vektorių dalį aibėje būtų sumuojami ne vektorių kiekiai, bet jiems priskirti svoriai. C4.5CS algoritme [Tin02], siekiant gauti brangių klaidų skaičių minimizuojantį medį, ši C4.5 savybė išnaudojama ir vektorių iš klasės i svoriai nustatomi taip:

$$w(i) = C[i] \frac{n}{\sum_{j \in classes} C[j] n_j} \ , \tag{32}$$

$C[i]$ apibrėžta (31), n - bendras vektorių skaičius, n_i - klasės i vektorių skaičius.

[ZL10] savo darbe teigia, kad kainų matricos būna subalansuotos arba ne, ir

pateikia metodiką optimaliems C4.5 algoritmo vektorių svoriams nustatyti remiantis šiuo suskirstymu. Teigiama, kad optimalūs vektorių, priklausančių klasėms i ir j, svoriai turėtų sutikti santykiu: $\tau_{ij} = w_i/w_j = C[i,j]/C[j,i]$. Taip gaunama C_n^2 sąlygų lygčių sistema $w_i/w_j = C[i,j]/C[j,i]$, i, $j \in classes$, turinti sprendinių tik su tam tikromis, vadinamomis nuosekliosiomis, kainų matricomis. Autorių siūlomas metodas yra nuosekliosioms kainų matricoms apskaičiuoti vektorių svorius išsprendžiant tiesinių lygčių sistemą, o nenuosekliosioms - išskaidyti sprendžiamą uždavinį į C_n^2 mažesnių, apmokant po klasifikatorių kiekvienai klasių porai su ją atitinkančiais poriniais klasių svoriais w_i ir w_j. Autoriai parodo, kad šitoks svorių C4.5 algoritmui parinkimo metodas yra efektyvesnis už klasifikavimą be svorių ar naudojant alternatyvų svorių parinkimą klasifikavimo kainos atžvilgiu.

Tiesioginis kainų naudojimas. Egzistuoja algoritmų kategorija, konstruojanti medį iš viršaus žemyn, tačiau mazgo testo pasirinkimui naudojanti ne informacijos išlošį, bet kainas tiesiogiai.

Pvz., algoritme Cost Minimization [PMM+94] tikėtina kaina nedalyti mazgo

$$\min_{c \in classes} \sum_{x \in D} C[class(x), c] \tag{33}$$

lyginama su tikėtina kaina jį padalijant testu T:

$$\sum_{D_i} \min_{c \in classes} \sum_{x \in D_i} C[class(x), c] \ , \tag{34}$$

D_i yra aibės D poaibis, su kuriuo testas T baigiasi i-ja baigtimi.

Dalinti nustojama, jei antroji reikšmė didesnė. Tai vadinamasis prieš-genėjimas.

Kelių kintamųjų sprendimo priėmimo mazgai. Priešingai nei dauguma metodų, sprendimo priėmimo mazguose tikrinančių vieno požymio reikšmę ir taip sukuriančių ašims statmenas klasių skiriamąsias hiperplokštumas, literatūroje iškelta idėja, kad geresnis klasifikavimas galėtų būti pasiektas naudojant kelių požymių (tiesinę arba netiesinę) kombinaciją.

Algoritmas, naudojantis tiek tiesinius, tiek netiesinius sprendimo priėmimo mazgus, yra CSNL [Vad05]. Tai dviem klasėms, kurioms taikoma normališkumo prielaida, skirtas atskirti klasifikatorius. Renkantis kiekvieno mazgo testą, išmėginamos visos įmanomos požymių kombinacijos, iš kurių kiekvieną panaudojant įvertinami normaliųjų skirstinių parametrai - kovariacinės matricos ir vidurkiai. Atsižvelgiant į kovariacinių matricų apibrėžtumą, mazge naudojamas kvadratinis, tiesinis ar pa-

prastas statmenas ašims klasifikatorius, minimizuojantis klasifikavimo klaidų kainą. Sukonstruotam medžiui pritaikomas sumažintos kainos genėjimo metodas.

Kainų naudojimas kombinacijoje su nepriklausomu medžiu

Tais atvejais, kai klaidų kainos kinta dažnai, gali praversti sprendimo medį užauginti vieną kartą ir paskui klasifikuojant naujus vektorius jį taikyti su dabartine kainų matrica.

Vienas iš tokiu požiūriu besivadovaujančių algoritmų yra metodas "I-gain (Cost-Laplace probability)" [PMM+94]. Konstruojant sprendimo medį, lapuose pagal jiems formuoti naudojamų mokymo duomenų poaibių klasines priklausomybes apskaičiuojami vektoriaus priklausymo kiekvienai klasei Laplace tikimybių įverčiai. Panaudojant kainų matricą, randama tikėtina klasifikavimo kaina, kurią tektų mokėti priskiriantį šį lapą pasiekusį vektorių konkrečiai klasei. Pasirenkama klasė, kuriai ši kaina būtų mažiausia.

3.2.2. Daugelio medžių metodai

Daugelio alternatyvių medžių metodai skirti įveikti trūkumams, kuriems nėra atsparūs vieno medžio metodai: būtent lokaliųjų minimumų problemai.

Genetiniai algoritmai

Genetiniai algoritmai yra paieškos euristika, leidžianti išspręsti optimizavimo uždavinį naudojantis populiacijos evoliucija, paremta stipriausiųjų išlikimu.

Tarp klasifikavimo kainą šiuo principu minimizuojančių algoritmų išsiskiria ICET [Tur95]. Populiaciją sudaro sprendimo medžio auginimo algoritmo parametrai. Medis gaunamas C4.5 algorimtu, kuriame vietoj informacijos išlošio požymio tinkamumui įvertinti naudojamas EG2 algoritmo ICF įvertis (29). Individų tinkamumas vertinamas kiekvieną jų panaudojus medžiui užauginti su mokymosi duomenimis ir gautiems medžiams įvertinus klasifikavimo klaidų kainą su testiniais duomenimis. Geriausiai įvertintiems individams pritaikomos standartinės kryžminimo ir mutacijos operacijos. Taip gaunama seka kainos prasme vis geresnių populiacijų, iš kurių po baigtinio iteracijų skaičiaus pasirenkamas geriausias individas - klasifikavimui naudojamas sprendimo medis.

Gaubiamieji metodai

Metodų grupė, kurioje stengiamasi jautrumą kainoms realizuoti per apvalkalą egzistuojantiems klasifikavimo klaidų skaičių minimizuojantiems algoritmams.

***Bagging* metodika.** *Bagging* metodikos idėja - sukurti daug tuo pačiu principu konstruojamų klasifikatorių iš skirtingų pradinės mokymo duomenų aibės poaibių ir sukombinuoti jų rezultatus.

Idėjas panaudoti į klaidų skaičiaus mažinimą orientuotą sprendimo medį ir sukombinuoti daugelio medžių rezultatus realizuoja **MetaCost** algoritmas [Dom99]. MetaCost algoritmas konkretų klasifikatorių padaro jautrų klaidų kainoms, apgaubdamas jį kainą minimizuojančia procedūra. Kainoms nejautrus klasifikatorius nėra modifikuojamas, kitaip sakant, traktuojamas kaip juodoji dėžė. MetaCost algoritmo esmė yra: mokymo vektorių klases pakeisti į įvertintas tų vektorių mažiausios kainos klases ir gautą duomenų aibę panaudoti klaidų kainoms nejautraus sprendimo medžio sudarymui.

Pirmiausia balsavimo būdu įvertinamos vektorių priklausymo klasėms tikimybės p_i. Tuo tikslu iš pirminės duomenų aibės atsitiktiniu būdu sugeneruojamas tam tikras skaičius imčių, kurių kiekviena nepriklausomai panaudojama sprendimo medžiui sukurti. Kiekvienas medis pateikia tikimybę, kad vektorius x priklauso klasei i, ir išvedamas medžių rezultatų vidurkis kiekvienai klasei - priklausymo jai tikimybės įvertis p_i.

Tuomet panaudojant gautąsias tikimybes, mokymosi vektoriai yra perklasifikuojami taip, kad būtų minimizuota tikėtina klasifikavimo klaidos kaina, t. y.:

$$class(x) = \arg\min_{c \in classes} \sum_{i \in classes} p_i C[i, c] \; . \tag{35}$$

Šie mokymosi vektoriai su naujais klasių numeriais panaudojami galutiniam - kainoms jautriam - sprendimo medžiui gauti įprastuoju klaidų skaičiaus minimizavimo algoritmu.

Stochastiniai sprendimai

Pasirinkto laiko algoritmai naudoja papildomą laiką klasifikavimo kokybei pagerinti. Stochastinių sistemų atveju, laiko sąnaudas lemia atsitiktinių egzempliorių skaičiaus r pasirinkimas. Kuo didesnis r, tuo mažesnės kainos medis gaunamas.

Šios kategorijos atstovas - ACT algoritmas [EM08], pasižymintis stochastiškumu kategoriniams, tačiau deterministiškumu tolydiesiems požymiams. Su kiekvienu potencialiu kategoriniu požymiu medžio šaknyje sukonstruojama po 1 medį EG2 algoritmu ir r-1 medį SEG2 algoritmu. Kita vertus, kiekvienam potencialiam tolydžiajam požymiui parenkama po r geriausią informacijos išlošį turinčių ribinių reikšmių, naudojamų galimoms požymio reikšmėms padalinti į dvi aibes, ir su kiekviena užauginamas medis EG2 algoritmu. Taip kiekvienam požymiui gaunama po

r medžių, kurie įvertinami, atsižvelgiant į tikėtiną testų ir klaidų kainą, tenkančią vienam mokymo vektoriui. Požymis su geriausiu įverčiu tampa medžio šaknimi.

Kalbant apie r medžiams sukonstruoti naudojamus algoritmus, EG2 medžio šakniai parenka maksimalaus ICF įverčio, jautraus testų kainoms ir informacijos išlošiui, požymį. Kita vertus, SEG2 yra stochastinė EG2 versija, parenkanti požymį atsitiktinai su tikimybėmis, proporcingomis požymių ICF įverčiams, dėl to kiekvieno paleidimo rezultatas yra vis kitoks medis.

Medis po auginimo nugenimas, pasinaudojant klaidomis grįsto genėjimo idėjomis.

3.2.3. C5.0 algoritmo analizė

Plačiai žinomas atviru kodu platinamas C5.0 [Res11] algoritmas yra C4.5 algoritmo kainoms jautrus išplėtimas. Šio darbo autoriai atliko C5.0 algoritmo kodo analizę, siekdami nustatyti, kaip klasifikavime naudojama kainų matrica 3 ir daugiau klasių atveju. Sprendimas atlikti kodo analizę priimtas, nes šio darbo autoriams nėra žinomi jokie literatūros šaltiniai, kuriuose būtų aprašomas kainų įvedimo į C5.0 metodas.

Buvo nustatyta, kad pagal jautrumą kainoms metodą būtų galima priskirti vieno medžio metodams, naudojantiems kainas su nepriklausomai sugeneruotu medžiu. Pagal nutylėjimą C5.0 vykdomas medžio konstravimas apima auginimo ir genėjimo stadijas, tačiau kai klasifikatoriui nurodoma kainų matrica, trijų ir daugiau klasių atveju genėjimas nėra atliekamas. Užauginto medžio lapuose esantys klasių pasiskirstymai panaudojami pigiausiai klasei šiame lape nustatyti:

$$c_{min\ cost} = \arg\min_{c \in classes} \left(\sum_{i \in classes} p_i C[i, c] \right) . \tag{36}$$

Čia p_i yra apibrėžta (22).

Apskritai C5.0 suteikiamos galimybės yra:

- Klaidų skaičių minimizuojančio klasifikatoriaus - medžio - konstravimas.
- Nežinomų reikšmių apdorojimas. Kai testuojamo požymio reikšmė nėra žinoma, algoritmas padalija vektorių ir siunčia po dalį visomis šakomis žemyn.
- Sukonstruoto klasifikatoriaus kokybės įvertinimas remiantis mokymosi arba testiniais duomenimis. Parodoma, kiek medyje turima lapų, kiek ir kokių klaidingų mokymosi (testinių) vektorių priskyrimų atlikta.
- Klasifikavimo taisyklių generavimas ir rikiavimas pagal naudingumą.

- Boosting galimybė.
- Naudingų klasifikavimui požymių atrinkimas (angl. attribute winnowing).
- Negriežtos (angl. soft) ribinės reikšmės.
- Išplėstinis genėjimo valdymas.
- Klasifikavimo kokybės vertinimas kryžminės patikros būdu.
- Klasifikavimo klaidų matricos pritaikymas.
- Svorių priskyrimas atskiriems mokymo vektoriams, siekiant jiems suteikti daugiau reikšmės mokymosi procese.

3.2.4. Laplace kainoms jautrus genėjimas

Ankstesniuose skyreliuose (3.2.1 ir 3.2.2) aptarėme įvairius medžio auginimo metodus, tačiau šio darbo kontekste prasminga taikyti medžio genėjimą. Siekiant sėkmingai inicializuoti hibridinį klasifikatorių reikia, kad sprendimo medis gebėtų kuo geriau apibendrinti ir būtų kuo mažesnis, kad tinklas netaptų pernelyg sudėtingas ir jautrus lokaliesiems minimumams.

Realizuotos C4.5 versijos auginamas medis dėl ypatybės kurti lapus iš daugiau kaip vieno nebūtinai tai pačiai klasei priklausančių mokymosi vektorių suteikia galimybę lapus pažymėti ne konkrečiomis klasėmis, o juose išsaugoti klasių pasiskirstymą. Vidiniuose medžio mazguose taip pat galima saugoti klasių pasiskirstymą, atitinkantį pomedžio lapų pasiskirstymų sumą. Viena vertus, saugant mazguose mokymo vektorių, priklausančių skirtingoms k klasių, skaičius, klasių tikimybės gali būti nusakomos remiantis klasių pasitaikymo santykiniais dažniais, kaip nurodyta (22).

Kita vertus, dėl to, kad medis konstruojamas siekiant kuo geriau atskirti skirtingoms klasėms priklausančius vektorius, mazguose tam tikrų klasių vektorių gali apskritai nebūti aptikta. Klasių skirstinį priartinti prie tolydžiojo, išvengiant ekstremalių reikšmių, galima atliekant tikimybių įverčių Laplaso korekciją [Goo65].

$$p_i = \frac{n_i + 1}{n + k} \,, \tag{37}$$

k - klasių skaičius.

Tokia timybių korekcija gali būti naudojama jautrumui klaidų kainoms užtikrinti. Tai atliekama medžio genėjimo stadijoje, naudojant alternatyvų klaidomis

grįstam genėjimo procesą, aprašytą [BKK+98]. Kiekviename medžio mazge apskaičiuojamos klasių tikimybės p_i taikant Laplaso korekcijos metodą (37). Kai mazgo padengiamų mokymosi vektorių skaičius yra n, jo tikėtinos klaidų kainos įvertis yra:

$$cost = n \times \min_{c \in classes} \left(\sum_{i \in classes} p_i C[i, c] \right) \tag{38}$$

Gautasis kainos įvertis lyginamas su vaikinių mazgų kainų įverčių suma ir jei yra už ją mažesnis, pomedis pakeičiamas lapu.

3.2.5. Kainoms jautrių medžių aptarimas

Pristatėme įvairius sprendimo medžių jautrumo kainoms užtikrinimo būdus. Kaip pastebi [LV13], kainos jautrūs sprendimo medžių algoritmai evoliucionavo nuo vieno medžio metodų link daugelio medžių metodų. Tai nulėmė santykinai silpnas pirmųjų veikimas. Vis dėlto pastebima, kad taikymuose, kuriuose labai svarbus veikimo greitis, dėl kurio galima sumokėti kiek didesnę klasifikavimo klaidų kainą, gali būti taikomi vieno medžio metodai.

Kadangi šiame darbe siekiama sprendimo medį naudoti dirbtiniam neuroniniam tinklui inicializuoti, ne visi iš pateiktųjų metodų yra tinkami šiam tikslui pasiekti. Kainų jautrumo medžiuose užtikrinimo metodas turėtų tenkinti šias savybes:

- Medis tinkamas tolydiesiems duomenims klasifikuoti: neuroninis tinklas sugeba apdoroti tik tolydžiuosius duomenis.
- Sukonstruotas medis gali būti tiesiogiai arba po nesudėtingos apibendrinamosios modifikacijos naudojamas keleto klasių duomenims klasifikuoti: pastebėta, kad dauguma metodų sėkmingai veikia su dviejų klasių duomenimis, tačiau prastai - su daugeliu klasių, dėl to siektinas patobulinimas šioje srityje.
- Mazgų testai tikrina vieno požymio reikšmę: reikalavimą iškelia planuojamas sprendimo medžių ir neuroninių tinklų apjungimo metodas (žr. skyrelį 4.1).
- Jautrumas testų kainoms nebūtinas: nenumatoma apdoroti trūkstamų reikšmių, dėl to testų atlikti neprireiks.
- Galutinis metodo pasiūlytas klasifikatorius yra vienas medis, o ne jų grupė: siekiant inicializuoti neuroninį tinklą, turi būti vienareikšmiškai aišku, kokiu medžiu pasiremti.

- Pageidautina, kad algoritmas stengtųsi minimizuoti medžio dydį (naudotų prieš/po genėjimą): mažesnis mazgų skaičius lems mažesnį tinklą, kurį reikia apmokyti (žr. skyrelį 4.1).

- Algoritmas turi veikti greitai: vienas iš darbo tikslų yra sumažinti DNT mokymo iteracijų skaičių, taip suefektyvinant mokymą.

Atsižvelgiant į išsakytus argumentus, tinkami kandidatai šio darbo kontekste būtų apriorinių tikimybių įverčius modifikuojantys, C4.5 vektorių svorius naudojantys ir tiesiogiai kainas naudojantys algoritmai. Vis dėlto į egzistuojančius algoritmus, kurie to nenumato, reikėtų mėginti įvesti kainoms jautrų genėjimą, kad rezultatas būtų optimalaus mazgų skaičiaus medis.

Tarp daugelio medžių metodų, keliame hipotezę, kad tiktų MetaCost algoritmas. Esant fiksuotai mokymo duomenų aibei, alternatyvūs medžiai yra konstruojami vieną kartą ir gali būti įsidėmėti. Kiekvieną kartą siekiant pritaikyti naują kainų matricą tektų sukonstruoti po vieną naują sprendimo medį iš viršaus žemyn principu. Vadinasi, papildomos laiko sąnaudos, lyginant su vieno medžio metodais, būtų patiriamos tik vieną kartą.

Kita galimybė - ACT algoritmas, kai r mažas. Autoriai pastebi, kad su $r = 3$ pasiekiamas optimalios kainos klasifikavimas. Reikėtų eksperimentiškai įvertinti, kokia būtų vykdymo trukmė.

4. SM ir DNT apjungimo apžvalga

4.1. Egzistuojantys apjungimo metodai

Iki šiol pristatėme du žinomų induktyvaus mokymosi būdų - konektyvistinio ir simbolinio - atstovus: dirbtinius neuroninius tinklus ir sprendimo medžius. Abu būdai turi savo privalumų ir trūkumų, kurie papildo vienas kitą.

Viena vertus, sprendimo medžiai yra greitai apmokomi, jais vaizduojamos taisyklės yra vaizdžios ir gali būti nesunkiai suprantamos dalykinės srities ekspertų. Vis dėlto šio klasifikatoriaus tikslumas nėra labai aukštas.

Kita vertus, dirbtiniai neuroniniai tinklai gerai klasifikuoja mokymui nenaudotus vektorius, esant triukšmui, tačiau jų mokymas lėtas ir sudėtinga nustatyti optimalias tinklo parametrų reikšmes, kaip kad tinklo topologija ar pradiniai svoriai.

Atsižvelgiant į šias ypatybes, galima apjungti šiuos metodus, tikintis gauti klasifikatorių, kurio apmokymo laikas trumpas, o tikslumas toks pat kaip daugiasluoksnio perceptrono. Apjungimas įgalintų išspręsti ir tinklo architektūros apibrėžimo problemą, nes nėra patikimo būdo sužinoti, kokia architektūra yra tinkamiausia specifiniam uždaviniui spręsti. Optimalus tinklas būtų mažiausias tinklas, sugebantis adekvačiai užfiksuoti duomenyse slypinčius sąryšius [TP11].

Idėja sujungti nagrinėjamus klasifikatorius literatūroje iškelta jau anksčiau [Rou06], tačiau pastebėta, kad daugeliui metodų būdinga tai, kad sukonstruoto hibridinio klasifikatoriaus tikslumas sutampa su pradinio sprendimo medžio tikslumu. Toks pastebėjimas verčia abejoti pačia metodų kombinavimo prasme, nes tikslumo prasme tuomet pakaktų naudoti tik medį. Vis dėl to literatūroje paskelbta darbų, pateikiančių įrodymų, kad hibridizacija tikslumą pagerina. Žemiau pateikiame jų apžvalgą.

Banerjee hibridinis klasifikatorius. Metodas, pasiūlytas [**Ban97**], pirmiausia C4.5 algoritmu [Qui93] sukuria sprendimo medį, kurio mazgų testai turi pavidalą $X < C$. Remiantis pastebėjimu, kad tokiu atveju kelias nuo šaknies iki medžio lapo gali būti išreikštas elementariąja konjunkcija, o kelių iki tos pačios klasės visuma (t. y. visi tą klasę atitinkantys lapai) - normaliąja disjunkcine forma, galima apibrėžti medį atitinkančią neuroninio tinklo architektūrą.

Kiekvienas mazgas atitinka dvi elementariąsias konjunkcijas: $q_i = (X < C)$ ir $\neg q_i = (X \geq C), i = 1,...,N$, N - medžio mazgų skaičius, kitaip sakant, du skirtingus sprendimus ir šakas iš to mazgo. Kiekvienai jų sukuriamas neuronas pirmajame tinklo sluoksnyje ir sujungiamas su atitinkamu požymiu X. Antrajame sluoksnyje kiekvienam unikaliam keliui medyje nuo šaknies iki lapo sukuriama po neuroną,

kuris sujungiamas su pirmojo sluoksnio neuronais, atitinkančiais sprendimus tame kelyje. Galiausiai trečiajame - išėjimo - sluoksnyje kiekvienai įmanomai klasei yra po vieną neuroną, kuris agreguoja visus kelius, kuriais galima į tą klasę patekti.

Be to, kad apibrėžiama tinklo architektūra, pateikiamos taisyklės, kaip apskaičiuoti jo jungčių svorius, kad gautoji struktūra savo prasme atitiktų normaliąją disjunkcinę formą kiekvienai klasei. Toliau, siekiant pagerinti tinklo klasifikavimo tikslumą, jis **klaidos sklidimo atgal** metodu mokomas su tais pačiais mokymo duomenimis. Tuo tikslu apibrėžtoje architektūroje įvedamos ir visos trūkstamos jungtys su santykinai mažais svoriais.

Šis metodas pasižymi keliomis geromis savybėmis. Pirmiausia ką tik inicializuotas tinklas veikia beveik taip pat tiksliai kaip ir jam konstruoti naudotas sprendimo medis. Antra, konversijos procedūra apibrėžia tinklo architektūrą. Pagaliau galutinis gautasis klasifikatorius veikia geriau nei to paties neuronų skaičiaus dvisluoksnis ar tos pačios architektūros tinklas su atsitiktiniais svoriais arba pirminis sprendimo medis pavieniui.

Kita vertus, kaip metodo trūkumas išskiriamas pastebėjimas, kad tinklo neuronų skaičius auga tiesiškai kartu su sprendimo medžio mazgų skaičiumi (vienas vidinis medžio mazgas atitinka tris nauju neuronus). Dėl to tam tikroms realių duomenų aibėms galima gauti sprendimo medį su dideliu skaičiumi mazgų, kas lemtų ir išaugusį tinklo sudėtingumą. Verta pastebėti, kad tinklas gali gautis didesnis negu pakaktų duomenų ypatumams atspindėti.

Kaip matyti, metodas kelia tam tikrus reikalavimus sprendimo medžiams. Pirma, kiekvienas mazgas turi tikrinti vieno tolydžiojo požymio reikšmę. Antra, medį kuriantis algoritmas turi stengtis užtikrinti kuo mažesnį mazgų skaičių.

Kubat radialinių bazinių funkcijų hibridinis klasifikatorius. Kaip parodoma [Kub98] autorių, sprendimo medžiai gali būti sėkmingai naudojami radialinių bazinių funkcijų neuroniniams tinklams inicializuoti.

Radialinių bazinių funkcijų tinklas turi vieną paslėptą sluoksnį, kurio neuronai žymi duomenimis būdingų klasterių centrus $\mu_i = [\mu_{i1}, \ldots, \mu_{ip}]$. Kiekvienam duomenų vektoriui įvertinami atstumai nuo šių centrų $\varphi_i(x)$ ir jų svertinės sumos $o_j = \sum_{i=0}^{n} w_{ij}\varphi_i(x)$, gautos išėjimo sluoksnio neuronuose, nulemia, kuriai klasei vektorius priskiriamas. Atstumai nuo centrų skaičiuojami remiantis daugiamačio normaliojo skirstinio pasiskirstymo funckijos reikšmėmis.

Šio tipo tinklų incializavimo problemos apima centrų ir standartinių nuokrypių parinkimą, klasifikatoriaus jautrumą neaktualiems požymiams ir per didelį jungčių skaičių tarp centrus ir klases reprezentuojančių sluoksnių. [Kub98] autoriai parodo,

kad visos jos gali būti išspręstos inicializuojant tinklą C4.5 algoritmu sukonstruotu sprendimo medžiu: pirmiausia užauginamas medis, kuris vektorių apibrėžimo erdvę padalija daugiamačiais stačiakampiais, tuomet jų geometriniai centrai tampa radialinių bazinių funkcijų centrais μ_i. Paskutiniame etape, užuot taikius iteratyvų tinklo apmokymą, optimalūs išėjimo sluoksnio neuronų svoriai randami apskaičiuojant pseudoinversinę matricą.

Metodo privalumas prieš jį inicializavusį sprendimo medį tas, kad inicializuoto tinklo klasifikavimo tikslumas aukštesnis esant įvairiems medžio genėjimo mastams, taigi inicializavimas pasiteisina.

Kubat antro sluoksnio sprendimo medžiams klasifikatorius. Sprendimo medis, turintis pavidalo $x < C$ testą, vienodai traktuoja tiek požymio $x = C + 0.1$, tiek $x = 1000C$ reikšmes - perduoda tolesnį vektoriaus apdorojimą dešiniajai medžio šakai, kaip pastebi [Kub96] autoriai. Iškeliama idėja įvertinti absoliutų skirtumą tarp testo ribinės reikšmės C ir požymio reikšmės x, sprendimo priėmimo ribą pakeičiant tolydžiąja funkcija:

$$f(x) = \frac{1}{1 + e^{-\|x-C\|}} \tag{39}$$

Toks pakeitimas įgalina ne tik suklasifikuoti duotąjį vektorių, tačiau įvertinti jo atstumą iki sprendimo priėmimo taisyklės, t. y. kelio medyje nuo šaknies iki lapo. Taip galima klasifikavimo procese atsižvelgti į visas medžio taisykles, o ne į vieną.

Būtent šia idėja ir remiasi [Kub96] pristatytas hibridinis sprendimo medžio ir daugiasluoksnio neuroninio tinklo metodas. Klasifikatorius konstruojamas pirmiausia užauginant sprendimo medį C4.5 algoritmu ir paverčiant jį normaliosios disjunkcinės formos taisyklių rinkiniu. Tuomet kiekvienam vektoriui apskaičiuojamas jo atstumas iki kiekvienos taisyklės pagal formulę:

$$R_k = \Pi_{j=1}^{T_k} r_j \tag{40}$$

Čia k yra taisyklės numeris, T_k - joje atliekamų testų skaičius, r_j - $j - jo$ taisyklės testo reikšmė funkcijai (39). Skaičiuojamos gautųjų atstumų svertinės sumos klasinei priklausomybei nustatyti.

Šiems veiksmams atlikti formuojamas neuroninis tinklas, kurio pirmame sluoksnyje yra po vieną neuroną kiekvienai iš n_T medžio taisyklių, o išėjimo sluoksnyje - kiekvienai duomenų klasei. Pirmojo sluoksnio neuronai skirti atstumams R_k nuo vektoriaus iki taisyklės apskaičiuoti. Išėjimo sluoksnio svoriai koreguoja taisyklių įtakos klasės parinkimui mastą: jais taisyklės sujungiamos su klasių numerius ati-

tinkančiais neuronais.

Neuroninio tinklo svoriai nustatomi ne iteratyvaus mokymo proceso, tačiau pseudoatvirkštinės matricos skaičiavimo būdu.

Parodoma, kad tokiu būdu kombinuojant sprendimo medžius ir dirbtinius neuroninius tinklus pagerinamas sprendimo medžio klasifikavimo tikslumas.

4.2. Siūlomo metodo hipotezė

Siekiant gauti kainoms jautrų klasifikatorių - hibridinį sprendimo medžių ir dirbtinio neuroninio tinklo metodą - pasirinkta kaip pagrindą naudoti [Ban97] siūlomą kombinacijos metodiką. Originaliame darbe nebuvo nagrinėjamas klaidų kainoms jautrus klasifikavimas bei išbalansuotų duomenų problema. Šiame darbe, siekiant į minimas problemas atsižvelgti, bus naudojami klaidų kainoms jautrūs medžio konstravimo ir dirbtinio neuroninio tinklo mokymo procesai. Realizuoti jų variantai aprašyti skyreliuose 5.2 ir 5.3.

5. Algoritmų realizacija

5.1. Programavimo aplinka

Projektui realizuoti buvo pasirinkta *Python* programavimo aplinka dėl šių priežasčių:

- *Python* yra atviro kodo programavimo aplinka.
- *NumPy* bibliotekos pagalba galima naudoti masyvų programavimo (angl. array programming) paradigmą.
- Siekiama, kad projekto rezultatas būtų viešai prieinamas. http://pypi.python.org/pypi paketų (bibliotekų) indeksavimo sistema suteikia galimybę savo atviro kodo paketą pateikti plačiai Python bendruomenei. Analogiškos paketų viešinimo sistemos neturi masyvų programavimo paradigmą įgalinantys analogai *R* bei *Matlab*.
- Vienas iš šio darbo tikslų yra pagreitinti klasifikatoriaus mokymo trukmę. *mpi4py* paketo pagalba galima išnaudoti daugiaprocesorinių architektūrų savybes, kas suteikia galimybę įvertinti ne tik nuosekliai vykdomo, bet ir išlygiagretinto algoritmo spartą.

5.2. Dirbtinių neuroninių tinklų realizacija

Norint geriau susipažinti su klasifikavimo klaidų kainų įvedimo metodais bei jų savybėmis, prieš kuriant SM ir DNT kombinuojantį klasifikatorių, buvo nuspręsta atskirai išsinagrinėti kainų įvedimo metodus į SM ir DNT klasifikatorius. Atlikus kainų įvedimo į DNT metodų analizę (žr. 2.2 skyrių) paaiškėjo, kad geriausias duomenų subalansavimo metodas yra (20), o praktikoje nusistovėjęs kainų įvedimo į DNT metodas yra (19), kuris ir realiuotas šiame darbe. Taip pat buvo atrastas naujas, literatūroje mažai tyrinėtas kainų įvedimo metodas (21), kuris pasirodė perspektyvus.

Nuspręsta pirmiausia Python aplinkoje realizuoti MLP klasę, kurios pagalba būtų galima kurti, apmokyti bei naudoti pasirinktos architektūros daugiasluoksnius perceptronus. Šios klasės pagrindinė paskirtis - susikurti patogią aplinką eksperimentavimui su daugiasluoksnio perceptrono architektūra.

Realizuojant MLP klasę buvo pasirinkti šie sprendimai:

1. Algoritmas realizuotas totaliniu metodu, nes iteracijos vykdymo trukmė yra mažesnė nei stochastinio metodo. Taip yra, nes totalinio metodo atveju vek-

torių daugyba yra atliekama panaudojant optimizuotą matricų daugybą, o ne sintaksinius darinius, skirtus iteruoti ir atlikti veiksmus su pavieniais vektoriais. Taip pat totalinio metodo privalumas yra tas, kad naudojant masyvų programavimo paradigmą kodas yra lakoniškesnis ir lengviau skaitomas.

2. Tinklas sukuriamas konstruktoriui perduodant sąrašą, sudarytą iš kiekvieno sluoksnio perceptronų skaičiaus arba iš kiekvieno sluoksnio perceptronų pradinių svorių (žr. (15)). Pirmuoju atveju tinklas yra inicializuojamas atsitiktiniais svoriais iš intervalo $[-1, 1]$.

3. Tinklui pateikiama $\boldsymbol{D}$ duomenų matrica, sudaryta iš duomenų vektorių, bei $\boldsymbol{T}$ matrica, turinti tiek stulpelių, kiek yra klasių, ir tiek eilučių, kiek yra duomenų vektorių. Kiekvieno $\boldsymbol{T}$ elemento reikšmė yra iš $\{0, 1\}$ aibės. 1, jeigu tos pačios eilutės duomenų vektorius priklauso elemento stulpelio klasei, 0 - priešingu atveju. Šis interfeisas atitinka $Matlab$ kalbos $NeuralNetworkToolbox$ duomenų pateikimo daugiasluoksniam perceptronui interfeisą.

4. Tinklas gali būti apmokomas dviem metodais:

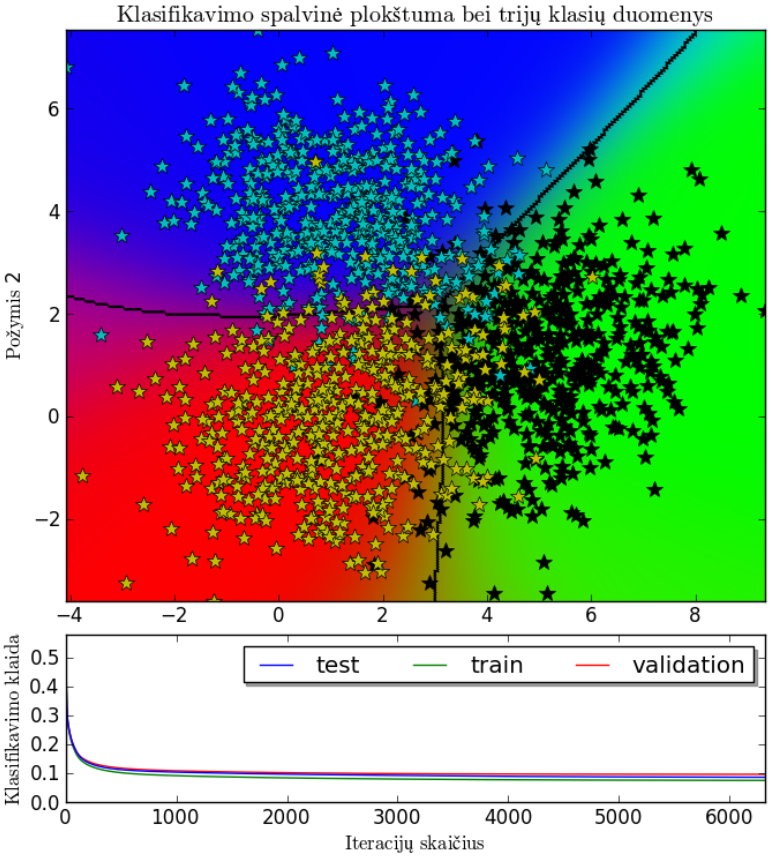
   (a) **Optimaliam sustojimui nustatyti naudojant validavimo duomenis** Tinklui pateikti duomenys yra suskaidomi į tris dalis: mokymo, validavimo bei testavimo. Duomenys į dalis yra skaidomi nuosekliai: mokymo duomenys yra pirmieji s elementų, validavimo duomenys - nuo s iki l, o testavimo duomenys - nuo l iki paskutinio elemento. Dėl šios specifikos skirtingų klasių duomenų vektorių pasiskirstymas pateiktųjų duomenų aibės skirtinguose poaibiuose turėtų būti panašus. Sugeneruotus sintetinius skirtingų klasių duomenis reikia atsitiktinai išmaišyti. Mokant daugiasluoksnį perceptroną, kiekvienos epochos metu yra apskaičiuojami nuostoliai su mokymo, validavimo. Nuostoliai su mokymo duomenimis nuolatos mažėja. Su validavimo duomenimis iš pradžių mažėja, bet po kurio laiko pradeda didėti, nes klasifikatorius per daug prisitaiko prie mokymo duomenų. Optimalus mokymo iteracijų skaičius yra randamas nustatant iteracijos indeksą, ties kuriuo nuostoliai su validavimo duomenim pradeda didėti. Naudojant nedidelę validavimo duomenų aibę, gali reikėti atlikti labai didelį iteracijų skaičių, kad klasifikavimo klaida pradėtų augti. Šiai problemai spręsti buvo pasirinkta tokia strategija: optimaliu iteracijų skaičiumi laikomas indeksas iteracijos, ties kuria nuostolių su validavimo duomenimis kitimas yra mažesnis nei pasirinkta konstanta $MAX\_VALIDATION\_ERROR\_SLOPE$. Kitaip sakant, mokymas

yra stabdomas, kai nuostoliai su validavimo duomenimis beveik nebekinta. Eksperimentų metu naudotos šios konstantos reikšmės yra 10^{-7} bei 0. Šią problemą buvo galima spręsti ir kitais metodais, pavyzdžiui, validavimo duomenų aibę praplečiant užtriukšmintais duomenimis. Visgi buvo pastebėta, kad triukšmo, kuris išlaiko užtriukšminamų duomenų skirstinį, generavimo realizacija reikalauja didelių laiko sąnaudų ir galėtų būti atskiro tyrimo objektas.

(b) **Mokant fiksuotą iteracijų skaičių ir parenkant mažiausią kainą su testiniais duomenimis gavusią iteraciją** Tinklui pateikti duomenys suskaidomi į dvi dalis: mokymo ir testavimo. Daugiasluoksnis perceptronas mokomas fiksuotą iteracijų skaičių, kiekvienos iteracijos metu yra apskaičiuojama kaina su testiniais duomenimis. Baigus mokymą yra parenkamas iteracijos numeris su geriausia kaina ir tinklas nuo tų pačių pradinių svorių pakartotinai apmokomas iki geriausios iteracijos.

5. Siekiant detaliai nagrinėti realizuoto klasifikatoriaus savybes, buvo realizuotas klasifikavimo skirstinio vizualizavimas dviejų bei trijų klasių atvejais (esant dviejų požymių duomenų vektoriams). Vizualizavimas atliekamas kiekvienam daugiasluoksnio perceptrono išėjimui priskiriant vieną iš RGB (raudona, žalia, mėlyna) spalvų. Visi mokymo duomenų vektoriai yra apibrėžiami stačiakampe plokštuma, kuri yra suskaidoma į smulkius kvadratėlius. Parenkama po vieną kiekvieno kvadratėlio tašką, apskaičiuojamos tų taškų spalvos bei kvadratėliai nuspalvinami jiems priklausančio taško spalva. Realizavus šį vizualizavimą buvo pastebėta, kad ne visai aišku, kurioje vietoje dominuojanti RGB spalva pasikeičia, todėl buvo pridėtas papildomas kvadratėlių spalvų modifikacijos algoritmas: vienas iš gretimų kvadratėlių, kurių dominuojanti spalva skiriasi, yra nuspalvinamas juodai. Taip vizualizuojama aiški riba tarp dviejų skirtingų klasių. Ant šios spalvinio klasių skirstinio plokštumos pavaizduojant duomenų vektorius galima matyti, kurie duomenų vektoriai kuriai klasei buvo priskirti (žr. 3 pav.).

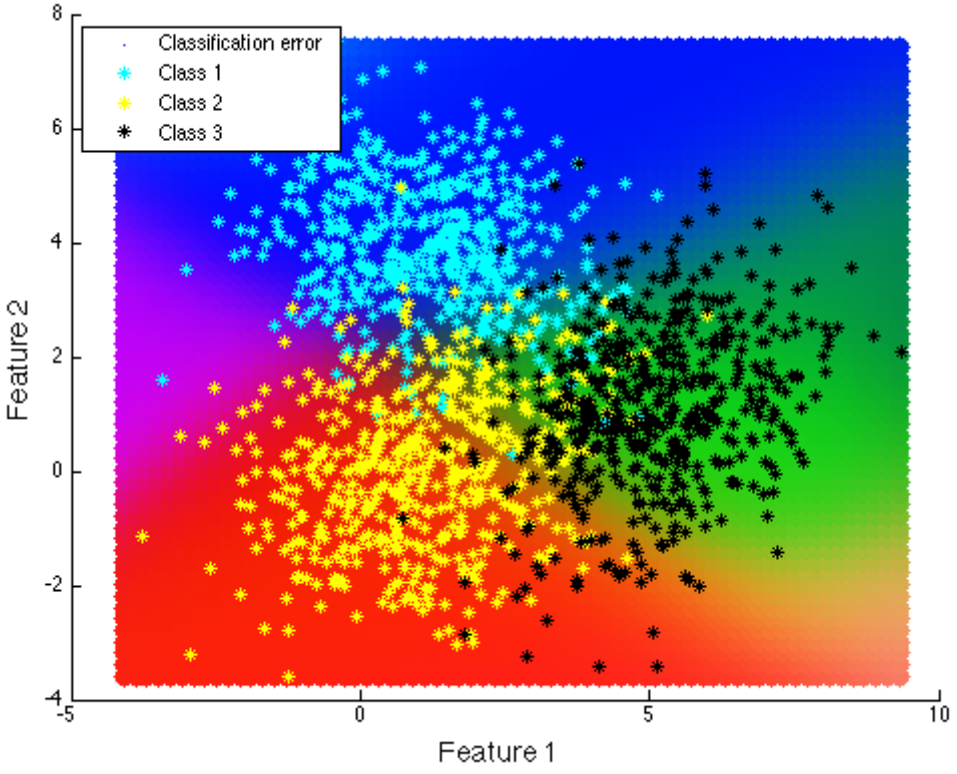
6. Gauto klasifikatoriaus patikimumui įvertinti pasirinkta naudoti vidutinę klasifikavimo kainą su testavimo duomenis. Kitaip sakant, su testiniais duomenimis yra apskaičiuojama vidutinė klasifikavimo kaina, kuri apskaičiuojama susumavus kiekvienos rūšies klaidas padaugintas iš tos rūšies klaidos kainos bei gautą sumą padalinus iš vektorių skaičiaus. Taip pat, kaip pagalbinės metrikas pasirinkta naudoti vidutinę klasifikavimo klaidą su mokymo duomenimis bei vidutinį klaidų skaičių su mokymo ir testavimo duomenimis.



3 pav.: Klasifikavimo skirstinio vizualizavimas trijų klasių atveju

Realizavus MLP klasę buvo sugeneruoti trijų klasių duomenys, turintys du požymius (žr. 3 pav.). Matlab aplinkoje sukurtas neuroninis tinklas, turintis 2 įėjimo, 7 paslėpto ir 3 išėjimo sluoksnio perceptronus. Pasirinkti 7 paslėpti perceptronai, nors trims klasėms atskirti užtenka ir 3 perceptronų, nes netinkamai inicializavus tinklą kai kurie perceptronai gali atlikti labai panašų klasifikavimą. Realizuotas Matlab aplinkai klasifikavimo skirstinio vizualizavimas. Daugiasluoksnis perceptronas apmokytas greičiausio nusileidimo metodu (angl. gradient descent):

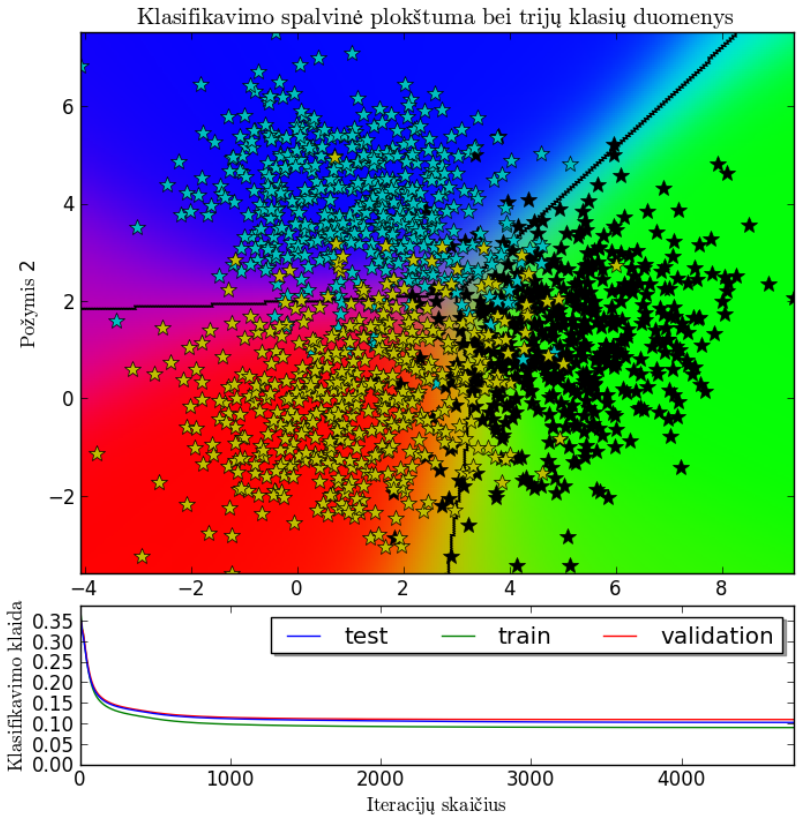
1. Įvykdyta 5000 mokymo epochų;
2. Mokymas truko 2:01 minutes;
3. Nuostoliai, gauti su pradiniais svoriais: 0.395;
4. Nuostoliai, gauti su mokymo metu gautais svoriais: 0.0831;
5. Gautas klasifikavimo skirstinys (žr. 4 pav.).



4 pav.: Matlab aplinkos daugiasluoksnio perceptrono ([2, 7, 3]), apmokyto greičiausio nusileidimo metodu, klasifikavimo skirstinys

MLP klasės pagalba, sukurtas analogiškas tinklas, kurio struktūra $[2, 5, 3]$. Šis tinklas apmokytas panaudojant tuos pačius trijų klasių duomenis (žr. 3 pav.). Mokymas buvo atliekamas naudojant greičiausio nusileidimo metodą (žr. (6)):

1. Įvykdyta 5000 mokymo epochų;
2. Mokymas truko 00:34 minutes;
3. Nuostoliai, gauti su pradiniais svoriais: 0.455;
4. Nuostoliai, gauti su mokymo metu gautais svoriais: 0.0867;
5. Gautas klasifikavimo skirstinys (žr. 5 pav.).



5 pav.: Python aplinkoje realizuoto daugiasluoksnio perceptrono ([2, 7, 3]), apmokyto greičiausio nusileidimo metodu klasifikavimo skirstinys

Svarbu pastebėti, kad *Matlab* aplinkos *NeuralNetworktoolbox* daugiasluoksnio perceptrono naudojamų svorių apibrėžimas skiriasi nuo įprasto svorių apibrėžimo (apibrėžėme 2.1.1 skyriuje). Todėl nebuvo galima įvykdyti eksperimento skirtingose aplinkose su vienodais pradiniais svoriais.

Gautuosius vykdymo rezultatus galima apibendrinti taip:

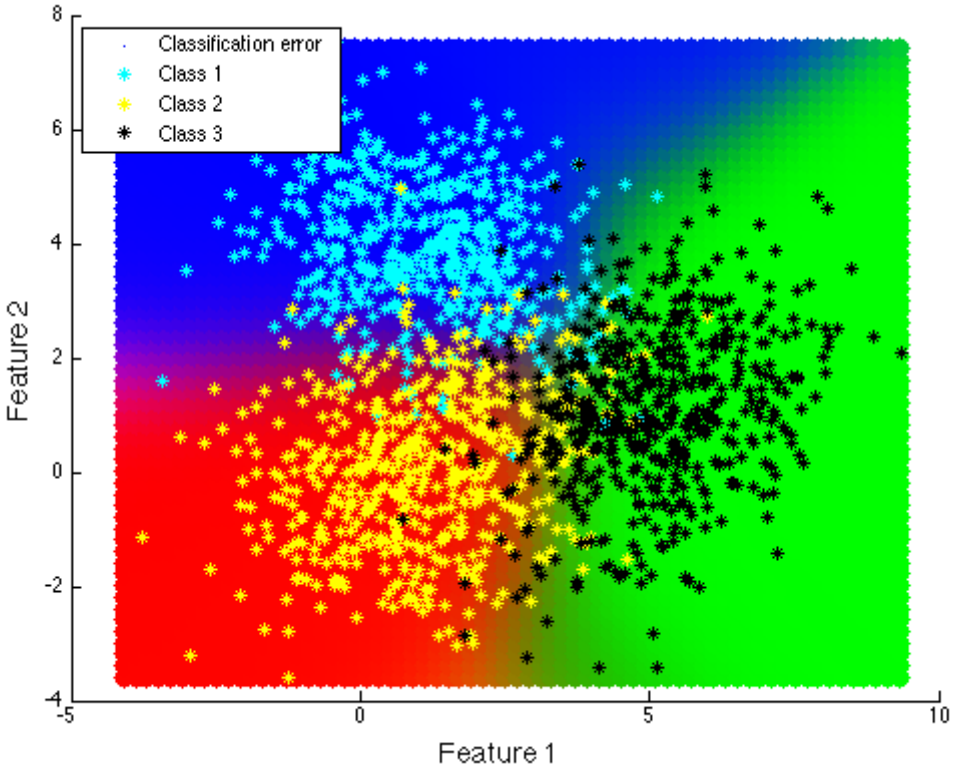
1. *MLP* klasės pagalba realizuoto tinklo epochos trukmė vidutiniškai keturis kartus trumpesnė nei Matlab aplinkoje realizuoto tinklo mokymo epochos.

2. Nuostoliai validavimo duomenims, gaunami su pradiniais abiejų aplinkų klasifikatorių svoriais, buvo panašūs, todėl pradinės mokymo sąlygos buvo artimos (Matlab aplinkoje aplinkybės net geresnės). Nuostoliai validavimo duomenims, gaunami su mokymo metu gautais svoriais, taip pat panašūs, todėl daroma išvada, kad šių klasifikatorių klasifikavimo galimybės yra analogiškos.

Matlab aplinkos *NeuralNetworkToolbox* pakete yra realizuoti keli tinklo apmokymo metodai, pavyzdžiui, *Levenberg-Marquardt* ar *Scaled conjugate gradient*.

Eksperimentas buvo dar kartą pakartotas *Matlab* aplinkoje, panaudojant kitą (*Scaled conjugate gradient*) tinklo apmokymo metodą:

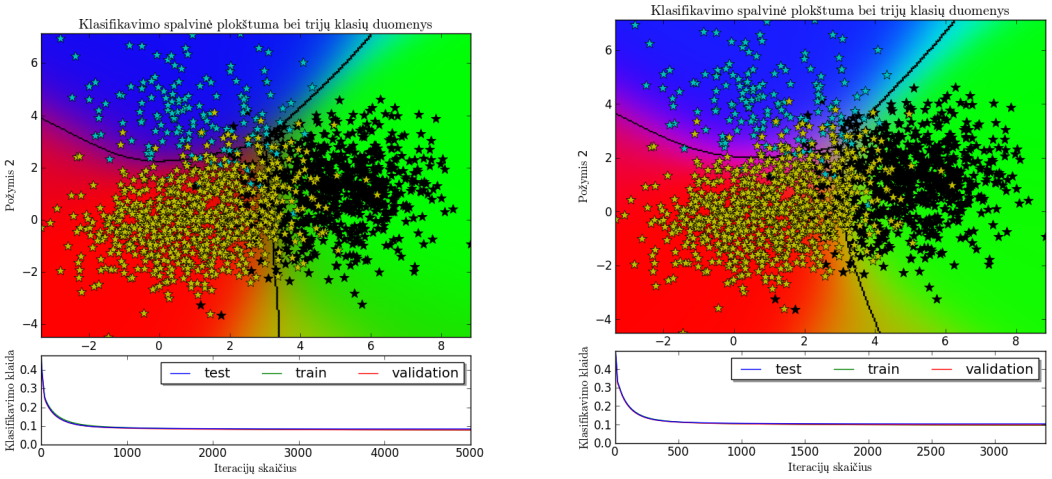
1. Įvykdytos 57 mokymo epochos;
2. Mokymas truko 00:02 minutės;
3. Nuostoliai, gauti su pradiniais svoriais: 0.393;
4. Nuostoliai, gauti su mokymo metu gautais svoriais: 0.0538;
5. Gautas klasifikavimo skirstinys (žr. 6 pav.).



6 pav.: Matlab aplinkos daugiasluoksnio perceptrono ([2, 7, 3]), apmokyto "Scaled conjugate gradient"metodu klasifikavimo skirstinys

Visų trijų eksperimentų nuostoliai su pradiniais svoriais yra panašūs, todėl galima teigti, kad šių metodų pradinės mokymosi sąlygos buvo identiškos. Skirtingose aplinkose išgauti *greičiausio nusileidimo* tinklo apmokymo metodų rezultatai yra labai panašūs, lyginant su *Scaled conjugate gradient* tinklo apmokymo metodo rezultatais. Iš šio eksperimento galima padaryti išvadą, kad *greičiausio nusileidimo* metodas konverguoja lėčiau, nei *scaled conjugate gradient* metodas.

Duomenų subalansavimas. *MLP* klasės realizacija buvo papildyta (20) duomenų subalansavimo savybe. Susikurti sintetiniai išbalansuoti duomenys. Tinklas apmokytas su savybe subalansuoti duomenis ir be jos. Vizualiai palyginti gautieji klasifikavimo skirstiniai (žr. 7a pav. bei 7b pav.). Vizualiai palyginus gautus klasifikavimo skirstinius, galima daryti išvadą, kad duomenų subalansavimo patobulinimas buvo įdiegtas sėkmingai.

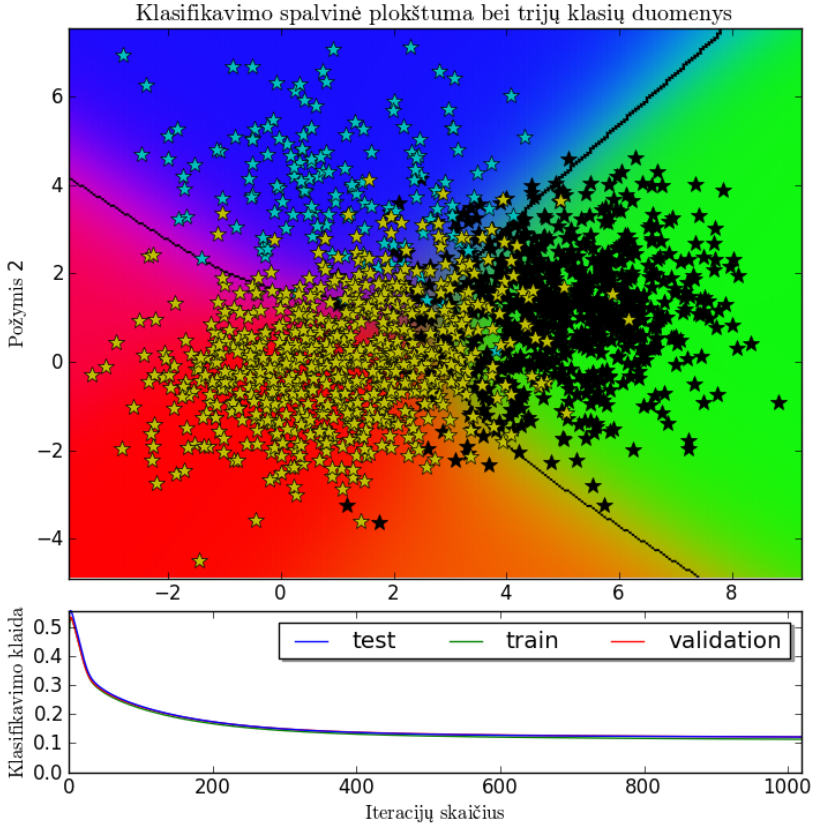


(a) Neįvertinant duomenų išbalansavimo

(b) Subalansuojant duomenis

7 pav.: Python aplinkoje realizuoto daugiasluoksnio perceptrono ([2, 7, 3]) klasifikavimo skirstinys.

Klasifikavimo klaidų kainų įtraukimas. *MLP* klasės realizacija papildyta (19) **paporinių** kainų matricos įvedimu. Sudaryta klasifikavimo klaidų kainų matrica, kurioje mažesniosios klasės kaina tris kartus didesnė nei kitų klasių. Vizualiai palyginti gautieji klasifikavimo skirstiniai (žr. 8 pav. ir 7b pav.). Matyti, kad klasifikavimo skirstinys pasikeitė mažosios klasės naudai taip, kad beveik neliko mažosios klasės elementų priskyrimo kitoms klasės klaidų. Padaryta išvada, kad klasifikavimo klaidų kainų įvedimas realizuotas sėkmingai.



8 pav.: Python aplinkoje realizuoto daugiasluoksnio perceptrono ([2, 7, 3]), subalansuojančio duomenis, bei įvertinančio klasifikavimo klaidų kainas (mažiausios klasės klaidos 3 kartus brangesnės už kitų), klasifikavimo skirstinys

Atsiradus poreikiui įvertinti dar nerealizuotas metrikas, MLP klasės realizacija buvo papildyta klasifikavimo klaidų skaičiaus radimo algorimu. Taip pat MLP mokymo algoritmas papildytas kitais mokymo sustojimo kriterijais: maksimalus iteracijų skaičius, maksimali vykdymo trukmė, nulinė klasifikavimo klaida.

5.3. Sprendimo medžių realizacija

Šiame skyriuje pateikiami realizuoti jautrių kainoms sprendimo medžių variantai. Jie pasirinkti remiantis skyrelyje 3.2.5 pateiktomis literatūros apžvalgos išvadomis. Realizacijos koncentruojasi ties baziniu C4.5 algoritmu ir jo modifikacijomis: klasių tikimybių skaičiavimu, genėjimu ir daugelio medžių kainoms jautriu kombinavimu.

5.3.1. C4.5 algoritmo realizacija

Šiame darbe sprendimo medis konstruojamas remiantis rašytinės C4.5 versijos [Qui93] ir C5.0 atviro kodo realizacijos [Res11] idėjomis. Pateikiame esminius analizuotų ir realizuoto medžio konstravimo algoritmų panašumus ir skirtumus.

Tiek C5.0, tiek C4.5 numato galimybę klasifikuoti vektorius, turinčius diskrečiųjų ir/arba tolydžiųjų požymių, taip pat jungti diskrečiuosius požymius į poaibius. Dėl to, kad siekiama sukurti hibridinį SM ir DNT klasifikatorių, šiame darbe realizuotas konkretus C4.5 algoritmo variantas, galintis klasifikuoti tik tolydžiuosius duomenis ir sukuriantis dvejetainius medžius. Kiekvienas medžio testas turi pavidalą $X \leq A$, kur X - tam tikras požymis, o $A \in \mathbb{R}$. Taip pat realizuotas ir klaidomis grįstas genėjimas. Abi algoritmo dalys aprašytos 3.1.1 skyrelyje.

Apribojimas naudoti tik tolydžiuosius požymius supaprastina informacijos teorijos formulių išraiškas. Realizacijoje naudoti tokie jų variantai:

1. Testo informacija (24) pakeičiama į:

$$Info_{test}(D_l, D_r, D) = \frac{|D_l|}{|D|} Info(D_l) + \frac{|D_r|}{|D|} Info(D_r) \ , \tag{41}$$

   D_l ir D_r yra kairiajam ir dešiniajam pomedžiui priklausančios D reikšmės, pritaikius testą $X \leq A$.

2. Dalijimo informacija (26) pakeičiama į:

$$Info_{split}(D_l, D_r, D) = -\frac{|D_l|}{|D|} log_2 \left(\frac{|D_l|}{|D|} \right) - \frac{|D_r|}{|D|} log_2 \left(\frac{|D_r|}{|D|} \right) \tag{42}$$

Realizuotas algoritmas nenumato praleistų reikšmių ir klasifikavimui nenaudotinų požymių galimybės.

Gautas medis C5.0 algoritme kiekviename lape pateikia faktinį juo padengiamų mokymo vektorių ir klasifikavimo klaidų skaičių. C4.5 algoritme rodomas ne faktinis klaidų skaičius, bet pesimistinis jo įvertis remiantis binominio pasisikirstymo parametro pasikliautiniu intervalu. C5.0 algoritmo realizacijoje ir šiame darbe genėjimas atliekamas ne tik pilnai užauginus medį, tačiau ir sukonstravus kiekvieną pomedį yra pirmiausia patikrinama, ar faktinis padarytų klaidų skaičius nebūtų mažesnis vietoj šio pomedžio kuriant lapą. C4.5 genėjimo versija realizuota ir šiame darbe.

[Qui93] bendrai paaiškinama, kaip parenkama ribinė reikšmė A tolydžiojo požymio testui $X \leq A$ vidiniame medžio mazge (28). Taip pat nurodoma, kaip tarpusavyje lyginti požymius kandidatus taikant santykinio informacijos išlošio kriterijų (27). Lieka neaišku, kokiu kriterijumi remiantis tarpusavyje lyginti vieno požymio

skirtingas įmanomas ribines reikšmes. Kodo analizė atskleidė, kad C5.0 tolydžiojo požymio geriausia ribine reikšme laikoma ta, kuri užtikrina didžiausią informacijos išlošį (25). Išsaugoma pati reikšmė, atitinkamas informacijos išlošis ir dalijimo informacija. Toks pat principas įgyvendintas ir šiame darbe.

Tiek C5.0 algoritme, tiek šio darbo realizacijoje tarpusavyje visi požymiai lyginami taikant santykinio informacijos išlošio kriterijų. Pirmiausia apskaičiuojamas teigiamų išlošių vidurkis visiems požymiams.

$$Avg\ gain(D) = \frac{1}{|Tests|} \sum_{T \in Tests} Gain(D,T)\ , \tag{43}$$

$Tests$ - visų įmanomų pavidalo $X \leq X^{thres}$ testų aibė.

Tuomet parenkamas požymis, turintis nemažesnį nei vidutinis informacijos išlošį ($Gain(D,T) \geq Avg\ gain(D)$), teigiamą dalijimo informaciją ($Info_{split}(D,T) > 0$) ir didžiausią iš visų santykinį informacijos išlošį $T = \arg\max_{T \in |Tests|} (Gain\ ratio(D,T))$.

Šio darbo realizacijoje nugenėtam medžiui, kaip ir C5.0 realizacijoje, atspausdinama skaičių pora (n/E), kur:

- n - kaip ir anksčiau, lapo padengiamų mokymo vektorių skaičius;
- E - klaidų skaičiaus lape įvertis remiantis binominio skirstinio pasikliautinio intervalo skaičiavimu su mokymosi duomenimis: $\bar{E}$, jei tai negenėtas lapas, arba $\bar{E}_l$, jei tai pomedis, pakeistas lapu.

C4.5 klaidomis grįsto genėjimo parametro p pesimistiniam įverčiui skaičiuoti naudota Agresti-Coull [Agr98] pasikliautinio intervalo skaičiavimo metodika:

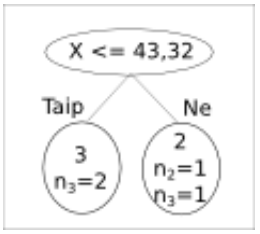
$$\bar{p}(E,n,\alpha) = \tilde{p} + z_{\alpha/2}\sqrt{\frac{\tilde{p}(1-\tilde{p})}{n}} \tag{44}$$

Čia $\tilde{p} = \frac{E+\frac{z_{\alpha/2}}{2}}{n+{z_{\alpha/2}}^2}$, o $z_{\alpha/2}$ yra standartinio normaliojo skirstinio $(1-\alpha/2)$-is kvantilis.

Pateikiame sprendimo medžio, pateikto 9 Pav., genėjimo pavyzdį. Medis turi vieną sprendimo priėmimo mazgą ir du lapus. Kiekviename jų išsaugota, kiek kurios klasės vektorių iš mokymo duomenų jie padengė: n_i žymi i-ai klasei priklausančių vektorių skaičių.

Medžio pavyzdyje antrasis lapas padengia $n = 1 + 1 = 2$ vektorius, iš kurių $E = 1$ klasifikuojamas klaidingai, lapui priskyrus 2 klasę. Dėl to klaidų skaičiaus įvertis šiame lape su pasikliovimo lygmenu $Q = 0.75$, čia $\alpha = 1 - Q = 0.25$, yra $\bar{E} = 2 \times \bar{p}(1, 2, 0.25) \approx 1.76$.

Medis pavyzdyje galėtų būti pakeistas lapu, padengiančiu $n = 2 + 1 + 1 = 4$ vektorius ir žyminčiu 3 (daugumos) klasę su klaida $E = 1$. Kai $Q = 0.75$, $\bar{E}_l = 4 \times \bar{p}(1, 4, 0.25) \approx 2.23$, o $\bar{E}_v = 2 \times \bar{p}(0, 2, 0.25) + 2 \times \bar{p}(1, 2, 0.25) \approx 2.72$. Kadangi $\bar{E}_l < \bar{E}_v$, vietoj medžio būtų paliktas lapas.



9 pav.: Sprendimo medžio, skirto genėti, pavyzdys.

5.3.2. Klasių tikimybių modifikavimas

Antrasis šiame darbe realizuotas metodas yra C4.5 modifikacija, atsižvelgianti į klaidų kainas medžio konstravimo fazėje, kai standartiniai santykiniais dažniais paremti apriorinių klasių tikimybių įverčiai (22) pakeičiami modifikuotaisiais pagal kainų matricą (30) aibės informacijos skaičiavimo formulėje (23). Kadangi tai originalaus algoritmo modifikacija, nėra žinoma, kaip medžio veikimą paveiks standartinis klaidomis grįstas genėjimas, dėl to turėtų būti atliekami eksperimentai su ir be jo.

5.3.3. Medžio konstravimas MetaCost algoritmu

Atlikus literatūroje rastų algoritmų analizę buvo pasirinkta naudoti daugelio medžių algoritmą MetaCost [Dom99], aprašytą 3.2.2 skyrelyje, kuris klaidų kainoms jautrų sprendimo medį išgauna remiantis *bagging* metodika. MetaCost algoritmas nenusako, kokį konkretų pavienių sprendimo medžių konstarvimo algoritmą (ar apskritai klasifikatorių) reikia naudoti. Šiame darbe buvo pasirinkta naudoti jau realizuotą C4.5 medžio konstravimo algoritmą.

Šiame darbe realizuoto MetaCost algoritmo pseudokodas pateiktas 1 algoritme. Įeities duomenys yra: S - mokymo duomenys, L - klasifikavimo algoritmas, C - kainų matrica, m - mokymo duomenų vektorių atsitiktinių rinkinių skaičius, n - vektorių skaičius atsitiktiniame mokymo duomenų poaibyje.

MetaCost algoritmas yra daugelio medžių (*bagging*) algoritmas, bet jo efektyvumas skiriasi tik per konstantą, atitinkančią balsuojančių medžių skaičių, nuo pavienio klaidų kainoms nejautraus klasifikatoriaus.

Testuojant sukurtą realizaciją pastebėta, kad MetaCost algoritmas geriau nei pavienis sprendimo medis susidoroja su užtriukšmintais duomenimis. Šis efektas gaunamas, nes mažesni duomenų rinkiniai nei pradinė mokymo duomenų aibė nufiltruoja triukšmą.

Taip pat pastebėta, kad šio algoritmo trūkumas yra atsitiktinumo faktorius, kuris lemia, kad MetaCost algoritmas dažniausiai, bet ne visada sugeneruoja rezultatą,

1 Algoritmas MetaCost algoritmo pseudokodas.

procedure MetaCost(S, L, C, m, n)
  for $i \leftarrow 1, n$ **do**
    S_i yra n ilgio vektorių rinkinys iš S aibės
    M_i yra klasifikatorius, gautas L algoritmą apmokius su S_i
  end for
  $S' \leftarrow \emptyset$
  for all $x \in S$ **do**
    for all $j \in classes$ **do**
      $P(j|x) \leftarrow \frac{1}{\sum_i 1} \sum_i P(j|x, M_i)$, kur
      $\forall i P(j|x, M_i) = \begin{cases} 1 & : \text{jei } M_i \text{ prognozuoja klasę j} \\ 0 & : \text{priešingu atveju} \end{cases}$
    end for
    x nauja klasė $\leftarrow \arg\min_i \sum_j P(j|x)C(i,j)$
    $S' \leftarrow S' \cup x$
  end for
  return Klasifikatorius, gautas apmokius algoritmą L su kainų atžvilgiu palankiausiai perklasifikuotais duomenimis S'
end procedure

kurio kaina mažesnė už pavienio klaidų kainoms nejautraus klasifikatoriaus. Ši problema gali būti išspręsta, įvykdžius algoritmą kelis kartus pakartotinai ir parenkant geriausią rezultatą.

5.3.4. Laplace genėjimo realizacija

Kadangi šio darbo kontekste sprendimo medis skirtas daugiasluoksniam perceptronui inicializuoti, naudinga, kad gauto medžio dydis, o dėl to ir tinklo dydis būtų optimalus. Dėl to ieškota tinkamo genėjimo varianto.

Realizuotas C4.5 klasifikatorius sugeba įvertinti klasių tikimybes lapuose ir vidiniuose mazguose. Be to, numatytoji klaidomis grįsta sprendimo medžio genėjimo metodika nėra jautri klaidų kainoms. Dėl to pasirinkta jam pritaikyti 3.2.4 skyrelyje aprašytą kainos jautrią Laplace genėjimo metodiką. Šiame darbe sukurta jos realizacija, kuri, taikoma kainoms nejautriam C4.5, padaro jį kainoms jautrų.

5.4. SM ir DNT apjungimo realizacija

Realizuotas [Ban97] siūlomas sprendimo medžių ir dirbtinių neuroninių tinklų apjungimo metodas (žr. skyrelį 4.1). Sukonstravus sprendimo medį, jis apeinamas metodu į gylį ir pakeliui suformuojamos trys daugiasluoksnio perceptrono svorių matricos, kurios naudojamas tinklui inicializuoti.

2 Algoritmas Hibridinio klasifikatoriaus DSP pradinių svorių konstravimo iš sprendimo medžio algoritmas.

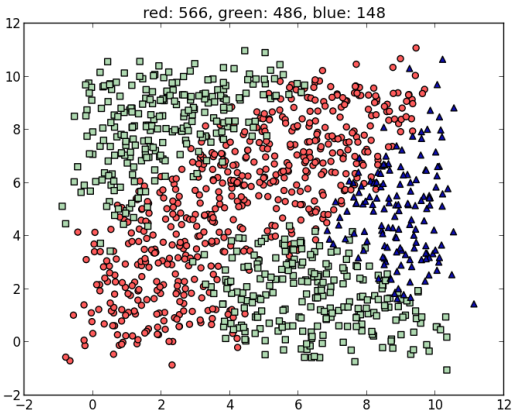
procedure Svorių Formavimas($tree, p, \sigma = 5.0, \beta = 0.025$)
  $rules \leftarrow \emptyset$
  for all $k \in$ keliai nuo šaknies iki lapo medyje $tree$ **do**
    $c \leftarrow$ lapo prognozuojama klasė
    $DNF_c \leftarrow$ disjunkcinės normaliosios formos taisyklė iš kelio k:
    pvz. $(x < A) \cap (y \geq B) \cap (z < C)$
    $rules \leftarrow rules \cup suprastinta\ DNF_c$
  end for
  ▷ *literal* sluoksnio formavimas
  $literal \leftarrow \emptyset$
  for all $rule \in rules$ **do**
    for all $(x > A) \in rule$ **do**
      $neuron \leftarrow$ neuronas, turintis jungtį σ su atributu x ir $\pm\beta$ su likusiais $p-1$ atributų ir $w_0 \leftarrow -\sigma \times A$
      $\_neuron \leftarrow$ neuronas, turintis priešingo ženklo svorius nei $neuron$ ir atitinkantis šaką $(x \leq A)$
      $literal \leftarrow literal \cup neuron \cup \_neuron$
    end for
  end for
  ▷ *conjunction* sluoksnio formavimas
  $conjunction \leftarrow \emptyset$
  for all $rule \in rules$ **do**
    $ntests \leftarrow$ testų skaičius taisyklėje $rule$
    for all $test \in rule$ **do**
      $neuron \leftarrow$ neuronas, turintis jungtį σ su *literal* sluoksnio neuronu, atitinkančiu testą $test$, jungtis $\pm\beta$ su visais kitais *literal* sluoksnio neuronais ir $w_0 \leftarrow -0.5\sigma(2 * ntests - 1)$
      $conjunction \leftarrow conjunction \cup neuron$
    end for
  end for
  ▷ *disjunction* sluoksnio formavimas
  $disjunction \leftarrow \emptyset$
  for all $c \in classes$ **do**
    $neuron \leftarrow$ neuronas, sujungtas su sluoksnio *conjunction* neuronais jungtimis σ, jei jie atitinka klasę c prognozuojančias taisykles, ir jungtimis $\pm\beta$ priešingu atveju, o $w_0 \leftarrow -0.5\sigma$
    $disjunction \leftarrow disjunction \cup neuron$
  end for
end procedure

Šiame kontekste DSP pirmasis paslėptas sluoksnis vadinamas *literal*, antrasis ir išėjimo atitinkamai - *conjuncion* ir *disjunction*. Tuomet medžio apėjimo pseudokodas pateiktas 2 algoritme. Įeities parametrai: *tree* - sprendimo medis, p - duomenų dimensiškumas, σ ir β - algoritmo parameterai, darantys įtaką jungčių svoriams.

5.5. Klasifikatoriaus konstravimo pavyzdys

Šiame skyrelyje vizualizuojame realizuotų hibridinių klasifikatorių konstravimo etapus.

Pavyzdžiai pateikiami su dvimačiais sintetiniais duomenimis, parodytais 10 pav. Naudota kainų matrica $C = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 10 & 10 & 0 \end{bmatrix}$, reiškianti, kad trečiosios (mažiausios, mėlynos) klasės klaidų kaina didžiausia ir jai turėtų būti skirtas didžiausias dėmesys. Pateikiama po du klasifikavimo rezultatus kiekvienai klasifikatoriaus fazei: klasifikavimas su mokymo ir testiniais duomenimis.



10 pav.: Sintetiniai duomenys, naudoti hibridinio klasifikatoriaus konstravimo pavyzdyje.

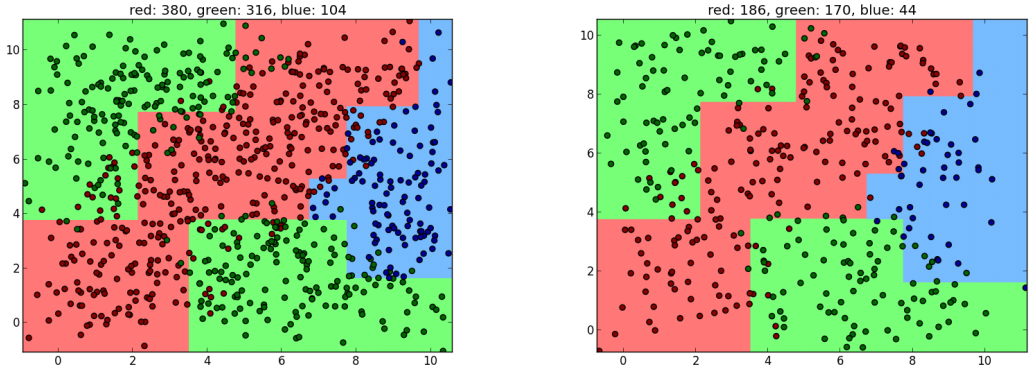
Pirmajame pavyzdyje (11 - 13 pav.) parodyti hibrido, inicializuoto iš sprendimo medžio, užauginto C4.5 algoritmu ir nugenėto kainoms jautriu Laplace genėjimu, konstravimo etapai. Toliau eksperimentuose jis vadinamas *Hybrid_Laplace*.

Kaip matyti iš 11 pav., sprendimo medis išskiria stambią kampuotą klasių struktūrą, padarydamas raudonos klasės klaidų tiek mėlyname, tiek žaliame regionuose. Toliau inicializuotas hibridinis klasifikatorius (12 pav.) atkartoja SM klasifikavimo plokštumą stambiais bruožais (laiptuota struktūra), tačiau ribų kampai aptakesni. Pilnai apmokyto hibrido, pateikto 13 pav., ribos dar labiau sušvelnintos ir nutolusios nuo pradinio SM, nors išlaikoma bendra plokštumos regionų kryptis ir įlinkimai anksčiau buvusių ryškių kampų vietose. Pilnai apmokytas hibridas gerokai sumažina raudonos klasės klaidų. Be to, hibrido klasifikavimo plokštumos regionai yra tolygiai nuspalvinti, o tai reiškia, kad klasifikatoriaus išėjimo sluoksnio perceptronų reikšmės skyrėsi pakankamai, priimant sprendimą dėl vektoriaus klasinės priklausomybės.

Antrajame pavyzdyje (14 - 16 pav.) etapais konstruojamas MetaCost sprendimo medžiu paremtas hibridas, toliau - *Hybrid_MetaCost*. Čia pradinis sprendimo medis konstruojamas iš mokymo vektorių, kurių klasinė priklausomybė pakeista pagal klasių matricą. Galutinis sprendimo medis užauginamas C4.5 metodu ir jam pritaikomas numatytasis klaidomis grįstas genėjimas, o jis laikomas kainoms jautriu būtent dėl naudotos modifikuotos vektorių aibės.

Pradinis sprendimo medis (14 pav.) skiriasi nuo prieš tai pateiktojo Laplace (11 pav.) tuo, kad klasių ribos stipriai dantytos ir prisitaikiusios prie mokymosi duomenų ypatybių. Dėl to, kad klasių ribos detalesnės, padaroma mažiau raudonos klasės klaidų. Taip gali būti dėl to, kad taikytas numatytasis klaidomis grįstas genėjimo metodas. Inicializuotas hibridas (15 pav.) sušvelnina klasių ribas, tačiau matyti, kad spalvos nevienalytės - yra regionų, kuriuose klasinė priklausomybė nustatoma sudėtingiau tikriausiai dėl ryškaus pradinio medžio dantytumo. Paskutiniame etape (16 pav.) klasifikatorius taip pat išlaiko banguotas klasių ribas (prastą apibendrinimo gebėjimą sąlygojantys išsikišimai, įdubimai), tačiau tampa vienareikšmiškesnis regionų viduje.

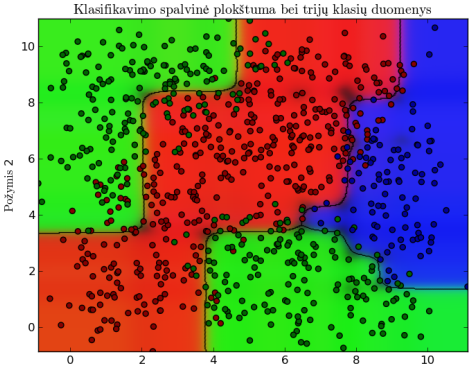
Iš pateiktų pavyzdžių matyti, kad *Laplace* medis dalija erdvę stambiais regionais, dėl to praranda tikslumo ten, kur klasės persidengia, o *MetaCost* - klasių regionų ribas padaro itin vingiuotas, dėl ko nukenčia klasifikatoriaus apibendrinimo pajėgumai. Atitinkamai medžiai padaro įtaką ir inicializuoto hibrido veikimui.



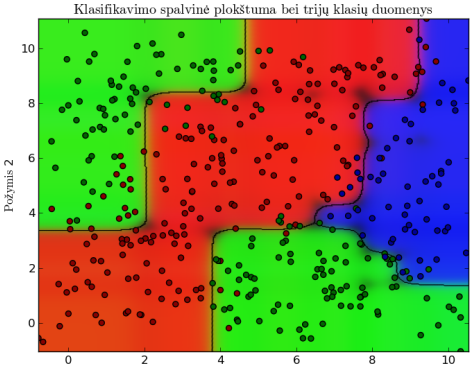
(a) Mokymo duomenys.

(b) Testiniai duomenys

11 pav.: Laplace sprendimo medis su mokymo ir testiniais duomenimis.

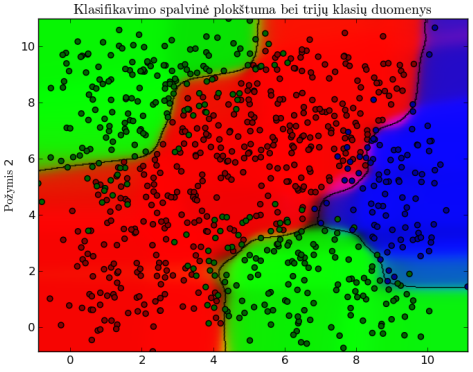


(a) Mokymo duomenys.

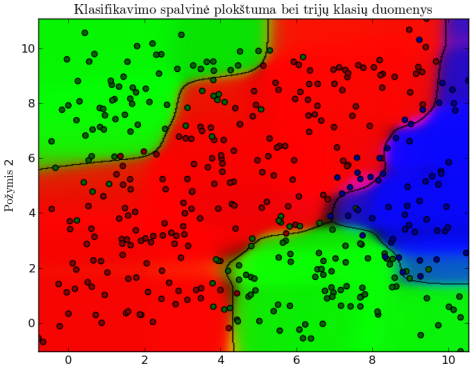


(b) Testiniai duomenys

12 pav.: Inicializuotas Laplace hibridas su mokymo ir testiniais duomenimis.

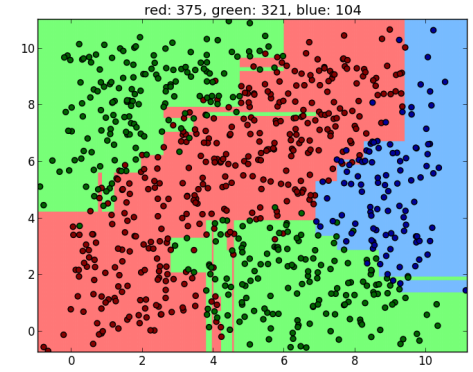


(a) Mokymo duomenys.

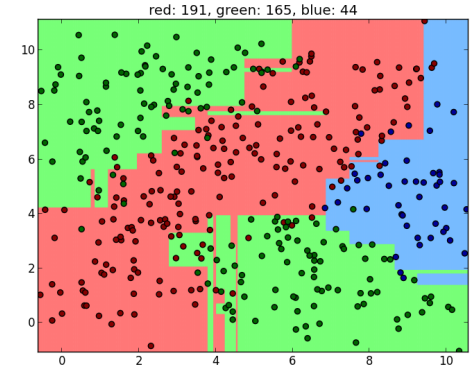


(b) Testiniai duomenys

13 pav.: Apmokytas Laplace hibridas su mokymo ir testiniais duomenimis.

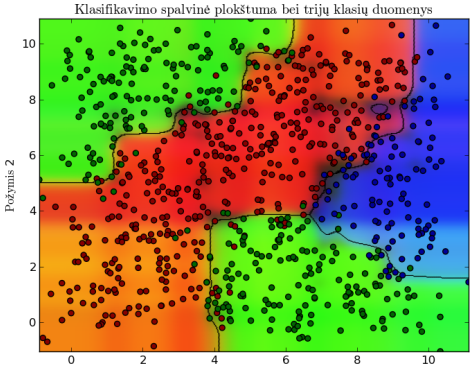


(a) Mokymo duomenys.

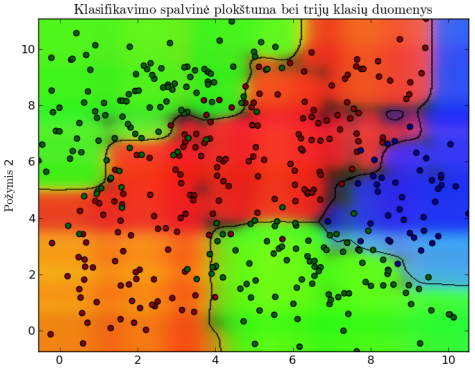


(b) Testiniai duomenys

14 pav.: MetaCost sprendimo medis su mokymo ir testiniais duomenimis.

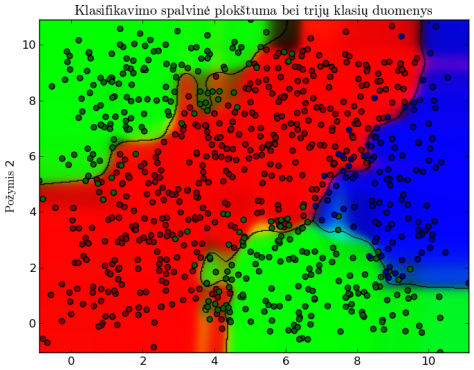


(a) Mokymo duomenys.

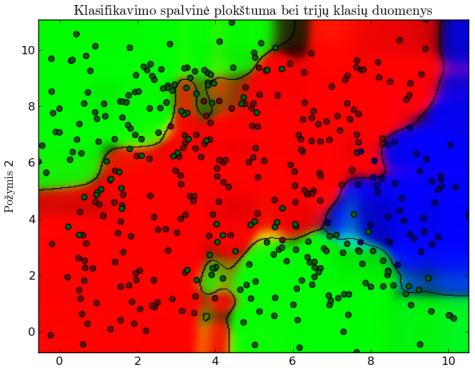


(b) Testiniai duomenys

15 pav.: Inicializuotas MetaCost hibridas su mokymo ir testiniais duomenimis.



(a) Mokymo duomenys.



(b) Testiniai duomenys

16 pav.: Apmokytas MetaCost hibridas su mokymo ir testiniais duomenimis.

6. Hibridinių klasifikatorių veikimo eksperimentinis tyrimas

6.1. Bendrieji eksperimentų nustatymai

Eksperimentais šiame skyrelyje siekiama nustatyti, ar apskritai yra prasminga kurti kainoms jautrų hibridinį klasifikatorių, panaudojant SM ir DNT hibridizacijos metodiką [Ban97] ir įvedant jautrumą kainoms kiekvienoje klasifikatoriaus dalyje nepriklausomai. Kad būtų prasminga, reikėtų parodyti, kad hibridinis klasifikatorius sugeba pasiekti mažesnę klasifikavimo kainą su testiniais duomenimis nei jį inicializavęs sprendimo medis per mažiau iteracijų nei tokios pat architektūros, tačiau atsitiktiniais svoriais inicializuotas DSP.

Šio darbo eksperimentuose tarpusavyje lyginami hibridiniai klasifikatoriai, pateikti 1 lentelėje. Jie skiriasi dalimis, iš kurių yra sudaryti, ir jautrumo kainoms užtikrinimo būdais, nurodytais dešiniajame lentelės stulpelyje:

1. **DSP**. Hibride naudojamas kainoms jautrus daugiasluoksnis perceptronas, aprašytas 5.2 skyrelyje.

2. **MetaCost**. Kainoms jautrus yra sprendimo medis, sukonstruotas gaubiamuoju metodu, aprašytu 5.3.3 skyrelyje.

3. **Genėjimas**. Kainoms jautrus Laplace medžio genėjimas, aprašytas 5.3.4 skyrelyje.

4. **Tikimybių modifikacija**. Klasių apriorinės tikimybės modifikuojamos, kad atitiktų kainų matricą, taip pat lapai pernumeruojami, kad atitiktų geriausią kainos atžvilgiu klasę, kaip aprašyta skyrelyje 5.3.2.

Verta pastebėti, kad *MetaCost* turi kaip bazinį klasifikatorių naudoti kainoms nejautrų klasifikatorių. Metodas suteikia galimybę perpanaudoti pirmoje jo stadijoje konstruojamą kainoms nejautrių klasifikatorių aibę kiekvienai naujai kainų matricai. Siekiant išsaugoti šią perpanaudojamumo galimybę, bazinis *MetaCost* naudojamas klasifikatorius neturi būti jautrus kainoms. Tiek genėjimas, tiek tikimybių modifikacija padarytų bazinį klasifikatorių jautrų kainoms, dėl to eksperimentuose jie nekombinuojami su *MetaCost*.

Papildomai į palyginimus įtraukti hibridai:

1. C5.0 sprendimo medžio klasifikatorius [Res11] bei juo inicializuotas hibridas *Hybrid_C5.0*. Sprendimo medis yra platinamas atviru kodu, o ne realizuotas šiame darbe.

2. Originalus kainoms nejautrus hibridas *Banerjee* [Ban97], aprašytas 4.1 skyrelyje. Sprendimo medžio daliai panaudotas šiame darbe realizuotas C4.5 algoritmas.

Be hibridų, eksperimentuojama su juos inicializavusiais medžiais. Taip pat į kai kuriuos eksperimentus įtraukiamas pigiausios klasės klasifikatorius, parenkantis vektoriui pigiausią pagal mokymo duomenis nustatytą klasę. Pigiausios klasės klasifikatorius naudojamas daugelyje kitų metodų kaip sudedamoji dalis ir yra pateiktas, pvz. (35) formulėje.

	Pavadinimas	Inicializuojantis medis	Kainoms jautrus algoritmo aspektas
1.	*Hybrid_C4.5*	C4.5 medis	DSP
2.	*Hybrid_Laplace*	C4.5 medis, Laplace genėjimas	Genėjimas, DSP
3.	*Hybrid_MetaCost*	MetaCost medis, naudojantis C4.5 medį kaip bazinį klasifikatorių	MetaCost, DSP
4.	*Hybrid_Mod_prob*	C4.5 medis, pakeitus apriorines klasių tikimybes ir lapų klases pagal kainų matricą	Tikimybių modifikacija ir lapų pernumeravimas, DSP
5.	*Hybrid_Mod_prob_err*	C4.5 medis, pakeitus apriorines klasių tikimybes pagal kainų matricą, klaidomis grįstas genėjimas	Tikimybių modifikacija, DSP
6.	*Hybrid_Mod_prob_lap*	C4.5 medis, pakeitus apriorines klasių tikimybes pagal kainų matricą, Laplaso genėjimas	Tikimybių modifikacija, genėjimas, DSP
7.	*Hybrid_C5.0*	C5.0 medis	C5.0 medis, DSP
8.	*Banerjee*	C4.5 medis	Kainoms nejautrus

1 lentelė.: Lyginami hibridiniai klasifikatoriai.

Eksperimentai atliekami su 2 lentelėje pateiktais sintetiniais ir realaus pasaulio duomenų rinkiniais. Realaus pasaulio duomenys eksperimentams pasirinkti iš UCI mašinų mokymo duomenų bazės [BKMM98], skirtos algoritmų kūrėjams. Žemiau pateikiame trumpus duomenų aprašymus.

Vertebral Column Data Set[4]. Trijų stuburo būklių pacientai, klasifikuojami pagal 6 biomechaninius požymius, nustatytus pagal dubens ir juosmeninės stuburo

[4]http://archive.ics.uci.edu/ml/datasets/Vertebral+Column, tikrinta 2012-12-01

dalies slankstelių padėtį ir orientaciją. Pateikti du rinkiniai, 2 ir 3 klasių parinktas 3 klasių.

Wine[5]. Trijų vyno rūšių cheminė sudėtis.

Seeds[6]. Pagal grūdų branduolių geometrinius matmenis, kaip plotas, perimetras, gautus iš Rentgeno nuotraukų, jie priskiriami vienai iš trijų rūšių.

Synth. Dvimačiai sintetiniai duomenys (žr. 17a pav.), sudaryti iš trijų persidengiančių normaliuoju skirstiniu pasiskirsčiusių klasių. Klasių vidurkiai yra $\mu_1 = [4, 5]$, $\mu_2 = [3, 2]$, $\mu_3 = [0, 5]$, o kovariacinės matricos lygios $\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$, kiekvienoje klasėje yra po $n = 350$ vektorių.

Set15. Dvimačiai, trijų klasių sintetiniai duomenys (žr. 17b pav.). Šie duomenys sugeneruoti taip: atsitiktinai sugeneruoti 400 antros klasės duomenų vektorių, kurių $x \in [0, 10]$, $y \in [0, 10]$ ir tenkina nelygybę $x + 1 < y$; atsitiktinai sugeneruoti 400 trečios klasės duomenų vektorių, kurių $x \in [0, 10]$, $y \in [0, 10]$ ir tenkina nelygybę $x - 1 > y$; atsitiktinai sugeneruoti 400 pirmosios klasės duomenų vektorių, kurių $x \in [0, 10]$, $y \in [0, 10]$ ir tenkina nelygybę $x + 3 > y > x - 3$. Prie visų duomenų vektorių koordinačių pridėtas triukšmas lygus $0{,}5 * n, n \sim N(0, 1)$.

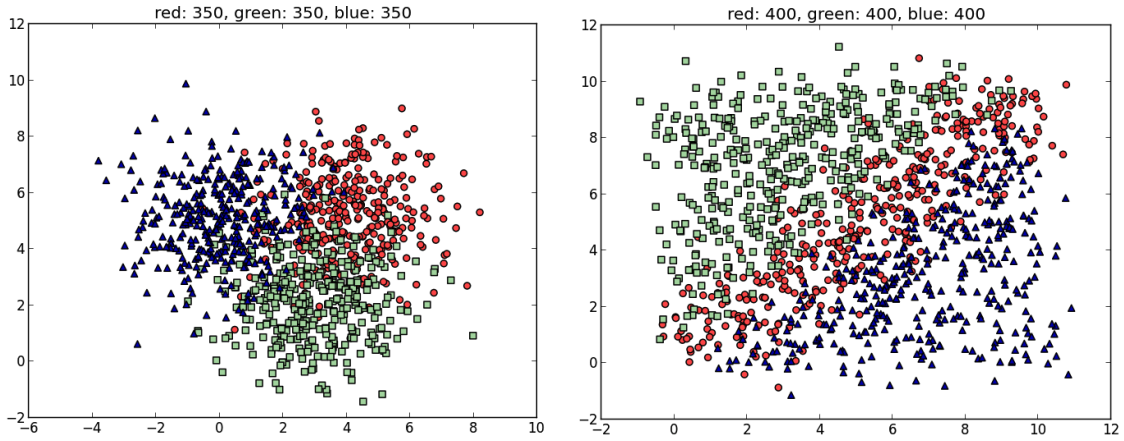
Nr.	Pavadinimas	Atributų	Vektorių	Klasių	Kiekvienos klasės
Realaus pasaulio duomenys					
1.	Vertebral	6	310	3	0: 60, 1: 150, 2: 100
2.	Wine	13	178	3	0: 59, 1: 71, 2: 48
3.	Tae	5	151	3	0: 49, 1: 50, 2: 52
4.	Seeds	7	210	3	0: 70, 1: 70, 2: 70
Sintetiniai duomenys					
5.	Synth	2	1050	3	0: 350, 1: 350, 2: 350
6.	Set15	2	1200	3	0: 400, 1: 400, 2: 400

2 lentelė.: Eksperimentuose naudotų duomenų rinkinių charakteristikos.

6.2. Eksperimentų vykdymas VU MIF superkompiuteryje

Algoritmai buvo realizuoti ir ištestuoti naudojant asmeninius kompiuterius. Pastebėjus, kad algoritmų apmokymas gali užtrukti ilgiau nei valandą, buvo pasinaudota Vilniaus Universiteto Matematikos ir Informatikos Fakultetas teikiama galimybe studentams naudotis superkomiuteriu[7]. Taigi, buvo sukurta eksperimentų vykdymo superkompiuteryje infrastruktūra, kuri įgalina eksperimentus vykdyti lygiagrečiai,

[5]http://archive.ics.uci.edu/ml/datasets/Wine, tikrinta 2012-12-01
[6]http://archive.ics.uci.edu/ml/datasets/seeds, tikrinta 2012-12-01
[7]http://mif.vu.lt/cluster/, tikrinta 2013-05-27



(a) Synth (b) Set15

17 pav.: Sintetiniai duomenų rinkiniai.

t.y. lygiagrečiai vykdyti nuosekliuosius algoritmus su skirtingais įeities parametrais. Bendri visiems algoritmams įeities parametrai: duomenų failas, kainų matrica, algoritmo išeigos direktorija ir tipas (generuoti paveikslėlius ar tik bendrą vykdymo statistiką).

Superkompiuteryje yra įdiegta:

1. SLURM[8] programinė įranga, naudota superkompiuterio užduočių valdymui

2. OpenMPI[9] programinė įranga užduočių vykdymui superkopiuteryje. Python programavimo aplinkoje naudojant OpenMPI apvalkalą MPI4PY[10] galima iš Python programavimo aplinkos valdyti OpenMPI programinę įrangą.

Taip pat, buvo realizuoti pagalbiniai įrankiai, leidžiantys vizualizuoti skirtingų algoritmų vykdymo rezultatus pagal pasirinktą metriką. Buvo registruojamos tokios algoritmų vykdymo metrikos: vidutinis klasifikavimo klaidų skaičius su mokymo ir su testavimo duomenimis, vidutinė klasifikavimo klaidų kaina su mokymo ir testiniais duomenimis, algoritmo vykdymo trukmė bei daugiasluosknį perceptroną naudojančių algoritmų iteracijų skaičius. Realizuoti tokie vizualizavimo įrankiai:

1. Skirtingų algoritmų su tuo pačiu duomenų rinkiniu ir ta pačia kainų matrica palyginamoji stulpelinė diagrama. Ši diagrama sudaroma suvidurkinant pasirinktą eksperimentų, tarpusavyje besiskiriančiu tik duomenų permaišymu, skaičių. Vizualizuojama pasirinktoji metrika.

[8]https://computing.llnl.gov/linux/slurm/, tikrinta 2013-05-27
[9]http://www.open-mpi.org/, tikrinta 2013-05-27
[10]http://mpi4py.scipy.org/docs/usrman/index.html, tikrinta 2013-05-27

2. Dviejų algoritmų palyginimui skirta sklaidos diagrama (angl. scatter plot). Tai dvimatė diagrama, kurios taškai žymi kiekvieno algoritmo metrikos įverčius, gautus atlikus eksperimentus, turinčius identiškas pradines sąlygas.

Realizuojant šią eksperimentų vykdymo infrastruktūrą buvo išspręstos šios techninės problemos:

1. Automatizuotas superkompiuterio užduočių generavimas ir paleidimas.
2. Automatizuotas rezultatų surinkimas, apibendrinimas ir vizualizavimas.
3. Skirtingų programavimo kalbų kodo vykdymas vienos užduoties metu superkompiuteryje (C5.0 sprendimo medžio realizacija pateikta C++ kalba, o visi kiti algoritmai realizuoti Python programavimo kalba).
4. Automatizuotas vykdymo rezulatatų parsisiuntimas į asmeninį kompiuterį iš superkompiuterio.

6.3. Eksperimentai su *Hybrid_Mod_prob*

6.3.1. Metodika

Pirmajame eksperimente, pasirinkus vieną iš pasiūlytų hibridų - *Hybrid_Mod_prob*, - atliekamas statistinis testas (t-testas vienodiems vidurkiams, kai dispersija nežinoma), siekiant nustatyti šių hipotezių teisingumą arba klaidingumą:

1.
   - H0 : sprendimo medžio ir juo inicializuoto hibrido vidutinė klasifikavimo klaidų kaina sutampa.
   - H1 : hibridinis klasifikatorius sumažina jį inicializavusio sprendimo medžio klasifikavimo klaidų kainą.
2.
   - H0 : originalaus kainoms nejautraus *Banerjee* ir hibrido *Hybrid_Mod_prob* vidutinė klasifikavimo klaidų kaina sutampa.
   - H1 : kainoms jautrus hibridas *Hybrid_Mod_prob* gauna mažesnę klasifikavimo klaidų kainą nei originalus *Banerjee* hibridas.
3.
   - H0 : atsitiktiniais svoriais inicializuotam DSP prireikia vidutiniškai tiek pat iteracijų pasiekti mažiausios kainos testiniams duomenims tašką kiek ir hibridui *Hybrid_Mod_prob*.
   - H1 : hibridas *Hybrid_Mod_prob* pasiekia geriausią būseną greičiau nei atsitiktinai inicializuotas DSP.

Siekiant patikrinti hipotezes, šiame eksperimente tarpusavyje lyginami keturi klasifikatoriai:

1. Kainoms jautrus sprendimo medis *Mod_prob*;
2. *Mod_prob* medžiu inicializuotas hibridinis klasifikatorius *Hybrid_Mod_prob*;
3. DSP, kurio struktūra (sluoksių ir neuronų skaičius) sutampa su *Hybrid_Mod_prob*, tačiau svoriai inicializuoti atsitiktinai;
4. Originalus kainoms nejautrus *Banerjee* hibridinis klasifikatorius.

Eksperimentuose naudojami du duomenų rinkiniai: *Synth* ir *Vertebral* (žr. 2 lentelę).

Siekiant teisingai suklasifikuoti mažiausios klasės vektorius, nes dažnai taikymuose jie būna patys svarbiausi, klaidų kainos, priskiriant jos vektorių kitai klasei, nustatytos didesnės už kitas:

$$C(i,j) = \begin{cases} 10: & \text{jei } i \text{ yra mažiausia klasė} \\ 0: & \text{if } i = j \\ 1: & \text{priešingu atveju} \end{cases} . \tag{45}$$

Kiekvienas duomenų rinkinys atsitiktinai permaišomas 100 kartų, tada kiekvienam duomenų permaišymui 2/3 duomenų naudojamos mokymui, o 1/3 - testavimui. Sprendimo medis apmokomas su mokymo duomenimis ir jo klasifikavimo klaidų kaina įvertinama su testiniais duomenimis. Hibridinis klasifikatorius inicializuojamas sprendimo medžiu, o DSP - ekvivalenčia struktūra, tačiau atsitiktiniais svoriais. Tiek hibridinis klasifikatorius, tiek DSP toliau mokomi su mokymo duomenimis iš anksto nustatytą kartą iteracijų: 2000 šiame darbe. Mokymo metu, apibendrinimo klasifikavimo klaidų kaina skaičiuojama su testiniais duomenimis. Tiek hibridui, tiek DSP geriausia kaina per visą procesą yra išsaugoma, o kartu ir ją atitinkančios iteracijos numeris. Tokiu būdu visiems trims klasifikatoriams gaunama po 100 klasifikavimo klaidų kainos įverčių, o hibridui ir DSP - papildomai dar po tiek pat geriausių iteracijų numerių.

Algoritmų palyginimo rezultatai pateikti 3 ir 4 lentelėse. Stulpelis *priimta* parodo, kuri iš hipotezių, suformuluotų šiame skyrelyje, priimta su $Q = 0{,}95$ patikimumo lygmeniu. Taip pat pateikti atitinkami dydžiai: $p-$reikšmė/2, T-statistika, palyginime dalyvavusių porų skaičius n, kiekvieno algoritmo rezultatų vidurkis ir dispersija. Hibridas *Hybrid_Mod_prob* lyginamas pagal klaidų kainą su sprendimo medžiu (1 hipotezės formuluotė) ir su originaliu hibridu *Banerjee* (2 hipotezės formuluotė) ir

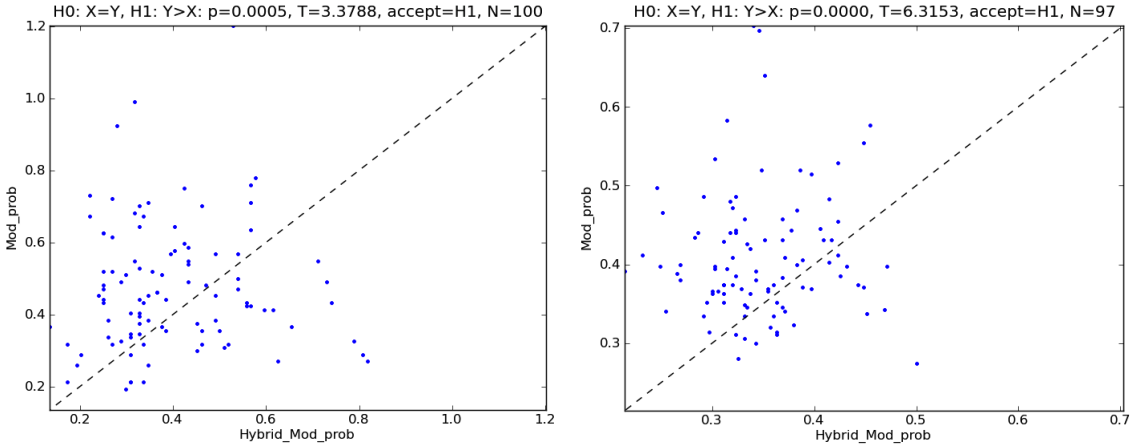
3 lentelė.: Algoritmų palyginimo rezultatai *Vertebral* duomenų rinkiniui.

Algoritmas	priimta	$p/2$	T	n	μ	s^2
Banerjee	H1	0.0000	9.6190	100	0.6753	0.2472
Hybrid_Mod_prob					0.3984	0.1479
Mod_prob	H1	0.0295	1.9107	100	0.4758	0.1747
Hybrid_Mod_prob					0.3984	0.1479
DSP	H1	0.000	7.7397	100	933.3899	668.4558
Hybrid_Mod_prob					285.0799	577.0025

4 lentelė.: Algoritmų palyginimo rezultatai *Synth* duomenų rinkiniui.

Algoritmas	priimta	$p/2$	T	n	μ	s^2
Banerjee	H1	0.0000	8.7049	97	0.4551	0.1035
Hybrid_Mod_prob					0.3477	0.0568
Mod_prob	H1	0.0000	6.3153	97	0.4102	0.0801
Hybrid_Mod_prob					0.3477	0.0568
DSP	H1	0.000	4.6431	97	226.6701	477.0487
Hybrid_Mod_prob					0.5773	0.5531

pagal iteracijų skaičių su atsitiktinai inicializuotu DSP (3 hipotezės formuluotė). Atitinkamos sklaidos diagramos pateiktos 18 - 20 pav.

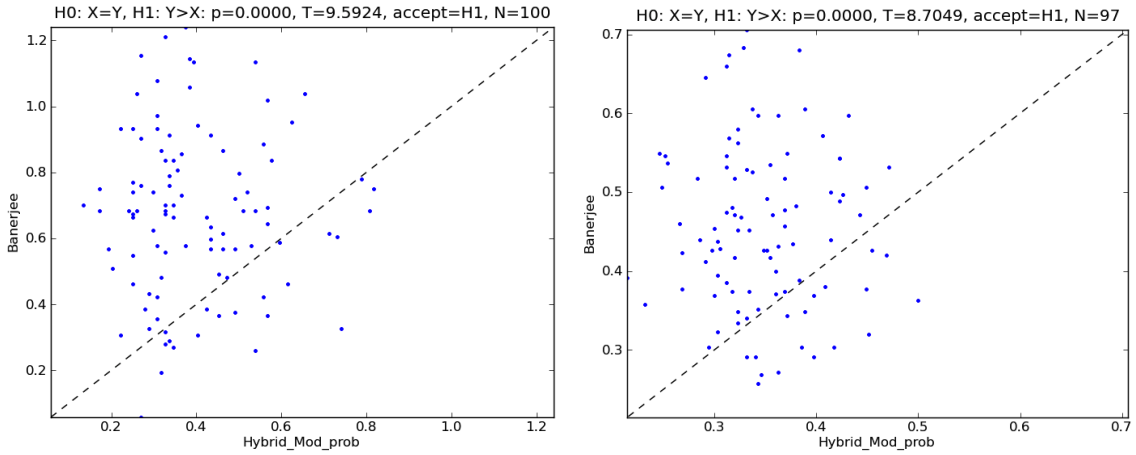


(a) *Vertebral* duomenų rinkinys. (b) *Synth* duomenų rinkinys.

18 pav.: Hibrido *Hybrid_Mod_prob* ir sprendimo medžio *Mod_prob* klaidų kainų palyginimas.

6.3.2. Aptarimas

Abiems duomenų rinkiniams kainoms jautrus hibridas *Hybrid_Mod_prob* gavo mažesnę už jį inicializavusio sprendimo medžio klasifikavimo klaidų kainą (3 ir 4 lentelėse ties *Mod_prob* ir *Hybrid_Mod_prob* skiltimis stulpelio *priimta* reikšmė yra



(a) *Vertebral* duomenų rinkinys. (b) *Synth* duomenų rinkinys.

19 pav.: Hibrido *Hybrid_Mod_prob* ir originalaus kainoms nejautraus hibrido *Banerjee* klaidų kainų palyginimas.

H1). Taip pat hibridas pagerino kainoms nejautraus *Banerjee* metodo klasifikavimą pagal klaidų kainą (ten pat, *Banerjee* ir *Hybrid_Mod_prob* skiltyse *priimta* H1). Be to, hibridinis klasifikatorius sugebėjo per mažesnį iteracijų skaičių nei atsitiktiniais svoriais inicializuotas DSP pasiekti mažiausios klasifikavimo klaidų kainos testiniams duomenims būseną (ten pat, *DSP* ir *Hybrid_Mod_prob* skiltyse *priimta* H1).

Iš pateiktų eksperimentų rezultatų galima daryti išvadą, kad egzistuoja duomenys, kuriems hibridinio klasifikatoriaus *Hybrid_Mod_prob* veikimas yra geresnis nei pavienių sprendimo medžio ar neuroninio tinklo.

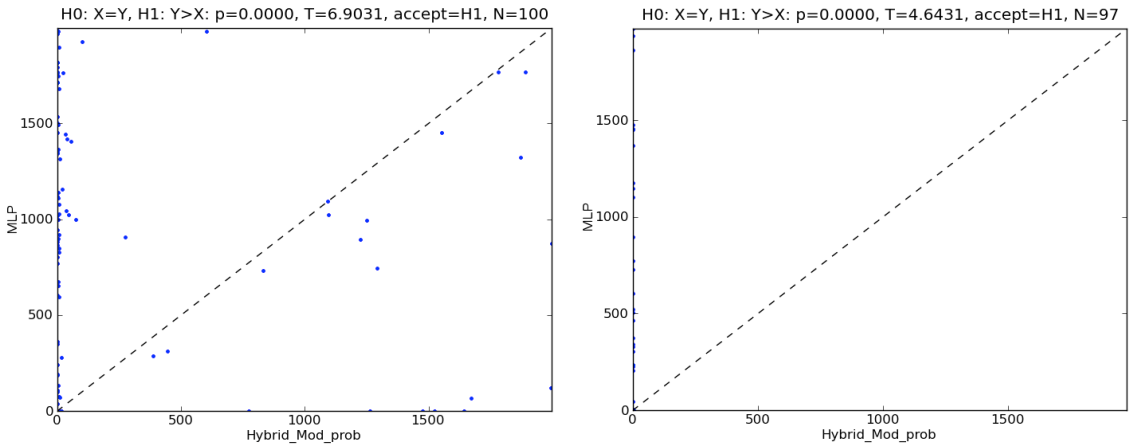
6.4. Visų hibridinių klasifikatorių lyginamieji eksperimentai

6.4.1. Metodika

Antruoju eksperimentu siekiama palyginti visų pasiūlytų hibridinių klasifikatorių veikimą ir tinkamumą pagal tai, kiek iš žemiau išvardytų teiginių jiems yra teisingi:

1. Hibridinis klasifikatorius patikslina jį inicializavusio sprendimo medžio klasifikavimą;
2. Hibridinis klasifikatorius pasiekia mažesnę už originalaus kainoms nejautraus *Banerjee* klaidų kainą;
3. Hibridinis klasifikatorius atlieka mažiau mokymosi iteracijų nei DSP.

Taip pat siekiama išsiaiškinti, kuris hibridinis klasifikatorius sugeba išgauti mažiausią klaidų kainą testiniams duomenims.



(a) *Vertebral* duomenų rinkinys. (b) *Synth* duomenų rinkinys.

20 pav.: Hibrido *Hybrid_Mod_prob* ir atsitiktiniais svoriais inicializuoto DSP palyginimas pagal iteracijų skaičių.

Eksperimentuojama su *Vertebral* ir *Wine* duomenimis iš 2 lentelės. Naudojama kainų matrica, leidžianti proporcingai įvertinti klasifikavimo klaidas proporcingai klasių dydžiams: $C(i,j) = \frac{n_i}{n_j}$, kur n_i ir n_j yra vektorių skaičius klasėse i ir j atitinkamai.

Taikoma v-kryžminės validacijos metodika (angl. v-fold cross validation). Duomenys padalijami į $v = 10$ dalių, tuomet paeiliui kiekviena dalis išskiriama testavimui, o devynios likusios - mokymui. Sprendimo medžiai apmokomi su testiniais duomenimis, o jų apibendrinimo klasifikavimo klaidų kaina įvertinama su testiniais duomenimis. Kiekvienam medžiui apskaičiuojamas kainos su testiniais duomenimis vidurkis per v duomenų padalijimų.

v hibridinių klasifikatorių incializuojami iš atitinkamų sprendimo medžių, kiekvienam iš padalijimų jie mokomi fiksuotą iteracijų skaičių, o rezultatams įvertinti parenkama iteracija su mažiausia kaina testiniams duomenims. Apskaičiuojamas šių (geriausių) iteracijų numerių vidurkis per v padalijimų, taip pat kainos testiniams duomenimis vidurkis.

Palyginimuose dalyvauja vienas DSP variantas, parinktas pagal kainoms nejautraus hibrido *Banerjee* architektūrą. Pirmiausia buvo sudarytos DSP architektūros, ekvivalenčios sudėtingiausioms *Banerjee* architektūroms. Pastebėjus, kad DSP veikimo laikas yra per didelis, ilgiausiai veikiančių DSP architektūros buvo pakeistos ekvivalenčiom paprasčiausiom *Banerjee* architektūrom.

Hibridams inicializuoti naudojami *Mod_prob*, *Mod_prob_err*, *Mod_prob_Laplace*, *MetaCost* ir *Laplace* sprendimo medžiai. Jais inicializuoti hibridai turi priešdėlį *Hybrid_*. Papildomai parodyti C4.5 medis ir jo kainoms

5 lentelė.: Vidutinė klaidų kaina ir geriausios iteracijos numeris testiniams duomenims.

	Vertebral		Wine	
Algoritmas	Kaina	Iteracijos	Kaina	Iteracijos
DSP	0.1565591	728.9	0.0170825	1524.4
Laplace	0.1725805	-	0.0400204	-
Hybrid_Laplace	0.1591397	382.0	0.0167448	1470.9
MetaCost	0.2147311	-	0.0400204	-
Hybrid_MetaCost	0.1727956	526.7	0.0167448	1411.2
Mod_Prob	0.2218279	-	0.0400204	-
Hybrid_Mod_Prob	0.1582795	562.1	0.0245626	1037.7
Mod_Prob_Err	0.2147311	-	0.0400204	-
Hybrid_Mod_Prob_Err	0.1632257	437.5	0.0178771	1236.4
Mod_Prob_Laplace	0.1725805	-	0.0400204	-
Hybrid_Mod_Prob_Laplace	0.1510752	439.0	0.0245626	1039.5
C4.5	0.2187096	-	0.0693958	-
Banerjee	0.1640859	247.4	0.0167448	1408.6
C5.0	0.076344	-	0.0232755	-
Cheapest class	0.3870967	-	0.5389539	-

nejautrus hibridas *Banerjee*, taip pat C5.0 medis bei pigiausios klasės klasifikatorius *Cheapest class*. Palyginti algoritmai ir jų rezultatai (vidutinės klaidų kainos ir vidutinis iteracijų skaičius per v duomenų padalijimų) parodyti 5 lentelėje.

6.4.2. Aptarimas

Su abiem duomenų rinkiniais šiame darbe pasiūlytieji hibridiniai klasifikatoriai sumažino sprendimo medžių klasifikavimo kainą. Tas pats pasakytina ir apie C4.5 bei juo inicializuotą *Banerjee*. Taip pat su abiem duomenų rinkiniais DSP prireikė daugiau iteracijų nei hibridiniams klasifikatoriams pasiekti mažiausią klaidų kainą testiniams duomenims.

Vertebral duomenims trys hibridiniai klasifikatoriai pasiekė mažesnę nei kainoms nejautrus *Banerjee* klasifikavimo kainą: *Hybrid_Mod_Prob*, *Hybrid_Mod_Prob_Laplace* ir *Hybrid_Laplace*. *Wine* duomenims tik *Hybrid_Laplace* ir *Hybrid_MetaCost* gavo tokią pačią klaidų kainą kaip kainoms nejautrus *Banerjee*.

Iš pasiūlytųjų hibridų mažiausią klaidų kainą su *Vertebral* duomenimis išgavo *Hybrid_Mod_Prob_Laplace* (0,1510752) hibridas, o su *Wine* - *Hybrid_Laplace* ir *Hybrid_MetaCost* (0,0167448).

Vertebral duomenims geriausias klasifikatorius pasirodė C5.0 medis, o *Wine* - keli hibridiniai klasifikatoriai. Visi klasifikatoriai veikė geriau nei pigiausios klasės klasifikatorius pagal klaidų kainą.

6.5. Papildomi lyginamieji eksperimentai

Vadovaujantis metodika, aprašyta 6.3.1 skyrelyje, buvo papildomai atlikta eksperimentų su hibridiniais klasifikatoriais: *Hybrid_C5.0*, *Hybrid_Laplace*, *Hybrid_MetaCost* ir *Hybrid_Mod_Prob*. Eksperimentuose naudoti šie duomenų rinkiniai: *Tae*, *Vertebral*, *Wine*, *Seeds* ir *Set15* (žr. 2 lentelę). Atitinkamų t-testų rezultatai pateikti priede - 6 - 10 lentelėse.

Kiekvienam duomenų rinkiniui, atliekami hibrido ir kito klasifikatoriaus (t. y. jį inicializavusio medžio, kainoms nejautraus *Banerjee* ir atsitiktiniais svoriais inicializuoto DSP) palyginimai. Stulpeliai po $a > b$ antrašte reiškia, kad keliama hipotezė, jog a raide pažymėtas metodas pagal atitinkamą metriką (klaidų kainą arba iteracijų skaičių, kaip paaiškinta 6.3.1 skyrelyje) vidutiniškai lenkia metodą b (analogiškai stulpeliams $b < a$).

Kaip matyti iš lentelių, hibridinis klasifikatorius ne su visais duomenimis pasiteisina, tačiau yra duomenų, su kuriais visos trys hipotezės po antrašte $a > b$ patvirtinamos (reikšmė T) - pvz., *Set15* su hibridu *Hybrid_Laplace*. Apskritai dažniausiai hipotezes patvirtina *Hybrid_Mod_Prob*, o rečiausiai - *Hybrid_MetaCost*.

Išvados

Šiame darbe realizuota:

1. Sukurta bazinė C4.5 algoritmo realizacija ir keli jautrumo kainoms joje užtikrinimo metodai: gaubiamasis MetaCost algoritmas, pakeistosios klasių tikimybės, Laplace genėjimas.

2. Sukurta į kainų matricą mokymosi ir klasifikavimo procesuose galinčio atsižvelgti daugiasluoksnio perceptrono realizacija.

3. Realizuota egzistuojančio hibridinio sprendimo medžio ir daugiasluoksnio perceptrono apjungimo metodika [Ban97].

4. Atviru kodu platinamas medžio pavidalo klasifikatorius C5.0 [Res11] adaptuotas eksperimentams šiame darbe, sukuriant hibridą *Hybrid_C50*.

Atlikus eksperimentus su sintetiniais ir realaus pasaulio duomenimis, gautos tokios išvados:

1. Parodyta, kad hibridinis kainoms jautrus klasifikatorius, paremtas [Ban97] hibridizacijos metodika ir jautrumo kainoms įvedimu į sprendimo medį bei daugiasluoksnį perceptroną atskirai, gali sumažinti inicializavusio sprendimo medžio klasifikavimo klaidų kainą su testiniais duomenimis. Taip pat parodyta, kad hibridas pasiekia geriausios kainos iteraciją greičiau nei analogiškos architektūros, tačiau atsitiktinių pradinių svorių daugiasluoksnis perceptronas.

2. Pastebėta, kad hibridinis klasifikatorius ne su visais duomenų rinkiniais pasiteisina, t. y. ne visuomet sumažina medžio klaidų kainą, klasifikuoja geriau už kainoms nejautrų *Banerjee* ir mokosi greičiau nei DSP. Dažniausiai šias hipotezes tenkina *Hybrid_Mod_Prob*, o dažniausiai netenkina - *Hybrid_Meta_Cost*.

3. Sprendimo medžiui turint daug mazgų, inicializuoto neuroninio tinklo architektūra gaunasi labai sudėtinga. Ji lemia, kad tinklas apmokomas labai lėtai.

4. Sprendimo medžio konstravimas stipriai sulėtėja, kai duomenų rinkinys turi daug požymių, neuroninis tinklas - kai yra daug mokymosi vektorių.

Ateityje reikėtų ištirti, kokių savybių duomenys tinkami klasifikuoti hibridiniu klasifikatoriumi. Taip pat reikėtų ištirti, ar būtų galima pagerinti jo veikimą atsižvelgiant į kainų matricos interpretacijos skirtumus kiekvienoje iš hibridinio klasifikatoriaus dalių - sprendimo medyje ir daugiasluoksniame perceptrone.

Literatūros sąrašas

[Agr98] Coull BA Agresti, A. Approximate is better than exact for interval estimation in binomial proportions, 1998.

[ASC08] Roberto Alejo, José Martínez Sotoca, and Gustavo A. Casañ. An empirical study for the multi-class imbalance problem with neural networks. In *CIARP*, pages 479–486, 2008.

[ASGV11] Roberto Alejo, José Martínez Sotoca, Vicente García, and Rosa Maria Valdovinos. Back propagation with balanced mse cost function and nearest neighbor editing for handling class overlap and class imbalance. In *IWANN (1)*, pages 199–206, 2011.

[Ban97] Arunava Banerjee. Initializing neural networks using decision trees. *Computational learning theory and natural learning systems*, IV:3–15, 1997.

[BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA, 1984.

[BKK+98] Jeffrey P. Bradford, Clayton Kunz, Ron Kohavi, Clifford Brunk, Carla E. Brodley, and Carla E. Brodley. Pruning decision trees with misclassification costs. In *ECML*, pages 131–136, 1998.

[BKMM98] C. Blake, E. Keogh, C.J. Merz, and C.J. Merz. Uci repository of machine learning algorithms databases. 1998.

[Cie09] David Cieslak. *Finding problems in, proposing solutions to, and performing analysis on imbalanced data*. PhD thesis, University of Notre Dame, July 2009.

[Dom99] Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. In *In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 155–164. ACM Press, 1999.

[EM08] Saher Esmeir and Shaul Markovitch. Anytime Induction of Low-cost, Low-error Classifiers: a Sampling-based Approach. *Journal of Artificial Intelligence Research*, 33:1–31, 2008.

[Goo65] I.J. Good. *The Estimation of Probabilities: An Essay on Modern Bayesian Methods*. M.I.T. Press, 1965.

[Hay04] Simon Haykin. *Neural Networks: A Comprehensive Foundation*, volume 2. Prentice Hall PTR, 2004.

[KK98] Matjaz Kukar and Igor Kononenko. Cost-sensitive learning with neural networks. In *ECAI*, pages 445–449, 1998.

[Kub96] Miroslav Kubat. Second tier for decision trees. In *ICML*, pages 293–301, 1996.

[Kub98] Miroslav Kubat. Decision trees can initialize radial-basis function networks. *Neural Networks, IEEE Transactions on*, 9(5):813–821, 1998.

[LV13] S Lomax and S Vadera. A survey of cost-sensitive decision tree induction algorithms. *ACM Computing Surveys*, 45, 2013.

[Mar00] Dragos D. Margineantu. When does imbalanced data require more than cost-sensitive learning? pages 47–50, 2000.

[MSH+12] Guangzhi Ma, Enmin Song, Chih-Cheng Hung, Li Su, and Dongshan Huang. Multiple costs based decision making with back-propagation neural networks. *Decision Support Systems*, 52(3):657–663, 2012.

[Núñ91] Marlon Núñez. The use of background knowledge in decision tree induction. *Machine learning*, 6(3):231–250, 1991.

[PMM+94] Michael J. Pazzani, Christopher J. Merz, Patrick M. Murphy, Kamal Ali, Timothy Hume, and Clifford Brunk. Reducing misclassification costs. In *ICML*, pages 217–225, 1994.

[PWK12] H.T. Pham, Y. Won, and J. Kim. Controlling multi-class error rates for mlp classifier by bias adjustment based on penalty matrix. In *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, page 24. ACM, 2012.

[Qui86] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, March 1986.

[Qui93] Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[Qui96] J. Ross Quinlan. Learning decision tree classifiers. *ACM Comput. Surv.*, pages 71–72, 1996.

[Rau08] Šarūnas Raudys. *Žinių išgavimas iš duomenų*. Klaipėdos universiteto leidykla, 2008.

[Res11] RuleQuest Research. C5.0, 2011. Prieiga per internetą: http://rulequest.com/see5-info.html. Žiūrėta: 2012-12-01.

[Rou06] Nathan Rountree. *Initialising Neural Networks with Prior Knowledge*. PhD thesis, University of Otago, Dunedin, New Zealand, September 2006.

[RR10] Sarunas Raudys and Aistis Raudys. Pairwise costs in multiclass perceptrons. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(7):1324–1328, 2010.

[SKW07] Yanmin Sun, Mohamed S. Kamel, Andrew K. C. Wong, and Yang Wang 0007. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, pages 3358–3378, 2007.

[SRD00] M. Skurichina, S. Raudys, and R. P. W. Duin. K-nearest neighbours directed noise injection in multilayer perceptron training. *IEEE Transactions on Neural Networks*, pages 504–511, 2000.

[SWKK09] Yanmin Sun, Andrew K. C. Wong, Mohamed S. Kamel, and Mohamed S. Kamel. Classification of imbalanced data: a review. *IJPRAI*, pages 687–719, 2009.

[Tin02] K. M. Ting. An instance-weighting method to induce cost-sensitive trees. *IEEE Trans. on Knowl. and Data Eng.*, 14(3):659–665, May 2002.

[TP11] Tina Tirelli and Daniela Pessani. Importance of feature selection in decision-tree and artificial-neural-network ecological applications. alburnus alburnus alborella: A practical example. *Ecological Informatics*, pages 309–315, 2011.

[Tur95] Peter D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, pages 369–409, 1995.

[Vad05] Sunil Vadera. Inducing cost-sensitive nonlinear decision trees. 2005. Prieiga per internetą: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Inducing+cost-sensitive+non-linear+decision+trees#0. Žiūrėta: 2012-06-06.

[ZL10] Zhi-Hua Zhou and Xu-Ying Liu. On multi-class cost-sensitive learning. *Computational Intelligence*, 26(3):232–257, 2010.

[ZLL06] Zhi-Hua Zhou, Xu-Ying Liu, and Xu-Ying Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Trans. Knowl. Data Eng.*, pages 63–77, 2006.

Priedas Nr. 1.
Papildomų eksperimentų rezultatų lentelės

6 lentelė.: Rezultatai su *Tae* duomenimis. *Banerjee* ir hibridiniai klasifikatoriai, taip pat medžiai ir jų hibridai lyginami pagal klaidų kainą, o DSP ir hibridai - pagal mokymosi iteracijų skaičių.

	$\bar{x}$	σ^2	$a > b$			$a < b$		
Algoritmas				p/2	T		p/2	T
a) Banerjee	1.6335	0.5584	N	0.0000	-5.1173	T	0.0000	5.1173
b) Hybrid_C50	1.7395	0.5647						
a) C50	1.7490	0.5819	N	0.3047	0.5109	N	0.3047	-0.5109
b) Hybrid_C50	1.7395	0.5647						
a) DSP	82.1200	244.7042	T	0.0112	2.2845	N	0.0112	-2.2845
b) Hybrid_C50	63.2800	202.4601						
a) Banerjee	1.6335	0.5584	T	0.0000	35.0309	N	0.0000	-35.0309
b) Hybrid_Laplace	0.7147	0.1800						
a) Laplace	0.6894	0.1159	N	0.0045	-2.6250	T	0.0045	2.6250
b) Hybrid_Laplace	0.7147	0.1800						
a) DSP	82.1200	244.7042	T	0.0000	12.4643	N	0.0000	-12.4643
b) Hybrid_Laplace	0.6100	1.3107						
a) Banerjee	1.6335	0.5584	N	0.0000	-8.4829	T	0.0000	8.4829
b) Hybrid_MetaCost	1.9445	0.4960						
a) MetaCost	1.2957	0.4814	N	0.0000	-18.9854	T	0.0000	18.9854
b) Hybrid_MetaCost	1.9445	0.4960						
a) DSP	82.1200	244.7042	N	0.0000	-11.1681	T	0.0000	11.1681
b) Hybrid_MetaCost	222.0800	351.4646						
a) Banerjee	1.6335	0.5584	T	0.0000	29.9994	N	0.0000	-29.9994
b) Hybrid_Mod_prob	0.6771	0.0814						
a) Mod_prob	1.1195	0.4557	T	0.0000	16.5131	N	0.0000	-16.5131
b) Hybrid_Mod_prob	0.6771	0.0814						
a) DSP	82.1200	244.7042	T	0.0000	5.1184	N	0.0000	-5.1184
b) Hybrid_Mod_prob	33.5640	157.2679						

7 lentelė.: Rezultatai su *Vertebral* duomenimis. *Banerjee* ir hibridiniai klasifikatoriai, taip pat medžiai ir jų hibridai lyginami pagal klaidų kainą, o DSP ir hibridai - pagal mokymosi iteracijų skaičių.

	$\bar{x}$	σ^2	$a > b$			$a < b$		
Algoritmas				p/2	T		p/2	T
a) Banerjee	0.6344	0.2366	N	0.0000	-36.2219	T	0.0000	36.2219
b) Hybrid_C50	1.0063	0.3590						
a) C50	1.0939	0.3914	T	0.0000	6.9203	N	0.0000	-6.9203
b) Hybrid_C50	1.0063	0.3590						
a) DSP	157.6800	366.3093	N	0.0000	-3.9695	T	0.0000	3.9695
b) Hybrid_C50	214.4300	435.5767						
a) Banerjee	0.6344	0.2366	T	0.0000	14.4320	N	0.0000	-14.4320
b) Hybrid_Laplace	0.4245	0.1779						
a) Laplace	0.3693	0.1144	N	0.0000	-5.6196	T	0.0000	5.6196
b) Hybrid_Laplace	0.4245	0.1779						
a) DSP	157.6800	366.3093	T	0.0000	15.2752	N	0.0000	-15.2752
b) Hybrid_Laplace	5.6400	53.0082						
a) Banerjee	0.6344	0.2366	N	0.0000	-13.5430	T	0.0000	13.5430
b) Hybrid_MetaCost	0.9211	0.2960						
a) MetaCost	0.5836	0.2158	N	0.0000	-18.4747	T	0.0000	18.4747
b) Hybrid_MetaCost	0.9211	0.2960						
a) DSP	157.6800	366.3093	N	0.0000	-10.4746	T	0.0000	10.4746
b) Hybrid_MetaCost	330.4200	488.0967						
a) Banerjee	0.6344	0.2366	T	0.0000	18.4482	N	0.0000	-18.4482
b) Hybrid_Mod_prob	0.3462	0.1074						
a) Mod_prob	0.4966	0.2040	T	0.0000	11.3254	N	0.0000	-11.3254
b) Hybrid_Mod_prob	0.3462	0.1074						
a) DSP	157.6800	366.3093	T	0.0000	10.8876	N	0.0000	-10.8876
b) Hybrid_Mod_prob	22.2700	143.7671						

8 lentelė.: Rezultatai su *Wine* duomenimis. *Banerjee* ir hibridiniai klasifikatoriai, taip pat medžiai ir jų hibridai lyginami pagal klaidų kainą, o DSP ir hibridai - pagal mokymosi iteracijų skaičių.

	$\bar{x}$	σ^2	$a > b$			$a < b$		
Algoritmas				p/2	T		p/2	T
a) Banerjee	0.9300	0.6277	N	0.0000	-8.1305	T	0.0000	8.1305
b) Hybrid_C50	1.1661	0.8262						
a) C50	0.3054	0.2156	N	0.0000	-37.6003	T	0.0000	37.6003
b) Hybrid_C50	1.1661	0.8262						
a) DSP	0.0250	0.1561	N	0.0000	-8.0757	T	0.0000	8.0757
b) Hybrid_C50	35.6625	157.8150						
a) Banerjee	0.9300	0.6277	N	0.3361	0.4234	N	0.3361	-0.4234
b) Hybrid_Laplace	0.9108	0.6139						
a) Laplace	0.2277	0.1366	N	0.0000	-21.8165	T	0.0000	21.8165
b) Hybrid_Laplace	0.9108	0.6139						
a) DSP	0.0250	0.1561	N	0.0000	-6.0892	T	0.0000	6.0892
b) Hybrid_Laplace	26.5500	145.7117						
a) Banerjee	0.9300	0.6277	N	0.0490	-1.6595	T	0.0490	1.6595
b) Hybrid_MetaCost	1.0287	0.7914						
a) MetaCost	0.3527	0.2761	N	0.0000	-14.3276	T	0.0000	14.3276
b) Hybrid_MetaCost	1.0287	0.7914						
a) DSP	0.0250	0.1561	N	0.0000	-7.1878	T	0.0000	7.1878
b) Hybrid_MetaCost	28.1000	120.9988						
a) Banerjee	0.9300	0.6277	T	0.0000	9.0896	N	0.0000	-9.0896
b) Hybrid_Mod_prob	0.5202	0.2592						
a) Mod_prob	0.3220	0.2492	N	0.0000	-8.6093	T	0.0000	8.6093
b) Hybrid_Mod_prob	0.5202	0.2592						
a) DSP	0.0250	0.1561	N	0.0000	-4.3636	T	0.0000	4.3636
b) Hybrid_Mod_prob	13.2975	85.9724						

9 lentelė.: Rezultatai su *Seeds* duomenimis. *Banerjee* ir hibridiniai klasifikatoriai, taip pat medžiai ir jų hibridai lyginami pagal klaidų kainą, o DSP ir hibridai - pagal mokymosi iteracijų skaičių.

	$\bar{x}$	σ^2	$a > b$			$a < b$		
Algoritmas				p/2	T		p/2	T
a) Banerjee	0.2200	0.2189	N	0.0000	-10.6680	T	0.0000	10.6680
b) Hybrid_C50	0.3284	0.2344						
a) C50	0.5090	0.2973	T	0.0000	10.7365	N	0.0000	-10.7365
b) Hybrid_C50	0.3284	0.2344						
a) DSP	1297.4667	506.7902	T	0.0000	18.6315	N	0.0000	-18.6315
b) Hybrid_C50	744.3333	687.7640						
a) Banerjee	0.2200	0.2189	N	0.0000	-4.0905	T	0.0000	4.0905
b) Hybrid_Laplace	0.2905	0.1646						
a) Laplace	0.3195	0.1831	N	0.0917	1.3363	N	0.0917	-1.3363
b) Hybrid_Laplace	0.2905	0.1646						
a) DSP	1297.4667	506.7902	T	0.0000	19.5899	N	0.0000	-19.5899
b) Hybrid_Laplace	422.4667	614.6402						
a) Banerjee	0.2200	0.2189	N	0.0053	-2.5972	T	0.0053	2.5972
b) Hybrid_MetaCost	0.2795	0.2078						
a) MetaCost	0.5090	0.2599	T	0.0000	7.4694	N	0.0000	-7.4694
b) Hybrid_MetaCost	0.2795	0.2078						
a) DSP	1297.4667	506.7902	T	0.0000	14.1029	N	0.0000	-14.1029
b) Hybrid_MetaCost	767.5000	600.8692						
a) Banerjee	0.2200	0.2189	N	0.0001	-3.9883	T	0.0001	3.9883
b) Hybrid_Mod_prob	0.3824	0.2205						
a) Mod_prob	0.4897	0.2605	T	0.0022	2.9216	N	0.0022	-2.9216
b) Hybrid_Mod_prob	0.3824	0.2205						
a) DSP	1297.4667	506.7902	T	0.0000	13.6967	N	0.0000	-13.6967
b) Hybrid_Mod_prob	562.2733	689.9852						

10 lentelė.: Rezultatai su *Set15* duomenimis. *Banerjee* ir hibridiniai klasifikatoriai, taip pat medžiai ir jų hibridai lyginami pagal klaidų kainą, o DSP ir hibridai - pagal mokymosi iteracijų skaičių.

	$\bar{x}$	σ^2	$a > b$			$a < b$		
Algoritmas				p/2	T		p/2	T
a) Banerjee	1.0633	0.1677	N	0.0000	-18.8881	T	0.0000	18.8881
b) Hybrid_C50	1.1924	0.2305						
a) C50	1.1711	0.2339	N	0.0030	-2.7542	T	0.0030	2.7542
b) Hybrid_C50	1.1924	0.2305						
a) DSP	150.7100	323.4681	T	0.0000	18.6260	N	0.0000	-18.6260
b) Hybrid_C50	0.0500	0.2179						
a) Banerjee	1.0633	0.1677	T	0.0000	109.9624	N	0.0000	-109.9624
b) Hybrid_Laplace	0.4901	0.0713						
a) Laplace	0.5089	0.0938	T	0.0000	5.7080	N	0.0000	-5.7080
b) Hybrid_Laplace	0.4901	0.0713						
a) DSP	150.7100	323.4681	T	0.0000	16.7853	N	0.0000	-16.7853
b) Hybrid_Laplace	0.0600	0.2375						
a) Banerjee	1.0633	0.1677	N	0.0014	-3.0004	T	0.0014	3.0004
b) Hybrid_MetaCost	2.0138	8.9452						
a) MetaCost	1.2112	0.1708	N	0.0057	-2.5358	T	0.0057	2.5358
b) Hybrid_MetaCost	2.0138	8.9452						
a) DSP	152.2323	324.7410	T	0.0000	12.3317	N	0.0000	-12.3317
b) Hybrid_MetaCost	0.0000	0.0000						
a) Banerjee	1.0633	0.1677	T	0.0000	107.3435	N	0.0000	-107.3435
b) Hybrid_Mod_prob	0.4987	0.0675						
a) Mod_prob	0.5503	0.0949	T	0.0000	15.5314	N	0.0000	-15.5314
b) Hybrid_Mod_prob	0.4987	0.0675						
a) DSP	150.7100	323.4681	T	0.0000	14.7176	N	0.0000	-14.7176
b) Hybrid_Mod_prob	0.0980	0.3105						