VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Duomenų dimensiškumo mažinimas ir klasifikavimas

Dimensionality reduction and classification

Bakalauro baigiamasis darbas

Atliko:	Donatas Kučinskas	(parašas)
Darbo vadovas:	Vytautas Valaitis	(parašas)
Recenzentas:	Julija Vysockytė	(parašas)

Vilnius – 2015

Santrauka

Glaustai aprašomas darbo turinys: pristatoma nagrinėta problema ir padarytos išvados. Santraukos apimtis ne didesnė nei 0,5 puslapio. Santraukų gale nurodomi darbo raktiniai žodžiai.

Raktiniai žodžiai: raktinis žodis 1, raktinis žodis 2, raktinis žodis 3, raktinis žodis 4, raktinis žodis 5

Summary

Santrauka anglų kalba. Santraukos apimtis ne didesnė nei 0,5 puslapio.

Keywords: keyword 1, keyword 2, keyword 3, keyword 4, keyword 5

TURINYS

Įvadas

Klasifikavimas – tai dažnai sutinkama užduotis, turintį įvairių sprendimo būdų. Šios uždavinio tikslas – identifikuoti, kuriai grupei priklauso tiriamas objektas. Tiriamieji objektai dažniausiai būna vienos rūšies, aprašomi tam tikrais parametrais, o grupės, kuriems jie yra priskiriami – iš anksto žinomos. Pavyzdžiui, galima klasifikuoti gyvūnus pagal tam tikras jų fizines savybes – kojų ilgį, storį, kitas kūno apimtis, kailio ilgį ir pan. Natūralu, kad kiekvienas net ir tos pačios rūšies gyvūnas turės šiek tiek kitokius parametrus, tačiau šie parametrai dažniausiai turi įvairius dėsningumus, pagal kuriuos galima bandyti atspėti, kuriai rūšiai tam tikras gyvūnas priklauso.

Norint išspręsti konkretų klasifikavimo uždavinį, akivaizdžiausias sprendimas galėtų būti šių grupių parametrų ištyrimas – pavyzdžiui, norint mokėti atskirti triušius nuo liūtų turint jų ūgius nėra sunki užduotis. Tačiau problema kyla, kai atskiriamos klasės yra labai panašios viena į kitą – tokiu atveju pastebėti tam tikrus dėsningumus ir juos sumodeliuoti bei realizuoti ir kur kas sunkiau. Be to, sprendžiant konkretų klasifikavimo uždavinį, tektų gilintis į klasifikuojamus objektus – pavyzdžiui, norint sukurti tam tikrų kiškių rūšių klasifikavimą, gilios žinios apie šias kiškių rūšių savybes būtų privalomos. Dažniausiai įvairūs dėsningumai apima ne vieną dydį, bet jų kombinaciją, kurią atrasti ir apskaičiuoti reikalautų daug pastangų. O norint efektyviai atskirti tam tikras objektų klases, gali prireikti daugybės skirtingų dėsningumų. Šios priežastys labai apsunkina efektyvaus klasifikavimo algoritmo kūrimą analizuojant klases, todėl yra retai naudojamas.

Vienas populiariausių klasifikavimo sprendimo metodų – klasifikavimas naudojant neuroninį tinklą. Turint pakankamai didelį tiriamų objektų duomenų kiekį, galima įžvelgti tam tikrus dėsningumus. Neuroninis tinklas leidžia tai automatizuoti – vietoje to, kad žmogus mokytųsi apie objekto savybes, tai atliekama su neuroniniu tinklu. Turimi duomenys panaudojami apmokyti neuroninį tinklą, kuris po to geba pats pasakyti, kuriai grupei tiriamas objektas priklauso. Žinoma, neuroninis tinklas nėra visada teisus, kadangi jis remiasi patirtimi, kurią įgijo pavyzdinių duomenų apmokymo metu, tačiau jei šie apmokymui buvo surinkta pakankamai korektiškų duomenų, neuroninis tinklas geba klasifikuoti objektus pakankamai tiksliai.

Dimensiškumo mažinimas (angl. *dimensionality reduction*) Savybių ištraukimas (angl. *feature extraction*)

TODO: TODO: pridėti paaiškinimą, kad dažniausiai turime daug duomenų?

TODO: neuroniniai tinklai? TODO: dimensiškumo mažinimas TODO: tyrimo tikslai?

TODO: Šaltiniai:

[HC94] [LG00] [LG00] [YL02]

1. Dirbtinių neuronų tinklas

Dirbtinis neuronų tinklas - tai tarpusavyje susijungusių dirbtinių neuronų tinklas, kurio užduotis yra spręsti tam tikrą užduotį arba užduotis. Dirbtinis neuronų tinklas gavęs pradinius užduoties duomenis, juos apdoroja ir taip gaunamas tam tikras atsakymas. Šis atsakymas nebūtinai yra teisingas - neuronų tinklai suprojektuoti taip, kad galėtų būti mokomi kai gauna neteisingą atsakymą.

1.1. Dirbtinis neuronas

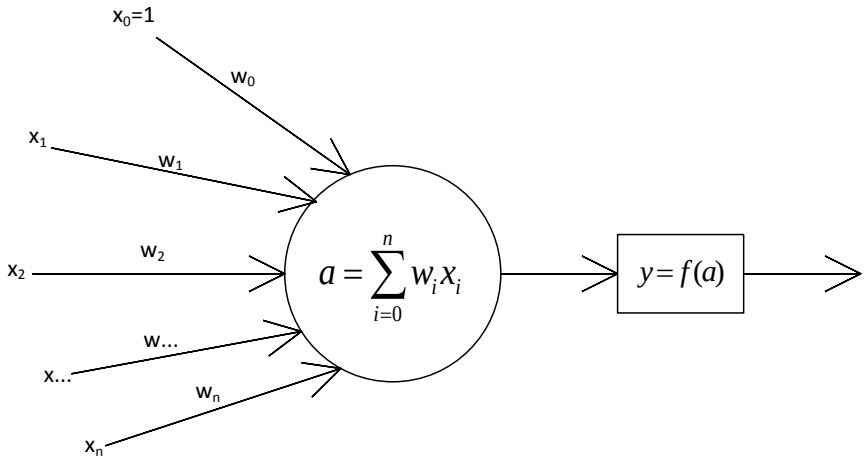
Dirbinių neuronų tinklas sudarytas iš daugybės dirbtinių neuronų, todėl norint suprasti tinklą, reikia pradėti nuo vieno dirbtinio neurono. Žmogaus smegenys sudarytos iš daugybės neuronų. Dirbtinis neuronas - tai supaprastintas šių biologinių neuronų modelis. Jo modelis pavaizduotas 1 paveiksliuke. Dirbtinio neurono veikimo principas gan paprastas - per kairėje esančias jungtis dirbtinis neuronas gauna signalus iš kitų dirbtinių neuronų - iš k-tosios jungties gaunamas x_k dydžio signalas. Šiuos signalus neuronas apjungia ir pertvarko, ir taip sugeneruojamas dirbtinio neurono išeinamasis signalas. Šis išeinamasis signalas gali būti siunčiamas daugybei kitų neuronų - dešinėje esančios jungtys yra neurono išeinamojo signalo jungtys, kuriomis ir yra siunčiamas išeinamasis signalas.

Dirbtinis neuronas generuoja išeinamąjį signalą pagal tam tikrą modelį. Pirmiausia, kiekviena įeinančioji jungtis k turi savo svorį w_k - šis svoris yra padauginamas iš įeinančio signalo dydžio x_k. Tada visos šios signalų dydžių ir svorių sandaugos yra susumuojamos - taip gaunamas sužadinimo signalas a (1 formulė). Tada sužadinimo signalas a yra paduodamas kaip argumentas tam tikrai funkcijai f ir gaunamas neurono išeities signalas $y = f(a)$. Šį funkcija f yra vadinama aktyvacijos funkcija - ją galima keisti pagal tai, kokio tikslo siekiama iš šio dirbtinio neurono. Populiariausios aktyvacijos funkcijos - slenkstinė, tiesinė, hiperbolinis tangentas bei sigmoidinė (2 formulė). Iš esmės akvyvacijos funkcija gali būti bet kokia funkcija, tačiau vėliau norint apmokyti dirbtinį neuronų tinklą, reikia rasti šios funkcijos išvestinę. Dėl šios priežasties dažniausiai pasirenkamos tokios aktyvacijos funkcijos, kurios ne tik tinkamai pertvarko signalą išvedimui, tačiau ir kurios išvestinės yra paprastos.

Įeinamosios neurono jungtys numeruojamos nuo 1 iki k. Norint a reikšmę padaryti tinkamesnę neuroninio tinklo funkcijoms, dažniausiai įvedama papildoma 0-inė jungtis su svoriu w_0 ir signalo stiprumu $x_0 = 1$. Tokiu būdu prie a (formulė 1) reikšmės papildomai pridedama $w_0 * x_0 = w_0$ reikšmė.

$$a = \sum_{k=1}^{N} w_k x_k \tag{1}$$

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



1 pav.: Dirbtinis neuronas

1.2. Dirbtiniai neuronai/tinklas?

TODO: finish

Dirbtinių neuronų tinklas (angl. *Artificial neural network*) - tai tinklas, kurį sudaro dirbtiniai neuronai bei jungtys, jungiančios kai kuriuos dirbtinius neuronus. Per kiekvieną jungtį gali eiti signalas, kuris perduoda vieno neurono išeinamąjį signalą kitam neuronui. Kiekvienas neuronas gali turėti bet kokį skaičių įeinančių ir bet kokį skaičių išeinančių jungčių.

Kai kurios jungtys gali būti prijungtos tik prie vieno neurono. Jungtys, kurios įeina į neuroną tačiau neišeina iš jokio neurono, naudojamos duomenų perdavimui - šiomis jungtimis neuronų tinklui perduodami signalai, atitinkantys duomenis. Jungtys, išeinančios iš neurono tačiau neįeinančios į jokį neuroną, naudojamos rezultato gavimui - kai per visą neuroninį tinklą pereina signalai, būtent šiose jungtyse ir yra gaunamas pateiktus duomenis atitinkantis atsakymas.

Dirbtinio neuronų tinklo užduotis - pagal pateikiamus duomenis sugeneruoti atsakymą. Pirmiausia duomenys pateikiami per tam skirtas jungtis. Neuronai, prijungti prie šių jungčių, gauna šiuos pradinius signalus, juos apdoroja ir pradeda skleisti tam tikro stiprumo signalą išeinančiomis jungtimis. Taip signalai sklinda tolyn ir visi tinklo neuronai būna apdorojami tol, kol galiausiai rezultatas būna gaunamas tam skirtose jungtyse.

Neuronų tinklo pateiktas atsakymas nebūtinai yra teisingas - neuronų tinklas suprojektuotas taip, kad jį būtų galima tobulinti pagal daromas klaidas. Vienas populiariausių būdų, kuris yra naudojamas šiame darbe - atgalinė propogacija (angl. *Backpropogation*). Tam, kad būtų galima apmokyti neuronų tinklą spręsti konkrečią problemą, reikia turėti nemažą šios problemos pavyzdinių duomenų rinkinį bei iš anksto žinoti kiekvienų duomenų teisingą atsakymą. Mokymas vyksta pažingsniui, vienu metu neuroninui tinklui pateikiant vienus duomenis iš turimo duomenų rinkinio. Neuroninis tinklas, gavęs duomenis, juos analizuoja ir pateikia tam tikrą atsakymą. Šis gautas atsakymas yra sulyginamas su teisingu, iš anksto žinomu atsakymu. Pagal tai, kaip neurininio tinklo pateiktas atsakymas skiriasi nuo teisingojo, neuroninis tinklas būna pertvarkomas. Pertvarkymas vyksta nagrinėjant neuroninį tinklą priešinga tvarka, nei buvo nagrinėjama, kai ne-

uroninis tinklas analizavo pateiktus duomenis. Nagrinėjant kiekvieną neuroninio tinklo neuroną, į jį įeinančių jungčių svoriai w_k (įskaitant ir w_0, kuris naudojamas kaip papildomas parametras) būna pakeičiami taip, kad neuroninis tinklas gautų panašesnį atsakymą į teisingąjį.

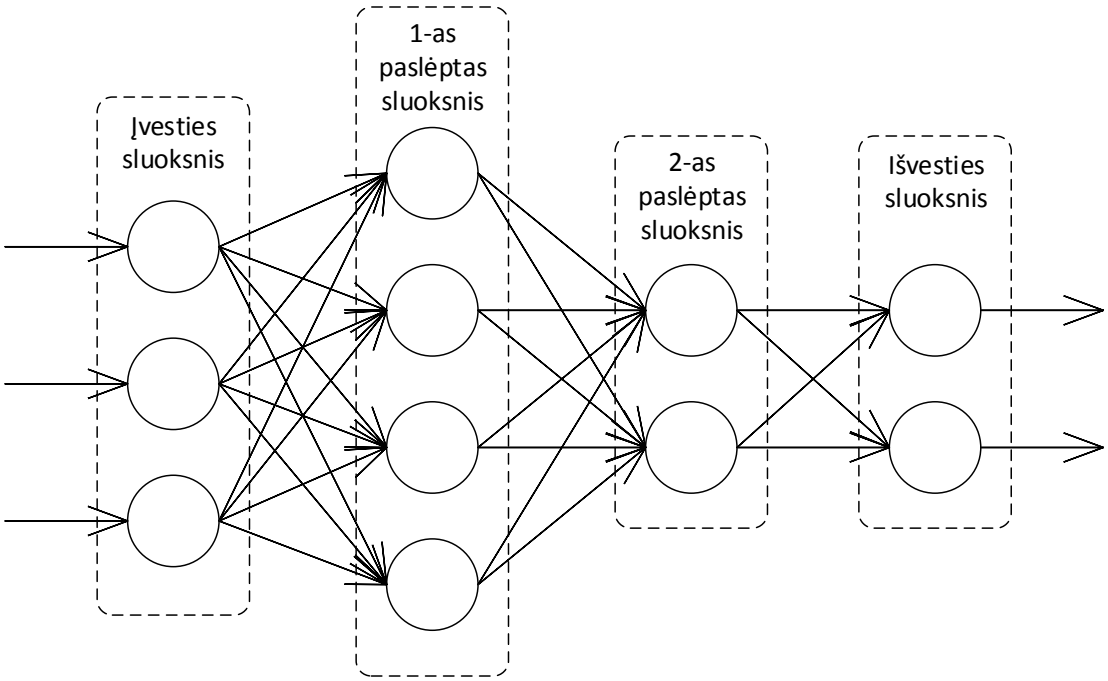
Taip atlikus vienų duomenų iš rinkinio apmokymą, imami kiti duomenys iš šio rinkinio ir mokymas kartojamas. Kiekvienus duomenis iš šio rinkinio rekomenduotina naudoti apmokymui bent kelis kartus, kadangi neuroninis tinklas ne iš karto teisingai išmoksta spręsti problemą su konkrečiais duomenimis. Taip pat apmokant tinklą su vienais duomenimis, neuroninio tinklo parametrai gali būti pakeisti taip, kad neuroninis tinklas nebegebės teisingai išspręsti prieš teisingai išspręstų duomenų. Tačiau per daug mokyti neuroninį tinklą taip pat nerekomenduotina, kadangi vėliau tinklas per daug prisitaiko prie jam apmokyti naudojamo duomenų rinkinio, o ne prie problemos. Nors ir gali pasirodyti, kad neuroninio tinklo pateikiami rezultatai vis labiau ir labiau panašėja į teisingus, tačiau išbandžius šį neuroninį tinklą su kitu šios problemos duomenų rinkiniu galima įsitikinti, kad rezultatai po kurio laiko pradeda blogėti.
TODO: įdėti (angl. *Validation group*)?
TODO: error'o skaičiavimas?
TODO: backpropogation?

1.3. Daugiasluoksnis perceptronas

Daugiasluoksnis perceptronas (angl. *Multilayer perceptron*) – tai tam tikromis savybėmis pasižymintis dirbtinių neuronų tinklas. Tai viena populiariausių dirbtinių neuroninių tinklų rūšis, kadangi savybės, kuriomis šis tinklas pasižymi, leidžia padaryti tam tikras skaičiavimo optimizacijas bei pakankamai lengvai realizuoti veikiantį neuroninį tinklą. Be to, galima keisti daugiasluoksnio perceptrono parametrus pritaikant jį konkrečiai sprendžiamai problemai.

Neuronų tinklą galima nagrinėti kaip grafą, kuriame dirbtiniai neuronai yra grafo viršūnės, o jungtys, jungiančios juos – kryptinės grafo briaunos. Jeigu neuronų tinklo grafe būtų bent vienas ciklas, tai reikštų, kad šiame cikle esančiomis jungtimis einantys signalai gali keistis ne kartą – atnaujinus tam tikro neurono išvedimo signalą, ciklu gali pakisti ir šio neurono įvedimo signalas. Tada reikėtų vėl atnaujinti šio neurono išvedimo signalą, o tai darant vėl gali pasikeisti bet kuris ivedimo signalas ir toks pasikeitimų ciklas gali kartotis labai daug kartų arba net ir niekada nesibaigti. Tai apsunkina dirbtinių neuronų veikimą, todėl dažniausiai naudojami neuronų tinklai, kuriais signalai skleidžiami pirmyn (angl. *Feedforward*). Pagal apibrėžimą, jeigu tinklo grafe nėra nei vieno ciklo, tinklas yra pirmyn skleidžiamas.

Tai, kad daugiasluoksnis perceptronas yra skleidžiamas pirmyn, suteikia nemažai privalumų. Norint apdoroti tam tikrą neuroną, privalu žinoti visus įeinančiųjų jungčių signalų dydžius, o tai reiškia, kad jau turi būti apdoroti visi neuronai, kurių išeinamosios jungtys įeina į apdorojamąjį neuroną. Grafe be ciklų rasti tokią neuronų seką, kuria būtų galima apdoroti tinklo neuronus nėra sunku – šį užduotis yra plačiai žinoma ir vadinama topologiniu rikiavimu. Yra žinoma, kad beciklį grafą visada galima topologiškai išrikiuoti, o tai reiškia, kad daugiasluoksnį perceptroną galima apdoroti tiesiog paeiliui apdorojant topologiškai išrikiuotų viršūnių seką. Šį savybė palengvina neuroninio tinklo apdorojimą.



2 pav.: Daugiasluoksnis perceptronas

Be to, daugiasluoksniai perceptronai yra organizuojami sluoksniais (žr. 2 paveiksliuką). Tinklas yra sudarytas iš perceptronų grupių, kurios yra vadinamos sluoksniais. Visi neuronų sluoksniai yra išsidėstę iš eilės, nuo kairės į dešinę. Kiekvieno sluoksnio visi neuronai turi išeinamąsias jungtis į visus neuronus iš sekančio sluoksnio, esančio dešinėje. Išimtis paskutinis sluoksnis - kadangi jis naudojamas išvedimui, todėl neturi jungčių į kitus neuronus, o jo jungtyse formuojamas atsakymas į analizuojamą problemą.

Pirmasis neuronų sluoksnis, kuriam paduodami duomenys, vadinamas įvesties sluoksniu (angl. *Input layer*). Po to gali būti vienas ar keli paslėpti sluoksniai (angl. *Hidden layer*), kurie naudojami tam, kad neuroninio tinklo struktūra būtų didesnė ir to pasekoje gebėti išmokti sudėtingesnius dalykus. Paprasčiausiuose tinkluose gali išvis nebūti paslėptų sluoksnių, tačiau tokio tinklo galimybės būtų labai ribotos. Paskutinysis sluoksnis, naudojamas rezultatų suformavimui, vadinamas išvesties sluoksniu.

Tokia neuronų organizacija į sluoksnius dar palengvina tinklo veikimą - signalus siųsti galima sluoksnis po sluoksnio, pradedant pirmuoju, kuris gauna problemos duomenis, siunčiant signalus visomis jungtimis antrąjam sluoksniui. Po to tas pats atliekama su antruoju sluoksniu siunčiant visus signalus trečiąjam ir t.t., kol galiausiai rezultatai gaunami paskutiniame sluoksnyje.

Savybė, kad kiekvieno sluoksnio, išskyrus paskutinio, visi neuronai turi išeinamąsias jungtis į visus sekančio sluoksnio neuronus taip pat yra naudinga - žinant tai, nereikia turėti jokio tinklo grafo, kadangi užtenka žinoti kiekvieno sluoksnio neuronus bei jungčių svorius. Be to, signalų siuntimas iš vieno sluoksnio į sekantį taip pat gali būti supaprastintas. Vietoj to, kad programiškai iteruoti per visas dviejų sluoksnių neuronų poras ir atlikti jungties dydžio skaičiavimą, tą galima padaryti su matricų aritmetika. Signalo skaičiavimo atliekamas operacijas, nurodytas 1 formulėje,

galima sutraukti į matricų daugybos operacijas, o aktyvacijos funkciją pritaikyti visai matricai. Taip daugiasluoksnio perceptrono tinklo modelis tampa dar tvarkingesnis ir paprastesnis.

TODO: dirbtinio neurono paveiksliukas

TODO: citata?

TODO: 110 iš knygos

TODO: Perkelti statinės struktūros aprašymą (pateiktą žemiau):

Prieš pradedant neuroninio tinklo apmokymą, visa jo struktūra yra statiškai aprašoma - parenkama, kiek sluoksnių jis turi. Taip pat aprašoma, kiek kiekvienas sluoksnis turi neuronų, kokios aktyvacijos funkcijos yra naudojamos, kaip duomenys yra paduodami ir panašiai. Visa tai kažkiek suvaržo neuroninio tinklo galimybes mokytis - prireikus išmokti sudėtingesnius duomenis, turima neuroninio tinklo struktūra gali to tiesiog nemokėti. Dėl to konkretus neuroninis tiklas turi tam tikras ribas, iki kurių gali tobulėti.

1.4. Neuroninio tinklo apmokymas

Neuroninių tinklų mokymo principas yra apmokyti jį mokant vieno pavyzdžio vienu metu. Pavyzdžio duomenys būna paduodami tinklui, gaunamas konkretus atsakymas, kuris yra sulyginamas su teisingu atsakymu. Pagal tai, koks buvo gautas atsakymas ir koks yra teisingas atsakymas, apskaičiuojama, kaip reikia pakeisti tinklo svorius, kad kitą kartą būtų gautas atsakymas, panašesnis į teisingą. Tai pasiekiama kiekvienam tinklo svoriui apskaičiavus, kaip ir kiek jį reikia pakeisti (padidinti ar pamažinti svorį). Tuomet koeficientas yra pakeičiamas pagal tai, kaip tuo metu jo pakeitimas pakeistų analizuojamo pavyzdžio atsakymą.

TODO: sutvarkyti pastraipą (ji perkelta iš momento koeficiento)

TODO: parašyti 3

$$w' = w - \eta \frac{\partial C}{\partial w} \tag{3}$$

1.5. Momento koeficientas

Momento koeficientas - tai itin populiari neuroninio tinklo modifikacija, skirta pagerinti tinklo mokymosi procesą. Ji remiasi 1.4 poskiryje aprašytu mokymu, tačiau šiek tiek praplečia svorių keitimą. Svoriai keičiami ne pagal 3 formulę, o pagal jos plėtinį 5 formulėje.

Svoriai keičiami ne tik pagal tai, kaip svorius keisti apsimoka būtent šiuo metu būtent šiam testui, tačiau ir pagal tai, kaip tai buvo keičiama praeitose mokymo iteracijose. Tai įgyvendinama pasinaudojant greičio ir įveikto atstumo fizikinį analogą - kai norima įveikti tam tikrą atstumą, kūnui reikia suteikti jėgos ir įgauti greitį, o greitis leis judėti. Panašiai veikia momento koeficientas - analizuojant, kaip reikia pakeisti svorius, šie dydžiai nėra tiesiog pritaikomi šiuo metu, tačiau jie yra panaudojami kaip greitis. Kiekvienam parametrui išsaugomas kitimo greitis, aprašytas 4 formulėje, kuris po kiekvienos apmokymo iteracijos pakeičiamas padidinant ar pamažinant tam tikra reikšme pagal tai, kaip šioje iteracijoje reikia keisti parametro dydį. Šis turimas greitis kiekvienos iteracijos metu yra pritaikomas kaip pokytis parametrui keisti.

Toks parametrų keitimo metodas suteikia kelis privalumus. Pirmiausia, mokymo procesas pagreitėja – jei kelias iteracijas parametrą reikia keisti ta pačia linkme, naudojant šį metodą prireiks mažiau iteracijų, kadangi pakeitimas sumuosis ir kaskart panaudojamas vis didesnis pakeitimo žingsnis. To pasiekti vien padidinus mokymo koeficientą nepavyks, kadangi didelis mokymo koeficientas atliks didelius pakeitimus net ir ten, kur jų nereikia. Tuo tarpu momentinis metodas iš pradžių pradės mokymą mažais žingsniais ir jeigu mokymo kryptis pasikeis, taip pat ir koeficiento kitimo greitis pradės keistis.

Taip pat šis metodas leidžia išvengti lokalių minimumų. Tai įvyksta natūraliai, kadangi įgavus pakankamai didelį parametro keitimo žingsnį, patekimas į lokalų minimumą nesugebės greitai pakeisti mokymo krypties. Žinoma, tai turės įtakos parametro keitimo žingsniui, tačiau jeigu lokalus minimumas yra iš tikro lokalus ir gan nereikšmingas, tada turimas parametro keitimo žingsnis leis kurį laiką tęsti parametro keitimą į tą pusę, į kurią jis buvo keičiamas senesnėse iteracijose.

Tinklo apmokymui naudojamas apmokymo rinkinys, sudarytas iš kelių (dažniausiai daugybės) skirtingų įrašų. Naudojant paprastą apmokymo metodą, kiekvieno įrašo apmokymas yra atskiras procesas. Naudojant momento koeficientą, parametrų keitimo tendencijos tarp skirtingų įrašų išlieka. Tai suteikia tam tikro pranašumo, kadangi labai išsiskiriantys ar nevisai korektiški įrašai, kurių paprastai yra nedaug, mažiau neigiamai paveiks tinklą. Taip yra todėl, kad jie keis ne pačias parametro reikšmes, bet greitį, kuris yra priklausomas ir nuo senesnių kitų įrašų iteracijų. Tai gali padėti išvengti tinklo pakeitimo tokia linkme, kuri kitiems įrašams gali stipriai pabloginti rezultatą – pavyzdžiui, patekimo į lokalų minimumą.

$$v' = \mu v - \eta \frac{\partial C}{\partial w} \tag{4}$$

$$w' = w + v' \tag{5}$$

TODO: function

TODO:

TODO: keitimo žigsnis/greitis - suvienodinti savokas?

2. Duomenys

2.1. Vilkdagių duomenys

Programuojant neuroninius tinklus, testavimui buvo panaudoti vilkdagių (angl. *Iris flower*) duomenys. Tai plačiai taikomi ir viešai pasiekiami duomenys, aprašantys 3 rūšių vilkdagius. Aprašyta po 50 kiekvienos rūšies vilkdagių. Kiekvienas vilkdagis aprašomas pateikiant 4 dydžius: taurėlapio ilgis, taurėlapio plotis, vainiklapio ilgis bei vainiklapio plotis. Šiuos vilkdagių duomenis sudaro 150 gėlių, kurių kiekviena aprašyta 4 parametrais bei priskirta vienai iš 3 vilkdagių grupių.

Šie duomenys puikiai tinka klasifikavimo tinklo apmokymui - tinklo tikslas yra kuo mažiau klystant pasakyti, kuriai iš 3 vilkdagių rūšių tam tikra gėlė su tam tikrais parametrais priklauso. Be to, yra pakankamai duomenų, kad būtų galima dalį jų panaudoti tinklo apmokymui, o kitą dalį - testavimui. Tokiu būdu bus užtikrinama, kad tinklas teisingai išmoko atskirti vilkdagių rūšis pagal parametrus, o ne tiesiog prisitaikė prie mokymui panaudotų duomenų.

TODO: nuoroda į vilkdagių duomenis

TODO: vilkdagių dimensijų mažinimo ir klasifikavimo diagramos?

2.2. Genų duomenys

TODO: atnaujinti čia panaudotų duomenų sąvokas kitur (įrašas).

Duomenis sudaro 12000 genų įrašų. Kiekvienas genas aprašytas 30-čia parametrų, apibūdinančių geno savybes. Visos parametrų reikšmės yra natūralieji skaičiai. Kiekvienas genas priklauso vienai iš 24 grupių. Visos genų grupės turi po 500 genų.

2.3. Duomenų normavimas

Analizuojami duomenų parametrai gali būti labai skirtingi, kadangi jie išreiškia skirtingas savybes, dydžius. Skiriasi parametrų mąsteliai - pvz. vieno parametro reikšmės gali kisti intervale $[0; 10]$, o kito $[10^3; 10^9]$. Dėl šių skirtumų tokias parametrų reikšmes tiesiai pateikti neuroniniui tinklui nėra praktiška. Taip atsitinka dėl neuroninio tinklo veikimo principo - parametrai, kurių reikšmės linkusios būti didesnėmis, turėtų didesnę įtaką rezultatui nei parametrai su mažesnėmis reikšmėmis, kadangi pradiniai duomenų signalai būtų daug stipresni parametrų su didesnėmis reikšmėmis. Tačiau daug naudingiau būtų visiems parametrams suteikti vienodą svarbą, kadangi parametro reikšmių intervalas neturi nieko bendro su parametro svarba - gali būti ir taip, kad dideles reikšmes įgyjantis parametras yra kur kas mažiau svarbus nei mažas reikšmes įgyjantysis. Dėl šios priežasties prieš pateikiant duomenis neuroniniui tinklui, juos svarbu normuoti.

Normuojant duomenis, kiekvienas duomenų parametras normuojamas atskirai. Norint sunormuotį parametrą, reikia išnagrinėti turimas šio parametro reikšmes bei pakeisti jas naujomis. Sunormalizavus duomenis, visi parametrai turėtų turėti beveik lygią svarbą neuroniniui tinklui - jų reikšmių intervalai turėtų būti lygūs. Paprasčiausias parametro normavimo metodas - rasti minimalią p_{min} ir maksimalią p_{max} parametro reikšmes ir kiekvieną i-tojo įrašo analizuojamo

parametro reikšmę p_i pakeisti pagal formulę:

$$p_i = \frac{p_i - p_{min}}{p_{max} - p_{min}} \tag{6}$$

Šis metodas užtikrina, kad naujos parametro reikšmės būtų intervale $[0; 1]$. Šis metodas buvo naudojamas tyrimo pradžioje, kadangi tai turbūt paprasčiausias ir lengvai sugalvojamas metodas. Visgi vėliau buvo pradėtas naudoti kitas metodas, naudotas ir aprašytas [RHM97] šaltinyje - Gauso normavimas (angl. *Gaussian Normalization*). Pagal šį metodą, radus parametro vidurkį p_{mean} ir vidutinį nuokrypį p_{std}, kiekviena i-tojo įrašo analizuojamo parametro reikšmė keičiama pagal formulę:

$$p_i = \frac{p_i - p_{mean}}{p_{std}} \tag{7}$$

Šiuo metodu gaunamų reikšmių vidurkis lygus nuliui, o dispersija lygi vienetui. Nors ir šiuo metodu gaunami duomenys nėra apibrėžti konkrečiame reikšmių intervale kaip pirmojo metodo, visgi yra 68% tikimybė, kad jie bus iš intervalo $[-1; 1]$. Bandymu metu šis metodas pasirodė tinkamesnis tiriamai problemai.

TODO: palyginti abiejų normavimų efektyvumą?

TODO: rezultatų palyginimas?

3. Duomenų grupės/Duomenų apmokymas/...

TODO: nuspręsti poskyrio pavadinimą

Neuroniniai tinklai mokymo procese yra linkę prisitaikyti prie konkrečių mokymui naudojamų duomenų. Taip atsitinka todėl, kad neuroninio tinklo gerumas yra matuojamas pagal tai, kokios klaidos gaunamos naudojant mokymui parinktus duomenis. Kadangi mokymui naudoti yra naudojama baigtinė aibė įrašų, todėl neuroninis tinklas stengiasi kuo labiau prie jų visų prisitaikyti, neatsižvelgdamas į tai, kad gali egzistuoti ir kitokie duomenys. Dėl to po tam tikro skaičiaus mokymo įtaracijų dažnai įvyksta persitaikymas (angl. *Overfitting*) - neuroninis tinklas siekia sumažinti mokomų duomenų pateikiamą klaidą taip stipriai, kad nuo to gali kentėti kitų, ne iš mokymo aibės pateikiamų įrašų atsakymai.

Viena priežastis, kodėl taip įvyksta - ne visi duomenys tobulai atitinka analizuojamą populaiciją. Dažniausiai duomenyse egzistuoja tam tikros anomalijos, būdingos tik keliems duomenų įrašas, kurios pirmiausia būna natūraliai ignoruojamos. Nors ir gaunamas blogas atsakymas su tokiomis anomalijomis, visgi tokių įrašų yra mažuma, todėl bendras rezultatas yra neblogas. Atliekant daugybę itaracijų su tais pačiais mokymo duomenimis, neuroninis tinklas nebeturi kur tobulėti, kaip tik prisitaikyti prie šių anomalijų, kadangi ir jos didina neuroninio tinklo klaidą. Tada neuroninio tinklo struktūra keičiasi taip, kad apimtų ir šią anomaliją, tačiau dėl to kenčia likusios populiacijos rezultatai.

Kita priežąstis, dėl kurios įvyksta persitakymas, yra lokalių minimumų problema. Prieš pradedant neuroninio tinklo apmokymą, visi neuronų jungčių svoriai yra parenkami atsitiktinai. Dėl to gaunamas atsitiktinis neuroninis tinklas, kuris jau pateikia tam tikrus atsakymus. Tikėtina, kad jo pateikiami atsakymai yra labai prasti ir gali būti pagerinti, kadangi tai visiškai naujas atsitiktinai sukurtas tinklas, dar nieko nežinantis apie tiriamą problemą. Dėl to mokymo metu jis tobulėja, mažindamas mokymo duomenų pateikiamą klaidą. Problema kyla dėl to, kad neuroninio tinklo tobulėjimas vyksta mažais žingsniais, iš dabartinių turimų svorių juos truputį pakeičiant taip, kad atsakymas pagerėtų. O kadangi visas mokymas prasideda nuo atsitiktinai sugeneruotų duomenų, todėl pats mokymas bus ne artėjimas prie idealaus tinklo, o prie panašaus į atsitiktinį tinklą, kuris veiktų geriau, nei dabar. Tačiau kartais norint pasiekti gerų rezultatų, gali prireikti drąstiškai pakeisti daugybę tinklo svorių. Visgi to tinklas padaryti negali, nes vienu metu jis analizuoja tik vieną duomenų įrašą, neatsižvelgiant apie kitus, todėl rimtesni pakeitimai sugadintų neuroninį tinklą sprendžiant tiriamą problemą. Dėl to apmokomas tinklas turi ribas, iki kiek gali iš tikro gali tobulėti - ši riba ir yra vadinama lokaliu minimumu. Artėdamas prie lokalaus minimumo, tinklas taip stipriai prisiderina prie mokymui panaudotų duomenų, kad jo struktūros teikiami rezultatai taip pat pradeda blogėti tiriamajai problemos populiacijai.

Paskutinė persitaikymo priežąstis, kuri šiek tiek paaiškina prieš tai buvusias priežąstis, yra ta, kad neuroninis tinklas vėlesnėse iteracijose per daug siekia tobulumo. Dažnai maža neuroninio tinklo pateikiama klaida nereiškia blogo atsakymo. Klasifikavimo uždavinyje atsakymas yra grupė, kuriai analizuojamas įrašas priklauso. Tačiau jo pateikiamo atsakymo formatas skiriasi - atsakyme yra gaunama, kiek šis įrašas turi kiekvienos grupės savybių. Iš šio tinklo pateikiamo atsakymo rezultatas gaunamas išrenkant grupę, kuri yra būdingiausia įrašui. Dėl to atsakymas bū-

tų idealus, jeigu analizuojamas įrašas būtų 100% priklausomas grupei, kuriai jis priklauso, ir 0% kitoms. Tačiau dažnai netgi patys duomenys gali būti tokios prigimties, kad nors jie ir priklauso vienai konkrečiai grupei, visgi turi ir kitų grupių savybių. Dėl to siekis, kad kiekvienas mokymo duomenų įrašas būtų pripažintas kaip 0% priklausomas kitoms grupėms, šiek tiek kenkia tinklo tobulėjimui - nors ir pasiekiamas teisingas atsakymas, visgi bandoma jį vis tobulint, siekiant pasiekti 0% ir 100% priklausomybę atitinkamoms grupėms. Čia ir įvyksta persitaikymas, nes po tam tikro iteracijų skaičiaus neuroninis tinklas tiek bando pasiekti minimalią galima klaidą, kad jo rezultatas su likusia problemos duomenų populiacija pradeda kristi.

Norint gauti neuroninį tinklą, kuris efektyviai spręstų ne tik mokymui parinktus duomenis, bet ir visą problemos populiaciją, reikia vengti persitaikymo. Tą galima padaryti naudojant validacijos rinkinį(angl. *validation set*). Apmokymui reikia parinkti ne tik apmokymui naudojamą rinkinį, bet taip pat ir validacijos rinkinį. Šie duomenų rinkiniai turi neturėti bendrų įrašų, priešingu atveju validacija nebus veiksminga. Taip pat tiek apmokymo, tiek validacijos rinkiniuose turėtų būti kuo įvairesnių ir skirtingesnių įrašų - jie turi apimti įvairius skirtingus duomenų variantus tam, kad neuroninis tinklas juos išmoktų bei kad validacija būtų efektyvi. Turint apmokymo ir validacijos rinkinius, apmokymas vyksta įprastai - tinklas mokomas spręsti po vieną įrašą. Tačiau vietoj to, kad tinklo progresas būtų sekamas pagal tai, kaip gerai ji sprendžia apmokymo rinkinį, yra panaudojamas validacijos rinkinys. Po kiekvienos apmokymo iteracijos, neuroniniui tinklui yra duodami validacijos rinkinio įrašai. Juos neuroninis tinklas turi išspręsti, tačiau be mokymosi žingsnio, nes kitaip validacijos rinkinys netektų prasmės. Gavus validacijos grupės atsakymų klaidą, ją galima naudoti stebint neuroninio tinklo progresą. Paprastai mokymo pradžioje validacijos grupės klaida mažėja, tačiau po tam tikro iteracijų skaičiaus ši klaida pradeda didėti, nors apmokymo rinkinio gaunama klaida vis dar mažėja. Būtent šioje vietoje ir prasideda persitaikymas, todėl vos pasiekus šią ribą arba dar po kelių iteracijų nuo jos geriausia neurono apmokymą nutraukti.

Mokymo rinkinys Validacijos rinkinys (Normavimas vyksta kartu)

TODO: paieškoti lokalių minimumų straipsnių

4. Dimensiškumo mažinimas

Klasifikavimo problema

Galimi sprendimai:

* statistinis sprendimas * neuroniniai tinklai * veikimas * apmokymas * validavimas? * Klasifikavimas požymių išskyrimui * Dimensiškumo mažinimas -> Klasifikavimas

4.1. Tiesinė diskriminantinė analizė

Vienas iš galimų dimensiškumo mažinimo sprendimo būdų - tiesinė diskriminantinė analizė (angl. *Linear discriminant analysis*).

TODO: Aprašyti veikimą

Pagrindinis šio metodo suvaržymas yra tai, kad kiekvienas naujas parametras yra gaunamas sudėjus originalių parametrų reikšmes, padaugintas iš atitinkamų koeficientų. Tai reiškia, kad naujieji parametrai turi tiesinę priklausomybę nuo senųjų, todėl toks dimensijų mažinimas yra labai elementarus. Jeigu norima sukurti naują parametrą, kurio išvedimas yra sudėtingesnis, to padaryti naudojant šį metodą nepavyks. Dėl to gaunamos parametrų reikšmės gali būti mažai naudingos.

TODO: statistinis dimensijų mažinimas - tiesinis (linear), o neuroniniai tinklai geba atlikti sudėting

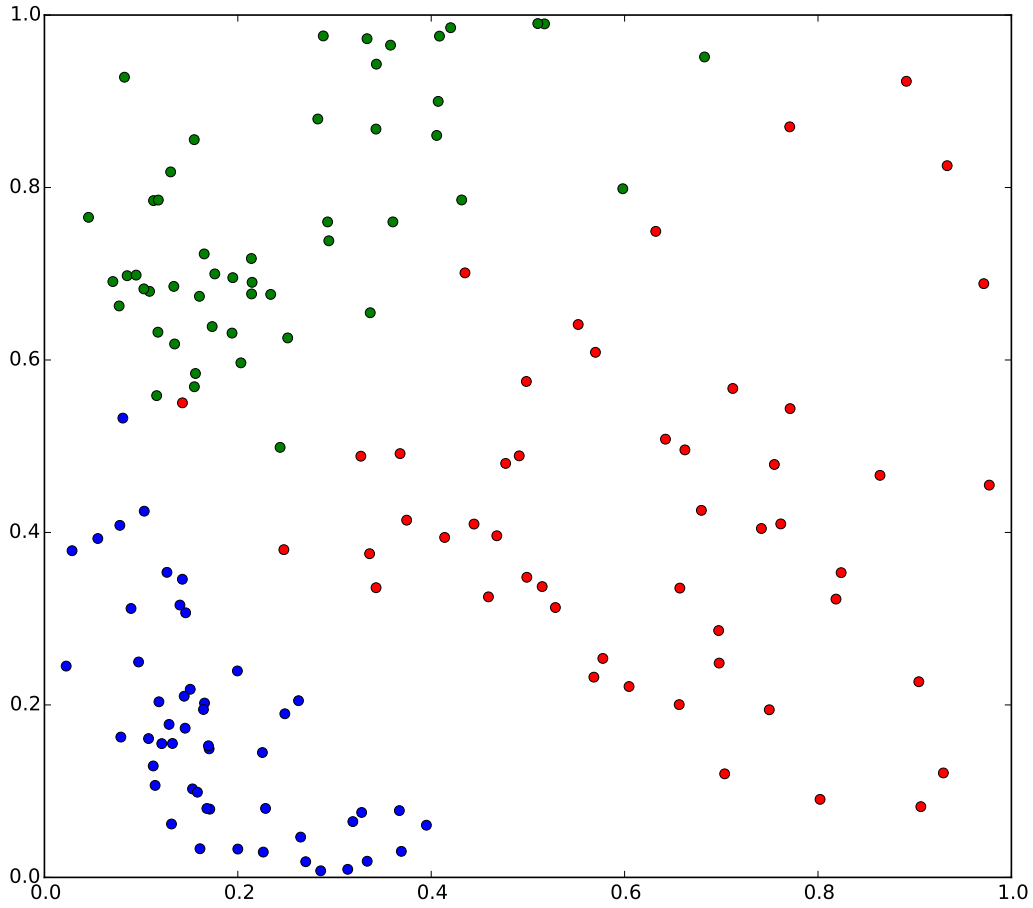
TODO: paaiškinti, kaip veikia tiesinis - kiekviena nauja savybė gaunama iš originalių savybių su ta

`http://en.wikipedia.org/wiki/Linear_discriminant_analysis#Face_recognition`

TODO: Pavyzdinio rezultato paveiksliukas

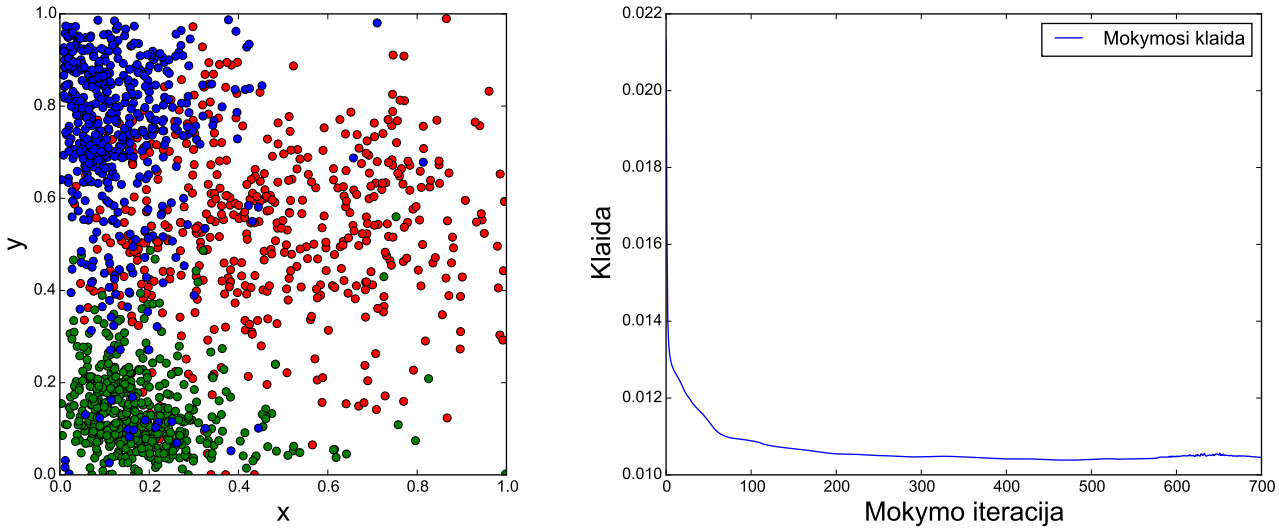
TODO: matlab, python script

TODO: SVD? `http://en.wikipedia.org/wiki/Singular_value_decomposition`



4.2. Dimensiškumo mažinimas neuroniniu tinklu

Dimensiškumo mažinimui taip pat buvo panaudotas neuroninis tinklas. Turint N dimensijų ir norint jas sumažinti iki M, kai $M < N$, tai buvo atliekama sukūrus neuroninį tiklą, kurio pirmąjame ir paskutinimae sluoksniuose yra po N neuronų, o viename iš vidinių sluoksnių - M (šį vidinį sluoksnį vadinkime kompresijos sluoksniu). Tokio neuroninio tinklo užduotis nėra tiesiog sumažinti dimensijų skaičių - tai daroma netiesiogiai. Šiam tinklui perduodant tam tikrus M dimensijų turinčius duomenis, iš jo tikimasi, kad išeities neuronuose susiformuos rezultatas, lygus pradiniams duomenims - tai yra neuronų tinklas šių duomenų nepakeis. Ši užduotis būtų paprasta, jeigu visi vidiniai turėtų bent N dimensijų - tada pateikiami duomenys galėtų būti tiesiog perkeliami iš vieno neuronų sluoksnio į kitą nepakeisti. Tačiau kompresijos sluoksnis turi tik M neuronų - vadinasi, duomenis reikės tam tikru būdu pertvarkyti, kad jie galėtų būti perduodami per šį sluoksnį prarandant kuo mažiau savybių. Būtent čia ir įvyksta dimensiškumo mažinimas - neuroninis tinklas yra apmokomas pateikti kuo panašesnius duomenis į pradinius, ko pasekoje kompresijos sluoksnyje su M neuronų yra gaunami duomenys, turintys mažiau dimensijų. Norint sumažinti tam tikro duomens dimensijas, užtenka šį duomenį paduoti apmokytui neuroniniui tinklui ir pažiūrėti, kokie duomenys susidarė kompresijos sluoksnyje. Nuskaičius šių neuronų



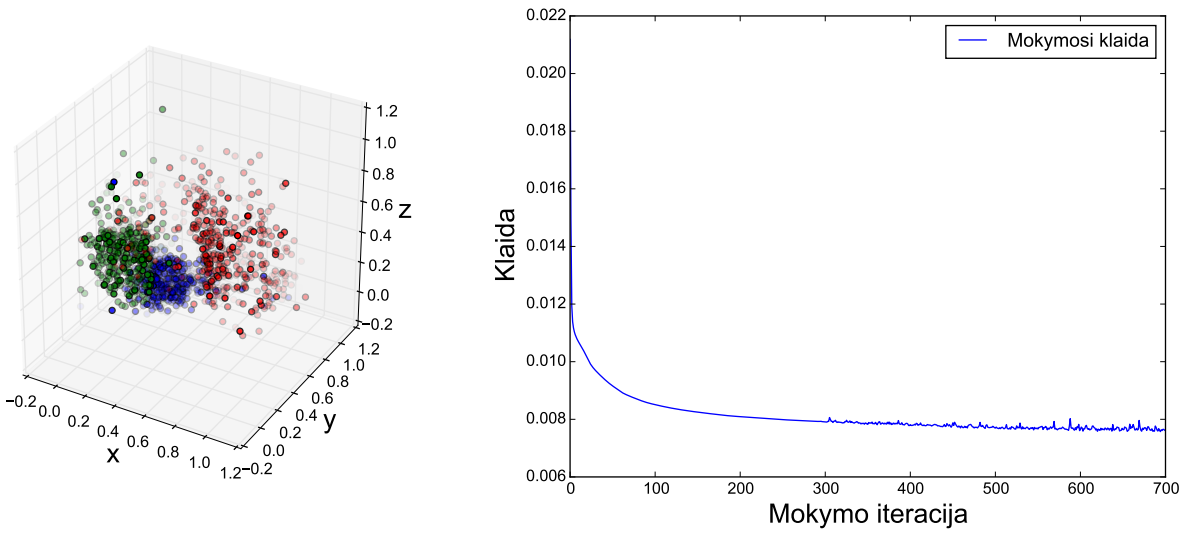
3 pav.: Dvimačiai duomenys

reikšmes ir bus gaunamas duomuo, turintis mažiau dimensijų.

Tokiame apmokytame neuroniniame tinkle visi sluoksniai, esantys kairėje nuo kompresijos sluoksnio, yra naudojami dimensiškumo mažinimui. Būtent per šiuos sluoksnius einant signalams ir yra sudaromas mažiau dimensijų turintis duomuo. Kadangi šio neuroninio tinklo tikslas yra pateikti rezultatą, kuris būtų kuo panašesnis į pateiktus duomenis, todėl galima teikti, kad sluoksniai, esantys dešinėje nuo kompresijos sluoksnio, yra naudojami pradinių duomenų atstatymui.

Turimi duomenų dimensiškumas buvo sumažintas šiuo būdu. Kadangi duomenys turi 30 dimensijų, duomenų dimensijos buvo mažinamos iki [1; 29] dimensijų. Pateikiami dvimačių (3 pav.) ir trimačių (4 pav.) duomenų dimensiškumo rezultatai - paimta po 500 įrašų iš 3 grupių, kurios atvaizduotos skirtingomis spalvomis. Iš pateiktų rezultatų matosi, kad dvimačiai duomenys susigrupavę ne taip gerai, kaip trimačiai. Kadangi turimi duomenys turi 30 dimensijų, todėl dimensijų sumažinimas iki 2 praranda didesnę dalį duomenų nei mažinant iki 3 dimensijų. Didesnių dimensijų rezultatų vizualiai pavaizduoti nepavyksta dėl natūralių priežasčių.

TODO: diagrama su dimensijų mažinimo neuroniniu tinku (kompresijos, dekompresijos pusės; N,



4 pav.: Trimačiai duomenys

5. Klasifikavimas mažinant dimensiškumą

Norint išsiaiškinti, ar galima pasiekti geresnius klasifikavimo rezultatus prieš tai sumažinant duomenų dimensiškumą, buvo atliktas tyrimas. Kadangi tiriamus duomenis sudaro 30 dimensijų, buvo bandoma sumažinti dimensijas iki visų galimų dimensijų dydžių - nuo 1 iki 29. Norint išsiaiškinti klasifikavimo efektyvumą pasirinkus į kiek dimensijų bus kompresuojami duomenys, atliekamas eksperimentas.

Eksperimento metu pirmiausia sukuriamas ir apmokomas daugiasluoksnis perceptronas, skirtas kompresuoti duomenis į pasirinktą dimensijų skaičių. Šį perceptroną sudaro 5 sluoksniai - įvedimo, išvedimo bei 3 paslėptieji sluoksniai. Įvedimo, išvedimo bei 2 paslėptieji sluoksniai turi po 30 neuronų. Trečiasis paslėptasis sluoksnis, esantys tinklo viduryje, turi tiek neuronų, į kiek dimensijų norima sukompresuoti duomenis - pagal anksčiau aprašytą metodą šiame sluoksnyje ir yra gaunami sukompresuoti duomenys. Šis sluoksnis tada yra apmokomas 300 kartų iteruojant per visus mokymo rinkinio įrašus. Apmokius tinklą, duomenys yra sukompresuojami pasinaudojant geriausią mokymo metu pasiektą rezultatą.

TODO: pridėti kompresijos tinklo schemą?

TODO: pridėti ref į kompresijos tinklo aprašymą vietoje "pagal anksčiau aprašytą"

Gavus sukompresuotus duomenis, pradedamas klasifikavimas. Pirmiausia sukuriamas klasifikavimui skirtas daugiasluoksnis perceptronas. Šį perceptroną sudaro 4 sluoksniai. Pirmiausia įvesties sluoksnis, turintis tiek neuronų, kiek dimensijų turi sukompresuoti duomenys. Tada du paslėpti sluoksniai, turintys po 30 neuronų. Galiausiai, išvesties sluoksnis, turintis tiek neuronų, kiek yra skirtingų galimų klasių. Kadangi šio eksperimento metu buvo analizuojami duomenys iš 3 skirtingų klasių, todėl išvesties sluoksnis turėjo 3 neuronus. Šis tinklas buvo apmokomas panaudojant iš kompresijos tinklo gautais kompresuotais duomenimis. Apmokymas vyko 400 kartų, kaskart mokymo metu panaudojant visus mokymo rinkinio įrašus. Baigus apmokymą, gaunamas rezultatas - klasifikavimo klaida.

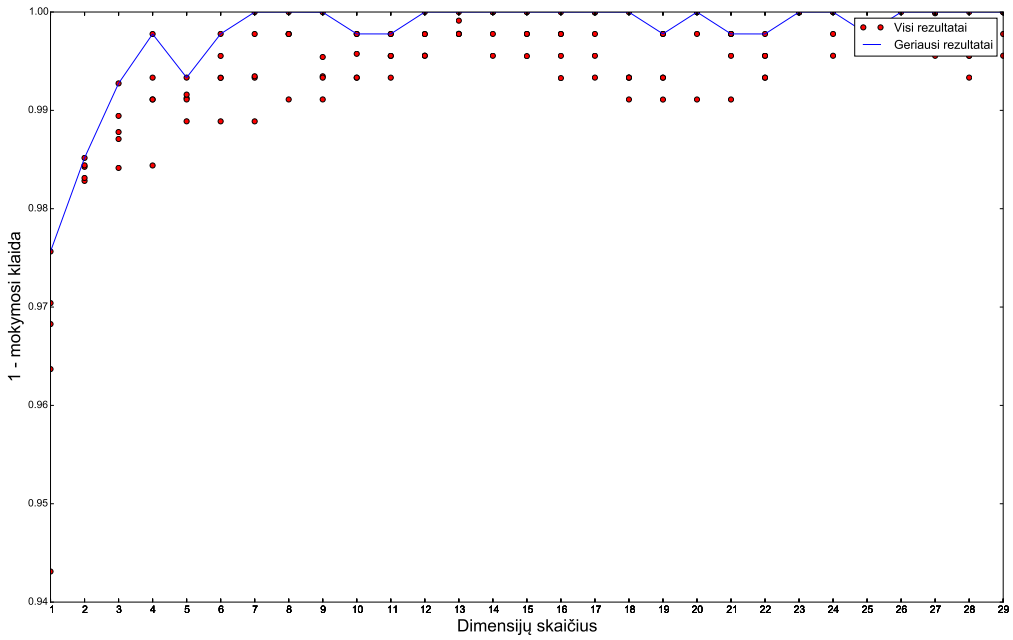
TODO: pridėti klasifikavimo tinklo schemą?

Kiekvienam galimam dimensijų skaičiui, į kurį norima sutraukti duomenis, šis eksperimentas buvo atliktas po 5 kartus. Taip buvo užtikrinama, kad gauti rezultatai bus patikimesni, kadangi kai kuriais atvejais atsitiktinai sugeneruoto perceptrono struktūra gali būti labai nepalanki ir gali būti gaunamas prastas rezultatas, neparodantis, kiek visgi toks kompresijos mažinimas gali padėti klasifikavimui.

5.1. Pirmasis eksperimentas

TODO: geresnis pavadinimas

Pirmiausia šis eksperimentas buvo atliktas su daliniais duomenimis. Iš kiekvienos grupės paimta po 50 įrašų, kurie buvo naudojami eksperimentuose tiek apmokymui, tiek ir pačiam tinklo klaidos skaičiavimui. Taip buvo pasielgta norint išsiaiškinti perceptronų galimybes - kadangi visi duomenys, kurie naudojami klaidos skaičiavime, yra naudojami taip pat ir apmokant, neuroniniui tinklui nelieka nežinomų duomenų. Pirmiausia šie duomenys buvo panaudoti apmokant komp-



5 pav.: Klasifikavimo mažinant dimensijas rezultatai

resijos tinklą, kuris po to buvo panaudotas šiems duomenims sukompresuoti. Tada šie sumažinto dimensiškumo duomenys buvo panaudoti klasifikavimo tinklo apmokymui. Apmokius klasifikavimo tinklą, gaunama klaida. Tai atlikus po 5 kartus visiems galimiems dimensijų skaičiams, gauti rezultatai, pavaizduoti 5 pav.

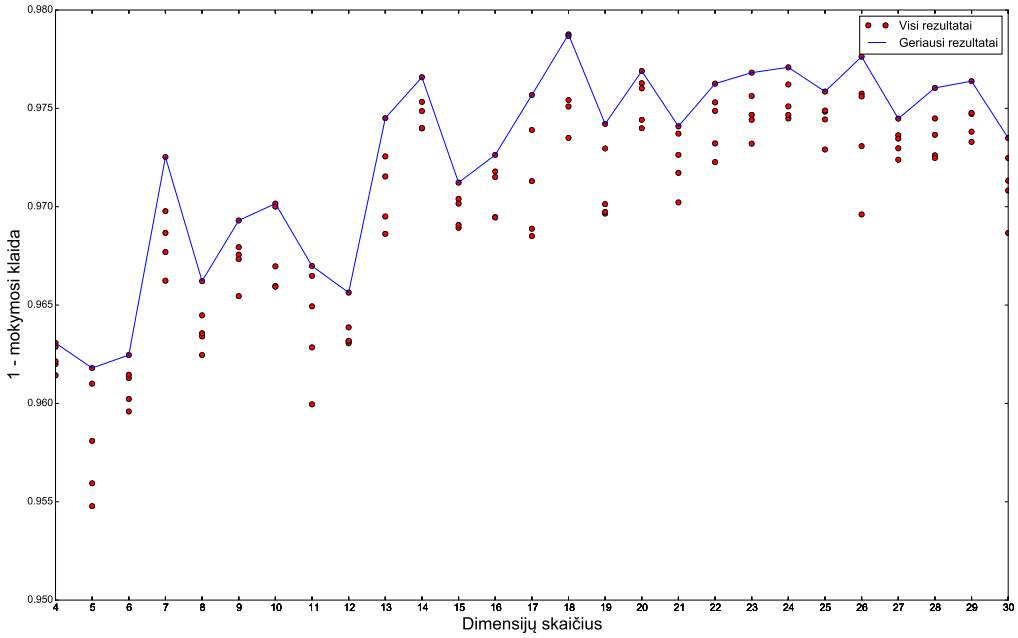
Iš rezultatų matosi, kad klasifikavimo tinklas gerai geba susidoroti su tokiais mažais duomenimis - rezultatai labai netoli vieneto, tai yra, klaida beveik lygi nuliui. Išimtis tyrimuose su mažais dimensijų skaičiais - kai dimensijų yra mažai, neuronų tinklas nesugeba suklasifikuoti sukompresuotų duomenų. Tai parodo, kad norint mažinti dimensijų skaičių iki labai mažo, prarandama šiek tiek informacijos apie duomenis. Dėl to juos klasifikuoti pasidaro sunkiau. Todėl norint padidinti klasifikavimo efektyvumą, reikia mažinti dimensijas bent iki keturių.

TODO: aprašyti, kaip išrinkti duomenys - kiek, kokios grupės, kas buvo naudojama apmokymui ir

5.2. Antrasis eksperimentas

TODO: geresnis pavadinimas

Sekančiame atliktame eksperimente buvo panaudoti visi įrašai iš tiriamų grupių - 500 įrašų iš 3 grupių. Kompresijos perceptrono apmokymui buvo panaudoti visi šie įrašai, kadangi tikslas yra kuo tiksliau ir autentiškiau sukompresuoti visus duomenis, o ne paruošti tinklą, kuris gebėtų kompresuoti dar tinklui nematytus duomenis. Apmokius kompresijos tinklą, gaunami kompresuoti duomenys. Tada iš kiekvienos grupės atsitiktinai pasirenkama po 50 įrašų, kurie naudojami klasifikavimo perceptrono apmokymui. Apmokius perceptroną, jo klaida skaičiuojama su visais duomenimis, o ne tik naudotais mokymo metu. Taip dauguma įrašų perceptronui yra nauji, to-



6 pav.: Klasifikavimo mažinant dimensijas rezultatai

dėl taip ištiriama, kaip gerai perceptronas sugebėjo išmokti duomenų savybes iš mažų mokymo grupių. Gauti rezultatai, pavaizduoti 6 pav.

Iš šių rezultatų matosi, kad su dimensijų skaičias, mažesniais nei 13, klasifikavimo perceptronas gauna šiek tiek didesnę klaidą nei su likusiais dimensijų skaičiais. Taip turbūt atsitinka dėl panašių priežąsčių, kaip ir pirmame tyrime su dimensijų skaičiais nuo vieno iki trijų - kompresija praranda dalį svarbios duomenų informacijos. To nesimato pirmame eksperimente, kadangi skirtumai gan maži - blogiausias tyrimas pasiektas su 5 dimensijomis, kurio metu gauta klaida apie $1 - 0.965 = 0.035$. Tuo tarpu praeitame eksperimente, kuriame buvo ištirti dar mažesni dimensijų skaičiai, blogiausias tyrimas pasiektas su viena dimensija, kurio klaida apie $1 - 0.943 = 0.057$ beveik dvigubai didesnė. Tai didelis skirtumas žinant, kad pirmame eksperimente klaidos skaičiavimui buvo naudojami visi apmokymui naudoti duomenys.

Optimaliausias šio eksperimento dimensijų skaičius - 18.

TODO: Apie didesnes, optimalias dimensijas, palyginant su nekompresuotais

TODO: 18 koord. optimalu

Rezultatai ir išvados

Rezultatų ir išvadų dalyje išdėstomi pagrindiniai darbo rezultatai (kažkas išanalizuota, kažkas sukurta, kažkas įdiegta), toliau pateikiamos išvados (daromi nagrinėtų problemų sprendimo metodų palyginimai, siūlomos rekomendacijos, akcentuojamos naujovės). Rezultatai ir išvados pateikiami sunumeruotų (gali būti hierarchiniai) sąrašų pavidalu. Darbo rezultatai turi atitikti darbo tikslą.

TODO: Suprogramuotas neuroninis tinklas

TODO: Išanalizuoti duomenys

TODO: Išbandyti skirtingi dimensiškumo mažinimo metodai

TODO: Išbandyti skirtingi klasifikavimo

TODO: Išbandyti skirtingi dimensiškumo mažinimo ir klasifikavimo apjungimo metodai

Šaltiniai

[HC94] J.C. Harsanyi ir Chein-I Chang. Hyperspectral image classification and dimensionality reduction: an orthogonal subspace projection approach. *Geoscience and remote sensing, ieee transactions on*, 32(4):779–785, 1994-07. ISSN: 0196-2892. DOI: 10.1109/36.298007.

[YL02] Chien-Cheng Yu ir Bin-Da Liu. A backpropagation algorithm with adaptive learning rate and momentum coefficient. *Neural networks, 2002. ijcnn '02. proceedings of the 2002 international joint conference on*. Tom. 2, 2002, p. 1218–1223. DOI: 10.1109/IJCNN.2002.1007668.

[LG00] S. Lawrence ir C.L. Giles. Overfitting and neural networks: conjugate gradient and backpropagation. *Neural networks, 2000. ijcnn 2000, proceedings of the ieee-inns-enns international joint conference on*. Tom. 1, 2000, 114–119 vol.1. DOI: 10.1109/IJCNN.2000.857823.

[RHM97] Yong Rui, T.S. Huang ir S. Mehrotra. Content-based image retrieval with relevance feedback in mars. *Image processing, 1997. proceedings., international conference on*. Tom. 2, 1997-10, 815–818 vol.2. DOI: 10.1109/ICIP.1997.638621.