VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Baigiamasis magistro darbas

Automatinis CERN CMS duomenų sertifikavimas

Atliko:
Monika Bijeikytė parašas

Vadovas:
dr. Valdas Rapševičius

Vilnius
2018

Turinys

Santrauka

Šio magistrinio darbo tikslas - sukurti duomenų klasifikavimo modelį patikimai klasifikuojantį CERN CMS duomenis pagal jų kokybės žymą. Toks modelis gerokai sumažintų duomenų kokybės ekspertų darbo poreikį, todėl yra itin reikalingas.

Darbo metu buvo apžvelgti egzistuojantys problemos sprendimo pasiūlymai bei jų trūkumai. Eksperimentinės dalies metu patikrinti keli skirtingi duomenų klasifikavimo metodai, palygintas jų apmokymo greitis bei tikslumą nusakantys parametrai. Nepavykus pasiekti pakankamai gero klasifikavimo tikslumo, buvo kuriamas duomenų klasifikatorius, naudojantis dirbtinį neuroninį tinklą.

Nustatyta, kad patobulinus neuroninio tinklo architektūrą, buvo sukurtas geriausius rezultatus rodantis CERN CMS duomenų klasifikatorius, kuris pasiekė daugiau nei 95% tikslumą.

Summary

Automated CERN CMS Data Certification

CMS (Compact Muon Solenoid) is one of the CERN LHC detectors. It is able to register particle collisions up to 40 million times in a second. All this data has to go through many complex tests and systems to reach scientist for physics analysis. Most of these tests are automated and doesn't need professional supervision. However, the final certification flag, that defines data quality, is done manually by data quality specialists. In order to ease their load, an automated data certification model has to be created.

The main goal of this work is to create a reliable model for CERN CMS data certification. A more detailed tasks were to have a look at current propositions for solving automated certification problem and investigate possible classification models in order to find the most suitable one for creating an automated certification model in the future.

After investigating related works from scientists of *Yandex* group and from University of Michigan, four classification models were examined. The first one, logistic regression classifier, was able to find optimal solution and reach accuracy of 87.55%. The second one, support vector machines, weren't able to find optimal solution even after increasing threshold for maximum iterations. It was able to reach quite similar accuracy of 87.12%. The third model that was examined in this work is boosted decision trees. Despite of the longest training time, it was able to reach high accuracy of 95.52%. The final model, which was created, is based on artificial neural networks. One of the goals of this work was to find the best learning parameters and the most suitable network architecture for CERN CMS data certification problem. After many invertigation the best neural network model was reached with a 95.15% accuracy.

After evaluating all the models, it was decided, that artificial neural network model suits this case the best. However, further research has to be done in order to make a more applicable model and possibly reach even best classification accuracy.

Įvadas

CMS (angl. *Compact Muon Solenoid*) - tai vienas iš didžiausio pasaulyje dalelių greitintuvo LHC (angl. *Large Hadron Collider*), kuris yra įsikūręs CERN mokslinėje laboratorijoje, detektorių. Šis detektorius registruoja duomenis iki 40 milijonų kartų per sekundę. Pagrindinis šių duomenų kokybės analizės įrankis - DQM (angl. *Data Quality Monitoring*) sistema, kurią sudaro dvi dalys: *Online* – skirta tiesioginiam duomenų stebėjimui realiu laiku ir *Offline* – skirta galutiniam gautų duomenų sertifikavimui. Nors duomenys nuo jų surinkimo detektoriuje iki pateikimo mokslininkų analizei prereina daug automatinių testų, galutinis duomenų sertifikavimas yra atliekamas rankiniu būdu, o tai reikalauja didžiulės žmonių darbo jėgos. Būtent todėl yra reikalingas patikimas modelis, kuris būtų apmokytas sertifikuoti šiuos duomenis automatiškai su tikslumu, artimu duomenų sertifikavimo eksperto darbui.

Šio baigiamojo magirstro darbo tikslas - sukurti patikimą modelį, gebantį automatiškai sertifikuoti CERN CMS duomenis. Nustatyti darbo uždaviniai - susipažinti su esamais CERN CMS duomenų automatinio sertifikavimo modeliais, ištirti uždavinio sprendimui tinkamus duomenų klasifikatorius bei atrasti klasifikatorių, kuris galėtų toliau būti naudojamas, kuriant duomenų sertifikavimo algoritmą.

Mokslo tiriamojo darbo metu buvo išbandyti trys skirtingi duomenų klasifikavimo modeliai, palygintas jų tikslumas ir kiti klasifikavimo kokybę nurodantys parametrai. Šiame darbe esantys teoriniai bei eksperimentiniai skyriai, susiję su logistinės regresijos, atraminių vektorių mašinų ir išpūstų sprendimų medžių klasifikatoriais, buvo paimti iš mokslo tiriamojo darbo projekto. Kandangi, nei vienas iš klasifikatorių nepasiekė tokio aukšto tikslumo ir patikimumo, kuris būtų patenkinamas praktiniam modelio naudojimui, baigiamojo magistro darbo metu buvo nuspręsta sukurti neuroninio tinklo modelį. Tiriant klasifikavimo tikslumo priklausomybę nuo neuroninio tinklo architektūros ir parametrų, buvo bandoma sukurti modelį, gebantį automatiškai sertifikuoti CERN CMS duomenis maksimaliai tiksliai.

Baigiamojo darbo metu buvo nustatyta, kad geriausi rezultatai yra gaunami naudojant sukurtą dirbtinį neuroninį tinklą. Tokiu klasifikatoriumi buvo pasiektas 95.12% tikslumas, išspręstos klasifikatoriaus prisitaikymo prie duomenyse esančių triukšmų problemos. Galima teigti, kad dibtinis neuroninis tinklas yra potencialiai tinkamas papildyti CERN CMS DQM sistemą siekiant automatizuoti galutinį duomenų sertifikavimo procesą.

Tolimesniuose skyriuose aprašyta CERN CMS duomenų kokybės stebėjimo sistema, jos veikimas, aptarti kompanijos *Yandex* bei Mičigano universiteto mokslininkų grupių sukurti duomenų klasifikavimo modeliai, fizikiniai parametrai, kurie svarbūs kuriant duomenų sertifikavimo modelį. Taip pat aprašyti naudotų klasifikatorių pagrindiniai principai bei neuroninių tinklų teorija. Eksperimentinėje dalyje aprašyti magistrinio tiriamojo darbo metu naudoti klasifikatoriai ir jais pasiektas klasifikavimo tikslumas. Taipogi aprašytas nauroninio tinklo naudojimas magistro darbo tikslui pasiekti bei jo tikslumui pagerinti atlikti tyrimai.

1. Teorinis įvadas

1.1. CERN CMS detektorius

CERN mokslinių tyrimų laboratorijoje įsikūręs, beveik 27 km ilgio dalelių greitintuvas LHC (angl. *Large Hadron Collider*) yra pats didžiausias ir galingiausias Žemėje. Jis gali įgreitinti protonus beveik iki šviesos greičio ir sukelti jų susidūrimus keturiose greitintuvo žiedo vietose. Vienoje iš jų yra įsikūręs CMS (angl. *Compact Muon Solenoid*) detektorius. Tai yra bendrosios paskirties detektorius, galintis registruoti beveik visų rūšių stabilias daleles, atsirandančias aukštos energijos protonų susidūrimų metu.

Viena pagrindinių detektoriaus konstrukcijos dalių - solenoidas (cilindrinė ritė), kuriantis magnetinį lauką, kuris yra daugiau nei 100 000 kartų stipresnis už Žemės magnetinį lauką. Tai padeda nustatyti po susidūrimo atsiradusių dalelių krūvį ir judesio kiekio momentą. Pasinaudojant dviejų rūšių kalorimetrais, detektorius gali išmatuoti įvairių dalelių energiją. CMS iš kitų detektorių išsiskiria tuo, kad, be įprastų dalelių, geba registruoti ir miuonus - elementariasias daleles, kurios turi neigiamą elektrinį krūvį ir yra 200 kartų sunkesnės už elektronus [11]. Įprasti kalorimetrai tokių dalelių sustabdyti negali, todėl tam reikalinga papildoma įranga.

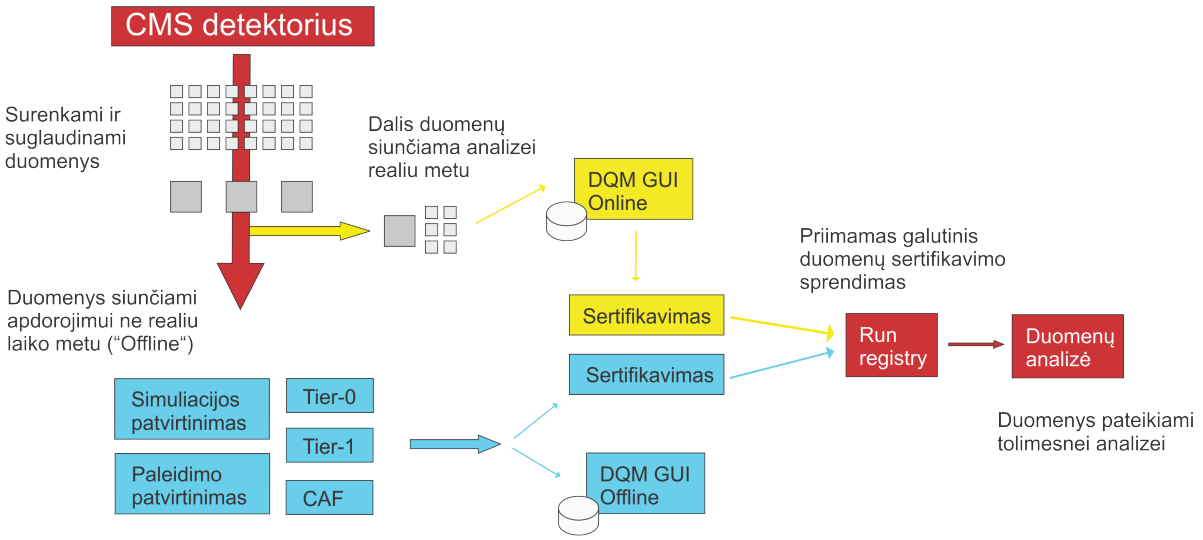
LCH greitintuve dalelių paketai susiduria maždaug 40 milijonų kartų per sekundę [3]. Detektoriuje veikianti sistema iš karto po susidūrimo atrenka fizikos mokslininkams potencialiai įdomius įvykius ir įrašo jų duomenis. Tačiau, net po šios atrankos lieka apie 100 įrašų per sekundę. Kiekvieną akimirką registruojant tokį didelį kiekį duomenų yra svarbu, kad detektorius veiktų nepriekaištingai, būtų nuolat pasiruošęs įrašyti duomenis ir juos įvertinti tolimesnei analizei. CMS detektoriaus atveju, šią užduotį atlieka duomenų kokybės priežiūros sistema DQM (angl. *Data Quality Monitoring*).

1.2. CMS DQM sistema

Jei kurioje nors greitintuvo ar detektoriaus dalyje atsiranda sutrikimas, tai iš karto atsispindi registruojamuose parametruose. Be to, ne visi duomenys yra tinkami tolimesnei fizikos mokslininkų analizei. Duomenų kokybės tikrinimo ekspertai remiasi apibrėžtomis taisyklėmis ir statistiniais metodais, kad nustatytų netinkamus fiksuojamų susidūrimų parametrų verčių nukrypimus. Šių taisyklių ir metodų apibrėžimas reikalauja puikiaus detektoriaus ir galimų anomalijų supratimo.

Duomenų kokybės stebėjimui buvo sukurta DQM (duomenų kokybės stebėjimo) sistema. Ji yra sudaryta iš dviejų dalių. Maža dalis susidūrimų metu registruojamų duomenų yra siunčiama į CMS valdymo centrą, kur tuo metu dirbantys asmenys realiu laiku analizuoja gautus duomenis ir pateikia išvadas apie duomenų kokybę. Antroji komanda yra įsikūrusi CMS centre CERN laboratorijose ir atlieka daug detalesnę duomenų analizę.

DQM sistemos supaprastinta schema pavaizduota 1 pav. Visų pirma, duomenys, gauti iš detektoriaus, yra filtruojami paliekant įrašus, kurie, tikėtina, bus naudingi tolimesnei analizei. Tuomet, dalis duomenų yra siunčiama į duomenų tyrimo realiu laiku centrą (*DQM Online*), kur įvairios sistemos ir algoritmai tikrina duomenis. Tai naudinga todėl, kad tokiu būdu galima iš karto pastebėti įvairius atsirandančius detektoriaus ar sistemos gedimus ir juos tvarkyti neuždelsiant ir neprarandant daug duomenų. Tuomet rezultatai yra įkeliami į *DQM GUI Online* sistemą, kur juos gali pasiekti sistemos vartotojai, ir yra atliekamas pradinis duomenų sertifikavimas. Kiekvieno įrašo kokybė yra pažymima kaip "gera", "bloga" ar "nežinoma" (jei turimiems duomenims nepavyksta tiksliai nustatyti kokybės žymos) ir nustatyti rezultatai yra įkeliami į eigos registrą (angl. *run*



1 pav. DQM sistemos principinė schema.

registry) [17].

Visi iš detektoriaus gauti tinkami duomenys siunčiami į antrąją DQM sistemos dalį, kurioje vyksta nuodugnus jų tyrimas, trunkantis savaites ar mėnesius. Duomenų tinkamumas tolimesniam panaudojimui labai priklauso nuo to, ar visa detektoriaus sistema yra teisingai sukalibruota. Dėl to, gauti duomenys pereina daug žingsnių, kurių metu nuolatos tikrinamas sistemos veikimas (simuliacijos ir paleidimo patvirtinimai, nulinės ir pirmosios duomenų apdorojimo pakopų (angl. *Tier-0*, *Tier-1*) kalibravimo tikrinimo sistemos, CAF (angl. *CERN Analysis Facility*) tikrinimai). Duomenims perėjus daug tikrinimo ir apdorojimo sistemų, jie įkeliami į *DQM GUI Offline* sistemą, kur jie pasiekiami mokslininkams. Į DQM GUI įkelti failai yra peremti ROOT objektų failų struktūra. ROOT - tai mokslinė programinės įrangos sistema, kuri leidžia naudotis funkcijomis, skirtomis apdoroti dideliams duomenų kiekiams, duomenis analizuoti, vaizduoti bei saugoti. Automatinius testuis perėję duomenys taip pat yra siunčiami sertifikavimui, kurio rezultatai su pastabomis yra įrašomi į eigos registrą.

Gavus sertifikavimo rezultatus iš abiejų DQM dalių, eigos registras dar karta peržiūrimas, sprendžiama dėl galutinio sertifikavimo rezultato. Tai atlieka tą pamainą dirbantys ekspertai. Šio rankinio sertifikavimo metu yra peržiūrimos pačios svarbiausios histogramos ir duomenų kokybė vertinama pagal tam tikras ekspertų parašytas instrukcijas. Kiekvienas duomenų įrašas yra pažymimas arba kaip geras, arba kaip blogas. Šios žymos įrašomos į eigos registrą ir yra pasiekiamos mokslininkams.

Nors šis galutinis sertifikavimas šiuo metu yra atliekamas žmonių ekspertų rankinio darbo metu, yra siekiama toliau tobulinti DQM sistemą ir šį procesą automatizuoti. Siekiant sukurti automatinio CERN CMS sertifikavimo modelį, yra naudojamasi įvairiais sistemos mokymosi metodais, kurie bus aprašyti tolimesniuose šio darbo skyriuose. Visi aprašyti metodai buvo naudojami ir šiame magistro darbe.

Šiuo metu egzistuoja pora kitų mokslininkų grupių, kurios užsiima automatinio CERN CMS duomenų sertifikavimo uždavinio sprendimo paieškomis. Jų pasiekti rezultatai buvo pristatyti įvairiose konferencijose ir pristatymuose, tačiau vieningos išvados, apie tinkamiausią automatinio sertifikavimo modelį, kol kas nėra. Minėtų mokslininkų grupių pasiekti rezultatai aprašyti tolimesniuose poskyriuose.

1.3. *Yandex* modelis

Viena iš didžiausią pažangą CERN CMS automatinio sertifikavimo srityje padariusių - tai *Yandex* duomenų analizės mokyklos mokslininkų grupė. Jie pasiūlė savo modelį, kuris yra apmokytas automatiškai sertifikuoti CERN CMS detektoriaus duomenis. Vietoje to, kad būtų aprašytos tikslios sąlygos, lemiančios duomenų kokybę, *Yandex* grupė pasiūlė modelį, kuris yra apmokomas jau egzistuojančiais ir įvertintais detektoriaus duomenimis. Jų aprašytu atveju, buvo naudojami 2010 metų LHC CMS detektoriaus rezultatai [1]. Šis modelio kūrimo būdas turi kelis privalumus. Visų pirma, sukurta sistema, naudodama jau ekspertų įvertintus duomenis, pati išmoksta taikyti įvairius satistinius metodus. Antra, tokia sistema gali būti nesunkiai pritaikoma pakeitus reikalingus eksperimento parametrus. Trečia, modelis yra nesunkiai suderinamas su duomenų kokybės ekspertų sukurtais statistiniais modeliais ir gali būti jais papildomas.

Pirminis *Yandex* siūlomos sistemos tikslas - sugebėti atrinkti pačius aiškiausius atvejus - kai duomenys yra tikrai geri, tinkami tolimesnei analizei ir nerodo jokių detektoriaus ar greitintuvo problemų ("baltoji zona"), bei tokius duomenis, kuriuose matomos anomalijos ("juodoji zona"). Duomenys, kurie netinka nei vienam iš šių regionų, yra priskiriami pilkąjai zonai ir reikalauja detalesnės duomenų kokybės ekspertų analizės. Kol modelis yra neapmokytas jau sertifikuotais duomenimis, jis visus atvejus priskiria pilkajai zonai [1]. Kuo labiau modelis apmokomas, tuo didesnę dalį duomenų gali tiksliai priskirti baltajai ar juodajai zonoms.

Kuriant automatinio sertifikavimo modelį, yra siekiama, kad būtų galima sertifikuoti kuo didesnę duomenų dalį, paliekant mažiau atvejų, kada yra reikalinga ekspertų nuomonė. Tai reikia atlikti esant kuo mažesnėms praradimų ir užterštumo santykių reikšmėms. Praradimo santykis - tai dydis, kuris parodo gerų duomenų kiekį, kurie buvo sertifikuoti kaip netinkami duomenys ir dėl to buvo prarasti. Užterštumo santykis - tai dydis, parodantis, kiek netinkamų duomenų buvo sertifikuoti kaip tinkami ir taip "užteršė" duomenis. Sertifikuota duomenų dalis yra apibrėžiama atmetimo santykiu:

$$RR = \frac{R}{T}, \tag{1.1}$$

kur RR - atmetimo santykis, R - nesertifikuotų duomenų kiekis (pilkoji zona), T - pilnas duomenų kiekis. Apribojimai sistemai:

$$LR = \frac{FN}{TP + FN}, \tag{1.2}$$

kur LR - praradimo santykis (angl. *loss rate*), FN - neteisingai netinamais sertifikuoti duomenys (angl. *false negative*), TP - teisingai tinkamais sertifikuoti duomenys (angl. *true positive*).

$$PR = \frac{FP}{TP + FP}, \tag{1.3}$$

kur PR - užterštumo santykis (angl. *pollution rate*), FP - neteisingai tinkamais sertifikuoti duomenys (angl. *false positive*). Kuriant modelį, yra nustatomos siekiamos konstantos L_0 ir P_0, atitinkančios maksimalias praradimo ir užteštumo santykių vertes, leidžiamas sistemai. Šių dydžių parinkimas nurodo automatinio duomenų sertifikavimo modelio tikslumą ir kokybę. Dėl šių parametrų, sistema yra priversta atrinkti tik pačius aiškiausius atvejus, o tuos, kurie kelia klausimų, palikti pilkoje zonoje.

Yandex grupės sukurtas modelis bando sertifikuoti gaunamus duomenis vienas po kito. Jei yra gaunamas duomenų įrašas, kurio sertifikuoti su duotais apribojimais modelis nesugeba, šis įrašas kartu su ekspertų nustatyta žyma yra siunčiamas modelio apmokymui. Tokiu būdu, kiekvieną kartą nepavykus sertifikuoti duomenų, modelis vis tobulinamas jį apmokant sudėtingesniais duomenimis. Kuo daugiau duomenų siunčiama modeliui, tuo "protingesnis" ir geriau apmokytas jis tampa.

Modelis buvo išbandytas esant įvairioms praradimo ir užterštumo santykių vertėms ir buvo nustatyta, kad modelis šių apribojimų nepažeidė. Sistema sugebėjo automatiškai sertifikuoti bent 20% gaunamų duomenų. Sertifikuojamų duomenų skaičius gerokai išauga, jei yra mažinami sertifikavimo kokybę nusakantys apribojimai.

1.4. Michigano universiteto tyrimas

Mokslininkai iš Michigano universiteto grupės 2016 metų pabaigoje pristatė savo tyrimus dėl CERN CMS duomenų klasifikavimo [10]. Jų tyrimo tikslas taip pat buvo panaudoti sistemos apsimokymo technikas siekiant sukurti modelį, gebantį atskirti aiškiausius duomenų kokybės atvejus, o duomenų kokybės ekspertų tyrimui palikti tik pačius netrivialiausius duomenų įrašus.

Mokslininkai naudojo 2016 metų duomenis kurdami dviejų rūšių modelius: atraminių vektorių mašinas bei išpūstus sprendimo medžius. Kiekvienu atveju buvo tikrinama modelio apmokymo trukmė ir tikslumas. Nustatyta, kad SVM modelio kūrimas trunka daugiau nei vieną valandą, o tuo tarpu BDT - 3 minutes. Ši mokslininkų grupė sukurto klasifikatoriaus kokybei tikrinti pasirinko kitą matą - preciziškumą. Tai yra teisingai klasifikuotų sertifikuotų duomenų santykis su šio dydžio ir nesertifikuotų duomenų, kurie buvo klasifikuoti kaip sertifikuoti, skaičiaus suma. Nors atraminių vektorių mašinų klasifikatoriaus apmokymo laikas buvo ilgesnis, šis klasifikatorius pasiekė ganėtinai didelį 97% preciziškumą. BDT pasiekiamas preciziškumas - 92%.

Nors šios grupės pasiektas preciziškumas yra ganėtinai didelis, yra norima ir toliau tobulinti naudojamus algoritmus siekiant sumažinti jų apmokymui skirtą laiką bei padidinti tikslumą. Šiam tikslui yra atliekama tolimesnė sistemų, kuriose atsiranda anomalijos ir nekokybiški duomenys, analizė bei ieškoma labiausiai šiam uždaviniui tinkančio algoritmo.

1.5. Fizikiniai duomenis nusakantys parametrai

Į DQM GUI sistemą ikėliami ROOT formato duomenys yra sudaryti iš daugybės histogramų. Siekiant atlikti duomenų analizę ir kurti automatinio sertifikavimo algoritmą apmokant metodą atpažinti duomenis, ROOT formato duomenų failas yra konvertuojamas įrašant svarbiausius histogramų parametrus. Šių parametrų žinojimas yra naudingas siekiant suprasti, kas lemia CERN CMS duomenų kokybę. Kokybės nusakymui svarbiausi parametrai buvo nustatyti eksperimentiniu būdu [1]. Analizei naudojamų parametrų sąrašas pateikiamas žemiau.

1. *runId* - detektoriaus paleidimo serijos numeris.
2. *lumiId* - segmento numeris. Detektoriaus paleidimo serija yra padalinama į mažesnius segmentus tam, kad atsiradus detektoriaus ar sistemos klaidoms ir jas greitai pataisius, neteiktų atsisakyti visos paleidimo serijos duomenų, o tik tam tikrų sugadintų segmentų. Tipiškai, vieno segmento duomenų registravimo trukmė yra apie 23 sekundes.
3. *lumi* - tai šviesis, gautas suintegravus viso segmento metu užfiksuotą spinduliavimo galią. Matuojamas vatais [W].

4. *isSig* - tai loginiu "taip" arba "ne" įrašyta žymė, kuri nusako bendrą įrašo kokybę. Reikšmė yra lygi "taip", jei duomenys yra tinkami analizei (visos sub-sistemos veikia tinkamai).

5. *qPFJetPt, qPFJetEta, qPFJetPhi* - tai vektoriai iš 7 dydžių (vidurkis, kvadratų vidurkio kvadratinė šaknis (RMS), kvantiliai), nusakantys atitinkamai šiuos parametrus dalelių-kandidačių trajektorijose, atsiradusiose po protono-protono susidūrimo:

   - skersinio impulso pasiskirstymą;
   - kampo tarp dalelės ir spindulio ašies pasiskirstymą;
   - azimutinio kampo pasiskirstymą.

   Pagal detektoriaus užfiksuotus rodmenis, įvairūs algoritmai geba atpažinti, kokios dalelės susidarė po susidūrimo (dalelės kandidatės). Apskaičiavus jų energiją, galima surasti trūkstamą energiją, kuri padeda spręsti apie neužfiksuotas daleles (pavyzdžiui, neutrinus).

6. *qMetPt, qMetPhi* - tai vektoriai iš 7 dydžių (vidurkis, kvadratų vidurkio kvadratinė šaknis (RMS), kvantiliai), nusakantys atitinkamai skersinio impulso bei azimutinio kampo pasiskirstymus MET (angl. *Missing transverse momentum*) tyrime. MET - tai visų užfiksuotų dalelių skersinio impulso disbalansas. Pagal impulso tvermės dėsnį, imulsas turi būti išlaikomas visomis kryptimis. Tačiau, po dalelių susidūrimo detektoriuje, atsiranda jo disbalansas, o tai leidžia numanyti, kad atsiranda neužfiksuojamų dalelių (pavyzdžiui, neutrinų).

7. *qNVtx* - tai vektorius iš 7 dydžių (vidurkis, kvadratų vidurkio kvadratinė šaknis (RMS), kvantiliai), nusakantis dalelių susidūrimų ir sklaidos įvykių skaičiaus pasiskirstymą.

8. *crossSection* - įvykių iš pirminio duomenų rinkinio skaičius padalintas iš segmento suintegruoto šviesio.

1.6. Sistemos mokymasis

Sistemos mokymasis (angl. *machine learning*) - tai duomenų mokslo technika, kuri leidžia kompiuteriams pasinaudoti jau egzistuojančiais duomenimis siekiant numatyti sistemos elgesį ateityje, galimas tendencijas ar rezultatus. Dažniausiai yra išskiriami trys sistemos mokymosi tipai:

- Prižiūrimas mokymasis (angl. *supervised learning*) - tai toks sistemos mokymosi tipas, kai yra turimi įėjimo kintamieji X, išėjimo kintamieji Y ir algoritmo pagalba yra siekiama rasti funkciją f, kuri kiekvienam įėjimo kintamąjam rastų teisingą išėjimo reikšmę: $Y = f(X)$. Tai yra bene dažniausiai naudojamas mokymosi metodas. Prižiūrimo mokymosi problemos gali būti toliau skirtomos į klasifikavimo ir regresijos problemas.

- Neprižiūrimas mokymasis (angl. *unsupervised learning*) - tai toks sistemos mokymosi tipas, kai yra turimi tik įėjimo kintamieji X ir jokių išėjimo kintamųjų. Šio mokymosi tikslas yra modeliuoti turimas duomenų struktūras ar pasiskirstymus duomenyse, kad būtų galima sužinoti apie juos daugiau informacijos. Šio tipo uždaviniai gali būti toliau skirstomi į klasterių analizės bei asociacijų.

- Mokymasis su pastiprinimu (angl. *reinforcement Learning*) - tai toks sistemos mokymosi tipas, kuriuo siekiama nustatyti idealų galimą sistemos elgesį, kad būtų gaunama maksimali galima nauda. Kiekvienos mokymosi iteracijos metu sistema gauna tam tikrą grįžtamąjį ryšį, kuris leidžia mokytis toliau ir šis ryšys yra vadinamas pastiprinimo ryšiu.

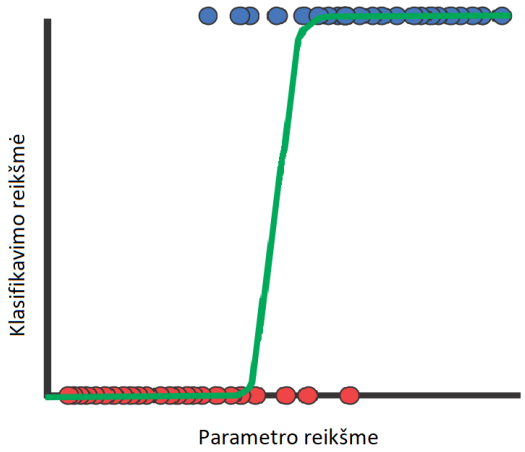
Šiame baigiamajame magistro darbe bus naudojami prižiūrimo sistemos mokymosi metodai, nes kiekvieno duomenų įrašo atveju yra žinoma duomenų kokybės žyma, kitaip - klasė, kuriai yra priskiramas konkretus įrašas. Tolimesniuose poskyriuose bus aprašyti dažniausiai duomenų klasifikatoriams naudojami sistemos mokymosi metodai. Visi minimi klasifikatoriai buvo naudojami ir baigiamojo magistrinio darbo eksperimentinėje dalyje.

1.6.1. Logistinė regresija

Logistinė regresija - tai toks modelis, kuriame tikimybė, kad dvejetainis kintamasis įgyją reikšmę "tiesa", yra modeliuojama kaip daugelio kintamųjų logistinė funkcija:

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}}, \tag{1.4}$$

kur e - natūralusis logaritmas, x_0 - funkcijos vidurio taško x koordinatės reikšmė, L kreivės maksimali vertė, k - kreivės statumo reikšmė. Logistinės funkcijos pavyzdys vieno parametro atveju yra pavaizduotas 2 pav.



2 pav. Logistinės funkcijos naudojimas vieno parametro atveju. Mėlini taškai žymi duomenis, kurių dvejetainis kintamasis turi reikšmę "tiesa", raudoni - "melas"; žalia kreivė žymi logistinę funkciją.

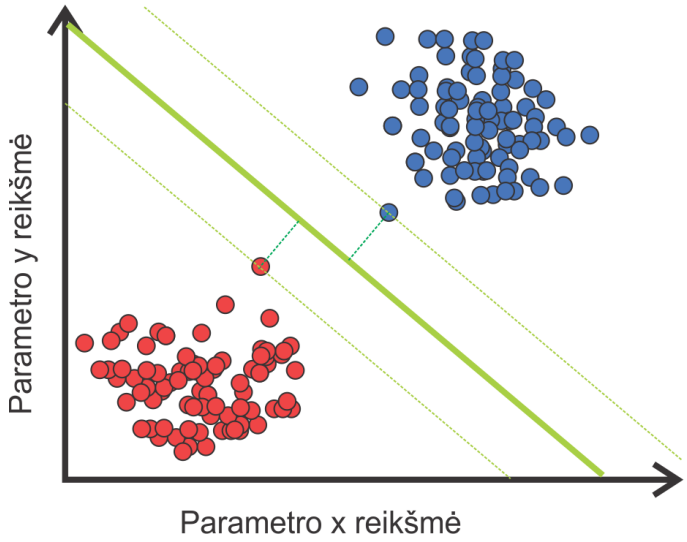
Naudojant logistinę regresiją galima klasifikuoti dvi galimas klases (ne daugiau). Esant trijų ar daugiau klasių problemai, galima naudoti daugelio klasių logistinę funkciją, tačiau šis metodas nėra taikomas taip dažnai. Logistinė funkcija savo forma yra panaši į Heaviside žingsnio funkcija. Tik šiuo atveju, dviejų reikšmių riba yra sušvelninta ir, turint tam tikrus įėjimo kintamuosius, nėra iš karto spėjama klasė, kuriai priklauso duomenys.

Logistinė regresija veikia modeliuodama tikimybes, kad duomenys priklauso numatytąjai klasei [8]. Tarkime, kad klasifikuojame CERN duomenis, kaip sertifikuotus ar nesertifikuotus ir sakome, kad numatytoji klasė yra sertifikuoti duomenys. Tuomet, remiantis logistinės regresijos metodu, yra skaičiuojama timybė, jog esant tam tikriems parametrams, duomenys bus priskirti sertifikuotų klasei. Jei ši tikimybė yra lygi ar didesnė nei 0.5, duomenų įrašas priskiriamas sertifikuotų duomenų klasei, jei mažesnė - nesertifikuotų duomenų klasei. Apmokant klasifikatorių, paremtą logistine regresija, yra keičiami logistinės funkcijos parametrai, siekant atrasti geriausiai konkretų uždavinį atitinkančią logistinės funkcijos formą.

Logistinės regresijos klasifikatorius yra tinkamiausias naudoti, kai uždavinyje egzistuoja riba, kuria galima atskirti dvi klases. Tačiau sprendžant sudėtingesnius uždavinius, šis klasifikatorius gali būti per daug paprastas ir nesugebėti pasiekti aukšto tikslumo.

1.6.2. Atraminių vektorių mašinos

Atraminių vektorių mašinų klasifikatorius - tai prižiūrimas sistemos mokymosi metodas. Kiekvienas įrašas yra vaizduojamas n-matėje erdvėje, kurioje yra ieškoma hiperplokštuma, geriausiai atskirianti nurodytas dvi klases [4]. Atraminis vektorius - tai ir yra kiekvienas duomenų įrašas, kurio parametrai atitinka atraminio vektoriaus koordinates. Pastaruoju metu SVM yra vis plačiau naudojamas dvejetainio klasifikavimo modelis, kuris dažnai naudojamas ir sprendžiant aukštos energijos fizikos problemas [13]. SVM naudojimas apmokant klasifikatorių dvimatėje erdvėje yra pavaizduotas 3 pav.



3 pav. Dvimačio klasifikatoriaus, naudojančio SVM, pavyzdys. Mėlyni ir raudoni taškai žymi skirtingoms klasėms priklausančius duomenis, žalia linija - sprendimo ribą, kuria pasinaudodamas klasifikatorius priskiria duomenis dviems klasėms.

Skirtingai nei logistinės regresijos atveju, šis klasifikatorius sukuria algoritmą, kuris iš karto priskira duomenis konkrečiai klasei (neskaičuodamas tikimybių). Jei galime sakyti, kad vienas duomenų įrašas gali būti vaizduojamas, kaip n-dimensinis vektorius, tai metodo tikslas bus rasti (n-1)-dimensinę hiperplokštumą, kuri atskirtų dvi turimas klases. Realioje sistemoje gali būti ir kelios hiperplokštumos, kurios atitinka šiuos reikalavimus. Tokiu atveju, geriausia hiperploštuma laikoma ta, kurios atstumas nuo abiejų klasių artimiausių taškų yra didžiausias.

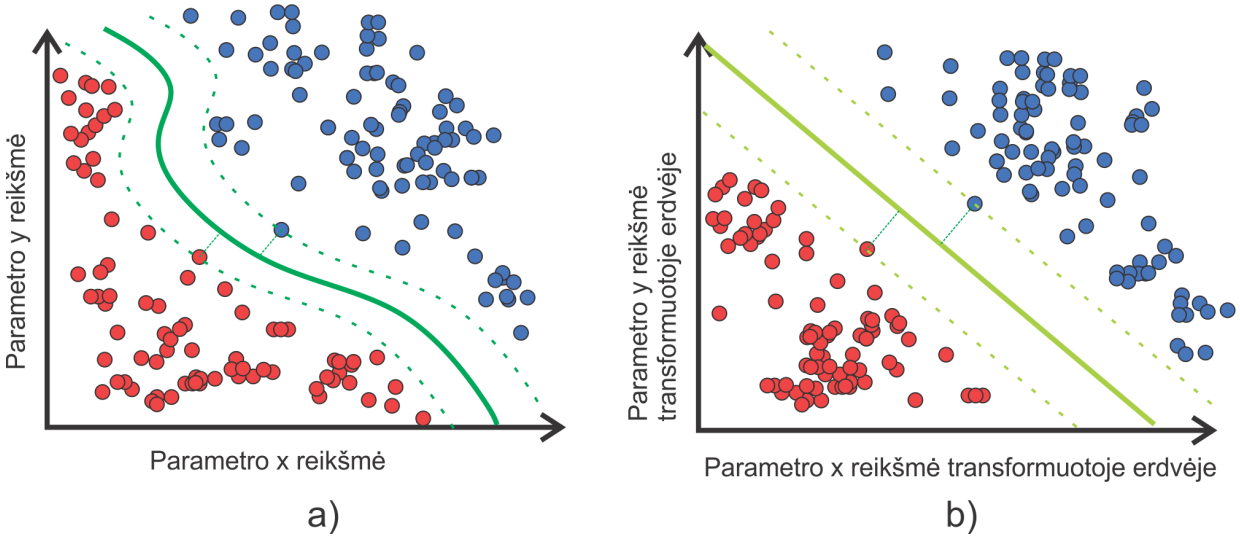
Tarkime, turime du parametrus: x ir y. Tuomet hiperplokštuma ieškoma tokia formule:

$$B0 + (B1 \cdot x) + (B2 \cdot y) = 0, \tag{1.5}$$

kur $B0$, $B1$ ir $B2$ - parametrai, kurių tinkamiausios vertės yra randamos modelio mokymosi metu. Radus šiuos parametrus, yra gaunama tiesė, kuri skiria dvi galimas klases. Jei duomenų įrašas įra vienoje hiperplokštumos pusėje, jis priskiriamas pirmajai klasei, jei kitoje - antrąjai.

Orginaliai, šis atraminių vektorių mašinų metodas buvo pasiūlytas spręsti tiesinėms problemoms [18]. Tačiau vėliau buvo pasiūlytas būdas pritaikyti šį metodą spręsti netiesiniams uždaviniams panaudojant Kernelio metodą [2]. Šio metodo naudojimas nekeičia pagrindinių atraminių

vektorių mašinių principų, tačiau algoritme esančios sandaugos yra keičiamos netiesine Kernelio funkcija. Tai sukuria transformuotą parametrų erdvę, kurioje radus hiperploštumą ir grįžus prie orginalios daugiadimensinės parametrų erdvės, hiperplokštuma yra netiesinė ir gali būti pritaikoma spęsti atitinkamiems uždaviniams (4 pav.).



4 pav. Dvimačio netiesinio klasifikatoriaus, naudojančio SVM, pavyzdys. a) paveikslėlyje pavaizduotas netiesinis uždavinys, kuriam spręsti erdvė yra transformuojama į tiesinę parametrų erdvę (b) paveikslėlis).

SVM pliusai: klasifikavimas šiuo metodu veikia itin tiksliai ir greitai, jei galima rasti gerai duomenis atskiriančią hiperploštumą. Šis metodas yra efektyvus ir daugelio dimensijų erdvėse, net jei dimensijų skaičius yra didesnis už duomenų įrašų skaičių. Atraminių vektorių mašinų pagrindinis minusas - jei yra turimas didelis duomenų skaičius, sistemos mokymasis trunka ilgai, palyginti su kitų klasifikavimo metodų mokymusi. Taip pat šis metodas gali duoti gana mažą tikslumą, jei daugiadimensinėje erdvėje duomenų įrašai persikloja, t.y. neįmanoma rasti tikslios ribos, skiriančios duomenų klases.

1.6.3. Išpūsti sprendimų medžiai

Išpūstų sprendimų medžių klasifikatorius (angl. *Boosted Decision Trees*, sutr. BTD) - tai sistemos mokymosi metodas, kurio metu yra naudojamasi srendimo medžiais, kad nuo duomenų parametrų verčių analizavimo būtų prieita prie išvadų apie duomenų klasę. Išpūsti sprendimų medžiai yra ypatingi tuo, kad kiekvienas iš kuriamų sprendimo medžių taiso prieš tai buvusio medžio klaidas, taip pasiekiant didesnį klasifikavimo tikslumą. Tai yra prižiūrimas sistemos mokymosi metodas. Per pastaruosius kelerius metus bavo padaryta didelė pažanga kuriant vis tikslesnius BDT metodus [14] ir šis metodas tampa vis dažniau naudojamas aukštos energijos fizikos srityje [12].

Pagrindiniai išpūstų sprendimų medžių elementai:

- Klaidos funkcija, kurią reikia optimizuoti
- Baziniai modeliai, kurie priima klasifikavimo sprendimus su palyginti mažu tikslumu
- Sudėtinis modelis, kuris sujungia bazinių modelių sprendimus.

Formaliai sudėtinį modelį galime užrašyti tokia formule:

$$g(x) = f_0(x) + f_1(x) + f_2(x) + \ldots, \tag{1.6}$$

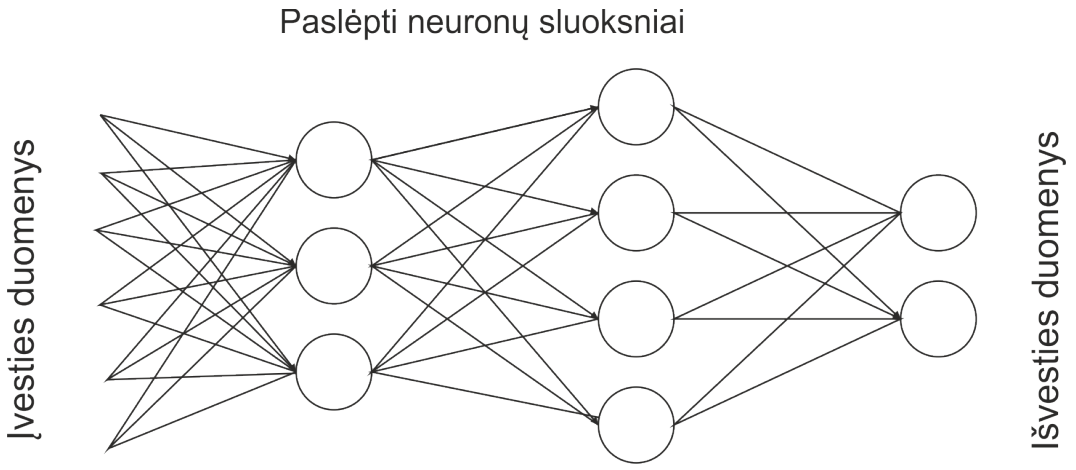
kur $g(x)$ - bazinių klasifikatorių $f_i(x)$ suma. Išpūstų sprendimų medžių atveju, kiekvienas iš šių bazinių klasifikatorių yra atskiras sprendimų medis. BDT mokymasis apima mokymasi sudėti šiuos bazinius klasifikatorius tokiu būdu, kad būtų kuo labiau sumažinta klaidos funkcija. Dažniausiai išpūstiems sprendimo medžiams yra naudojamas gradientinio mažėjimo principas, kuris šiuo atveju lemia tai, kad atskiri baziniai modeliai yra pridedami prie bendro išpūstų spendimo medžio taip, kad būtų judama klaidos funkcijos gradiento mažėjimo kryptimi.

Išpūstų sprendimų medžių didžiausi privalumai prieš kitus sistemos mokymosi metodus yra tokie, kad šis metodas reikalauja mažesnio duomenų paruošimo, gali dirbti ne tik su skaitiniais, bet ir su kategoriniais duomenimis. Taip pat modelis gerai veikia su dideliais duomenų kiekiais bei geriausiai atitinka žmonių sprendimų priėmimo modelį.

Kita vertus, šis modelis dažnai gali būti ne toks tikslus ir greitai pritaikomas naujiems atvejams. Net ir mažas pokytis duomenyse gali lemti didelius pokyčius sprendimo medžio struktūroje. Taip pat, naudojant šį metodą dažnai pasitaiko situacijos, kai yra sukuriami per daug sudėtingi medžiai, kurie atsižvelgia į menkiausius triukšmus mokymosi duomenyse, todėl gali pasiekti labai didelį jų klasifikavimo tikslumą. Tačiau šį sukurtą modelį naudojant testavimo duomenims, tikslumas gana drąstiškai krenta. Tai galima spręsti nurodant, kiek šakų medis gali turėti arba koks minimalus duomenų skaičius turėtų priklausyti kiekvienam iš regionų. Dar vienas išpūstų sprendimo medžių minusas - labai didelis kompiuterinės atminties naudojimas (arba itin ilgas mokymosi laikas).

1.7. Dirbtiniai neuroniniai tinklai

Dirbtinis neuroninis tinklas (angl. *artifitial neural network*, sutr. ANN)- tai skaičiavimo sistema, sudaryta iš paprastų, tačiau stipriai tarpusavyje sujungtų elementų, vadinamų dirbtiniais neuronais. ANN yra paremti biologinių neuroninių tinklų veikimo principu. Dažniausiai ANN yra sudaryti iš įėjimo sluoksnio, keletos paslėptų neuronų sluoksnių ir išėjimo sluoksnio neuronų. Supaprastintas dirbtinio neuronų tinklo modelis yra pavaizduotas 5 pav.



5 pav. Dirbtinio neuronų tinklo modelis

Sluoksnių jungtims yra priskiriami tam tikri svoriai, kurie vėliau naudojami matematinėse funkcijose, kurios nurodo neurono aktyvavimą [7]. Keičiant dirbtinių neuronų svorius galima

pasiekti, kad tinklui gavus tam tikrą įėjimo duomenų rinkinį, išėjime būtų gaunamas teisingas rezultatas. Paslėpto sluoksnio neurono *i* išėjimo vertė h_i yra lygi:

$$h_i = \sigma\left(\sum_{j=1}^{N} W_{ij}x_j + T_i^{hid}\right), \tag{1.7}$$

kur $\sigma()$ - aktyvacijos funkcija, N - įėjimo neuronų skaičius, W_{ij} - neuronų svoriai, x_j - įėjimo neuronų vertės, T_i^{hid} - paslėpto sluoksnio neurono slenkstinė vertė [19]. Aktyvacijos funkcijai gali būti naudojamos įvairios matematinės funkcijos: tiesinė, apribota tiesinė, slenkstinė, sinusiodinė, tangentinė, hiperbolinė, sigmoidinė (arba logistinė) ir kitos. Detaliau dirbtinio neuronų tinklo veikimas bus aprašytas tolimesniuose skyriuose.

1.7.1. Dirbtinių neuroninių tinklų mokymasis

Kadangi dirbtinius neuroninius tinklus įprastai sudaro šimtai ar tūkstančiai neuronų, yra kuriami reikalingi algoritmai, randantys teisingus tinklo neuronų svorius. Šis teisingų svorių ieškojimo procesas yra vadinamas dirbtinio neuroninio tinklo mokymusi ir tam dažniausiai yra naudojami sistemos mokymosi algoritmai. Procesas pradedamas su atsitiktiniais neuronų svoriais ir mokymosi metu jie yra keičiami taip, kad galutinio rezultato paklaida būtų minimali.

Dažniausiai neuroniniai tinklai yra apmokomi naudojant jungčių svorių gradientinio nusileidimo metodą. Šis metodas yra paremtas tuo, kad skaliarinio lauko gradientas visuomet rodo greičiausią lauko augimo kryptį, o jo antigradientas rodo greičiausią mažėjimo kryptį. Gradientui skaičiuoti yra naudojama klaidos funkcija (angl. *loss function*). Nors galutinį sukurto neuroninio tinklo gerumą vertiname skaičiuodami jo tikslumą (klasifikavimo atveju - kokia dalis duomenų buvo teisingai klasifikuoti), šis dydis nėra tinkamas vertinti modeliui mokymosi metu ir naudoti skaičiuojant gradientą. Taip yra todėl, kad apmokant neuroninį tinklą, svoriai yra keičiami po truputį ir mažas jų pokytis gali visiškai nepakeisti klasifikavimo tikslumo. Dėl šios priežasties yra naudojama klaidos funkcija, kurios vertę gali pakeisti net ir nedidelis tinklo svorių pokytis. Tai leidžia aiškiau matyti, į kurią pusę yra teisinga keisti svorius.

Dar viena sąvoka, reikalinga kalbant apie dirbtinių neuroninių tinklų mokymąsį - grįžtamasis perdavimas (angl. *backpropapagation*). Visi duomenys iš mokymuisi skirto duomenų rinkinio yra padalinami į paketus pagal sistemoje nustatytą paketo dydį. Vienos mokymosi iteracijos metu tinklui yra siunčiamas vienas duomenų paketas. Vyksta priekinis perdavimas (angl. *forward pass*), duomenys pereina visus dirbtiniame neuronų tinkle esančius sluoksnius, spėjamos klasės, kurioms duomenys turėtų priklausyti, ir pagal formules skaičiuojama visa paklaida. Tuomet vyksta antrasis žingnis - atgalinis perdavimas (angl. *backwards pass*). Šio žingsnio tikslas atnaujinti visų tinkle esančių neuronų jungčių svorius bei slenkstines vertes, kad gaunama paklaida būtų kuo mažesnė. Tam pasiekti naudojamos gautos vertės ir tinklu einant nuo galo link pradžios, pagal nustatytas funkcijas keičiamos svorių bei slenksčių vertės. Vienas grįžtamasis perdavimas, sudarytas iš šių dviejų žingsnių, yra vadinamas viena iteracija, o visos reikalingos iteracijos, kai per neuroninį tinklą pereina visi duomenų rinkinį sudarantys paketai, yra vadinama viena epocha.

1.7.2. Mokymosi parametrai

Kalbant apie neuroninio tinklo mokymąsi naudojant gradientinio nusileidimo metodą, vienas iš svarbiausių naudojamų parametrų yra mokymosi greitis. Tarkime, turime klaidos funkciją

$C(v_1, v_2)$, kuri priklauso nuo dviejų parametrų v_1 ir v_2. Jei šiek tiek pakeičiame šių parametrų vertes (pavyzdžiui, pakoreguojame neuroninio tinklo svorius), tuomet funkcijos $C(v_1, v_2)$ reikšmės pokytis gali būti nusakomas tokia formule:

$$\Delta C \approx \frac{\partial C}{\partial v_1}\Delta v_1 + \frac{\partial C}{\partial v_2}\Delta v_2. \tag{1.8}$$

Kadangi uždavinio tikslas yra sumažinti klaidos funkciją, yra reikalinga parinkti tokius v_1 ir v_2, kad ΔC būtų neigiamas. Funkcijos C gradientą galime aprašyti kaip dalinių išvestinių vektorių:

$$\nabla C = \left(\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2}\right). \tag{1.9}$$

Perrašome parametrų v_1 ir v_2 pokyti, kaip vektorių $\Delta v = (v_1, v_2)$. Tokiu būdu 1.8 formulę galima perrašyti:

$$\Delta C \approx \nabla C \cdot \Delta v. \tag{1.10}$$

Tokiu atveju, norėdami gauti neigiamą ΔC reikšmę, pasirenkame parametrų v pokytį, lygų:

$$\Delta v = -\eta \nabla C, \tag{1.11}$$

kur η - mokymosi greitis. η visada yra mažas ir teigiamas parametras. Tuomet pertvarkę 1.10 lygtį gauname:

$$\Delta C \approx -\eta \nabla C \cdot \nabla C = -\eta \left\|\nabla C\right\|^2. \tag{1.12}$$

Kadangi η ir $\|\nabla C\|^2$ visada yra teigiami dydžiai, toks parametrų v pokyčio pasirinkimas lemia klaidos funkcijos C mažėjimą, ko ir siekiame. Tokiu būdu metodą kartojame tiek kartų, kiek reikia norint maksimaliai sumažinti klaidos funkciją.

Tam, kad gradientinio nusileidimo metodas veiktų tinkamai, turime tinkamai pasirinkti mokymosi greitį η. Jis turi būti pakankamai mažas, kad 1.12 aproksimacija būtų pakankamai tiksli. Tuo pačiu metu, mokymosi greitis negali būti per daug mažas, nes tuomet net ir pakankamai dideli parametrų pokyčiai lems mažas klaidos funkcijos pokyčių vertes ir algoritmas veiks labai lėtai.

Papildomai, kuriant dirbtinį neuroninį tinklą su GraphLab Create paketo funkcijomis, galima nurodyti parametrą "*learning_rate_schedule*". Šis parametras leidžia nustatyti skirtingą mokymosi greičio vertę skirtingų mokymosi iteracijų metu. Artėjant prie globalaus klaidos funkcijos minimumo yra naudinga jo siekti kuo mažesniais žingsniais. Mokymosi greičio grafiko parametro naudojimas leidžia pradėti dirbtinio neuroninio tinklo mokymąsi su didele mokymosi greičio verte, greičiau priartėti prie klaidos funkcijos minimumo ir tada siekti geriausio klasifikatoriaus tikslumo mažais žingsniais dėl palaipsniui mažinamos mokymosi greičio vertės.

Mokymosi greičio grafiko parametras turi dvi galimas reikšmes: "*exponential_decay*" ir "*polynomial_decay*". Šios reikšmės skiriasi tuo, pagal kokią funkciją yra mažinama mokymosi greičio vertė. Eksponentinio mažinimo atveju yra naudojama tokia funkcija:

$$new\_lr = lr \cdot lr\_gamma^{epoch/lr\_step}, \tag{1.13}$$

kur new_lr - nauja mokymosi greičio vertė, lr - esama mokymosi greičio vertė, lr_gamma - papildomas parametras, kuris sistemos numatytuoju atveju yra lygus 0.1, $epoch$ - epochos numeris, lr_step - papildomas parametras, kuris sistemos numatytuoju atveju yra lygus 1. Polinominio mokymosi greičio mažėjimo atveju yra naudojamasi tokia funkcija:

$$new\_lr = lr \cdot (1 + (epoch/lr\_step) \cdot lr\_gamma) - lr\_alpha, \tag{1.14}$$

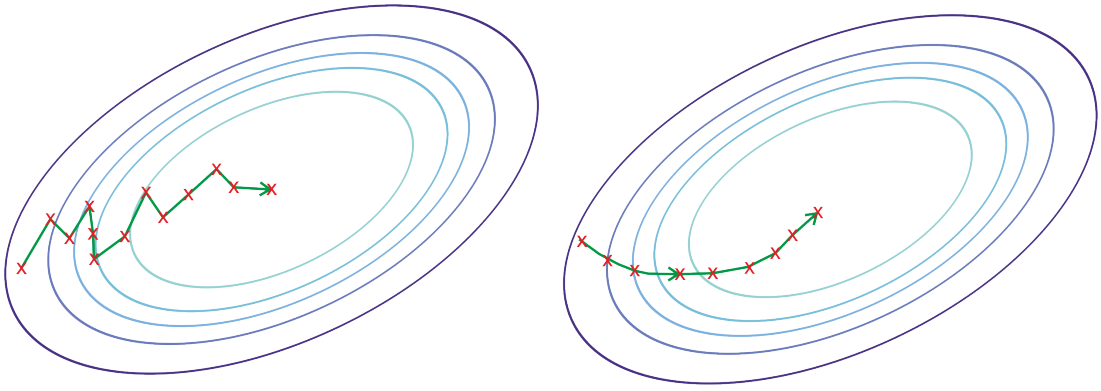
kur lr_alpha - papildomas parametras, kuris sistemos numatytuoju atveju yra lygus 0.5. Dirbtinio neuroninio tinklo kūrimo metu galima keisti parametrų lr_gamma, lr_alpha ir lr_step parametrų vertes. Taip pat papildomai galima nurodyti parametro $min\_learning\_rate$ vertę, kuri sistemos numatytuoju atveju yra lygi 0.00001 ir nurodo, kokią minimalią mokymosi greičio vertę pasiekus, ji toliau mažėti nebegali.

Vienareikšmiško atsakymo, kokia turėtų būti mokymosi greičio vertė, nėra. Šis parametras turi būti pritaikytas kiekvieno konkretaus uždavinio atveju.

Dar vienas mokymosi parametras - momentas, yra reikalingas siekiant pasiekti greitesnį neuroninio tinklo konvergavimą į klaidos funkcijos globalų minimumą. Momento naudojimo apmokant neuroninius tinklus idėja yra ta, kad atnaujinant tinklo svorių vertes būtų atsižvelgiama ir į pokyčius, kurie buvo atlikti buvusiose iteracijose. Grįžtant prie anksčiau aprašyto klaidos funkcijos pokyčio skaičiavimo pavyzdžio, parametrų v pokytis n-tosios iteracijos metu būtų:

$$\Delta v(n) = -\eta \nabla C + \alpha \Delta v(n-1), \tag{1.15}$$

kur α - momento vertė, nuodanti, kiek įtakos praeitos iteracijos pokyčiai turėtų turėti esamai iteracijai [9]. Momentas neuroninio tinklo mokymuisi prideda "slopinamąjį" efektą, sumažina osciliacijas artėjant prie klaidos funkcijos minimumo. Taip pat šio parametro naudojimas padeda modeliui neapsistoti klaidos funkcijos lokaliuose minimuose. Artėjimas prie klaidos funkcijos minimumo naudojant momento parametrą ir be jo, yra pavaizduoti 6 pav.



a) b)

6 pav. Artėjimas prie klaidos funkcijos minimumo mokantis be momento parametro (a) paveikslėlis) ir su momento parametru (b) paveikslėlis).

1.7.3. Aktyvacijos funkcijos

Aktyvacijos funkcijos - tai tokios funkcijos, kurios paima sluoksnio įėjimo reikšmes ir jas transformuoja į išėjimo signalą. Dažniausiai, uždaviniai sprendžiami dirbtiniais neuroniniais tink-

lais yra ganėtinai sudėtingi ir jų neįmanoma aprašyti tiesinėmis funkcijomis. Tokiu atveju turi būti naudojamos netiesinės aktyvacijos funkcijos, kurios leidžia sukurti dirbtinius neuroninius tinklus, gebančius išspręsti sudėtingus klasifikavimo ar kitokius uždavinius. Tiesinės aktyvacijos funkcijos gali būti naudojamos tik itin retais ir paprastais atvejais.

Dirbtiniai neuroniniai tinklai yra apmokomi naudojant grįžtamąjį perdavimą, todėl yra reikalinga naudoti tokias aktyvacijos funkcijas, kurios yra diferencijuojamos, t.y. funkcija turi išvestinę bet kuriame taške iš viso verčių intervalo. Grįžtamasis perdavimas yra naudojamas tam, kad neuroninio tinklo jungčių svoriai galėtų būti tinkamai koreguojami kiekvienos mokymosi iteracijos metu, jis yra reikalingas norint suskaičiuoti klaidos funkcijos gradientą.

Aktyvacijos funkcijos yra svarbios dirbtinių neuroninių tinklų kokybei, todėl svarbu žinoti, kokia funkcija geriausiai tinka konkrečiu tinklo kūrimo atveju. Dažniausiai naudojamos aktyvacijos funkcijos bus plačiau aprašytos kitame skyriuje, kalbant apie dirbtinio neuronų tinklo sluoksnius.

1.7.4. Dirbtinių neuronų sluoksniai

Dirbtiniai neuroniniai tinklai yra sudaryti iš įvairių sluoksnių, kurių pagrindiniai bus aprašyti šiame skyriuje. Kiekvienas dirbtinių neuronų tinklas yra sudarytas iš įėjimo, paslėptųjų ir išėjimo sluoksnių. Įėjimo sluoksnis yra pasyvus, nes priima vertes ir perduoda tolimesniems sluoksniams jų visiškai nekeisdamas. Tuo tarpų paslėptieji ir išėjimo sluoksniai yra aktyvūs. Tai reiškia, kad juose reikšmės yra keičiamos. Pavyzdžiui, dauginamos iš jungčių svorių arba keičiamos kitokių funkcijų [15].

Šiame baigiamajame magistriniame darbe dirbtiniams neuroniniams tinklams buvo naudojamasi GraphLab Create paketu, kuriame prieinami tokie ANN sluoksniai:

- FullConnectionLayer - tai toks jungiamasis dirbtinių neuroninių tinklų sluoksnis, kuris sujungia kiekvieną į sluoksnį įeinančią vertę su kiekvienu šio sluoksnio vienetu (dirbtiniu neuronu). Todėl šis sluoksnis yra vadinamas pilnai sujuntu sluoksniu. Kiekvienas sluoksnio dirbtinis neuronas taip pat turi tam tikrą slenkstinę vertę, kuri yra pridedama prie į sluoksnį įeinančios vertės. Šios slenkstinės vertės yra automatiškai keičiamos apmokant tinklą. Pilnai sujungtas sluoksnis turi kelis galimus parametrus:
  - num_hidden_units - dirbtinių neuronų skaičius sluoksnyje
  - init_bias - pradinė dirbtinių neuronų slenkstinė vertė. Numatytoji reikšmė - 0.
  - init_random - pasiskirstymas iš kurio inicijuojami sluoksnio parametrai. Galimos reikšmės: 'gaussian', 'xavier'.
  - init_sigma - jei sluoksnio parametrai yra inicijuojami pagal Gauso skirstinį, init_sigma parametras nurodo standartinio nuokrypio dydį. Nuo jo priklauso parametrų skirstinio kreivės amplitudė bei plotis.

  Dėl didelio parametrų skaičiaus, šis sluoksnis gali numatyti net labai sudėtingus ryšius tarp duomenų parametrų reikšmių, tačiau gali būti sunkiai apmokomas.

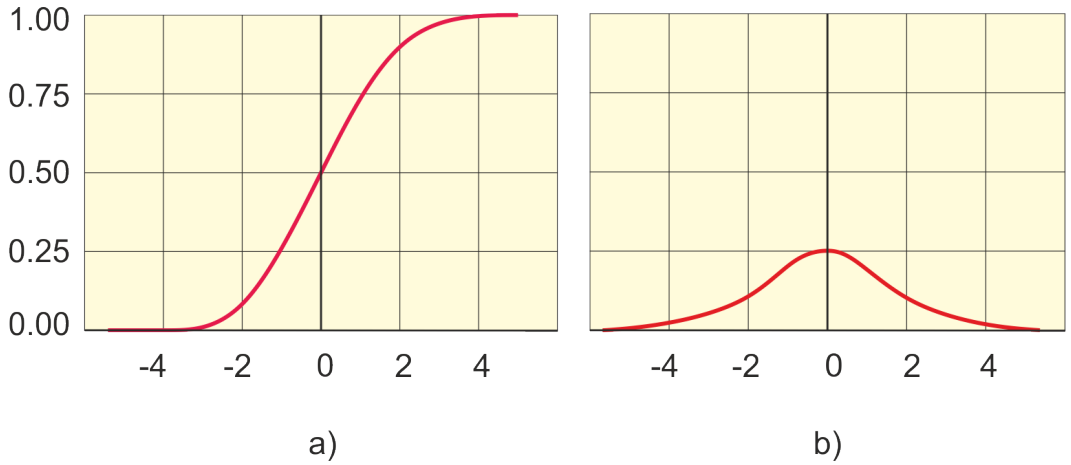
- SigmoidLayer - tai aktyvacijos sluoksnis, kuris transformuoja visus įėjimus su tokia aktyvacijos funkcija:

$$f(x) = 1/(1 + e^{-x}). \tag{1.16}$$

Ši transformacija sukuria netiesiškumą dirbtiniame neuroniniame tinkle. Sluoksnis dažniausiai yra naudojamas po pilnai sujungto dirbtinių neuronų sluoksnio. Šios aktyvacijos funkcijos minusas - dingstantys gradientai. Kai funkcija pasiekia tam tikrą vertę, tinklas toliau mokosi labai lėtai, nes net ir didelis įėjimo vertės pokytis sukelia mažą išėjimo vertės pasikeitimą. Funkcijos pliusas - lengvai skaičiuojama išvestinė:

$$f'(x) = f(x)(1 - f(x)). \tag{1.17}$$

Sigmoidinė funkcija ir jos išvestinė pavaizduotos 7 pav.



7 pav. Sigmoidinė funkcija (a) paveikslėlis) ir jos išvestinė (b) paveikslėlis).

- TanhLayer - tai aktyvacijos sluoksnis, kuris transformuoja visus įėjimus su tokia aktyvacijos funkcija:

$$f(x) = tanh(x). \tag{1.18}$$

Ši transformacija sukuria netiesiškumą dirbtiniame neuroniniame tinkle. Sis sluoksnis dažniausiai yra naudojamas po pilnai sujungto dirbtinių neuronų sluoksnio. Funkcijos išvestinė skaičiuojama pagal funkciją:

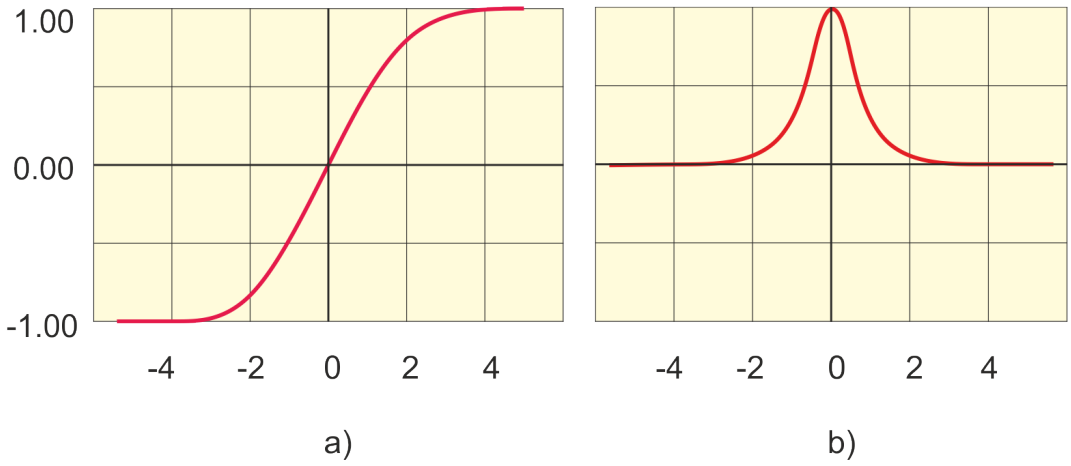
$$f'(x) = 1 - tanh^2(x) \tag{1.19}$$

Hiperbolinė tangentinė funkcija ir jos išvestinė pavaizduotos 8 pav.

Nors hiperbolinės tangentinės funkcijos išvestinės forma panaši į sigmoidinės, ši funkcija yra centruota ties nuliu ir jos išvestinė yra didesnė. Be to, hiperbolinio tangento aktyvacijos funkcija geriau veikia su neigiamomis įėjimo vertėmis.

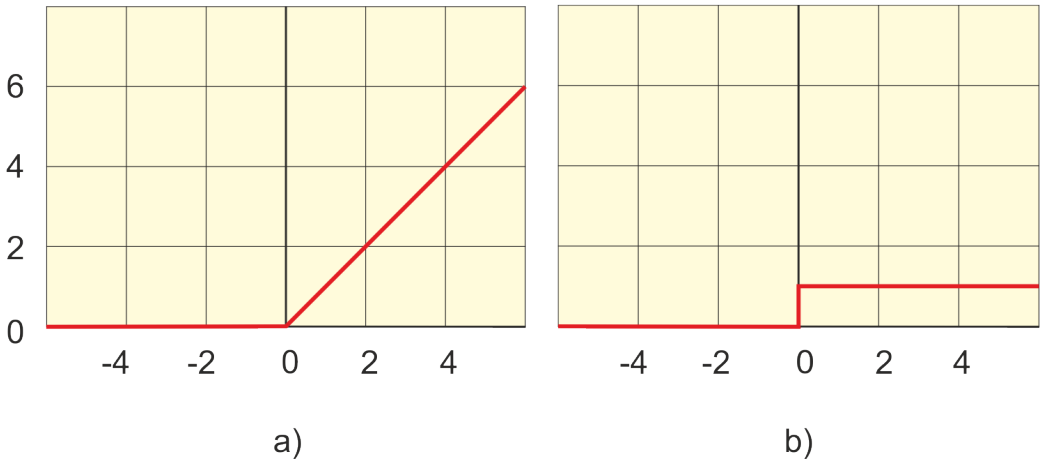
- RectifiedLinearLayer - tai aktyvacijos sluoksnis, kuris transformuoja visus įėjimus su tokia aktyvacijos funkcija:

$$f(x) = max(0, x). \tag{1.20}$$



8 pav. Hiperbolinė tangentinė funkcija (a) paveikslėlis) ir jos išvestinė (b) paveikslėlis).

Ši transformacija sukuria netiesiškumą dirbtiniame neuroniniame tinkle. Šis sluoksnis dažniausiai yra po pilnai sujungto dirbtinių neuronų sluoksnio. Ištaisyta tiesinė funkcija grąžina išėjimo reikšmę lygią įėjimui, jei įėjimo vertė yra didesnė už nulį. Tai yra paprasčiausia netiesinė aktyvacijos funkcija, kuri yra tinkama naudoti kuriant neuroninį tinklą. Funkcija ir jos išvestinė pavaizduotos 9 pav.



9 pav. Ištaisyta tiesinė funkcija (a) paveikslėlis) ir jos išvestinė (b) paveikslėlis).

Šios funkcijos pagrindinis pliusas - labai greitas apmokymas. Dažnai naudojamos ir funkcijos atmainos. Galimas funkcijos minusas - jei per tokį dirbtinių neuronų sluoksnį pereina didelio gradiento duomenys, tai gali sukelti tokį jungčių svorių pakitimą, kad toliau mokantis, nepriklausomai nuo įeinančių duomenų, neuronas nebebus aktyvuotas. Kitaip sakant, neuronas "mirs". Tokiu būdu, jei tinklo mokymosi greitis yra per didelis, dalis neuronų gali būti nepataisomai "mirę". Tačiau, šiai problemai spęsti yra keli pasiūlymai, iš kurių dažniausiai naudojamas - tiesiog nustatatyti mažesnį tinklo mokymosi greitį.

- PoolingLayer - vienas iš svarbių sluoksnių tipų, tai sujungimo (angl. *pooling*) sluoksniai. Tai yra tam tikra netiesinė duomenų skaičiaus sumažinimo forma. Egzistuoja keli netiesiniai būdai, kaip galima atlikti duomenų sujungimą (jų skaičiau sumažinimą). Bene dažniausiai tam naudojamas dirbtinių neuronų sluoksnis - MaxPoolingLayer. Šis sluoksnis padalina įėjimo duomenis į tam tikras dalis, kurias nurodo sluoksnio parametras $kernel\_size$. Iš kiekvienos dalies atiduodamos tos dalies maksimalios duomenų parametrų vertės.

Kiti dažnai naudojami sluoksnių tipai: SumPoolingLayer (iš kiekvienos dalies grąžinamos tos dalies duomenų parametrų verčių sumos), AveragePoolingLayer (iš kiekvienos dalies atiduodami tos dalies duomenų parametrų verčių vidurkiai). Dažniausiai šie sluoksniai yra naudojami po kelių aktyvacijos sluoksnių.

Sujungimo sluoksnių naudojimas padeda sumažinti neuroniniam tinklui reikalingų duomenų parametrų skaičių. Dažnai tai gelbsti nuo neuroninio tinklo prisitaikymo prie triukšmo esančio duomenyse (daugiau apie tai kitame poskyryje).

- DropoutLayer - šis sluoksnis priima visus įėjimo duomenis ir juos prilygina nuliui su tikimybe lygia sluoksnio parametrui $treshold$. Tai apsaugo tinklą nuo vienos dažniausiai pasitaikančių problemų kuriant neuroninį tinklą - tinklo prisitaikymo prie duomenyse esančių triukšmų. Atmetimo sluoksnis yra vienas iš dažniausiai naudojamų metodų minėtai problemai spręsti.

  Šis sluoksnis naudojamas tik apmokant neuroninį tinklą su mokymuisi skirtais duomenimis. Naudojant tinklą su testavimo duomenimis, atmetimo sluoksnis yra praleidžiamas. Įvairūs tyrimai įrodė, kad atmetimo sluoksnio naudojimas gali gana ženkliai padidinti dirbtinių neuroninių tinklų, naudojamų prižiūrimam sistemos mokymuisi, tikslumą bei greitį [16].

- SoftmaxLayer - tai išėjimo sluoksnis, kuris transformuoja kiekvieną įėjimo vertę tokia funkcija:

$$f(x_i) = e^{x_i} / \sum_k e^{x_k}, \tag{1.21}$$

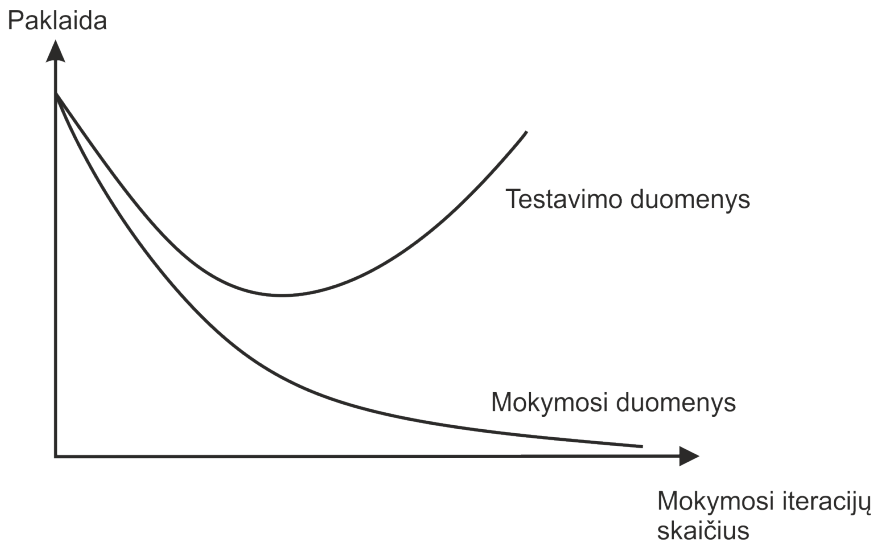
  kur x_i tai i-tojo išėjimo reikšmė susumuota per visus įėjimus k. Visos sluoksnio išėjimo vertės yra intervale [0,1], o jų suma yra lygi 1. Klasifikavimo atveju, kiekvienas išėjimas žymi skirtingą klasę. Dirbtinio neurono išėjimo reikšmė rodo tikimybę, kad duomuo priklauso atitinkamai klasei. Sluoksnis, esantis prieš SoftmaxLayer sluoksnį, turi būti pilnai sujungtas sluoksnis, turintis tiek dirbtinių neuronų, kiek yra galimų duomenų klasių.

  Apibendrinant, Softmax sluoksnis neprisideda prie teisingo dirbtinio neuroninio tinklo klasifikavimo uždavinio sprendimo. Šis sluoksnis tiesiog transformuoja jau esančias išėjimo vertes į aiškiau suvokiamas vertes - tikimybes, kad duomenų įrašas priklauso konkrečiai klasei.

1.7.5. Dažniausiai pasitaikantys sunkumai

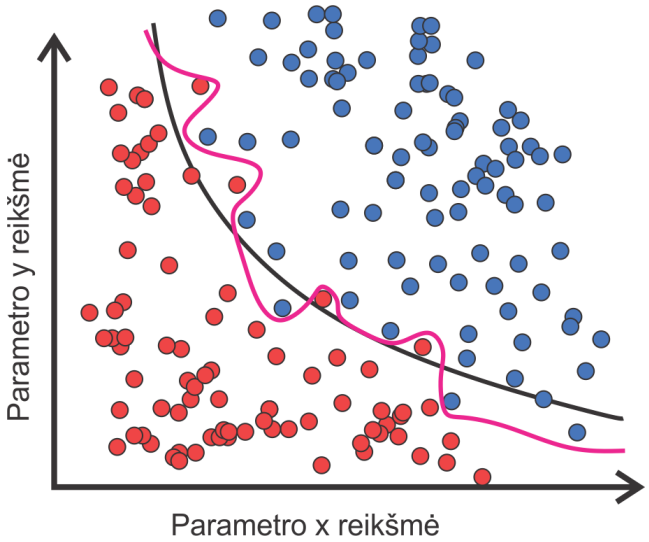
Viena iš dažniausiai pasitaikančių problemų apmokant neuroninius tinklus - tai neuroninio tinklo prisitaikymas prie triukšmo duomenyse (angl. *overfitting*). Ši situacija dažnai pasitaiko, kai palyginti paprastam uždaviniui spręsti yra sukuriamas sudėtingas dirbtinis neuroninis tinklas. Šios problemos pažymius galime pastebėti jei, didėjant mokymosi iteracijų skaičiui, tinklo klasifikavimo tikslumas mokymosi duomenims didėja, tačiau testavimo duomenims - mažėja (10 pav.). Tai parodo, kad tinklas prisitaiko prie menkiausių pokyčių mokymuisi skirtuose duomenyse, net tų, kurie nėra reikšmingi ar svarbūs.

Atsiradus tokiai problemai yra patariama supaprastinti tinklo architektūrą, sumažinti sluoksnių ar mokymosi iteracijų skaičių. Taip pat į tinklo architektūrą galima įtraukti tam tikro tipo sluoksnius, pavyzdžiui, atmetimo sluoksnį (*DropoutLayer*).



10 pav. Neuroninio tinklo klaidos dydžio priklausomybė nuo mokymosi iteracijų skaičius testavimo bei mokymosi duomenims, kai tinkle pasireiškia prisitaikymo prie triukšmo duomenyse problema.

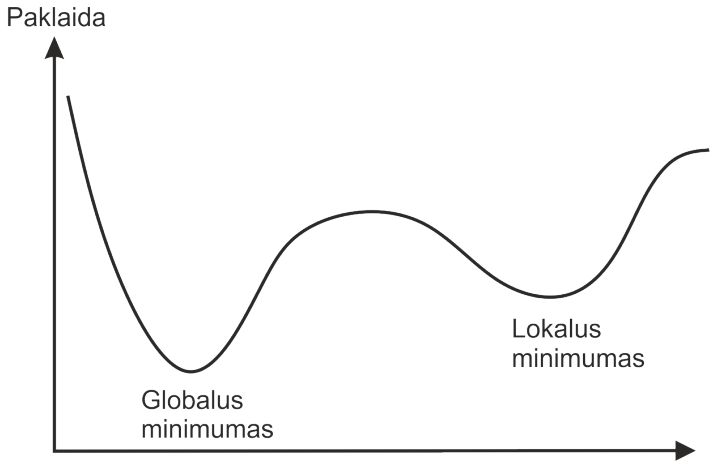
Šis efektas pastebimas tada, kai modelis įsimena mokymuisi skirtus duomenis, o ne mokosi atrasti juose esančias tendencijas (11 pav). Be jau minėtų prisitaikymo pasekmių, dėl šios problemos taip pat kyla modelio naudojamų kompiuterio resursų dydis bei tinklo architektūros sudėtingumas, todėl modelis darosi sunkiai pritaikomas uždaviniui spręsti. Kadangi sistemos mokymosi metodų tikslas yra, stebint mokymuisi skirtus duomenis, sukurti apibendrintą modelį, gebantį dirbti su kitais duomenimis iš to pačio šaltinio, prisitaikymas prie trukšmų yra gana rimta ir nepageidaujama problema, kuriai spręsti yra skiriama daug dėmesio.



11 pav. Neuroninio tinklo prisitaikymas prie triukšmo mokymosi duomenyse. Juoda linija žymi reguliuojamą modelį, kuris geriausiai atitinka turimus duomenis. Rožinė linija žymi per daug prie mokymosi duomenų prisitaikiusį modelį. Nors mokymosi duomenims šis modelis turės didžiausią tikslumą, jį naudojant su testavimui skirtais duomenimis, yra didelė tikimybė, kad tikslumas bus gerokai mažesnis.

Dar viena problema, pasitaikanti kuriant neuroninius tinklus - tai lokalūs minimumai klaidos funkcijoje (12 pav.). Jei dirbtinio neuroninio tinklo apmokymo metu, keičiant svorius yra paten-

kamai į lokalaus minimumo zoną, modelis tai priima kaip minimumą ir stengiasi į jį konverguoti. Tuo tarpu teisingas sprendimo būdas būtų keisti svorius tiek, kad klaidos funkcija šiek tiek padidėtų ir taip būtų galima pasiekti globalų minimumą.



12 pav. Lokalus ir globalus minimumai klaidos funkcijoje.

Lokalių minimumų funkcijoje gali būti ne vienas, todėl itin svarbu naudoti papildomas priemones, kad šios problemos būtų išvengiama. Vienas iš dažniausiai naudojamų šios problemos sprendimo būdų - momento, kaip dirbtinio neuronų tinklo mokymosi parametro, naudojimas. Momento naudojimas didina dirbtinio neuronų tinklo mokymosi žingnių dydį, kas lemia tai, kad dauguma atvejų, patekus į lokalaus minimumo tašką iš jo tiesiog "iššokama". Be to, naudojant momento parametrą, neuroninis tinklas taip pat atsižvelgia ir į praėjusių iteracijų įtaką, taip gerindamas tinklo veikimą. Reikia pastebėti, kad jei yra naudojama didelė momento vertė, mokymosi greitis turėtų būti sumažinamas. Kitu atveju didėja tikimybė, kad minimumai bus peršokti labai dideliu žingsniu ir bus sunku pasiekti klaidos funkcijos globalų minimumą.

2. Eksperimentinė dalis

Eksperimentinė darbo dalis buvo atliekama Python programavimo kalba. Papildomai buvo naudojamasi GraphLab Create Python paketu. GraphLab Create leidžia atlikti didelio mąsto duomenų analizę, dirbti su duomenų struktūromis bei naudotis sistemos mokymosi įrankiais. Kodas buvo rašomas Jupyter Notebook aplikacijoje.

2.1. Duomenų analizė ir paruošimas

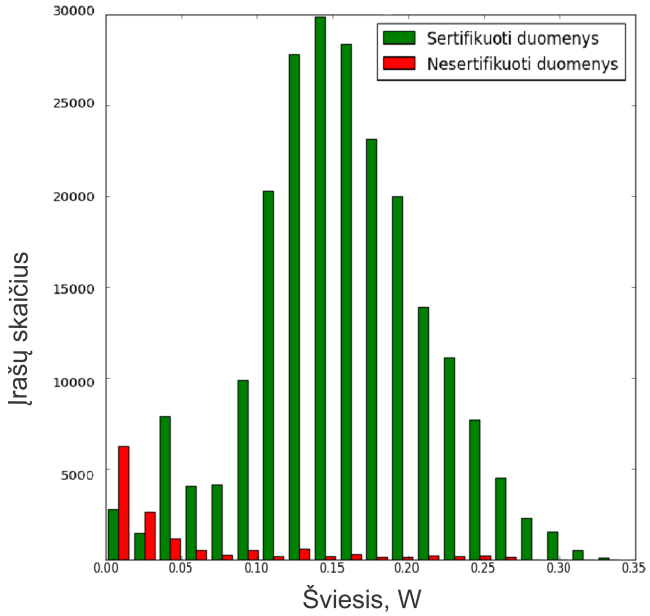
Programoje yra naudojami 2016 metų CMS detektoriaus duomenys iš JetHT pirminio duomenų rinkinio. Pirminis duomenų rinkinys - tai detektoriaus užregistruotų įvykių srautas, atrinktas naudojant HLT (angl. *High Level Trigger*) gaunamus rezultatus [6]. Tam, kad detektoriaus užregistuotas įvykis būtų įrašytas saugojimui, jis turi ne tik praeiti pirminį techninės įrangos patikrinimą, bet ir atitikti daug sudėtingų HLT sąlygų [5].

Iš .root tipo duomenų konvertuoti įrašai yra įkeliami SFrame formatu. Tam, kad klasifikatorius nebūtų apmokomas klaidingais duomenimis, yra pašalinami įrašai, turintys "NA" reikšmių. Kadangi duomenų įrašus sudaro ne tik parametrai, lemiantys duomenų kokybę, bet ir kiti dydžiai, tokie kaip įrašo ID, įvykių skaičius ir pan., yra nusakoma, kurie dydžiai yra laikomi modeliui kurti reikalingais parametrais. Šie parametrai buvo plačiau aprašyti 1.5 poskiryje. Taip pat yra nusakomas rodiklis, skiriantis sertifikuotus bei nesertifikuotus duomenis - "isSig".

Pradinis sertifikuotų ir nesertifikuotų duomenų skaičius labai skiriasi: visus turimus duomenis sudaro beveik 95% sertifikuotų duomenų ir 5% nesertifikuotų. Jei sistemos mokymosi metodui būtų duodami duomenys, kurių skaičius klasėse taip skiriasi, didelė tikimybė, kad modelis išmoktų gerai klasifikuoti didesnės klasės duomenis ir visus tolimesnius jam siunčiamus duomenus priskirtų didesniajai klasei. Siekiant to išvengti galima naudoti du metodus. Pirmasis - duomenų skaičių sulyginti atmetant atsitiktinę dalį tinkamais sertifikuotų duomenų. Tokiu atveju yra sukuriamas subalansuotas duomenų rinkinys, kuriame lieka dvi klasės su vienodu skaičiumi duomenų. Tačiau šio metodo pagrindinis minusas yra tas, kad tai lemia didelį duomenų skaičiaus sumažėjimą ir mažesnį skaičių įrašų, skirtų modelio apmokymui. Dėl to krinta klasifikatoriaus tikslumas. Antra galima priemonė klasių duomenų skaičiaus disbalanso problemai spręsti - GraphLab Create modelių papildomas parametras *class_weights = 'auto'*. Šio parametro reikšmė yra tokia, kad kuriant modelį klasėms yra priskiriami skirtingi svoriai, atsižvelgiant į klasių pasikartojimą apmokymui skirtuose duomenyse. Šio magistrinio baigiamojo darbo metu buvo išbandomi abu galimi problemos sprendimo būdai.

Vienas iš pagrindinių duomenis apibūdinančių parametrų yra šviesis. Nors jis nėra tiesiogiai naudojamas tolimesniuose sistemos mokymosi metoduose, jis lemia duomenis nusakančių histogramų parametrus. Duomenų pasiskirstymas pagal šviesį yra pavaizduotas 13 pav. Atskirai pavaizdavus sertifikuotų bei nesertifikuotų duomenų pasisikirstymus matome, kad didžioji dalis nesertifikuotų duomenų yra mažo šviesio zonoje. Tuo tarpu, sertifikuotų duomenų didžioji dalis priklauso segmantams, kurių šviesis yra didesnis. Tai leidžia tikėtis, kad mažesnio šviesio segmentai yra su anomalijomis ar rodo tolimesnei analizei netinkamus įrašus. Tačiau, dėl pasiskirstymų persiklojimo, negalima teigti, kad šviesis vienareikšmiškai nusako duomenų kokybę. Vis dėlto, šios priklausomybės grafinis pavaizdavimas leidžia gariau suprasti, su kokiais duomenimis yra dirbama.

Po duomenų paruošimo likę įrašai yra atsitiktinai padalinami į dvi dalis: 80% duomenų yra skirta modelio apmokymui, o likę 20% - jau sukurto modelio testavimui. Skirtingai nei *Yandex*



13 pav. Sertifikuotų ir nesertifikuotų duomenų pasiskirstymas pagal šviesį.

grupės siūlyme, šiuo atveju nebuvo naudojamos užterštumo ar praradimo santikių ribinės vertės. Visi duomenys buvo klasifikuojami ir tuomet tikrinamas modelio tikslumas dar modelio nematytiems - testavimo - duomenims, lyginant klasifikavimo rezultatus su duomenyse nurodyta duomenų kokybės ekspertų sertifikavimo verte.

2.2. Duomenų klasifikavimas naudojant sistemos mokymosi metodus

Mokslo tiriamojo darbo metu buvo išbandyti trys skirtingų rūšių duomenų klasifikatoriai paremti sistemos mokymosi metodais. Naudoti klasifikatoriai: logistinės regresijos, atraminių vektorių mašinų, išpūstų sprendimų medžių. Visi šie klasifikatoriai yra plačiau aprašyti šio darbo 1.6 skyriuje. Kiekvieno iš klasifikatorių atveju, buvo nustatytas maksimalus galimų mokymosi iteracijų skaičius. Kai vienos iteracijos metu vykstantis pokytis yra mažesnis, nei tam tikra sistemos numatyta riba, yra laikoma, kad buvo sukurtas optimalus modelis. Jei praėjus maksimaliam mokymosi iteracijų skaičiui optimalus modelis yra nerandamas, sistemos mokymasis yra stabdomas, tačiau sukurtu modeliu vis tiek galima naudotis. Rezultatai, gauti naudojantis minėtais klasifikatoriais yra pateikti tolimesniuose poskyriuose.

Sukurti modeliai buvo lyginami pasitelkiant kelis galimus tikslumo įvertinimo būdus. Kiekvieno modelio atveju buvo skaičiuojamas jo tikslumas (teisingai klasifikuotų duomenų santykis su visais duomenimis) su testavimui skirtais duomenimis. Kadangi yra siekiama sukurti modelį, kuris vienodai gerai klasifikuotų tiek sertifikuotus, tiek nesertifikuotus duomenis, buvo skaičiuojami klasifikavimo tikslumai ir šioms dviems klasėms atskirai. Dėl duomenų skaičiaus skirtingose klasėse disbalanso, modeliai buvo kuriami ir naudojant parametrą *class_weights = 'auto'*, ir atmetant dalį sertifikuotų duomenų. Skirtingų klasių duomenų klasifikavimui stebėti buvo naudojama ir painiavos matrica. Joje yra išskiriama, kiek kiekvienos klasės duomenų buvo klasifikuoti teisingai ir kiek - neteisingai.

Dar vienas dydis, skirtas tikrinti sukurto modelio klasifikavimo kokybei, tai F1-rezultatas. Šis dydis - tai tikslumo matas, atsižvelgiantis į praradimo bei užterštumo santykius. F1-rezultatas

gali kisti nuo 0 iki 1, kur 0 reiškia blogiausią klasifikatoriaus parametrą, o 1 - geriausią įmanomą. Kadangi tiriamu atveju skirtingų klasių duomenų skaičius yra nesubalansuotas, šis parametras gana gerai tinka spręsti apie sukurto modelio tinkamumą.

2.2.1. Duomenų sertifikavimas naudojant logistinę regresiją

Visų pirma buvo išbandytas klasifikatorius paremtas logistine regresija. Plačiau apie šio klasifikatoriaus veikimą aprašyta 1.6.1 poskyryje.

Pirmiausia logistinis klasifikatorius buvo kuriamas naudojant pilnus klasių duomenis ir papildomą parametrą *class_weights = 'auto'*. Šiuo atveju, optimalus modelis buvo sukurtas jau po 7 skaičiavimo iteracijų. Modelio kūrimas truko 10.4 sekundes. Naudojant testavimui skirtus duomenis buvo įvertintas modelio tikslumas, kuris yra lygus 92.23%. Gauta painiavos matrica pavaizduota 1 lentelėje.

Tikroji sertifikavimo reikšmė	Modeliu nustatyta sertifikavimo reikšmė	Pasikartojimų skaičius
1	1	40917
1	0	3112
0	1	518
0	0	2195

1 lentelė. Logistiniu klasifikatoriumi sukurto modelio painiavos matrica.

Naudojantis painiavos matrica galime suskaičiuoti, kad sertifikuotų duomenų klasifikavimo procentas yra lygus 92.93%, o nesertifikuotų - 80.91%. Nors gana greitai yra sukuriamas optimalus modelis, šio klasifikatoriaus tikslumas yra ganėtinai mažas. Dėl itin mažo nesertifikuotų duomenų klasifikavimo tikslumo, modelis taip pat buvo kuriamas naudojant balansuotą duomenų rinkinį. Tokiu atveju optimalus sprendimas buvo rastas jau po 6 iteracijų (1.8 sekundės). Bendras klasifikatoriaus tikslumas testavimo duomenims - 87.55%, o painiavos matrica pavaizduota 2 lentelėje.

Tikroji sertifikavimo reikšmė	Modeliu nustatyta sertifikavimo reikšmė	Pasikartojimų skaičius
1	1	2606
1	0	175
0	1	506
0	0	2182

2 lentelė. Logistiniu klasifikatoriumi sukurto modelio painiavos matrica.

Naudojantis šia lentele galima suskaičiuoti, kad sertifikuotų duomenų klasifikavimo procentas yra lygus 93.71%, o nesertifikuotų - 81.18%. Taigi matome, kad net naudojant subalansuotą duomenų rinkinį, logistinis klasifikatorius gerokai sunkiau geba atskirti nesertifikuotus duomenis.

2.2.2. Duomenų klasifikavimas paremtas atraminių vektorių mašinomis

Antrasis išbandytas duomenų klasifikatorius - atraminių vektorių mašinų. Plačiau šis klasifikatorius yra aprašytas 1.6.2 poskyryje.

Nesubalansuoto duomenų rinkinio atveju, SVM optimalaus modelio rasti nepavyko net padidinus klasifikatoriaus kūrimo iteracijų skaičių iki 1500. Numatytasis GraphLab modelio kūrimo iteracijų skaičius - 10 iteracijų. Kadangi daug iteracijų trunkančio modelio kūrimas kainuoja daug resursų, ko yra siekiama išvengti, itaracijų skaičius programoje buvo sumažintas iki 100. Toks skaičius buvo pasirinktas dėl nedidelio tikslumo kitimo toliau didinant iteracijų skaičių.

Atraminių vektorių mašinų klasifikatoriaus sukūrimas užtruko 55 sekundes. Sukūrus modelį jis buvo tikrinamas naudojant duomenis skirtus testavimui. Įvertintas modelio tikslumas bei painiavos matrica. Aprašytu atveju gautas modelio tikslumas - 94.30 %. Gauta painiavos matrica pavaizduota 3 lentelėje.

Tikroji sertifikavimo reikšmė	Modeliu nustatyta sertifikavimo reikšmė	Pasikartojimų skaičius
1	1	41988
1	0	2041
0	1	623
0	0	2090

3 lentelė. SVM klasifikatoriumi sukurto modelio painiavos matrica.

Kaip matoma iš painiavos matricos, sukurtas modelis gerokai tiksliau veikia su sertifikuotais duomenimis (jų klasifikavimo tikslumas yra lygus 95.36%), nei su nesertifikuotais (klasifikavimo tikslumas - 77.03%).

Kuriant atraminių vektorių mašinų modelį su subalansuotu duomenų rinkiniu, buvo pasiektas 87.12% procentų bendras tikslumas testavimo duomenims. Klasifikavimo tikslumas sertifikuotų duomenų klasei - 95.72%, o nesertifikuotų duomenų klasei - 78.24%. Matome, kad, kaip ir logistinės regresijos atveju, naudojant subalansuotą duomenų rinkinį yra pasiekiamas šiek tiek (apie 1%) didesnis klasifikavimo tikslumas nesertifikuotų duomenų klasei.

2.2.3. Duomenų sertifikavimas naudojant išpūstus sprendimų medžius

Trečiasis išbandytas klasifikatorius - išpūstų sprendimų medžių. Plačiau apie šį klasifikatorių yra aprašyta 1.6.3 poskyryje.

Kaip ir atraminių vektorių mašinų atveju, kuriant šį algoritmą optimalaus modelio sukurti nepavyko net padidinus iteracijų skaičių. Siekiant išlaikyti optimalų modelio apsimokymo laiką, buvo nuspręsta pasirinkti ribinį iteracijų skaičių lygų 100. Kuriant modelį su pilnu duomenų rinkinių, jo kūrimas užtruko 747 sekundes. Aprašytu atveju, modelis pasiekė itin didelį 98.22% tikslumą su testavimo duomenimis. Šiuo atveju gauta painiavos matrica nurofyta 4 lentelėje.

Tikroji sertifikavimo reikšmė	Modeliu nustatyta sertifikavimo reikšmė	Pasikartojimų skaičius
1	1	43438
1	0	591
0	1	237
0	0	2476

4 lentelė. BDT klasifikatoriumi sukurto modelio painiavos matrica.

Kaip ir prieš aukščiau aprašytų klasifikatorių atvejais, didesnis tikslumas buvo pasiektas klasifikuojant sertifikuotus duomenis (98.65%). Nesertifikuotų duomenų klasifikavimo tikslumas - 91.26%.

Naudojant subalansuotą duomenų rinkinį, modelio kūrimas truko 87.9 sekundės ir buvo pasiektas 95.52% klasifikavimo tikslumas. Gauti tikslumai sertifikuotų ir nesertifikuotų duomenų rinkiniams - atitinkamai 97.61% ir 93.12%. Matome, kad šiuo atveju pasiektas didžiausias bendras klasifikavimo tikslumas, bei geriausias tikslumas nesertifikuotų duomenų rinkiniui.

Tačiau buvo pastebėta, kad modelio mokymosi metu yra pasiekiamas itin aukštas 99.9% tikslumas (mokymosi duomenų rinkiniui). Tai reiškia, kad šio modelio mokymosi metu yra itin gerai išmokstami mokymosi duomenis ir prisitaikoma net prie juose esančių triukšmų. Tai gali būti ženklas, kad atsiradus naujiems duomenims, šis modelis veiks vis prasčiau.

2.2.4. Modelių palyginimas

Mokslo tiriamojo darbo metu, siekiant nustatyti, ar kuris nors iš išbandytų sistemos mokymosi klasifikatorių yra tinkamas automatinio sertifikavimo uždaviniui spręsti, buvo lyginami modelių apmokymo laikai bei jų tikslumas. Kadangi visų klasifikatorių atveju buvo gautas mažesnis tikslumas nesertifikuotiems duomenims, be bendro modelio tikslumo, sukurtų modelių kokybei lyginti buvo naudojamas ir nesertifikuotų duomenų klasifikavimo tikslumas. Gauti rezultatai pateikiami 5 lentelėje.

Klasifikatorius	Iteracijų skaičius	Apmokymo trukmė (s)	Bendras tikslumas (%)	Tikslumas nesertifikuotiems duomenims (%)	F1-rezultatas
Logistinis	7	10.4	92.23	80.91	0.95
Logistinis (balansuota)	6	1.8	87.55	81.18	0.88
SVM	100	55	94.30	77.03	0.96
SVM (balansuota)	100	9	87.12	78.24	0.88
BDT	100	747	98.22	91.26	0.99
BDT (balansuota)	100	87.9	95.52	93.12	0.96

5 lentelė. Duomenų klasifikavimo modelių palyginimas.

Kaip matome iš šios lentelės, visų klasifikatorių atveju, naudojant subalansuotą duomenų rinkinį, buvo gautas mažesnis bendras klasifikavimo tikslumas, tačiau nesertifikuoti duomenys buvo klasifikuojami tiksliau. Kadangi yra svarbu galėti atmesti visus blogus duomenis, kaip nesertifikuojamus, į pastarąjį sertifikavimo tikslumą buvo kreipiama daugiau dėmesio. Todėl nuspręsta, kad, šios problemos atveju, tinkamesnis klasių duomenų skaičiaus disbalanso problemos sprendimo būdas yra atmesti didžiąją dalį sertifikuotų duomenų taip išlyginant klasių dydžius.

Lyginant klasifikavimo metodus, optimalų modelį pavyko sukurti tik logistinio klasifikatoriaus atveju. Tačiau šio modelio atveju buvo pasiekiamas ganėtinai mažas 87.55% tikslumas, bei 0.95 F1-rezultatas.

Atraminių vektorių mašinų atveju buvo pasiektas taip pat nedidelis bendras klasifikavimo tikslumas (87.12%) bei žemiausias klasifikavimo tikslumas nesertifikuotiems duomenims (78.24%). Kadangi vienas iš pagrindinių CERN duomenų kokybės stebėjimo sistemos tiklsų yra atmesti tolimesei analizei netinkamus duomenis, šio klasifikatoriaus naudojimas sistemoje būtų netinkamas.

Geriausi rezultatai buvo pasiekti išpūstų sprendimų medžių klasifikatoriumi. Pasiektas aukštas bendras sertifikavimo tikslumas (95.52%) bei F1-rezultatas (0.96) leidžia manyti, kad išpūstų sprendimo medžių klasifikatorius galėtų būti potencialiai pritaikomas CMS DQM sitemoje. Vis dėlto, vis dar išlieka pakankamai didelis skirtumas tarp sertifikuotų ir nesertifikuotų duomenų klasifikavimo tikslumo (atitinkamai 97.19% ir 93.12% procentai). Modelio kūrimo metu buvo pasiektas itin aukštas 99.9% tikslumas mokymosi duomenims, kas leidžia manyti, kad modelis gerokai prisitaiko prie visų mokymosi duomenyse esančių triukšmų. Be to, net ir padidinus maksimalų modelio kūrimo iteracijų skaičių, optimalus sprendimas pasiektas nebuvo. Bandant pritaikyti išpūstų sprendimų medžių klasifikatorių realioje duomenų kokybės stebėjimo sistemoje, dėl šių priežasčių galėtų kilti problemų su klasifikavimo stabilumu ir patikimumu. Kadangi BDT klasifikatorius lengvai prisitaiko ir prie triukšmų duomenyse, reikėtų atidžiai stebėti šio klasifikatoriaus veikimą su naujais duomenimis ir pritaikyti papildomus metodus, leidžiančius sumažinti šį prisitaikymą.

Kadangi nei vienu iš sistemos mokymosi metodų nepavyko sukurti modelio, kuris vienodai kokybiškai klasifikuotų tiek sertifikuotus, tiek nesertifikuotus duomenis ir būtų patikimas, baigiamojo magistro darbo metu buvo nuspręsta išbandyti ir dirbtinius neuroninius tinklus CERN CMS duomenų sertifikavimo problemai spręsti. Kol kas literatūroje nėra duomenų, apie tokį minėto uždavinio sprendimo būdą, o dirbtiniai neuroniniai tinklai tampa vis plačiau naudojami klasifikatoriams kurti.

2.3. Duomenų klasifikavimas naudojant neuroninius tinklus

Dirtinio neuroninio tinklo veikimas labai priklauso nuo įvairių jo mokymosi parametrų bei architektūros. Kaip ir aprašyta šio darbo 1.7 skyriuje, naudojantis įvairiais neuronų sluoksniais, galima sukurti nuo paprastų, iki itin sudėtingų neuroninių tinklų su skirtingomis aktyvacijos funkcijomis, bei kitais parametrais. Baigiamojo magistrinio darbo metu buvo tiriama kuriamo modelio klasifikavimo tikslumo priklausomybė nuo tinklo sluoksnių skaičiaus, tipų bei mokymosi parametrų.

Naudojant dirbtinius neuroninius tinklus, visi duomenų parametrai buvo papildomai paruošiami. Visu darbo su dirbtiniais neuroniniais tinklais metu buvo naudojamas subalansuotas duomenų rinkinys (vienodas sertifikuotų bei nesertifikuotų duomenų rinkinių dydis). Kaip ir ankstesniu atveju, visi duomenys buvo padalinami į dvi dalis: 80% - modelio apmokymui skirti duomenys ir 20% - testavimui skirti duomenys.

Prieš dirbant su dirbtiniais neuroniniais tinklais, duomenys buvo centruojami iš kiekvieno parametro atimant jo vidurkį pagal funkciją:

$$data\_balanced[f] = data\_balanced[f] - data\_balanced[f].mean(), \tag{2.1}$$

kur $data\_balanced$ - visi duomenys iš subalansuoto duomenų rinkinio, f - kiekvienas iš klasifikavimui reikalingų duomenų parametrų, $data\_balanced[f].mean()$ - duomenų parametro f vidurkis. Tuomet visi duomenys buvo normuojami. Kiekvieno iš parametrų f reikšmės kinta skirtinguose intervaluose, todėl prieš kuriant dirbtinį neuroninį tinklą, reikia suvienodinti šių reikšmių mąstelius. Tai buvo atliekama kiekvieno iš parametrų vertes dalinant iš standartinio nuokrypio pagal formulę:

$$data\_balanced[f] = data\_balanced[f]/data\_balanced[f].std(), \tag{2.2}$$

kur $data\_balanced[f].std()$ duomenų parametro f standartinis nuokrypis.

2.3.1. Sistemos numatytojo tinklo kūrimas

Klasifikavimui naudojamas dirbtinis neuroninis tinklas buvo kuriamas naudojantis GraphLab Create "deeplearning" moduliu. Šis modulis yra skirtas kurti skirtingų neuroninių tinklų architektūras bei jomis manipuliuoti. Klasifikatoriaus kūrimas buvo pradedamas nuo sistemos numatytojo dirbtinio neuronų tinklo kūrimo.

Visų pirma buvo sukurtas NeuralNet klasės objektas, kurio paskirtis yra nusakyti neuroninio tinklo architektūrą bei mokymosi parametrus, tokius kaip mokymosi greitis ir momentas. Objekto kūrimui buvo panaudoti modelio apmokymui skirti duomenys, nurodomi reikalingi parametrai bei kokybės žymė "$isSign$". Sukurtas sistemos numatytas tinklas, kurio architektūra sudaryta iš 4 sluoksnių su atitinkamais parametrais:

1. FullConnectionLayer sluoksnis
    - init_sigma = 0.01
    - init_random = gaussian
    - init_bias = 0
    - num_hidden_units = 10
2. SigmoidLayer sluosknis
3. FullConnectionLayer sluoksnis
    - init_sigma = 0.01
    - init_random = gaussian
    - init_bias = 0
    - num_hidden_units = 2
4. SoftmaxLayer sluoksnis

Numatytieji neuroninio tinklo mokymosi parametrai:

- Mokymosi greitis: 0.001
- Momentas: 0.9

Pasinaudojant šiuo neuroniniu tinklu, buvo kuriamas NeuralNetClassifier klasės objektas. Modelio kūrimui buvo naudojami mokymui skirti duomenys, nustatytas pradinis maksimalus iteracijų skaičius - 50 iteracijų. Tokio klasifikatoriaus sukūrimas užtruko 6.11 sekundžių. Įvertinus sukurto klasifikatoriaus tikslumą ir painiavos matricą buvo gauti tokie rezultatai: bendras tikslumas testavimo duomenims - 87.52%, tikslumas sertifikuotiems duomenims - 94.53%, tikslumas nesertifikuotiems duomenims - 80.28%, painiavos matrica pavaizduota 6 lentelėje.

Iš šių rezultatų matome, kad net ir sistemos numatytasis dirbtinis neuroninis tinklas pasiekia ganėtinai aukštus klasifikavimo tikslumo rezultatus. Vis dėlto, kaip ir ankstesnių klasifikatorių atveju, nesertifikuotų duomenų klasifikavimo tikslumas yra gerokai mažesnis, nei sertifikuotų. Norint pasiekti geresnį sukurto klasifikatoriaus tikslumą, aukščiau aprašytas tinklas buvo koreguojamas keičiant jo mokymosi parametrus ir architektūrą. Tolimesniuose poskyriuose yra aprašyti baigiamojo magistro darbo metu atlikti pakeitimai, siekiant sukurti tokį neuroninį tinklą, kuris gebėtų klasifikuoti sertifikuotus ir nesertifikuotus CERN CMS duomenis kuo tiksliau.

Tikroji sertifikavimo reikšmė	Modeliu nustatyta sertifikavimo reikšmė	Pasikartojimų skaičius
1	1	2629
1	0	152
0	1	520
0	0	2158

6 lentelė. Sistemos numatytojo dirbtinių neuroninių tinklų klasifikatoriaus sukurto modelio painiavos matrica.

2.3.2. Klasifikatoriaus tikslumo priklausomybė nuo mokymosi parametrų

Siekiant sukurti geriausiai turimą uždavinį atitinkantį dirbtinį neuronų tinklą, visų pirma buvo tikrinama klasifikatoriaus tikslumo priklausomybė nuo mokymosi parametrų: mokymosi greičio bei momento. Kadangi šie du parametrai yra iš dalies susiję vienas su kitu, tai ir klaisifikatoriaus tikslumo priklausomybė buvo tiriama nuo šių parametrų kartu. Buvo tiriamos mokymosi greičio vertės nuo 0.0001 iki 0.1 (7 vertės iš šio intervalo), bei momento vertės nuo 0.5 iki 1 (5 vertės iš šio intervalo). Esant kiekvienai mokymosi greičio vertei, tinklui buvo priskiriama kiekviena iš momento reikšmių ir kuriamas klasifikatorius su neuroniniu tinklu, turinčiu tokias mokymosi parametrų reikšmes. Tokiu būdu buvo patikrinta 35 galimos parametrų kombinacijos.

Be bendro klasifikatoriaus tikslumo papildomai buvo tikrinama ir teisingai klasifikuotų nesertifikuotų duomenų skaičius. Kadangi prieš tai tirtais atvejais daugiausia problemų kilo su nesertifikuotų duomenų atskyrimu, šis parametras darbo atveju yra taipogi svarbus. 7 lentelėje nurodytos mokymosi greičio bei momento reikšmių kombinacijos, kurioms esant buvo pasiektas didžiausias tikslumas. Taip pat nurodomas teisingai klasifikuotų nesertifikuotų duomenų skaičius.

Modelio tikslumas (%)	Mokymosi greitis	Momentas	Modelio tikslumas nesertifikuotiems duomenims (%)
94.40	0.05	0.9	92.29
94.17	0.01	0.9	92.07
93.97	0.1	0.7	92.07
93.86	0.05	0.5	91.11
93.73	0.05	0.7	91.23

7 lentelė. Klasifikatoriaus tikslumo priklausomybė nuo mokymosi parametrų, bei teisingai klasifikuotų nesertifikuotų duomenų skaičius.

Iš šios lentelės matome, kad didžiausias tikslumas (94.40%) yra pasiekiamas su mokymosi greičio reikšme 0.05 bei momento reikšme lygia 0.9. Ši mokymosi greičio reikšmė yra didesnė nei numatytoji. Didelė mokymosi greičio reikšmė reiškia, kad yra greičiau judama link klaidos funkcijos minimumo, tačiau su tokia verte gali būti sunku pasiekti pačią tiksliausią minimumo vertę. Kadangi tiriant tikslumo priklausomybę nuo mokymosi parametrų buvo reikalinga sukurti daug atskirų klasifikatorių ir lyginti jų tikslumus, tai reikalavo daug kompiuterio skaičiavimo resursų. Dėl šios priežasties buvo pasirinktas neuroniniams tinklams palyginti nedidelis maksimalus iteracijų skaičius lygus 100. Norint naudoti mažesnį mokymosi greitį, šį iteracijų skaičių reikėtų didinti, nes mokantis mažu žingsniu, tokios iteracijų skaičiaus vertės gali nepakakti pasiekti globaliam minimumui.

Vis dėlto, modelio tolimesniam kūrimui buvo pasirinktos vertės, su kuriomis pasiekiama didžiausia tikslumo vertė. Tačiau toliau darbe, pakeitus dirbtinio neuroninio tinklo architektūrą, mokymosi parametrų vertės bus dar kartą peržiūrėtos, siekiant pasiekti aukščiausią galimą klasifikavimo tikslumo vertę su jau modifikuotos architektūros dirbtiniu neuroniniu tinklu.

Taigi, toliau naudojamos mokymosi parametrų vertės:

- Mokymosi greitis - 0.05;
- Momentas - 0.9.

2.3.3. Klasifikatoriaus tikslumo priklausomybė nuo tinklo sluoksnių parametrų

Tiriant klasifikatoriaus tikslumą nuo dirbtinio neuroninio tinklo architektūros pradedama tai daryti su turimu sluoksnių skaičiumi, keičiant sluoksnių tipus ar parametrus. Visų pirma vyksta tikrinimas, kuri iš aktyvacijos funkcijų geriausiai tinka CERN CMS duomenims klasifikuoti. Tam yra naudojamos trys skirtingos aktyvacijos funkcijos: sigmoidinė, ištaisyta tiesinė bei hiperbolinio tangento. Kiekvienu atveju yra kuriamas klasifikatorius, tikrinamas jo tikslumas visiems testavimo duomenims ir atskirai skaičiuojamas tikslumas nesertifikuotiems duomenims. Gauti rezultatai aprašyti 8 lentelėje.

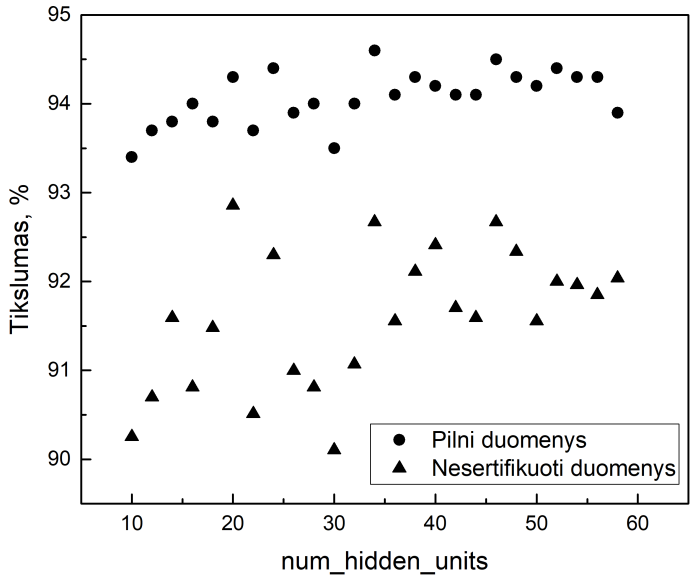
Aktyvacijos funkcijos tipas	Modelio tikslumas (%)	Modelio tikslumas nesertifikuotiems duomenims (%)
Sigmoidinė funkcija	92.97	90.98
Ištaisyta tiesinė funkcija	93.87	91.59
Hiperbolinio tangento funkcija	94.51	92.04

8 lentelė. Klasifikatoriaus tikslumo (visiems testavimo duomenims bei atskirai nesertifikuotiems duomenims) priklausomybė nuo aktyvacijos funkcijos.

Iš šios lentelės matome, kad visų funkcijų atveju yra pasiekiamas ganėtinai panašus bendras klasifikavimo tikslumas. Vis dėlto, hiperbolinio tangento funkcija pasiekia didžiausią tikslumą tiek visam duomenų rinkiniui, tiek nesertifikuotiems duomenims. Dėl šios priežasties toliau bus naudojama ši aktyvacijos funkcija.

Sistemos numatytoje dirbtinio neuroninio tinklo architektūroje be aktyvacijos sluoksnio taip pat yra ir du pilnai sujungtų paslėptų neuronų sluoksniai. Antrasis iš jų eina prieš pat išėjimo SoftmaxLayer sluoksnį, todėl jo paslėptų neuronų skaičių nurodantis parametras "num_hidden_units" turi būti lygus galimų klasių skaičiui, taigi, šio uždavinio atveju - 2. Tuo tarpu pirmojo pilnai sujungto sluoksnio paslėptų neuronų skaičių galime keisti. Todėl buvo nuspręsta ištirti ir klasifikatoriaus tikslumo priklausomybę nuo šio skaičiaus. Šiai priklausomybei tirti buvo parinktas "num_hidden_units" reikšmių intervalas nuo 10 iki 60 su žingsniu lygiu 2.

Gauta klasifikatoriaus tikslumo priklausomybė nuo pilnai sujungto sluoksnio paslėptų neuronų skaičiaus (pilnam duomenų rinkiniui, bei nesertifikuotų duomenų rinkiniui) yra pavaizduota 14 pav. Iš šio grafiko matome, kad keičiant vieno pilnai sujungto sluoksnio paslėptų neuronų skaičių, sukurto klasifikatoriaus tikslumas gali keistis net daugiau nei per vieną procentą. Tačiau aiškios pastovios priklausomybės, kaip didėja tikslumas, didinant paslėptų neuronų skaičių sluoksnyje, iš grafiko nematome. Kadangi matome išskirtinai aukštą tikslumą nesertifikuotų duomenų rinkiniui, kuomet pilnai sujungtą sluoksnį sudaro 20 dirbtinių neuronų, pasirenkame šį skaičių.



14 pav. Klasifikatoriaus tikslumo priklausomybė nuo pilnai sujungto sluoksnio paslėptų neuronų skaičiaus.

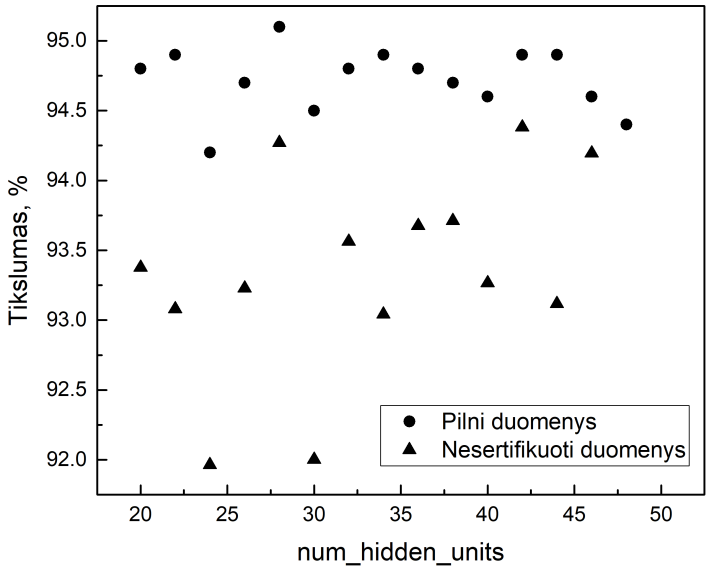
Šiuo atveju buvo pasiektas 94.24% tikslumas pilnam testavimo duomenų rinkiniui ir 92.86% tikslumas nesertifikuotų duomenų rinkiniui.

2.3.4. Klasifikatoriaus tikslumo priklausomybė nuo tinklo architektūros

Siekiant pasiekti aukštesnį klasifikavimo tikslumą, į turimą dirbtinį neuroninį tinklą buvo nuspręsta įtraukti daugiau sluoksnių. Visų pirma į neuroninio tinklo pradžią buvo pridėti du papildomi sluoksniai: vienas pilnai sujungtų neuronų sluoksnis ir papildoma hiperbolinio tangento aktyvacijos funkcija. Kadangi iš 14 pav. matome, kad nuo parametro num_hidden_layers reikšmės tikslumas gali keistis gana nemažai, pridėje papildomą pilnai sujungtą sluoksnį vėl tikriname klasifikatoriaus priklausomybė nuo paslėptų neuronų skaičiaus. Tikrindami šią priklausomybę neuronų skaičių keičiame tik naujame sluoksnyje, laikome, kad anksčiau sukurtų sluoksnių dirbtinių neuronų skaičius jau yra parinktas tinkamai. Gauta priklausomybė pavaizduota 15 pav. Šiame grafike matome, kad paslėptų neuronų skaičiaus vertės, su kuriomis yra pasiekiamas geriausias tikslumas, skiriasi nuo 14 pav. matomų verčių. Tai tik įrodo, kad kiekvieno sluoksnio atveju, paslėptų neuronų skaičius turi būti pritaikomas atskirai, o ne nustatomas toks pat.

Kaip matome iš 15 pav., didžiausias tikslumas buvo pasiektas sluoksnyje esant 28 paslėptiems neuronams. Todėl šis skaičius buvo užfiksuotas ir toliau nekeičiamas. Tokiu atveju sukurtas klasifikatorius pasiekė 95.1% tikslumą pilnam testavimo duomenų rinkiniui ir 94.27% tikslumą nesertifikuotų duomenų rinkiniui. Kaip matome, po šio sluoksnio pridėjimo, klasifikatorius jau pasiekė didesnį klasifikavimo tikslumą nesertifikuotų duomenų rinkiniui, nei išpūstų sprendimų medžių klasifikatorius. Kadangi papildomas sluoksnis turėjo gerą įtaką klasifikavimo tikslumui, buvo nuspręstą pridėti dar daugiau papildomų sluoksnių su skirtingomis paslėptų neuronų skaičiaus vertėmis.

Pridėjus dar vieną papildomą pilnai sujungtų neuronų sluoksnį bei hiperbolinio tangento aktyvacijos funkciją, vėl buvo tikrinama tikslumo priklausomybė nuo naujojo sluoksnio paslėptų neu-



15 pav. Klasifikatoriaus tikslumo priklausomybė nuo pilnai sujungto sluoksnio paslėptų neuronų skaičiaus.

ronų skaičiaus. Šiuo atveju geriausius tikslumo rezultatus rodė sluoksnis, kurio paslėptų neuronų skaičius lygus 40. Šiuo atveju buvo gautos tokios tikslumo vertės:

- Pilnam duomenų rinkiniui - 94.55%
- Sertifikuotų duomenų rinkiniui - 94.79%
- Nesertifikuotų duomenų rinkiniui - 94.31%

Kaip matome iš šių duomenų, tinklo architektūra leido pasiekti beveik vienodą abiejų klasių sertifikavimo tikslumą. Tačiau bendras sertifikavimo tikslumas šiek tiek mažesnis, nei esant prieš tai buvusiai tinklo architektūrai. Vis dėlto buvo nuspręsta pabandyti pridėti dar vieną pilnai sujungtą sluoksnį bei hiperbolinio tangento aktyvacijos funkciją. Vėlgi buvo tikrinama kuriamo klasifikatoriaus tikslumo priklausomybė nuo sluoksnyje esančių dirbtinių neuronų skaičiaus.

Tačiau šiuo atveju buvo gauta, kad, nepriklausomai nuo dirbtinių neuronų skaičiaus, klasifikatorius sugebėjo klasifikuoti tik sertifikuotus duomenis. Painiavos matrica, gauta pridėjus paskutinį papildomą pilnai sujungtą sluoksnį, yra pavaizduota 9 lentelėje.

Tikroji sertifikavimo reikšmė	Modeliu nustatyta sertifikavimo reikšmė	Pasikartojimų skaičius
1	1	2781
0	1	2688

9 lentelė. Sudėtingos architektūros dirbtinio neuroninio tinklo klasifikatoriaus painiavos matrica.

Atsižvelgus į visus gautus rezultatus, buvo nuspręsta tikti prie tokios architektūros dirbtinio neuroninio tinklo:

1. FullConnectionLayer sluoksnis

- init_sigma = 0.01
- init_random = gaussian
- init_bias = 0
- num_hidden_units = 28

2. TanhLayer sluosknis
3. FullConnectionLayer sluoksnis
    - init_sigma = 0.01
    - init_random = gaussian
    - init_bias = 0
    - num_hidden_units = 20
4. TanhLayer sluosknis
5. FullConnectionLayer sluoksnis
    - init_sigma = 0.01
    - init_random = gaussian
    - init_bias = 0
    - num_hidden_units = 2
6. SoftmaxLayer sluoksnis

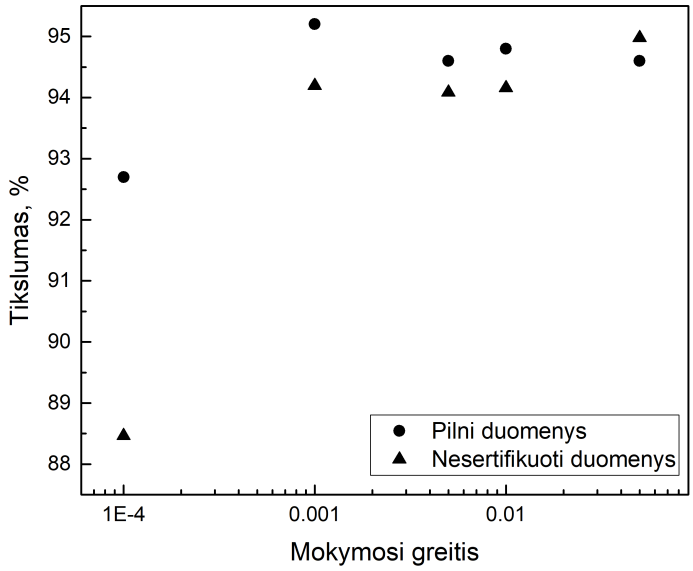
2.3.5. Klasifikatoriaus mokymosi greičio parinkimas

Atradus geriausią tinkamą neuroninio tinklo architektūrą, buvo dar kartą siekiama parinkti geriausią tinkamą mokymosi greitį. Ši kartą buvo tiriama klasifikatoriaus tikslumo priklausomybė tik nuo mokymosi greičio vertė. Buvo parinktos 5 potencialiai tinkamos mokymosi greičio vertės: 0.0001, 0.001, 0.005, 0.01, 0.05. Kadangi, esant mažoms mokymosi greičio vertėms, klasifikatoriui reikia daugiau laiko (iteracijų), kad būtų pasiektas globalus minimumas klaidos funkcijoje, šio priklausomybės tyrimo metu buvo nustatytas gerokai didesnis maksimalus mokymosi iteracijų skaičius, lygus 2000. Gauta tikslumo (pilnam duomenų rinkiniui, bei nesertifikuotų duomenų rinkiniui) priklausomybė nuo mokymosi greičio yra pavaizduota 16 pav.

Iš gautos priklausomybės matome, kad geriausi klasifikavimo rezultatai yra pasiekiami su mokymosi greičio verte lygia 0.001. Ši vertė yra 5 kartus mažesnė, nei prieš tai naudotoji. Papildomai, siekiant, kad artėjant prie klaidos funkcijos minimumo, žingsnis būtų mažesnis, buvo naudojamasi ir dirbtinių neuroninių tinklų mokymosi parametru "learning_rate_schedule". Išbandžius eksponentinį mokymosi greičio mažėjimo grafiką buvo nustatyta, kad mokymosi greičio vertė per daug greitai sumažėja iki minimalios vertės. Tokiu atveju klaidos funkcijos minimumas nėra pasiekiamas net po daugelio iteracijų ir klasifikatoriaus tikslumas lieka palyginti nedidelis.

Pasirinkus polinominį klaidos funkcijos mažėjimo grafiką buvo gauti geresni rezultatai. Papildomo parametro "learning_rate_alpha" vertė buvo padidinta nuo 0.5 (sistemos numatytoji vertė) iki 0.9. Tokiu atveju buvo gautos tokios klasifikatoriaus tikslumo vertės:

- Mokymosi duomenims - 96.35%



16 pav. Klasifikatoriaus tikslumo priklausomybė nuo pilnai klasifikatoriaus mokymosi greičio.

- Pilnam testavimo duomenų rinkiniui - 94.58%
- Sertifikuotų duomenų rinkiniui - 95.29%
- Nesertifikuotų duomenų rinkiniui - 93.86%

Matome, kad šiuo atveju bendras klasifikavimo tikslumas pilnam testavimo duomenų rinkiniui buvo pasiektas šiek tiek mažesnis, nei prieš tai tirtu atveju. Vis dėlto, klasifikavimo tikslumas nesertifikuotų duomenų atveju buvo gautas didesnis. Tačiau, šiuo atveju taip pat reikėtų atkreipti dėmesį į padidėjusį klasifikatoriaus tikslumą mokymosi duomenims. Visais prieš tai tirtų klasifikatorių atvejais, klasifikavimo tikslumas mokymosi duomenims bei pilnam testavimo duomenų rinkiniui buvo apytiksliai lygus. Tačiau šiuo atveju stebime beveik dviejų procentų tikslumo skirtumą. Tokią situaciją stebime po dirbtinio neuroninio tinklo architektūros pakeitimo, pridedant daugiau neuronų sluoksnių, bei po papildomų mokymosi parametrų pridėjimo. Dėl šios priežasties galima įtarti, kad modifikuojamas dirbtinių neuronų tinklas tapo pakankamai sudėtingas, kad pradėtų prisitaikyti prie testavimo duomenyse esančių triukšmų, kurie nėra naudingi siekiant nusakyti duomenų klasę. Plačiau apie klasifikatoriaus prisitaikymą prie mokymuisi skirtuose duomenyse esančių triukšmų aprašyta 1.7.5 skyriuje.

2.3.6. Klasifikatoriaus apsaugojimas nuo prisitaikymo prie triukšmų

Pastebėjus, kad, esant sudėtingesnei klasifikatoriaus naudojamo dirbtinio neuroninio tinklo architektūrai bei papildomiems mokymosi parametrams, klasifikatorius pradėjo prisitaikyti prie mokymosi parametruose esančių triukšmų, buvo nuspręsta išbandyti kovojimui su šia problema rekomenduojamus metodus. Nors baigiamajame magistro darbe pastebėtas klasifikatoriaus tikslumo mokymosi bei testavimo duomenims skirtumas yra palyginti nedidelis, tai gali būti ženklas, rodantis apie tolimesnes galimas problemas. Todėl baigiamojo magistrinio darbo metu buvo išbandyti tokie metodai: L2 reguliavimo parametro padidinimas bei papildomo atmetimo sluoksnio įtraukimas į dirbtinio neuroninio tinklo architektūrą.

Visų pirma buvo išbandytas L2 reguliavimo padidinimas. Kalbant apie dirbtinius neuroninius tinklus, L1 ir L2 reguliavimas - tai technikos, kuomet, skaičiuojant klaidos funkcijos vertę yra pridedamas papildomas narys. Šis narys skiriasi priklausomai nuo to, ar yra naudojamas L1 ar L2 reguliavimas. L1 reguliavimo atveju pridedamo parametro dydis:

$$\lambda \sum_{i=1}^{k} |w_i|, \tag{2.3}$$

kur λ - reguliavimo parametro dydis, w_i - dirbtinių neuronų svorių reikšmės. L2 reguliavimo atveju pridedamas narys:

$$\lambda \sum_{i=1}^{k} w_i^2. \tag{2.4}$$

Kuriamame dirbtiniame neuroniniame tinkle yra naudojamas L2 reguliavimas. Sistemos numatytoji reguliavimo parametro λ reikšmė yra lygi 0.0005. Siekiant sumažinti dirbtinio neuroninio tinklo prisitaikymą prie triukšmų, L2 reguliavimas buvo didinamas. Tai buvo daroma padidinus parametrą λ tris kartus - iki 0.0015. Sukūrus klasifikatorių su padidintu L2 reguliavimu, buvo gautos tokios tikslumo vertės:

- Mokymosi duomenims - 95.22%
- Pilnam testavimo duomenų rinkiniui - 94.49%
- Sertifikuotų duomenų rinkiniui - 96.51%
- Nesertifikuotų duomenų rinkiniui - 92.41%

Kaip matome iš šių duomenų, pilno testavimo duomenų rinkinio tikslumas tapo artimesnis mokymosi duomenų klasifikavimo tikslumui, bet skirtumas vis dar išliko. Beto, padidinto L2 reguliavimo atveju, gana smarkiai krito klasifikavimo tikslumas nesertifikuotiems duomenims. Dėl šių priežasčių buvo nuspręsta grįžti prie sistemos numatytosios reguliavimo parametro vertės ir išbandyti kitus būdus kovoti su klasifikatoriaus prisitaikymu prie mokymosi duomenų.

Antrasis išbandytas metodas - tai atmetimo sluoksnio pridėjimas. Kadangi tiriamu atveju prisitaikymas prie triukšmų nėra didelis, buvo nustatyta nedidelė šio sluoksnio parametro *threshold* vertė lygi 0.1. Tai reiškia, kad duomenims praeinant pro sluoksnį, kiekvienas iš duomenų atsitiktiniu būdu gali būti nustatytas lygus nuliui su tikimybe lygia 0.1. Sukūrus klasifikatorių, naudojantį tokios architektūros dirbtinį neuroninį tinklą, buvo gautos tokios tikslumo vertės:

- Mokymosi duomenims - 95.28%
- Pilnam testavimo duomenų rinkiniui - 95.37%
- Sertifikuotų duomenų rinkiniui - 96.44%
- Nesertifikuotų duomenų rinkiniui - 94.27%

Matome, kad šiuo atveju buvo efektyviai sumažintas skirtumas tarp klasifikavimo tikslumo mokymosi ir testavimo duomenims. Be to, buvo išlaikytas pakankamai aukštas klasifikavimo tikslumas ir nesertifikuotų duomenų rinkiniui, kas tiriamu atveju yra svarbu.

2.3.7. Dirbtinių neuroninių tinklų klasifikatoriaus apibendrinimas

Klasifikatoriaus, naudojančio dirbtinį neuroninį tinklą kūrimas buvo pradėtas nuo sistemos numatytojo tinklo kūrimo. Buvo gautos tokios tikslumo vertės: tikslumas pilnam testavimo duomenų rinkiniui - 87.52%, tikslumas sertifikuotiems duomenims - 94.53%, tikslumas nesertifikuotiems duomenims - 80.28%. Šios vertės yra labai panašios į gautas naudojant logistinės regresijos bei atraminių vektorių mašinų klasifikatorius.

Baigiamojo magistro darbo metu, minėtasis dirbtinis neuronų tinklas buvo tobulinamas keičiant jo mokymosi parametrus bei architektūrą. Susidūrus su klasifikatoriaus prisitaikymu prie mokymosi duomenyse esančių triukšmų, buvo išbandyti du metodai, skirti šiai problemai sumažinti. Neuroninio tinklo architektūra, su kuria buvo sukurtas geriausius rezultatus rodantis klasifikatorius:

1. FullConnectionLayer sluoksnis
    - init_sigma = 0.01
    - init_random = gaussian
    - init_bias = 0
    - num_hidden_units = 28
2. TanhLayer sluosknis
3. DropoutLayer sluoksnis
    - threshold = 0.1
4. FullConnectionLayer sluoksnis
    - init_sigma = 0.01
    - init_random = gaussian
    - init_bias = 0
    - num_hidden_units = 20
5. TanhLayer sluosknis
6. FullConnectionLayer sluoksnis
    - init_sigma = 0.01
    - init_random = gaussian
    - init_bias = 0
    - num_hidden_units = 2
7. SoftmaxLayer sluoksnis

Naudojami mokymosi parametrai:

- Mokymosi greitis - 0.001
- Momentas - 0.9

Remiantis šia tinklo architektūra bei mokymosi parametrais, buvo sukurtas klasifikatorius, kuris po 5000 mokymosi iteracijų pasiekė tokias tikslumo vertes: tikslumas pilnam testavimo duomenų rinkiniui - 95.18%, tikslumas sertifikuotiems duomenims - 94.63%, tikslumas nesertifikuotiems duomenims - 95.74%.

Gauti rezultatai leidžia teigti, kad dirbtinių neuroninių tinklų naudojimas gali būti rekomenduotas kaip metodas CERN CMS automatinio sertifikavimo uždaviniui spręsti.

Išvados ir rekomendacijos

- Mokslo tiriamojo darbo metu, tiriant klasifikatorius paremtus trimis sistemos mokymosi metodais, buvo nustatyta, kad geriausi rezultatai yra pasiekiami išpūstų sprendimų medžių klasifikatoriumi. Pasiektas ganėtinai aukštas 95.52% tikslumas leidžia teigti, kad modelis potencialiai galėtų būti pritaikomas CERN CMS duomenų automatinio sertifikavimo uždaviniui spręsti. Vis dėl to, buvo pastebėta, kad sukurtas klasifikatorius labai prisitaiko prie mokymosi duomenyse esančių triukšmų, todėl, norint naudoti šį klasifikatorių, turėtų būti atlikti išsamūs tyrimai ir patobulinimai, siekiant šios problemos išvengti.
- Naudojant dirbtinių neuroninių tinklų klasifikatorių buvo pasiektas 95.18% tikslumas pilnam duomenų rinkiniui ir 95.74% tikslumas nesertifikuotų duomenų rinkiniui. Susidūrus su klasifikatoriaus prisitaikymo prie mokymosi duomenyse esančių triukšmų problema, ji buvo išspręsta. Dėl šių priežasčių CERN CMS automatinio sertifikavimo uždaviniui spręsti yra rekomenduojama naudoti dirbtinius neuroninius tinklus.

Ateities darbų gairės

Baigiamojo magistrinio darbo metu buvo nustatyta, kad CERN CMS automatinio sertifikavimo uždaviniui spręsti yra tinkamiausi klasifikatoriai naudojantys išpūstus sprendimų medžius bei dirbtinius neuroninius tinklus. Tačiau, kiekvienu iš šių atveju, modeliai turi būti tobulinami siekiant juos pritaikyti realiam naudojimui CMS DQM sistemoje. Rekomenduojamos tokios tolimesnės darbų gairės:

- Išpūstų sprendimų medžių atveju svarbiausia užduotis būtų išspręsti klasifikatoriaus prisitaikymo prie mokymosi duomenyse esančių triukšmų problemą. Tai galima bandyti padaryti apribojant modelio mokymąsi mažinant leistinų iteracijų skaičių ar pridedant papildomus parametrus, tokius kaip kuriamų medžių gylio maksimali vertė ar minimalus vienam regionui tenkančių duomenų skaičius. Taip pat vertėtų pagerinti nesertifikuotų duomenų rinkinio klasifikavimo tikslumą.

- Dirbtinių neuroninių tinklų atveju būtų galima išbandyti daugiau tinklo architektūros variantų, įterpti papildomų sluoksnių, siekiant padidinti klasifikavimo tikslumą.

- Dar viena rekomendacija ateities darbams - gauti daugiau kuo naujesnių CERN CMS detektoriaus duomenų ir patikrinti, kaip sukurti klasifikatoriai veikia su jais. Tam, kad modelis galėtų būti pritaikomas realioje sistemoje, klasifikatorius turėtų atskirti naujų duomenų klases panašiu tikslumu, kaip ir iki šiol naudotų 2016 metų duomenis.

Literatūros šaltiniai

[1] M. Borisyak. Towards automation of data quality system for cern cms experiment. *J. Phys.: Conf. Ser.*, 2017.

[2] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144--152, New York, NY, USA, 1992. ACM.

[3] The CMS Collaboration. The cms experiment at the cern lhc. *JINST*, 2008.

[4] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273--297, 1995.

[5] G. Franzoni. Dataset definition for cms operations and physics analyses. *Nuclear and Particle Physics Proceedings 273–275*, 2016.

[6] V. Gori. The cms high level trigger. *PIC2013 conference*, 2014.

[7] N. Gupta. Artificial neural network. *Network and Complex Systems*, 2013.

[8] D. Hosmer and S. Lemeshow. *Applied Logistic Regression*. John Wiley & Sons, 2000.

[9] M. Moreira and E. Fiesler. Neural networks with adaptive learning rate and momentum terms. *IDIAP*, 1995.

[10] H. Otalora, N. Akchurin, and Dr. F. de Guio. Data quality monitoring with machine learning at cms. In *Data Quality Monitoring at CMS with Machine Learning*, 2016.

[11] C. Patrignani and Particle Data Group. Review of particle physics. *Chinese Phys. C*, 2016.

[12] B. Roe, H. Yang, J. Zhu, Y. Liu, I. Stancu, and G. McGregor. Boosted decision trees as an alternative to artificial neural networks for particle identification. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 543(2):577 -- 584, 2005.

[13] M. Sahin, D. Krücker, and I.-A. Melzer-Pellmann. Performance and optimization of support vector machines in high-energy physics classification problems. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 838:137 -- 146, 2016.

[14] R. Schapire. *The Boosting Approach to Machine Learning: An Overview*, pages 149--171. Springer New York, New York, NY, 2003.

[15] S. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 1997.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.

[17] L. Tuura, A. Meyer, I. Segoni, and G. Della Ricca. CMS data quality monitoring: Systems and experiences. *J. Phys. Conf. Ser.*, 219:072020, 2010.

[18] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[19] S. Wang. *Interdisciplinary Computing in Java Programming*. Springer US, 2003.