УНИВЕРЗИТЕТ У БЕОГРАДУ

ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА

ЗАВРШНИ РАД

Јаков Петровић

Београд 2020.

УНИВЕРЗИТЕТ У БЕОГРАДУ

ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА

ИНФОРМАЦИОНИ СИСТЕМИ И ТЕХНОЛОГИЈЕ

ПОСЛОВНА ИНТЕЛИГЕНЦИЈА

ЗАВРШНИ РАД

АПРОКСИМАТИВНИ АЛОГОРИТАМ НАЈБЛИЖИХ СУСЕДА ЗА ВЕЛИКЕ ПОДАТКЕ

Јаков Петровић 2017/3029

Београд 2020.

**Сагласност чланова комисије за одбрану**

*Попуњавају чланови Комисје за одбрану:*

Комисија која је прегледала рад

кандидата Петровић (ГОРАН) ЈАКОВ

под насловом АПРОКСИМАТИВНИ АЛГОРИТАМ НАЈБЛИЖИХ СУСЕДА ЗА ВЕЛИКЕ ПОДАТКЕ и одобрила одбрану:

Ментор: др Милош Јовановић,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члан: др Милан Вукићевић,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члан: др Гордана Савић,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

БИОГРАФИЈА

Овде иде биографија

**Изјава о академској честитости**

*Попуњава кандидат:*

|  |  |
| --- | --- |
| Петровић Горан Јаков  Презиме, име једног родитеља, име | /3029  Број индекса |

**Студијски програм: Информациони системи и технологије**

**Модул:**Пословна интелигенција

**Аутор завршног рада под насловом:** **АПРОКСИМАТИВНИ АЛГОРИТАМ НАЈБЛИЖИХ СУСЕДА ЗА ВЕЛИКЕ ПОДАТКЕ**

Чија је израда одобрена на Седници Већа студијских програма мастер академских студија одржаној: 18. јуна 2018.

Потписивањем изјављујем:

* + Да је рад искључиво резултат мог сопственог истраживачког рада;
  + Да сам рад и мишљења других аутора које сам користио у овом раду назначио или цитирао у складу са Упутством;
  + Да су сви радови и мишљења других аутора наведени у списку литературе/референци који су саставни део овог рада и писани су у складу са Упутством;
  + Да сам довио све дозволе за коришћење ауторског дела који се у потпуносзи/целости уносе у предати рад и да сам то јасно навео;
  + Да сам свестан да је плагијат коришћење туђих радова у било ком облику (као цитата, парафраза, слика, табела, дијаграма, дизајна, планова, фотографија, филма, музике, формула, веб сајтова, компјутерских програма и сл.) без навођења аутора или представљање туђих ауторских дела као мојих, кажњиво по закону (Закон о ауторским и сродним правима, Службени гласник Републике Србије, бр. 104/2009, 99/2011, 119/2012), као и других закона и одговарајућих аката Универзитета у Београду и Факултета организационих наука;
  + Да сам свестан да плагијат укључује и представљање, употребу и дустрибуирање рада предавача или других студената као сопствених;
  + Да сам свестан последица које код доказаног плагијата могу проузроковати на предати завшни мастер рад и мој статус;
  + Да је електронска верзија завршног рада идентична штампаном примерку и пристајем на његово објављивање под условима прописаним актима Универзитета и Факултета.

Београд, \_\_\_\_\_\_\_\_\_\_

Потпис студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_

АПСТРАКТ

Алгоритам најближих суседа припада наједноставнијим предиктивним алгоритмима машинског учења. Класификацију или регресију, алгоритам врши на основу скупа најближих познатих инстанци. Претрага најближих суседа (енг. *Nearest neighbour search*) је временски најкритичнија фаза алгоритма. Фаза претраге је и рачунски веома захтевна, па утиче на скалабилност алгоритма при раду са већим количинама података. На скалбилност не утиче само количина података, већ и број димензија које поседују. Како би се побољшале преформансе алгоритма, уводе се апроксимативне методе претраге суседа. Одабране апроксимативне технике биће представљене и анализиране у раду. Експерименталним путем тестираће се преформансе апроксимативних алгоритама над скуповима података под различитим параметрима.

Кључне речи: апроксимативни к-НН, претрага најближих суседа, апроксимативна претрага најближих суседа, машинско учење

ABSTRACT

The algorithm of k-nearest neighbors is among the simplest predictive algorithms of machine learning. The algorithm performs classification and regression, based on a set of the nearest known instances. Nearest neighbor search is the most time-critical phase of the algorithm. The search phase is also very computationally demanding, so it affects the scalability of the algorithm when working with large amounts of data. Scalability is affected not only by the amount of data, but also by the number of dimensions they possess. Approximate neighbor search methods are introduced in order to improve the performance of the algorithm. Selected approximate techniques will be presented and analyzed in the paper. The performance of approximate algorithms over data sets under different parameters will be tested experimentally.

Key words: approximate kNN, nearest neighbour search, approximate nearest neighbour search, machine learning

АКРОНИМИ

|  |  |
| --- | --- |
| *k-NN* | к-најближих суседа (енг. *k-Nearest Neighbours*) |
| *k-D tree* | к-Д стабло (енг. *k-D tree*) |
| *vp-Tree* |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

ИЛУСТРАЦИЈЕ

[Илустрација 1: Апроксимативни алгоритам 9](#_Toc49360252)

[Илустрација 2: к-Д подела простора 13](#_Toc49360253)

[Илустрација 3: к-Д Бинарно стабло 13](#_Toc49360254)

ТАБЕЛЕ

Садржај

[1. Увод 7](#_Toc49360702)

[2. Дефиниција проблема и преглед стања у области 9](#_Toc49360703)

[3. Апроксимативне технике претраге суседа 13](#_Toc49360704)

[3.1 Подела простора и стабла 13](#_Toc49360705)

[3.1.1 к-Д стабла 14](#_Toc49360706)

[3.1.2 ВП-стабла 16](#_Toc49360707)

[3.1.3 Друге апроксимативне методе 17](#_Toc49360708)

[3.2 Графовски приступ 17](#_Toc49360709)

[3.2.1 HNSW 17](#_Toc49360710)

[4. АНАЛИЗА МЕТОДА 19](#_Toc49360711)

[5. ОРГАНИЗАЦИЈА И МЕТОДЕ ИСТРАЖИВАЊА 20](#_Toc49360712)

[6. Опис дела света који ће бити изучаван 23](#_Toc49360713)

[7. Сврха и циљеви истраживања 23](#_Toc49360714)

[8. РЕФЕРЕНЦЕ 23](#_Toc49360715)

# Увод

Рад се бави истраживањем апроксимативних техника за налажење најближих суседа над већим скуповима података. Претрага најближег суседа у метричком просотру подразумева проналажење инстанце који има најмању удаљеност од инстанце у односу на коју се претрага врши.

Претрага најближих суседа преставља значајан проблем у областима попут биомедицине, генетике, маркетинга, социјалних медија, машинског учења и оптимизације, при обради слика и снимака, препознавању образаца и облика те анализи и обради просторних и геолошких података. У већини случајева, записи из ових база се могу представити у векторском простору који има неколико десетина па чак и хиљаду димензија. (Indyk & Motwani, 1998). У раду се са истим значењем користе појмови тачка у простору и вектор у простору.

Алгоритам к- најближох суседа (енг. *k-nearest neighbours*) је предиктивни алгоритам машинског учења, који непознату класу или вредност инстанце, одређује на основу особина из скупа најближих познатих инстанци и широко је коришћен метод класификације и регресије.

Предвиђање будућих догађаја је један од кључних изазова са којима се суочавају друштва, предузећа и појединци. У складу са предвиђањима, ентитети могу предузети акције у циљу повећања прихода или смањења штете.

Хоће ли ће нови комитент банке бити у могућности да отплати кредит? Хоће ли ће пацијент имати проблем при уградњи зубног импланта? Који жанр музике ће бити предложен потенцијалном купцу?

Добијање благовремених резултата применом алгоритама машинског учења, уз ефикасно коришћење ресурса, је императив. Једноставност је врлина КНН алгоритма, али је може бити узрок слабијих перформанси код података са шумовима, или кад је потребно обрадити велики број димензија и инстанци.

Претрага суседа је кључна и временски најкритичнија фаза алгоритма. Из претходно наведеног, намеће се закључак да су класификација и регресија завршне и једноставније фазе које следе захтевнији и сложенији проблем претраге најближих суседа (енг. *Nearest neighbour search*). С тога проблем брзог налажења најблиших суседа и структура које ту претрагу омогућавају представља окосницу овог рада. Укорењено је мишљење да ће потпуна, односно егзактна претрага суседа увек бити спорија од апроксимативне претраге. Апроксимативна претрага подразумева проналажење суседа који могу, али и не морају бити најближи суседи инстанце претраге. Најзначајнија особина апроксимативног алгоритма је да постигне готово исту прецизност као оригинални приступ са потпуном претрагом, али за значајно краћи временски интервал.

Управо је та чињеница мотивисала су аутора рада на изучавање апроксимативних метода претраге суседа. Резултат истраживања ће приказати рад алгоритма, преглед и анализу његових апроксимативних варијанти. Утврдиће се разлика у перформансама апроксимативних техника над скуповима података различитих величина и димензија, на корист будућим истраживачима.

У другом поглављу бавимо се формалним дефинисањем појмова претраге најближих суседа и апроксимативне претраге суседа.

# Дефиниција проблема и преглед стања у области

Алгоритам к- најближих суседа (КНН) je предиктивни алгоритам машинског учења који непознату инстанцу класификује на основу најсличнијих познатих података. У формалном облику, проблем налажења најближег суседа се састоји у следећем:

Нека је дат скуп тачака у простору . Потребно је претражити скуп , како би се нашла тачка која је најближа задатој тачки то јест:

где је функција раздаљине метричког простора.

За разлику од алгоритама који испрва изводе генерализацију података и генеришу класификациони модел, те на основу модела брзо процењују припадност непознате инстанце одређеној класи (енг. *eager learners*), овај метод припада групи лењих алгоритама (енг. *lazy learners*). Одлика лењих алгоритама је да се инстанце просто складиште у фази тренирања или се извршава минимално претпроцесирање података. То значи да ће фаза тренирања алгоритма бити краћа, међутим предвиђање алгоритма ће трајати дуже. Како се фаза тренирања код ових алгоритама састоји у простом чувању инстанци, ова група алгоритама се назива и инстанцно-оријентисаном. Овај алгоритам je често веома рачунски интензиван и захтева ефикасне технике складиштења података, које треба да омогуће паралелно извршавање поступка проналажења суседа (Han, Pei, & Kamber, 2012).

Иако је проблем налажења суседа наизглед једноставан, постоје два изразита проблема у вези са скалабилношћу при раду са великом количином података:

* Време извршавања: Алгоритамска сложеност проналажења само једног најближег суседа је , где је број тачака у простору и број димензија. Алгоритам је рачунски захтевнији када је потребно пронаћи више суседа, што је најчешћи случај. Најзад, цео процес је потребно извести сваки пут када се појави инстанца коју треба класификовати.
* Меморија: За брзо израчунавање удаљености, неопходно је да је цео скуп постојећих тачака учитан у РАМ меморију, што може представљати изазов код великих база података. (Maillo, Ramirez, Triguero, & Herrera, 2017)

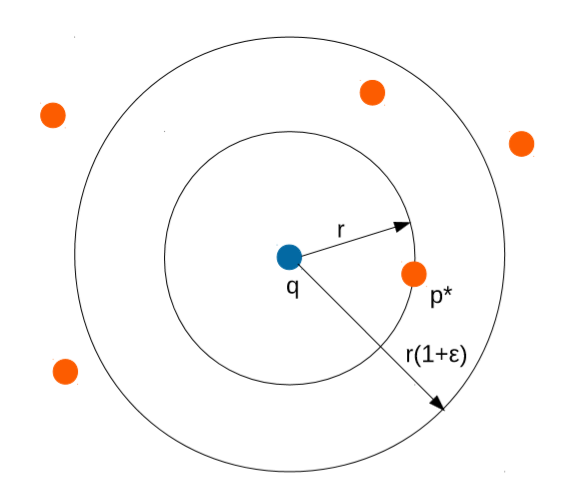
Ефикасно претраживање најближих суседа постаје учестали захтев према великом броју база података. Алгоритми који се директно ослањају на индексне структуре дају задовољавајуће резултате на малим или средњим количинама података, али не задовољавају потребе ефикасности када је реч o великим и високо димензионираним скуповима. (Seidl & Kriegel, 1998).

Очигледна тешкоћа да се осмисле алгоритми засновани на претрази суседа који ће бити ефикасни у случају када је број димензја већи (код неких аутора када је број димензија већи од 10), наводи да је потребно размотрити апроксимативне методе које ће проналазити приближно најближе суседе.

У том случају, можемо проширити претходно дату дефиницију:

Нека је дат скуп тачака у простору и нова тачка којој треба одредити суседа. За дато , тачка је приближан сусед ако је:

Где је најближи сусед задате тачке . У општем случају, када je , поступак се своди на проналажење приближних најближих суседа, где за сваког - тог важи да је апроксимација одговарајућег најближег суседа тачке (Arya, Mount, Netanyahu, Silverman, & Wu, 1998).



Илустрација 1: Апроксимативни алгоритам

Важно је напоменути да се међу имплементацијама апроксимативне претраге, ретко налазе оне, које дају гаранције да ће се пронађен сусед наћи у околини најближег суседа.

Проблем налажења суседа био је у пољу интересовања истраживача од педесетих година двадесетог века, међутим, област долази у фокус почетком деведесетих година када је објављено мноштво значајних радова из ове области. Према наводу (Kushilevitz, Ostrovsky, & Rabani, 2000), када је реч о веома димензионираним подацима, проблем је разматран од стране Добкина и Липтона (*Dobkin, D., & Lipton, R. J. (1976). Multidimensional searching problems*) који су генерисали алгоритам чија сложеност претраге експоненцијално зависи од броја димензија . Представљени метод је касније побољшан, али време извршења je још увек било у експоненцијалној зависности од димензије простора.

Крајем века, (Kleinberg, 1997) је представио два алгоритма који дају значајна побољшања у односу на дотадашња достигнућа. Први алгоритам нуди сложеност претраге од и други, чија сложеност тежи линеарној (Kleinberg, 1997). Алгоритми подразумевају пресликавање вектора у простор са нижим бројем димензија и креирање графовске структуре.

Другачији приступ решавању проблема донели су (Indyk & Motwani, 1998) и објавили метод који има полиномну сложеност у зависности од и , дакле . Идеја је да се пронађе метод којим ће се међусобно слични објекти сврставати у исте категорије. Хеширање на основу положаја или локацијски-сензитивно хеширање (енг. *Locality-sensitive hashing,* у даљем тексту ЛСХ) је назив представљене технике. Дефиниција коју је дао (Samet, 2006) даје ближе објашњење:

Циљ ЛСХ алгоритма је да се пронађе функција хеширања која ће сачувати информацију о удаљености инстанци, или приближне удаљености уз одређену толеранцију. Општи кораци ЛСХ алгоритма представљени су у раду под називом „Претраживање сличности при великом броју димензија путем хеширања“ (Gionis, Indyk, & Motwani, 1999).

Већина до данас представљених решења подразумева претпроцесирање постојећих инстанци, како би се над њима касније извршила претрага. Читав сет апроксимативних метода предвиђа преуређење скупа података тако што би се формирала структура стабла (*k-D tree*, *R-tree, VP-tree, SА-tree* и друга), чиме се обезбеђује убрзање алгоритма.

Друге методе предлажу хеширање инстанци како би се спровело груписање међусобно сличних инстанци.

При одабиру апроксимативног алгоритма, пожељно је размотрити време, сложеност, као и меморијске захтеве обраде података. Велики број аутора даје одличан теоријски допринос, међутим, недостатак многих радова огледа се у изостанку свеобухватне експерименталне анализе. Често се практична примена изводи на вештачки креираним скуповима података, а понекад изостаје упоредна анализа са изворним алгоритмом који се ослања на комплетну претрагу података.

Новија истраживања представљају решења проблема апроксимације ослањајући се на технологије за паралелизацију процеса и дистрибуиране рачунарске архитектуре. Неколико дистрибуираних алтернатива су предложене са циљем да омогуће апроксимативном алгоритму да обради велику количину података. Већина решења се базира на програмској парадигми *MapReduce* и његовој имплементацји отвореног кода – *Hadoop*. На тај начин се паралелизује извршавање алгоритма и ублажавају ефекти меморијских и рачунских ограничења. Недавно је представљен нови дизајн (Maillo, Ramirez, Triguero, & Herrera, 2017) који превазилази стандардне *Hadoop*-*MapReduce* приступе и обезбеђује флексибилну шему за класификацију великог броја непознатих инстанци над великим подацима, користећи *Apache Spark* архитектуру (Triguero, Maillo, Luengo, García, & Herrera, 2016).

Поменути радови објављени у прошлом веку инспирисали су истраживаче да осмисле нова, или дају значајна побољшања постојећих метода. Сва досадашња решења за претрагу суседа можемо оквирно сврстати у четири групе:

1. Приступи засновани на подели простора и стаблима
2. Графовски приступ
3. Приступ заснован на хеширању
4. Приступ заснован на векторској квантизацији

Подела је групо дефинисана и могуће је наћи примере који користе мешовите приступе попут (Douze, Sablayrolles, & Jégou, 2018) који комбинују графовске технике са приступима векторске квантизације. У наредним поглављима дати су описи неких представника ових група, који су имплементирани у програмским библиотекама отвореног кода.

# Апроксимативне технике претраге суседа

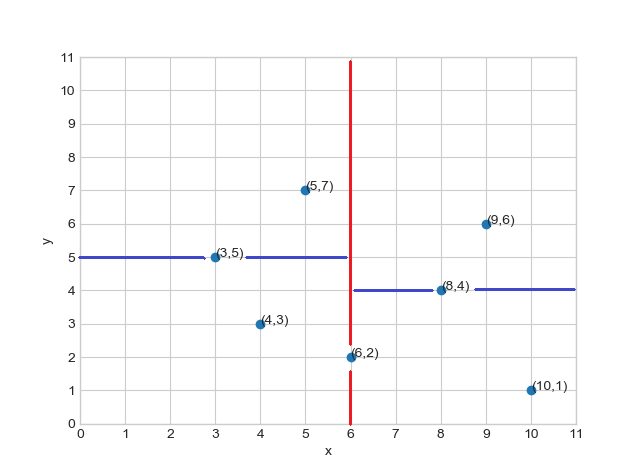
У претходном поглављу дат је осврт на најзначајније правце истраживања у области претраге суседа. У поглављима која следе даје се детаљан опис најзначајнијих предтавника сваке класе.

## Подела простора и стабла

Како би убрзали поступак налажења суседа и смањили обим претраге, истраживачи су се окренули техникама за поделу простора. У највећем броју случајева, ради се о бинарној подели. Бинарна подела простора подразумева рекурзивно дељење простора на два дела, све док је испуњен критеријум поделе. Најмањи подпростор добијен на овим путем се често назива ћелијом. Бинарно стабло је структура података која је погодна за организацију елемената из тако подељеног простора. Тачке сваког дела, постају елементи левог и десног подстабла, респективно. Исправно је претпоставити да ће се најближи суседи наћи у ћелији којој припада тачка претраге, или пак у суседним ћелијама. Један од најбитинијих фактора по којима се разврставају алгоритми ове класе, је метод поделе простора. Код најпознатијег представника групе, к-Д стабла, простор у сваком кораку дели хиперраван која је нормална на осу неке димензије к. Преставници ове групе су и РП – стабла (енг. *RP-tree)*, где је хиперраван насумично одређена, 2–М стабла (енг. *2-Means tree*) где се подела спроводи на основу алгоритма калстеризације, ПЦА-стабло (енг. *PCA-tree*) које простор дели на основу анализе главних компоненти, ВП- стабло (енг. *VP-tree*) и друга.

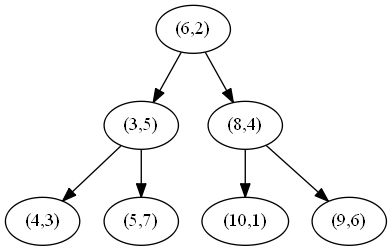
### к-Д стабла

к-Д стабло је један је од првих и најпознатијих предствника групе. Посматрајмо једноставан пример дводимензионалног простора (). У основном облику, алгоритам налази медијалну вредност међу кординатама одабране осе, у овом случају *x* осе. Медијална вредност ће дефинисати хиперраван. У случају када имамо две димензије, хипераван ће заправо бити права, која дели простор на два дела.



Илустрација 2: к-Д подела простора

Како је медијална вредност по *x* оси једнака 6, у првој итерацији, простор је подељен у односу на тачку са координатама (6,2) - која постаје корен стабла. Остале тачке из просотра постају чворови левог или десног подстабла, у завиности од положаја у односу на праву . У наредној итерацији простор се дели у односу на *y* осу за сваки претходно добијен подпростор. Праве и деле леви и десни подпростор, па тачка (3,5) постаје корен левог подстабла, а тачка (8,4) десног. Бинарно стабло добијено на овај начин представљено је на Илустрацији 3: к-Д Бинарно стабло.



Илустрација 3: к-Д Бинарно стабло

Алгоритам је лако проширити на случај када је број димензија *k* већи () . У свакој итерацији простор се дели у односу на димензију к. Услов за заустављање поделе може бити дат корз ораничавање броја нивоа које стабло боже имати, дефинисање минималног броја елемената који ће се наћи у листовима, или на неки други начин.

к-Д је егзактан метод, што значи да ће резултат претраге бити тачке који су прави суседи тачке претраге. Из тог разлога, алгоритам не проверава само саджај ћелије којој припада тачка претраге, већ и суседних ћелија уколико је потребно (Алгоритам 1.). Вратимо се на дводимензиони пример. Нека тачка претраге има координате (7,9). Претрага започиње од корена стабла који ћемо прогластити најближим. Како корени чвор није лист у стаблу, услов из 2. линије није задовољен. Алгоритам се наставља у десном подстаблу. Рекурзивним позивима долази се до чвора (9,6), који се налази у ћелији којој припада тачка претраге и који се проглашава најближим. Претрага се поново наставља у кореном чвору где се утврђује да је дистанца између тачке претраге и тачке (9,6), већа од удаљености од осе поделе (линија 7), па постоји могућност да се најближи сусед налази у левом подстаблу. На исти начин, алгоритам пролази кроз лево подстабло и налази правог најближег суседа (5,7).

|  |  |
| --- | --- |
| Алгоритам 1. Претрага у к-Д стаблу | |
| 1: | pretraga(q, n, p, d) |
| 2: | if n.l = ∅ {\displaystyle \varnothing } ∅ ∧ n.r = ∅ |
| 3: | dc ≔ dist(q,n) |
| 4: | if dc < d then p ≔ n; d≔dc; return p,d; |
| 5: | else |
| 6: | if q.ax ≤ n.ax |
| 7: | If q.ax – d ≤ n.ax then p,d ≔ pretraga(q, n.l, p, d) |
| 8: | If q.ax + d > n.ax then p,d ≔ pretraga(q, n.r, p, d) |
| 9: | if q.ax > n.ax |
| 10: | If q.ax + d > n.ax then p,d ≔ pretraga(q, n.r, p, d) |
| 11: | If q.ax – d ≤ n.ax then p,d ≔ pretraga(q, n.l, p, d) |

Читалац ће приметити да се егзактан метод лако може претворити у апроксимативни, тако што ће се ограничити број чворова који се посећује, односно спречити претрага суседних ћелија. У повољном случају, сложеност претраге к-Д стабла је , али се потврдило да сложеност може бити линеарна, што су потврдила и емпиријска истраживања (Дај референцу на истраживања) Метод је показао неефикасност када је број димензија већи (Marimont & Shapiro, 1979). Један од изазова је проналажење медијалне врености

### ВП-стабла

Једна од најпознатијих подврста Лопта-стабала (Ball-tree) je ВП-стабло (енг. *Vantage point tree*). Испрва је потребно из скупа тачака изабрати тачку која ће бити пивот - . Потом се израчунава медијална удаљеност осталих тачака у односу на пивот и скуп дели на два дела по принципу:

тако да су тачке из скупа унутар полупречника , док су остале тачке ван овог пречника. Рекурзивном применом овог правила гради се бинарно стабло где су унутрашњи чворови пивоти, чија су елементи левог и десног подстабла унутар и изван одговарајуће лопте (Samet, 2006). Пивот се може изабрати насумично, или на неки други начин. Један од предложених метода, од којег се очекује да резултира креирањем стабла које пружа добре преформансе, дефинисао је (Yianilos, 1993). Идеја је да се из насумично одабраног подскупа тачака пронађе она, која има најбољу расподелу удаљености у односу на друге тачке. Такав метод форсира избор оне тачке која је удаљена од средине, односно скрајнута на ивицама скупа. Поступак креирања стабла може се дефинисати једноставним алгоритмом:

|  |  |
| --- | --- |
| Алгоритам 2. Креирање ВП стабла | |
| 1: | kreirajVP() |
| 2: | if = ∅ {\displaystyle \varnothing } ∅ then return ∅ |
| 3: | = ∅ {\displaystyle \varnothing } ∅ |
| 4: | = ∅ {\displaystyle \varnothing } ∅ |
| 5: | n ≔ kreirajCvor() |
| 6: | n.pt ≔ izaberiPivot() |
| 7: | n.med ≔ Medijana(dist(n.pt, )) |
| 8: | For each |
| 9: | If dist(, n.pt) < n.med then .add() |
| 10: | If dist(, n.pt) ≥ n.med then .add() |
| 11: | n.l = kreirajVP() |
| 12: | n.l = kreirajVP() |
| 11: | return n |

Претрага се врши на сличан начин као код к-Д стабала, па и овде важи аналогија да се апроксимативна варијанта алгоритма може добити ограничавањем броја чворова који порцедура пролази.

### Друге апроксимативне методе

K-d- randomized

RP tree

I njihove forest verzije

## Графовски приступ

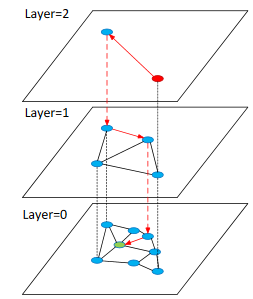
Овде иде мали преглед

### HNSW

HNSW се сврстава у алгоритме који користе графовске структуре за проналажење најближих суседа у метричком простору. Тачке у простору представљају чворове графа, а неусмерене гране престављају везе међу суседима. Алгоритам који врши претрагу структуре користи се и при додавању нових чворова. Наиме, алгоритам претражује структуру у потрази за најближим суседима, када су услови претраге задовољени, елемент постаје чвор графа са гранама које га везују за суседе.

Аутори представљају хијерархијски организовану структуру графова, за коју тврде да има логаритамску сложеност претраге. Главни чиниоци који доприносе логаритамској сложености су, експлицитна селекција улазног чвора у графовску струкуру и напредна хеуристика за брзо налажење суседа у оквиру структуре. Структура се назива хијерархијском јер је организована у више међусобно повезаних нивоа. На свим нивоима хијерархије формира се граф, са тим што се број чворова који формирају граф смањује са порастом нивоа. На најнижем нивоу, све тачке векторског простора формирају граф. На сваком наредном нивоу, граф ће формирати подскуп тачака из претходног нивоа. Формалнији опис претходно описане структуре може се дати на следећи начин:

Нека је дат скуп тачакa векторског простора и нека је дат скуп нивоа хијерархије . Све тачке скупа биће организоване у графовску структуру на најнижем нивоу хијерархије те можемо записати . За скп тачака које су саставни део графа на нивоу важи да је . У општем случају .



Из оваквог описа, јасно је да ће се поједине тачке векторског простора јављати на неколико нивоа хијерархије. Важно је напоменути да међу нивоима структуре постоје везе. Тачка која представља чвор графа на нивоу имаће референцу на чвор исте те тачке на нивоу .

Конструисање структуре

Графови се конструишу итеративним додавањем елемената из скупа . Елемент постаје чвор графа и повезује се са најближих суседа. Аутори наводе да је фаза креирања графа погодна за паралелизацију.

Испрва се за сваки елемнт који се додаје у структуру одреди макслимални ниво на којем ће се појавити. Вероватноћа да се елемент нађе на вискоим нивоима је експоненцијално опадајућа, чиме се обезбеђује мали број чворова у вишим слојевима хијерархије. Процес додавања у структуру се угубо може поделити у две фазе:

* Проналажење улазног чвора почевши од највишег нивоа, све до максималног дефинисаног нивоа на ко ком се чвор додаје
* Проналажење најближих суседа на максималном нивоу за елемент и његовим поднивоима и повезивање са њима

Треба напоменути да се хеуристика која служи за претрагу нивоа користи у обе фазе. Разлика је што се у првој фази параметри постављају тако да се проналази само један најближи сусед.

Прва фаза додавања започиње у највишем нивоу где се проналазе најближи суседи за елемент који се додаје. Најближи сусед пронађен у највишем нивоу постаје улазни чвор за претрагу на нижем нивоу, где се пoступак претраге понавља. Суседи улазног чвора који су ближи елементу који се додаје у граф постају кандидати за најближег суседа. Како је у овој фази потребно наћи најближег суседа елемента који се додаје, хеуристика „прождрљиво“ бира само једног кандидата из листе суседа. У следећем кораку посматрају се суседи одабраног кандидата. На овај начин хеуристика у сваком чвору бира суседа који задовољава критеријум да је најближи елементу који се додаје. Претрага се завршава у чвору чији су суседи нису ближи елементу који се додаје. Тада се прелази на нижи хијерархијски ниво.

Друга фаза започиње када алгоритам спусти на ниво хиејарахије где ће елемент први пут посати чвор графа.

Старије верзије алгоритма нису имале хијерархијску структуру. Како би се избегло да се алгоритам претраге заустави у неком локалном минимуму, аутори су саветовали да се претрага започне у чворовима који имају висок степен повезаности (енг. 'hubs'). Како је број високо повезаних чворова растао са повећањем броја елемената, претрага је тежила полилогаритамској сложености. Увђењем концепта хијерархије, очекивало се смањење сложености претраге и могућност заустављања у локалном минимуму када се ради о калстеризованим подацима. Наведене претпоставке су се потврдиле у досадашњим експериментима.

# АНАЛИЗА МЕТОДА

Анализа перформанси и примене апроксимативних техника за претрагу најближих суседа представља срж проблема овог рада. Време је један од кључних ресурса данашњих предузећа. У складу са тим, расте потреба за брзим генерисањем резултата за предвиђање. Осим времена и ефикасног коришћења ресурса, тачност предвиђања је један од критичних захтева. Управо су дате чињенице показатељ вредности изучавања ове области машинског учења.

Најважнији резултат истраживања би могла бити препорука за правилан одабир алгоритма и његових параметара. У раду ће бити имплементирани и промењени одабрани апроксимативни алгоритми. Резултат истраживања обухватиће детаљан приказ неколико врста апроксимативних алгоритама и структура података на које се ослањају. У истраживању ће се спровести упоредна анализа алгоритама и њихових перформанси.

Истраживање треба да одговори на следећа питања:

* Шта је апроксимативни алгоритам најближих суседа?
* Које су најпопуларније апроксимационе технике?
* Колика је разлика у времену потребном за спровођење изворног алгоритма и апроксимативних алгоритама?
* Колика је разлика у резултатима предвиђања апроксимативних решења и изворног КНН-а?
* Да ли је, и у којој мери је могуће паралелизовати извршење алгоритма?
* Колики је утицај структуре података на брзину извршавања и резултате предвиђања апроксимативних варијанти алгоритма?
* Које ће перформансе и резултате апроксимативни алгоритми постићи при раду над скуповима различитих величина и броја атрибута?
* Да ли је могуће извести препоруке за коришћење одговарајућег апроксимативног алгоритма у зависности од података и корисничких захтева?

# ОРГАНИЗАЦИЈА И МЕТОДЕ ИСТРАЖИВАЊА

Резултати истраживања ће се добити експерименталним путем. Алгоритми ће се примењивати нaд скуповима података различитих димензија под различитим параметрима. Оквирни план истраживања обухвата четири фазе:

* Изучавање одабране литературе о алгоритмима који ће се применити у раду
* Измена постојећих и израда сопствених имплементација алгоритама
* Припрема података и тестирање алгоритма на пресоналном рачунару и на рачунарском кластеру уколико је то могуће
* Квантитативна анализа добијених резултата
* Извођење закључака и препорука

Добру основу за приступ проблему дао је Ханан Самет (Samet, 2006). Аутор пружа теоријску и практичну базу за примену алгоритма над различитим структурама података, превасходно стаблима. Читаво поглавље, посвећено је искључиво двема техникама претраге најближих суседа – инкременталној и техници претраге у дубину. Међу корисним радовима који ће бити проучени је (Liu, Moore, Yang, & Gray, 2005) где се презентују алгоритми базирани на метричким стаблима. Аутори истичу једноставност алгоритама и сугеришу да је могуће постићи резултате компаративне алгоритму дефинисаном од стране (Indyk & Motwani, 1998) уз могућу већу ефикасност. Рад (Abbasifard, Ghahremani, & Naderi, 2014) који пружа поређење различитих функција удаљености и сложености алгоритама ће такође утицати на избор метoда које ће се имплементирати.

Препоруке за избор структуре погодне за претрагу у зависности од карактеристика података који се обрађују дали су (Muja & Lowe, 2009). Утврђено је да два алгоритма од којих се оба заснивају на стаблима (енг. *randomized kd-tree* и *hierarchical k-means tree*) пружају најбоље резултате. Значајно је напоменути да би број инстанци код К-Д стабла морао бити значајно већи у односу на број димензија , јер у супротном не би било побољшања у времену у односу на комплетну претрагу свих инстанци (Arya, Mount, & Narayan, Accounting for boundary effects in nearest-neighbor searching, 1996).

Следећи корак у изради истраживања засниваће се на проучавању литературе које се тиче примене алгоритама у случајевима када подаци над којима се извршавa имају већи број димензија.

Значајно другачији приступ у проналажењу најближих суседа даје рад (Indyk & Motwani, 1998).

Наводи се да када је број димензија између 10 или 20, К-Д стабла и њима сличне структуре генеришу резултате претраге тек незнатно брже од линеарне претраге.

Раније је напоменуто да ће време бити кључни идентификатор перформанси. Друге мере евалуације које би се могле дефинисати су:

* Грешка код функције удаљености – разлика у вредности функције удаљености од непознате инстанце између апроксимативних и правих суседа
* Грешка у идентификацији – број идентификованих правих суседа у односу на број тражених
* Трошкови претпроцесирања – размотриће се време, сложености поступка и меморисјки захтеви алгоритма
* Тачност – проценат правилно класификованих инстанци

Садржај мастер рада ће бити подложан променама у зависности од тока истраживања. Следи оквирни садржај:

1. Увод
2. Апроксимативни алгоритми најближих суседа
   1. Преглед стања у предметној области
   2. Алгоритам заснован на К-Д стаблима
      1. Дефиниција К-Д стабла
      2. Попуњавање К-Д стабла
      3. Претраживање К-Д стабла
   3. Алгоритам заснован на хеш функцијама
   4. Дистрибуиране технологије за апроксимацију
3. Имплементација
   1. Анализа скупова података
   2. Тестирање алгоритама
4. Резултати истраживања
   1. Упоредни приказ резултата
   2. Препоруке
5. Закључак
6. Прилози

Алгоритми ће се имплементирати у Пајтон (eng. *Python*) програмском језику. У разматрање ће се узети постојећа решења и библиотеке попут *scipy.spatial.KDTree* или *sklearn.neighbors.kd\_tree* и друге. При истраживању могућности имплементације над *Apache Spark* архитектуром користиће се *PySpark* програмски интерфејс.

# Опис дела света који ће бити изучаван

Претпоставка аутора је да ће се користити скупови података доступни на адреси https://archive.ics.uci.edu/ml/datasets.html (UCI Machine Learning Repository). Међу кандидатима за почетну анализу налазе се мањи скупови који имају више од сто хиљада инстанци попут *Covertype Data Set*. Изазован кандидат за анализу представља скуп под називом *HIGGS,* где задатак разлучивање да ли сигнал означава појаву Хигсовог бозона или не.

# Сврха и циљеви истраживања

Сврха овог истраживања је да прикаже могућности унапређења стандардног алгоритма за проналажење најближих суседа применом метода апроксимације, што ће омогућити истраживачима и другим заинтересованим странама да брже и ефикасније спроводе предиктивне анализе над већим скуповима података. Аутор се нада да ће мастер рад обогатити скроман фонд литературе која се бави поменутом проблематиком на српском језику и олакшати будућим мастер студентима упознавање са темом. Истраживање ће дати преглед и поређење досадашњих резултата и указати на њихове предности и мане. Циљ рада је унапређење и експериментално поређење перформанси апроксимативних алгоритама који се извршавају над великим подацима, као и теоријски осврт на кораке и сложеност таквих алгоритама.

# РЕФЕРЕНЦЕ

Abbasifard, M. R., Ghahremani, B., & Naderi, H. (2014). A survey on nearest neighbor search methods. *International Journal of Computer Applications, 95*, 39-52.

Arya, S., Mount, D. M., & Narayan, O. (1996). Accounting for boundary effects in nearest-neighbor searching. *Discrete & Computational Geometry, 16*, стр. 155-176.

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., & Wu, A. Y. (1998). An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Јournal of the ACM (JACM), 45*, 891-923.

Douze, M., Sablayrolles, A., & Jégou, H. (2018). Link and code: Fast indexing with graphs and compact regression codes. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (стр. 3646-3654).

Gionis, A., Indyk, P., & Motwani, R. (1999). Similarity search in high dimensions via hashing. *Vldb*, *99*, стр. 518-529. Edinburgh.

Han, J., Pei, J., & Kamber, M. (2012). *Data mining: concepts and techniques* (3rd изд.). Waltham, Massachusetts, United States of America: Elsevier.

Indyk, P., & Motwani, R. (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. ACM.

Kleinberg, J. M. (1997). Two algorithms for nearest-neighbor search in high dimensions. *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing* (стр. 599--608). El Paso,Texas, USA: АCM.

Kushilevitz, E., Ostrovsky, R., & Rabani, Y. (2000). Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing, 30*, 457-474.

Liu, T., Moore, A. W., Yang, K., & Gray, A. G. (2005). An investigation of practical approximate nearest neighbor algorithms., (стр. 825-832).

Maillo, J., Ramirez, S., Triguero, I., & Herrera, F. (2017). kNN-IS: An Iterative Spark-based design of the k-Nearest Neighbors classifier for big data. *Knowledge-Based Systems, 117*, 3-15.

Marimont, R., & Shapiro, M. (1979). Nearest neighbour searches and the curse of dimensionality. *IMA Journal of Applied Mathematics*, стр. 59-70.

Muja, M., & Lowe, D. G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 331-340.

Samet, H. (2006). *Foundations of multidimensional and metric data structures.* (J. Gray, Ур.) San Francisco, California, United States of America: Morgan Kaufmann.

Seidl, T., & Kriegel, H.-P. (1998). Optimal Multi-Step k-Nearest Neighbor Search. *27.* Seattle: ACM SIGMOD Record.

Triguero, I., Maillo, J., Luengo, J., García, S., & Herrera, F. (2016). From big data to smart data with the k-nearest neighbours algorithm.

*UCI Machine Learning Repository*. (n.d.). Преузето са University of California, Irvine: https://archive.ics.uci.edu/ml/datasets.html

Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. *e 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, (стр. 311-321). Austin, USA.