Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matej Klemen

Izboljšave metode globokih naključnih gozdov

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: prof. dr. Marko Robnik Šikonja

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Na več področjih strojnega učenja, predvsem pri obdelavi slik, besedil in zaporedij, so globoke nevronske mreže v zadnjem času po uspešnosti presegle klasične modele. Kot alternativo temu pristopu, ki zahteva mnogo učnih primerov in zmogljive računalnike, je nastala metoda globokih naključnih gozdov, ki v kaskade združuje uspešne in robustne modele naključnih gozdov. Implementirajte metodo globokih naključnih gozdov in preizkusite nekaj potencialnih izboljšav. Metodo ovrednotite na več podatkovnih množicah.

Zahvaljujem se mentorju prof. dr. Marku Robniku Šikonji za strokovno pomoč in podporo pri pisanju naloge. Za omogočen dostop do strežnika za testiranje algoritmov gre zahvala Laboratoriju za kognitivno modeliranje. Poleg tega bi se rad zahvalil svoji družini ter prijateljem, ki so mi vedno stali ob strani.

Kazalo

Povzetek

Naslov: Izboljšave metode globokih naključnih gozdov

Avtor: Matej Klemen

Na področju globokega učenja je vodilna metoda globokih nevronskih mrež. Te za učenje potrebujejo veliko podatkov, njihova uspešnost pa je odvisna od uporabljenih parametrov. Ena izmed alternativ je model globokih naključnih gozdov. V delu s svojo implementacijo algoritma globokih naključnih gozdov preverimo ponovljivost rezultatov originalnega članka (Zhou in Feng, 2017). Raziščemo, ali lahko napovedno točnost modela izboljšamo z dodajanjem gozdov naključnih podprostorov ali z uporabo zlaganja za kombiniranje napovedi modelov zadnjega nivoja kaskade gozdov.

Osnovni implementaciji ter implementacijo z našimi dodatki testiramo na petih podatkovnih množicah in predstavimo dosežene rezultate. Z zlaganjem na vseh podatkovnih množicah dosežemo enako oziroma boljšo povprečno napovedno točnost. Gozdovi naključnih podprostorov na treh podatkovnih množicah dosežejo slabše, na dveh pa boljše rezultate od osnovnega algoritma.

Ključne besede: globoko učenje, globoki naključni gozdovi, ansambli, zrnato skeniranje, kaskada gozdov, zlaganje modelov.

Abstract

Title: Improving deep random forests

Author: Matej Klemen

The most frequently used deep learning models are deep neural networks. Although they have been successfully applied to various problems, they require large training sets and careful tuning of parameters. An alternative to deep neural networks is the deep forest model, which we independently implemented to verify the replicability of results in (Zhou and Feng, 2017). We test if the accuracy of deep forest can be improved by including random subspace forests or by using stacking to combine predictions of cascade forest's last layer.

We evaluate the original implementation and our improvements on five data sets. The algorithm with added stacking achieves equal or better results on all five data sets, whereas the addition of random subspace forests brings worse results on three data sets and better results on two data sets.

Keywords: deep learning, deep forest, ensemble methods, multi-grained scanning, cascade forest, stacking.

Poglavje 1

Uvod

Živimo v času, v katerem zbiramo vse več podatkov, poleg tega pa imamo na voljo tudi veliko količino zgodovinskih podatkov. Velika količina podatkov v kombinaciji z večanjem zmogljivosti računskih enot je v zadnjem času omogočila hiter razvoj tehnik strojnega učenja. Predvsem so popularne postale tehnike globokega učenja - področja strojnega učenja, ki učenje kompleksnih konceptov doseže z nivojsko hierarhijo učenja. Začetni nivoji modela se naučijo preprostih konceptov, naslednji nivoji pa se s kombiniranjem že naučenih konceptov naučijo kompleksnejših zakonitosti [4].

Izmed tehnik globokega učenja je najbolj popularna uporaba modelov globokih nevronskih mrež, ki so uporabni na različnih problemskih domenah [10] [8] [5]. Kljub uspehom globokih nevronskih mrež pa imajo tudi nekaj slabosti. Ker so mreže sestavljene iz več nivojev, imajo mnogo uteži, ki jih je potrebno "pravilno" nastaviti - tako, da se model čim bolje nauči konceptov iz učnih podatkov in jih zna aplicirati na nove, še ne videne podatke. Da je to mogoče, potrebujejo za učenje veliko podatkov. Drug problem pri globokih nevronskih mrežah je veliko število parametrov (nastavitvenih opcij). S pravilnim nastavljanjem parametrov lahko velikokrat izboljšamo uspešnost napovednega modela, vendar lahko veliko število parametrov obenem oteži primerjavo napovednih modelov.

Ena izmed predlaganih alternativ globokim nevronskim mrežam je mo-

del globokih naključnih gozdov (imenovan tudi gcForest), ki nivoje gradi na osnovi naključnih gozdov. Ta na večjih podatkovnih množicah dosega rezultate, ki so konkurenčni rezultatom globokih nevronskih mrež, na manjših podatkovnih množicah pa dosega boljše rezultate [20]. V primerjavi z globokimi nevronskimi mrežami ima relativno malo parametrov, obenem pa je model zmožen samodejno določiti optimalno število nivojev, kar pomeni, da nam ni treba definirati vsakega nivoja posebej. Algoritem je bil od objave (leta 2017) uporabljen na problemih, kot je na primer detekcija prekrivajočih se zvokov [18].

Ko je govora o globokem učenju, se največkrat omenjajo različni tipi globokih nevronskih mrež. Kljub temu obstajajo različni poskusi kreiranja globokih struktur na osnovi "klasičnih" algoritmov strojnega učenja - poleg globokih naključnih gozdov [20], ki so tema tega dela, omenimo še globoko mrežo podpornih vektorjev [9]. Grajenje nivojev, sestavljenih iz naključnih gozdov, uporablja tudi arhitektura Forward Thinking Deep Random Forest [13]. Predstavljena struktura je podobna kaskadi gozdov v gcForest, gradnja novega nivoja pa je odvisna zgolj od izhoda prejšnjega nivoja strukture, ne pa tudi od prvotnih vhodnih podatkov.

Z diplomskim delom raziščemo globoke naključne gozdove in preverimo, če je rezultate, navedene v članku avtorjev algoritma [20], možno ponoviti. Poleg tega preizkusimo, če lahko z dodajanjem nove vrste gozdov izboljšamo točnost algoritma globokih naključnih gozdov. V ta namen smo razvili svojo implementacijo algoritma, ki vsebuje implementacijo osnovnega algoritma in naših idej v programskem jeziku python 3.

Diplomsko delo vsebuje pet poglavij. V 2. poglavju pričnemo z opisom nekaterih konceptov, ki so pomembni za razumevanje dela. V 3. poglavju opišemo našo implementacijo globokih naključnih gozdov in ideje za izboljšanje algoritma. V 4. poglavju preverimo, če so rezultati, navedeni v originalnem članku, ponovljivi z našo implementacijo algoritma in predstavimo rezultate algoritma z našimi izboljšavami. V 5. poglavju na kratko povzamemo opravljeno delo in podamo ideje za nadaljnje delo.

Poglavje 2

Ansambli odločitvenih dreves

V tem poglavju najprej predstavimo koncepte, ki so pomembni za razumevanje algoritma globokih naključnih gozdov - **odločitvena drevesa**, **naključne** in **izjemno naključne gozdove** kot osnovo za učenje iz podatkov ter **zlaganje** kot način kombiniranja napovedi posameznih modelov. Na koncu predstavimo globoke naključne gozdove, ki so glavna tema tega dela.

Ker se v diplomskem delu ukvarjamo s klasifikacijskimi problemi, pri opisih v večini primerov izpustimo prilagoditve algoritmov za regresijske probleme.

2.1 Odločitveno drevo

Odločitveno drevo (angl. *decision tree*) je algoritem, ki z drevesno strukturo predstavlja relacije med vhodnimi atributi in izhodom.

Algoritem kot vhod prejme množico n-dimenzionalnih vhodnih primerov $X = \{x^{(0)}, x^{(1)}, \ldots, x^{(m-1)}\}$ ter pripadajočo množico izhodov $y = \{y^{(0)}, y^{(1)}, \ldots, y^{(m-1)}\}$. Izmed n vhodnih atributov glede na kriterij ocenjevanja pomembnosti atributov izbere najboljši atribut ter prag za delitev množice. Nato algoritem vhodno množico (in pripadajoče izhode) razdeli na dve podmnožici glede na izbran atribut ter postopek rekurzivno ponavlja, dokler ne doseže ustavitvenega pogoja. Ustavitveni pogoj ni enolično določen, po-

gosto uporabljena kriterija pa sta na primer doseganje "čiste" podmnožice (kjer vhodni primeri pripadajo izhodom zgolj enega razreda) in preseganje največje (v naprej določene) globine drevesa [15].

Za ocenjevanje pomembnosti atributov obstaja mnogo kriterijev, kot so na primer Gini indeks, informacijski prispevek ter razmerje informacijskega prispevka [15]. Tukaj omenimo zgolj Gini indeks, ki je definiran z enačbo 2.1, v kateri C predstavlja množico razredov, prisotnih v trenutni podmnožici izhodov.

$$Gini\ indeks = \sum_{c \in C} p_c \cdot (1 - p_c) \tag{2.1}$$

Dobra lastnost odločitvenih dreves je, da jih je preprosto vizualizirati in s pomočjo vizualizacije interpretirati odločitve drevesa. Njihova slabost pa je nestabilnost - majhne spremembe podatkov lahko povzročijo gradnjo povsem drugačnega drevesa [6].

2.2 Naključni gozdovi

Eden izmed načinov za zmanjšanje nestabilnosti odločitvenih dreves je uporaba naključnih gozdov (angl. *random forest*) [1].

Ideja naključnih gozdov je v gradnji večjega števila med seboj čim manj koreliranih dreves in združevanju njihovih napovedi v končno napoved. Bolj specifično, algoritem kot vhod prejme množico n-dimenzionalnih vhodnih primerov $X = \{x^{(0)}, x^{(1)}, \ldots, x^{(m-1)}\}$, pripadajočo množico izhodov $y = \{y^{(0)}, y^{(1)}, \ldots, y^{(m-1)}\}$ in število dreves T. Za vsakega izmed dreves naključno (s ponavljanjem) izbere vzorec vhodnih primerov velikosti m, na katerem nato zgradi drevo. Gradnja drevesa poteka podobno kot je bilo opisano v poglavju 2.1 - razlika je zgolj v vsebini množice atributov, izmed katerih se izbira najboljšega. Namesto, da algoritem pred vsako delitvijo podatkov poišče najboljšega izmed vseh n atributov, najprej naključno izbere podmnožico $k \leq n$ atributov in nato izmed izbranih atributov izbere najboljšega (ter najboljši prag delitve).

Napoved za nov primer je sestavljena iz napovedi posameznih dreves, kombiniranih z glasovalno funkcijo. V primeru klasifikacije je to izbor večinske napovedi, kar pomeni, da je končna napoved tista, ki je najbolj pogosta med napovedmi posameznih dreves [6].

Naključni gozdovi v praksi dosegajo dobre rezultate, a imajo to slabost, da je njihove odločitve težko interpretirati ter vizualizirati [14].

2.3 Izjemno naključni gozdovi

Še korak dlje pri uporabi naključnosti pri gradnji dreves gre algoritem izjemno naključnih gozdov, ki gozd zgradi iz izjemno naključnih dreves (angl. *extremely randomized trees*) [3].

Algoritem za gradnjo izjemno naključnih dreves se od že opisanih algoritmov razlikuje v dveh podrobnostih:

- namesto uporabe naključnega vzorca podatkov (izbranega s ponavljanjem), se pri gradnji izjemno naključnih dreves uporabi celotna množica podatkov,
- poleg tega, da algoritem ob vsaki delitvi množice pri gradnji drevesa naključno izbere $k \leq n$ atributov, za vsakega izmed izbranih atributov še naključno izbere eno pragovno vrednost (izmed vrednosti, ki jih zavzema posamezen atribut) [3].

Izjemno naključne gozdove, kjer je $k = 1$, v nadaljevanju omenjamo pod imenom **povsem naključni gozdovi** (angl. *completely random forests*). V tem primeru se pri vsakem deljenju podatkovne množice pri gradnji drevesa povsem naključno (ne glede na kriterij pomembnosti, kot je na primer Gini indeks) izbereta naključni atribut ter naključni prag za izbran atribut.

2.4 Zlaganje

Pri opisu algoritma naključnih gozdov smo omenili, da se za kombini-

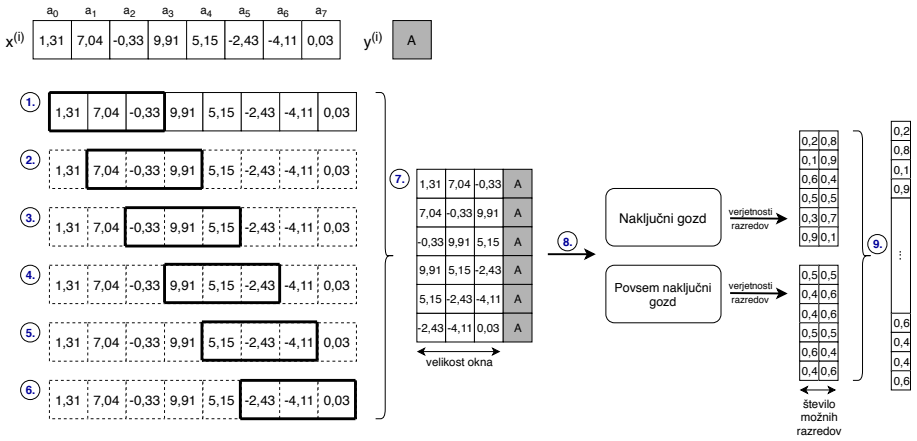
ranje napovedi posameznih dreves uporablja glasovalna funkcija. Pri klasifikacijskih problemih je ta funkcija ponavadi izbor večinske napovedi, pri regresijskih problemih pa povprečje napovedi posameznih dreves. Zlaganje (angl. *stacking*) je še ena metoda, ki je namenjena združevanju napovedi posameznih modelov [17].

Metoda zlaganja kot vhod dobi napovedi posameznih modelov ter pravilne napovedi. Na teh podatkih zgradi nov model, ki ugotovi, kako povezati posamezne napovedi tako, da bo skupna napoved čim bližje pravilni napovedi. Da ne bi povzročili prevelikega prileganja učnim podatkom, so vhodne napovedi pridobljene s prečnim preverjanjem [14].

2.5 Globoki naključni gozdovi

Vsi do sedaj omenjeni koncepti igrajo pomembno vlogo v algoritmu globokih naključnih gozdov [20]. Ta je sestavljen iz dveh delov - zrnatega skeniranja vhoda (angl. *multi-grained scanning*) ter kaskade gozdov (angl. *cascade forest*). Prvi del avtomatsko kreira atribute, ki smiselno predstavljajo koncepte v vhodnih podatkih, drugi pa se iz ustvarjenih atributov nauči zakonitosti v podatkih.

Slika 2.1 prikazuje potek zrnatega skeniranja za en vhodni primer. Zato, da poenostavimo primer, slika prikazuje proces za binarno klasifikacijo. Na začetku algoritem z uporabo drsečega okna izreže dele vhodnih podatkov in si zabeleži pripadajoči razred. V naslednjem koraku se s k-kratnim prečnim preverjanjem pridobi vektorje verjetnosti razredov za vsak primer v novo nastali podatkovni množici. Natančneje, podatkovna množica se razdeli v k skupin, nato pa se v vsaki iteraciji $k - 1$ skupin uporabi za gradnjo naključnega in povsem naključnega gozda, za 1 skupino pa se z zgrajenima gozdovoma pridobi napovedi verjetnosti razredov. Nato se na isti (celotni) podatkovni množici zgradita naključni in povsem naključni gozd, ki se shranita za kasnejšo uporabo (za transformiranje novih primerov). Na koncu se vektorje z napovedmi verjetnosti za posamezne primere "splošči" v en vektor,



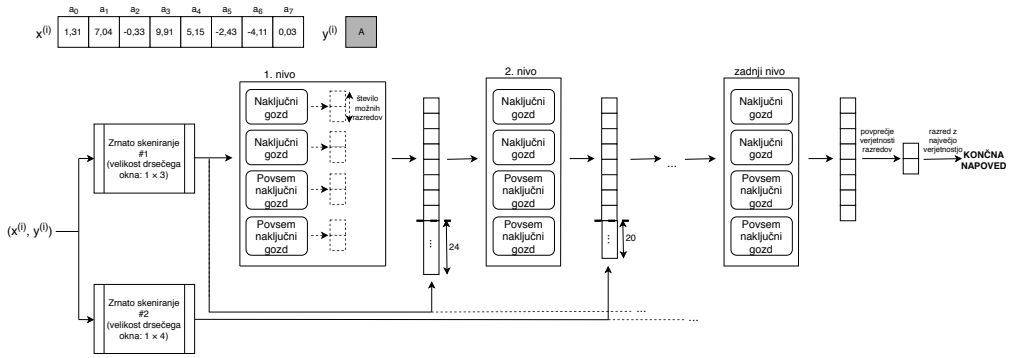
Slika 2.1: Primer zrnatega skeniranja z uporabo 1-dimenzionalnega drsečega okna: (1. - 7.) ekstrakcija delov vhodnega primera z drsečim oknom, (8.) pridobivanje vektorjev verjetnosti razredov s prečnim preverjanjem, (9.) preoblikovanje napovedanih verjetnosti v skupen vektor.

ki vsebuje ustvarjeno predstavitev konceptov v vhodnem primeru.

Opisani postopek lahko ponovimo tudi večkrat, vsakič z drugačno velikostjo drsečega okna, v nekaterih primerih (na primer takrat, ko so vhodni atributi konstruirani ročno) pa je bolj smiselno postopek izpustiti in kaskado zgraditi na prvotni, neprocesirani podatkovni množici. Z vsako ponovitvijo zrnatega skeniranja nastane nova različica prvotne podatkovne množice, ki drugače predstavi lastnosti vhodnih podatkov.

Postopku zrnatega skeniranja sledi gradnja kaskade gozdov, kjer se iz novo nastalih transformiranih podatkov zgradijo nivoji, sestavljeni iz naključnih in povsem naključnih gozdov. Primer postopka gradnje kaskade gozdov za en vhodni primer je prikazan na sliki 2.2. V tem primeru smo določili, da se v vsakem nivoju zgradita 2 naključna in 2 povsem naključna gozdova. Pred gradnjo kaskade pa sta bili izvedeni dve ponovitvi zrnatega skeniranja - prvič z drsečim oknom velikosti 1×3, drugič pa 1×4. Tudi v tem primeru predpostavimo, da gre za postopek binarne klasifikacije.

Vhodni podatki za gradnjo prvega nivoja kaskade so bodisi rezultat prvega izvajanja zrnatega skeniranja bodisi neprocesirani vhodni podatki (če je



Slika 2.2: Primer gradnje kaskade gozdov in uporabe zgrajene kaskade za napoved novega vhodnega primera pri binarni klasifikaciji.

bil postopek zrnatega skeniranja izpuščen). Iz teh podatkov se s prečnim preverjanjem (na enak način kot pri zrnatem skeniranju) za vsak model v nivoju pridobi napovedi verjetnosti razredov za vsak primer v podatkovni množici. V primeru, prikazanem na sliki 2.2, se na ta način pridobi 8-dimenzionalen vektor (sestavljen iz napovedi 4 modelov za primer binarne klasifikacije). Nato se vsi gozdovi v trenutnem nivoju ponovno zgradijo, tokrat na vseh podatkih, ki so bili podani kot vhod trenutnemu nivoju. Po gradnji novega nivoja se oceni klasifikacijska točnost celotne kaskade. Če je točnost kaskade z novo zgrajenim nivojem višja kot točnost kaskade brez novega nivoja, se postopek nadaljuje. V tem primeru se vektorji, ki vsebujejo napovedane verjetnosti razredov, združijo z:

- izhodom $(1 + i_L \ mod \ N_L)$-te ponovitve zrnatega skeniranja, kjer je i_L zaporedna številka trenutnega nivoja kaskade, N_L pa število vseh ponovitev zrnatega skeniranja, ali

- s prvotnimi, neprocesiranimi vhodnimi podatki (če je bil postopek zrnatega skeniranja izpuščen).

Dodajanje novih nivojev se nadaljuje, dokler uspešnost kaskade z novo dodanim nivojem ne začne upadati. Takrat se zadnji dodani nivo odstrani iz kaskade, s tem pa se postopek gradnje kaskade zaključi. Končna napoved

za posamezen primer se iz kaskade dobi tako, da se povprečijo napovedane verjetnosti posameznih modelov v zadnjem (veljavnem) nivoju kaskade in vzame razred z največjo povprečno verjetnostjo [20].

Rezultati, navedeni v originalnem članku [20], kažejo, da globoki naključni gozdovi na večini podatkovnih množic dosegajo konkurenčne ali boljše rezultate od globokih nevronskih mrež. Globoke nevronske mreže dosegajo občutno boljše rezultate na podatkovni množici CIFAR-10 [11]. Algoritem globokih naključnih gozdov pri vseh opravljenih poskusih avtorjev doseže boljše rezultate od klasičnih algoritmov strojnega učenja.

Poglavje 3

Implementacija algoritma in opis novosti

Algoritem globokih naključnih gozdov smo implementirali v programskem jeziku python 3. V implementaciji uporabljamo knjižnico NumPy, ki omogoča učinkovito delo z vektorji in matrikami, ter knjižnico scikit-learn, ki vsebuje metode strojnega učenja. Poleg osnovnih gradnikov, opisanih v prejšnjem poglavju, naša implementacija vsebuje dve novosti, opisani v nadaljevanju poglavja. Izvorna koda implementacije je prosto dostopna na spletu[1].

Prva novost je uporaba zlaganja za združevanje napovednih verjetnosti posameznih modelov zadnjega nivoja kaskade gozdov v končno napoved razreda. S tem preprostejši postopek povprečenja nadomestimo z algoritmom, ki na podlagi podatkov natančneje določi uteži posameznih napovedi. Pri zlaganju uporabimo model logistične regresije.

Naša implementacija omogoča uporabo novega modela na osnovi odločitvenih dreves - gozdove naključnih podprostorov [7]. Model je na voljo za uporabo tako pri postopku zrnatega skeniranja, kot tudi pri gradnji kaskade gozdov.

Gozdovi naključnih podprostorov so primer ansambelskih algoritmov. Se-

[1] `https://github.com/matejklemen/deep-rf`

stavljeni so iz dreves naključnih podprostorov. Gradnja dreves naključnih podprostorov je podobna gradnji navadnih odločitvenih dreves, ki je opisana v poglavju 2.1. Dodaten korak, ki nastopi pred gradnjo dreves naključnih podprostorov, je izbira naključnega podprostora. To pomeni, da izmed n možnih atributov naključno (s ponavljanjem) izberemo $d < n$ atributov, nato na celotni podatkovni množici in izbrani podmnožici d atributov zgradimo odločitveno drevo. Pri napovedi novega primera vsako drevo napove verjetnosti, da primer pripada posameznemu razredu. Končne napovedane verjetnosti so povprečne napovedane verjetnosti vseh dreves, končni napovedani razred pa je tisti, ki doseže največjo povprečno verjetnost [7].

Poglavje 4

Poskusi in rezultati

V tem poglavju predstavimo rezultate testiranja naše implementacije globokih naključnih gozdov in predstavljenih dodatkov. Najprej na kratko predstavimo uporabljene podatkovne množice in parametre algoritma. Nadaljevanje vsebuje rezultate, ki so razdeljeni na dva dela. Prvi del predstavlja primerjavo naše implementacije algoritma z implementacijo avtorjev globokih naključnih gozdov [20]. S tem preverimo, ali so rezultati, navedeni v njihovem članku, ponovljivi. Drugi del predstavlja rezultate testiranja novosti, ki smo jih vključili v našo implementacijo.

Rezultati naše implementacije so pridobljeni na 50 ponovitvah izvajanja in so predstavljeni s povprečjem in standardnim odklonom točnosti. Navedemo tudi p-vrednosti, pridobljene z Wilcoxonovim testom predznačenih rangov. Te vrednosti uporabimo za preverjanje, če so razlike v doseženih točnostih posameznih modelov statistično značilne. Za odločanje o statistični značilnosti pojava uporabimo stopnjo značilnosti $\alpha = 0{,}05$.

4.1 Uporabljeni podatki in parametri

Pri testiranju uporabljamo pet podatkovnih množic, in sicer YEAST, ADULT, LETTER [2], ORL [16] in MNIST [12]. Prve tri množice predstavljajo primere nizkodimenzionalnih podatkov, kjer pri testiranju ne upo-

rabljamo postopka zrnatega skeniranja, saj bi s tem uničili informativne atribute. Preostali dve množici predstavljata visokodimenzinalne podatke (slike), kjer posamezni atributi, ki predstavljajo intenzivnost slikovnih pik, ne predstavljajo informativnih lastnosti podatkov. Da iz teh podatkov pridobimo uporabne atribute, uporabimo zrnato skeniranje.

Pri testiranju podatke razdelimo podobno, kot so to naredili avtorji originalnega članka [20]. Zaradi časovnih omejitev namesto celotne podatkovne množice MNIST uporabimo vzorec s 15000 primeri - učna množica vsebuje 10000 primerov, testna množica pa preostalih 5000 primerov. Na podoben način razdelimo množico LETTER - 16000 primerov gre v učno, 4000 pa v testno množico. Množica ORL vsebuje slike obrazov 40 oseb (10 slik za vsako osebo). Pri testiranju množico razdelimo tako, da 5 slik vsake osebe uporabimo za treniranje modela (skupaj 200 slik), preostalih 5 slik pa uporabimo v testni množici. Množico YEAST razdelimo tako, da 1038 primerov (70%) uporabimo za učenje, 446 (30%) pa za testiranje. Množica ADULT je že podana v obliki ločenih množic - učna množica vsebuje 32561 primerov, testna pa 16281 primerov.

Parametri, ki jih uporabljamo pri testiranju, so navedeni v tabeli 4.1. Za določanje optimalnega števila nivojev kaskade gozdov namesto validacijske množice uporabljamo 3-kratno prečno preverjanje. Parametri so izbrani tako, da so čim bolj podobni tistim, ki so bili uporabljeni v originalnem članku [20], saj s tem lažje primerjamo implementaciji. Obenem želimo preveriti, če lahko tudi brez natančnega nastavljanja parametrov dosežemo dobre rezultate na različnih podatkovnih množicah.

Posebnosti, ki veljajo zgolj za posamezne poskuse, opišemo v pripadajočih poglavjih.

4.2 Primerjava implementacij

Tabela 4.2 prikazuje povprečne klasifikacijske točnosti ter pripadajoče standardne odklone obeh implementacij na petih domenah. Ker v članku

Tabela 4.1: Parametri, uporabljeni pri testiranju.

Parameter	**Uporabljena vrednost**
Velikosti drsečih oken pri zrnatem skeniranju	$\{\lfloor n/16 \rfloor, \lfloor n/8 \rfloor, \lfloor n/4 \rfloor\}$; $n \ldots$ število atributov
Število gozdov (posameznega tipa) v nivoju kaskade gozdov	4
Število gozdov (posameznega tipa) v posamezni iteraciji zrnatega skeniranja	1
Število dreves v posameznem gozdu	500
Število skupin pri prečnem preverjanju	3

avtorjev algoritma globokih naključnih gozdov [20] ni bil naveden način testiranja, smo testiranje na čim bolj podoben način kot za našo implementacijo opravili tudi za njihovo.

Če primerjavo algoritmov opravimo na vsaki domeni ločeno, ne moremo statistično potrditi, da sta implementaciji ekvivalentni. Sklepamo, da so razlike posledica različnih odločitev pri implementaciji algoritma in naključnosti, ki je del zasnove algoritma. V naši implementaciji se na primer gradnja kaskade gozdov zaključi, ko se validacijska točnost 1 iteracijo ne zviša, implementacija avtorjev gcForest pa je glede tega manj striktna - dovoljuje na primer 4 zaporedne iteracije brez zvišanja validacijske točnosti, preden se gradnja zaključi. Ta podrobnost v implementaciji včasih omogoči premagovanje lokalnih ekstremov, a obenem občutno poveča čas gradnje kaskade gozdov.

Kljub temu, da obstajajo razlike za posamezne podatkovne množice, te

Tabela 4.2: Dosežene klasifikacijske točnosti obeh implementacij in p-vrednosti Wilcoxonovega testa pri ničelni hipotezi, da obe implementaciji dajeta enake rezultate.

Podatkovna množica	**Implementacija avtorjev gcForest**	**Naša implementacija**	**p-vrednost**
YEAST	62,68% (0,92%)	62,26% (0,98%)	0,05
LETTER	97,29% (0,08%)	97,51% (0,06%)	$\ll 0{,}01$
ADULT	85,93% (0,14%)	85,83% (0,13%)	$\ll 0{,}01$
MNIST (15K)	97,87% (0,04%)	97,74% (0,04%)	$\ll 0{,}01$
ORL	93,68% (0,82%)	94,10% (1,08%)	$\ll 0{,}01$

razlike praktično niso pomembne. Če algoritma primerjamo na vseh petih domenah skupaj in razlike testiramo z Wilcoxonovim testom, ne moremo zavrniti hipoteze, da sta algoritma ekvivalentna.

4.3 Evalvacija novosti

Tabela 4.3 prikazuje rezultate naše implementacije z dodanim zlaganjem. Za zlaganje uporabljamo model logistične regresije. Rezultati kažejo, da na dveh podatkovnih množicah (MNIST in ORL) dosežemo statistično značilne razlike v napovedni točnosti, a so te razlike minimalne. Sklepamo, da je temu tako, ker se že prej, v skoraj vsakem koraku algoritma globokih naključnih gozdov, uporablja zlaganje in so zato napovedi modelov že pravilno utežene. Ker novost prinese dodatno kompleksnost modela, je smiselnost njene uporabe odvisna od razpoložljivih virov. Če je cilj doseči čim boljše rezultate na račun nekaj daljšega časa treniranja, potem je ta dodatek smiselno uporabiti. Sicer (kadar je čas izvajanja algoritma kritičen faktor) uporabe dodatka ne priporočamo.

Tabela 4.4 predstavlja rezultate z dodanimi modeli gozdov naključnih

Tabela 4.3: Klasifikacijske točnosti, dosežene z globokimi naključnimi gozdovi z dodanim zlaganjem. Za lažjo primerjavo rezultatov so v tabeli prikazani tudi rezultati osnovne implementacije (brez novosti).

Podatkovna množica	**Osnovna implementacija**	**Dodano zlaganje**	**p-vrednost**
YEAST	62,26% (0,98%)	62,26% (0,73%)	0,66
LETTER	97,51% (0,06%)	97,53% (0,08%)	0,13
ADULT	85,83% (0,13%)	85,83% (0,14%)	0,81
MNIST (15K)	97,74% (0,04%)	97,78% (0,00%)	$\ll 0{,}01$
ORL	94,10% (1,08%)	94,35% (1,49%)	$\ll 0{,}01$

Tabela 4.4: Točnosti, dosežene z globokimi naključnimi gozdovi, ki imajo dodane modele gozdov naključnih podprostorov. Za lažjo primerjavo rezultatov so v tabeli prikazani tudi rezultati osnovne implementacije (brez novosti).

Podatkovna množica	**Osnovna implementacija**	**Dodani gozdovi naključnih podprostorov**	**p-vrednost**
YEAST	62,26% (0,98%)	62,50% (1,23%)	0,10
LETTER	97,51% (0,06%)	97,40% (0,07%)	$\ll 0{,}01$
ADULT	85,83% (0,13%)	86,53 % (0,13%)	$\ll 0{,}01$
MNIST (15K)	97,74% (0,04%)	97,73% (0,03%)	0,19
ORL	94,10% (1,08%)	94,01% (1,73%)	0,01

podprostorov. Pri testiranju uporabimo 1 dodaten gozd pri zrnatem skeniranju ter 4 dodatne gozdove naključnih podprostorov pri gradnji vsakega nivoja kaskade gozdov. Na dveh podatkovnih množicah (LETTER in ADULT) dosežemo statistično značilne razlike v napovedni točnosti - pri prvi slabše, pri drugi pa boljše. Rezultati namigujejo na to, da se dodani gozdovi naključnih podprostorov v večini primerov niso sposobni naučiti nobenih novih zakonitosti iz podatkov. Možen razlog za neuspeh je v tem, da so drevesa v zgrajenih gozdovih naključnih podprostorov morda korelirana z drevesi ostalih uporabljenih gozdov, kar pomeni, da delajo enake napake kot ostali gozdovi. Glede na dobljene rezultate ter dejstvo, da občutno povečajo časovno kompleksnost treniranja, uporabo dodatka gozdov naključnih podprostorov ne priporočamo.

Poskuse smo izvedli na sistemu z 8-jedrnim procesorjem (Intel Xeon E5-2630) in 32 GB pomnilnika. Osnovna implementacija je za zrnato skeniranje in gradnjo 5 nivojev kaskade gozdov na uporabljenem vzorcu množice MNIST potrebovala 5005 sekund, implementacija z dodanim zlaganjem 5184 sekund, implementacija z dodanimi gozdovi naključnih podprostorov pa 15407 sekund.

Poglavje 5

Zaključek

V okviru diplomskega dela smo ustvarili lastno implementacijo algoritma globokih naključnih gozdov in jo primerjali z implementacijo avtorjev originalnega članka [20]. Osnovni algoritem smo dopolnili z gozdovi naključnih podprostorov ter preizkusili uporabo zlaganja za kombiniranje napovedi posameznih modelov v zadnjem nivoju kaskade gozdov.

Z dodatkom zlaganja dosežemo enake ali boljše rezultate kot z osnovnim algoritmom, z dodatkom gozdov naključnih podprostorov pa na treh podatkovnih množicah dosežemo slabše rezultate, na dveh pa boljše. Glede na kompleksnost dodatkov in dosežene rezultate ugotavljamo, da je dodatek zlaganja smiselno uporabiti, če čas treniranja modela ni kritičen faktor. Dodatek gozdov naključnih podprostorov časovno kompleksnost občutno poveča, zato tega dodatka v večini primerov ni smiselno uporabiti.

Algoritem globokih naključnih gozdov je še relativno nov, zato je možnosti za nadaljnje raziskovanje veliko. Smiselno bi bilo preveriti, če kateri izmed preostalih ansambelskih algoritmov, ki jih nismo uporabili v tem delu, izboljša napovedno točnost globokih naključnih gozdov. Primer takega algoritma so gozdovi s pragovnimi konstrukti tipa X-izmed-N [19]. Ti so vključeni v našo implementacijo. Prvi rezultati algoritma s tem dodatkom so obetavni. Na vzorcu 25 ponovitev izvajanja na množici YEAST je algoritem dosegel višjo povprečno točnost ($62,97\%$) in nižji standardni odklon ($0,51\%$) od

osnovne implementacije. Poskusov na preostalih podatkovnih množicah nam zaradi časovnih omejitev ni uspelo opraviti.

V nalogi smo preizkusili uporabo zlaganja za združevanje napovedi modelov zadnjega nivoja kaskade gozdov. Smiselno bi bilo preizkusiti tudi uporabo zlaganja za kombiniranje napovedi istoležečih modelov v vseh nivojih kaskade gozdov. Tako bi na primer v kaskadi gozdov s 3 nivoji, kjer vsak nivo vsebuje 2 modela, dobili uteženi kombinaciji za vse prve modele in vse druge modele. Končna napoved bi bila bodisi utežena vsota bodisi povprečje dobljenih kombinacij.

Ker je bil algoritem razvit kot alternativa globokim nevronskim mrežam, mora biti zmožen učenja na velikih podatkovnih množicah. Modeli, ki so trenutno del globokih naključnih gozdov, ne podpirajo inkrementalnega učenja iz podatkov, kar omejuje uporabnost algoritma za učenje na velikih podatkovnih množicah. Da bi to omejitev odpravili, bi morali trenutno uporabljene algoritme zamenjati z njihovimi inkrementalnimi različicami.

Literatura

[1] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository. Dosegljivo: `http://archive.ics.uci.edu/ml`. Dostopano: 13. 9. 2018.

[3] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, volume 1. MIT press, Cambridge, 2016.

[5] Alex Graves, Abdel-Rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013*, pages 6645–6649. IEEE, 2013.

[6] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009.

[7] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8), August 1998.

[8] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with

convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[9] Sangwook Kim, Swathi Kavuri, and Minho Lee. Deep network with support vector machines. In *International Conference on Neural Information Processing*, pages 458–465. Springer, 2013.

[10] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, 2014.

[11] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[13] Kevin Miller, Chris Hettinger, Jeffrey Humpherys, Tyler Jarvis, and David Kartchner. Forward thinking: Building deep random forests. *arXiv preprint arXiv:1705.07366*, 2017.

[14] Kevin Murphy. *Machine learning: A probabilistic perspective*. Cambridge, MA, 2012.

[15] Lior Rokach and Oded Maimon. *Data mining with decision trees: theory and applications (2nd edition)*, volume 81. World scientific, 2nd edition, 2015.

[16] Ferdinando Samaria and Andy Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of the Second IEEE Workshop on Applications of Computer Vision, 1994.*, pages 138–142. IEEE, 1994.

[17] David Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[18] Chun-Yan Yu, Huang Liu, and Zi-Ming Qi. Sound event detection using deep random forest. Technical report, DCASE2017 Challenge, September 2017.

[19] Zijian Zheng. Constructing X-of-N attributes for decision tree learning. *Machine learning*, 40(1):35–75, 2000.

[20] Zhi-Hua Zhou and Ji Feng. Deep forest: Towards an alternative to deep neural networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, (IJCAI-17)*, pages 3553–3559, 2017.