

# Primena genetskog programiranja u klasifikaciji spama

Milena Stojić

26. septembar 2021.

## 1 Uvod

Problem klasifikovanja neželjene pošte ili spama je problem koji od početka ere komunikacije putem elektronske pošte privlači veliku pažnju. Osim što neželjene poruke pune naše sanduče i zatrpavaju potencijalno važne poruke, spam poruke nanose veliku finansijsku štetu kompanijama [6], a mogu sadržati i virus [4]. Prema jednoj studiji [13] čak preko 70% svih poslovnih mejlova je spam.

Veliki se naponi ulažu da se napravi što pouzdaniji klasifikator koji će znati da prepozna i ukloni poruku koja je spam, ali koji u isto vreme neće pogrešne mejlove klasifikovati kao spam.

Većina poznatih algoritama klasifikacije poput SVM i neuronskih mreža je primenjeno na ovaj problem i svi su dali odlične rezultate sa veoma malim greškama klasifikacije [9].

Pored standardnih algoritama klasifikacije postoji pokušaj razvoja klasifikatora i genetskim programiranjem [12]. U tom pristupu klasifikator je bio u formi algebarskog izraza čija se vrednost računala nad transformisanim vrednostima atributa i na osnovu te vrednosti izraza klasifikovala instanca. Algoritam je sam davao malo lošije rezultate od standardnih algoritama klasifikacije, ali ansambl ovih klasifikatora je davao bolje rezultate od polovine postojećih algoritama.

Genetsko programiranje (skraćeno *GP*) je metaheuristika koja je, poput genetskih algoritama (skraćeno *GA*), inspirisana evolucijom. [2]. Pronalazi primene u mašinskom učenju i često se koristi kao alat u preprocesiranju podataka. [3] Mi ćemo u ovom radu direktno konstruisati stabla odlučivanja, to jest klasifikatore, genetskim programiranjem, probati za moguće vrednosti parametara i potom uporediti dobijene rezultate sa rezultatima iz prethodnih istraživanja.

U poglavlju 2 ćemo dati detaljniji opis opšteg algoritma genetskog programiranja, a u poglavlju 3 konkretan opis našeg pristupa. Eksperimentalni rezultati sa upoređivanjem sa postojećim rezultatima će biti u 4 i u 5 zaključak sa razmatranjem mogućih načina daljeg unapređivanja.

## 2 Genetsko programiranje

Zamisao genetskog programiranja je bila evolucija računarskih programa koji za korisnika obavljaju određeno izračunavanje. [2] Kao i kod svih ostalih evolutivnih algoritama, inspirisanih Darvinovom teorijom evolucije, i u genetskom programiranju imamo koncepte kao što su selekcija, ukrštanje i mutacija.

Prva istraživanja koja su isla u tom pravcu su započela početkom 80-ih godina prošlog veka. [2] Neki od doprinosa iz tog perioda su Smitov sistem učenja zasnovan na genetskim algoritmima [11] (koji je autoru bio tema doktorske disertacije) i Forsitov kompjuterski paket za generisanje pravila odlučivanja u oblasti forenzike [7]. Jedno od prvih važnih publikacija na temu genetskog programiranja je "Genetsko programiranje - o programiranju računara prirodnom selekcijom" [8] čiji je autor J. Koza, profesor univerziteta Stanford koji se među prvim bavio istraživanjima na temu genetskog programiranja i koji ima velike naučne doprinose u ovoj oblasti. Danas GP ima raznovrsne primene u najrazlicitijim oblastima, od problema masinskog učenja, kao što je, recimo, predviđanje cena akcija, do raznih primena u inženjerstvu. [10]

### 2.1 Opsti princip

Programi (bilo kakve vrste) se najprirodnije predstavljaju hijerarhijskom, drvolikom reprezentacijom. Tako da su jedinke u populaciji (koje predstavljaju programe) predstavljene u formi drveta. Za razliku od jedinki u genetskim algoritmima koje se predstavljaju linearnom strukturom fiksne veličine. U drvetu obično jasno razlikujemo dva tipa cvorova:

- **unutrasnji (neterminalni) cvorovi** - obično predstavljaju neke funkcije ili operacije koje se primenjuju u toku izračunavanja
- **terminalni cvorovi (u listovima)** - obično predstavljaju vrednosti koje učestvuju u izračunavanjima. Deo terminalnih cvorova može biti namenjen ulaznim podacima (koji će učestvovati u izračunavanju), a u ostalim konstantne vrednosti koje učestvuju u svakom izračunavanju.

U našem slučaju u listovima će biti klase koje se mogu dobiti izračunavanjem. (dakle izlazne vrednosti izračunavanja)

Selekcija u GP-u je obično ruletska. (srazmerna fitnessu) Pri formiranju jedinke iz nove generacije obično se slučajno bira da li će se izvršiti mutiranje odabrane jedinke (sa veoma malom verovatnoćom, obično reda  $\sim 0.05$ ) ili ukrcanje dve odabrane jedinke iz tekuće populacije. Određen broj jedinki sa najboljom vrednošću fitness funkcije se može iz tekuće generacije direktno prenositi u narednu kao i kod genetskih algoritama. (koncept elitizma) Mutacija kao slučajna promena u stablu se može obavljati na različite načine. Ukrcanje se radi razmenom podstabala između jedinki koje se ukrstaju.

### 3 Opis moje metode

Posto u resavanju koristimo genetsko programiranje, potrebno je da objasnimo šta će nam biti jedinke i kako će tačno biti predstavljene. Takođe je potrebno da objasnimo šta će predstavljati fitness funkciju i na koji način je izračunavamo.

Jedinke će nam biti pojedinačni klasifikatori koji će biti stabla odlučivanja. U stablima odlučivanja terminalni cvorovi su atributi, a u terminalnim cvorovima su predviđene klase.

Klasifikacija instance se vrši na sledeći način: polazi se od korena stabla odlučivanja, ispituje se vrednost atributa iz korena u toj instanci i u zavisnosti od te vrednosti (tačan uslov zavisnosti je obično definisan u samom cvoru ili na neki drugi način) nastavlja se ispitivanje u levom ili u desnom podstablu. I tako radimo za svaki unutrašnji cvor (i atribut u njemu) sve dok ne stignemo do terminalnog cvora. Instanci dodeljujemo onu klasu koja se nalazi u terminalnom cvoru u koji smo stigli.

### 3.1 Implementacija jedinki

#### 3.1.1 Korisceni atributi - geni

Atributi koje ćemo koristiti u neterminalnim cvorovima stabala odlučivanja su preuzeta iz skupa atributa iz *Spam-base Data Set-a* [1]. Skup podataka se sastoji od 4601 instanci koje predstavljaju mejlove. U skupu atributi su (ne računajući ciljni atribut) reci, slova, kao i prosedna dužina sekvenci uzastopnih velikih slova, najveća dužina sekvence uzastopnih velikih slova i ukupna dužina svih sekvenci uzastopnih velikih slova. Za nas su u tom skupu podataka već izračunate frekvencije reci i slova, zbog čega je taj skup podataka zgodan. Ne moramo da se bavimo *textmining*-om koji nije fokus ovog rada.

Od tih atributa koristili smo deo atributa čija je prosedna vrednost veća za mejlove koji su spam. Ideja je

bila da se koriste atributi čije veće vrednosti su obično karakteristične za spam mejlove.

#### 3.1.2 Sirova implementacija stabla odlučivanja

Stabla odlučivanja smo interno implementirali nizovima fiksne dužine. Definirali smo minimalnu i maksimalnu moguću dubinu stabla. Maksimalan broj cvorova u stablu dubine  $d \geq 1$  je

$$1 + 2^1 + 2^2 + \dots + 2^d = \frac{2^{d+1} - 1}{2 - 1} = 2^{d+1} - 1$$

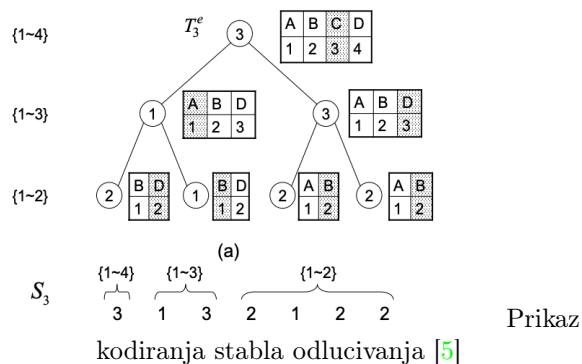
tako da su nam nizovi interno uvek dužine

$$2^{\text{najveća\_dubina}} - 1$$

. U zavisnosti od dubine stabala koristimo određene delove niza ili ceo niz. Članovi niza koji se ne koriste imaju nedefinisano (*None*) vrednost.

Ako bismo jednostavno čuvali atribut u stablu, takva stabla bi dobro radila u klasifikaciji, ali ne bi bila pogodna za operatore ukrstavanja. Na primer, desavalo bi se da često pri ukrstavanju dobijamo stablo u kome bismo na putu od korena do nekog lista imali više puta ponovljen atribut. Drugim rečima, bila bi više puta ispitivana vrednost istog atributa. Da bismo ovo izbegli, moramo da drugacije kodiramo atribut u stablu. Tako da i dalje imamo ispravna stabla odlučivanja, ali i da ukrstanjem takođe dobijamo ispravna stabla odlučivanja.

Da bismo prevazisli ovaj problem u kodiranju, iskoriscena je (samo) početna ideja iz jednog konferencijskog članka koji sam citala radeci na drugom projektu. [5] Umesto da koristimo direktno atribut, koristimo njihove indekse u *trenutnom nizu atributa*. Naime, u korenu ćemo čuvati indeks atributa u (celom) nizu atributa. U neterminalnom cvoru na nekoj dubini  $d \geq 1$  ćemo čuvati indeks atributa u nizu svih atributa iz koga su uklonjeni atributi u precima cvora. Ideja je jasnije prikazana na slici ispod.



U cvoru dubini  $d$  mogu biti vrednosti indeksa od 0 do  $(n - 1) - d$ , gde je  $n$  ukupan broj atributa koji se koristi u izgradnji stabla.

### 3.1.3 Omotac klasa jedinke

Objekat klase koja implementira jedinke je sacinjen od stabla odlucivanja i vrednosti fitnes funkcije. Za svaku jedinku znamo da izracunamo vrednost fitnes funkcije. Dva stabla se upoređuju po vrednosti fitnes funkcije.

## 3.2 Fitnes funkcija

Nama je cilj da ovim algoritmom dobijemo stablo koje ce imati sto bolju klasifikacije. Tako da nam fitnes funkcija moze biti neka od mera kvaliteta klasifikacije ili kombinacija vise aspekata kvaliteta klasifikacije. (linearna kombinacija ili bilo kakva drugacija funkcija vise merila klasifikacije)  
Ovde ce nam fitnes funkcija biti tacnost (eng. *accuracy*). Ta mera predstavlja udeo dobro klasifikovanih instanci, to jest:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

## 3.3 Inicijalizacija populacije

Populacija se slucajno generise ogranicenim pristupom. To znaci da dubina stabala moze da varira izmedju minimalne i maksimalne vrednosti. Pri samom kreiranju jedne instance stabla njen sadrzaj se unapred slucajno generise. Tako da je dovoljno da samo kreiramo zeljeni broj stabala i imamo spremnu pocetnu populaciju.

## 3.4 Selekcija

Selekcija jedinki za ukrstanje ili mutaciju je ruletska ili fitnes srazmerna selekcija. Bice primenjen i elitizam, jer ako dodjemo do jako dobrih jedinki, ne zelimo da ih izgubimo.

## 3.5 Operatori ukrstanja i mutacije

Ukrstanje je razmena podstabala izmedju dve jedinke, kao u svakom GP algoritmu.  
Operatorom mutacije mi menjamo slucajno izabrani cvor (terminalni ili neterminalni) slucajno izabranom vrednoscu odgovarajuceg tipa.

## 3.6 Dodatno

Posle izvršavanja svih iteracija, osim evaluacije klasifikacije najbolje jedinke na test podacima ispitivali smo kakvu klasifikaciju nam daje i ansambl odredenog broja najboljih jedinki. Ovaj pristup je koriscen pri poboljšavanju metode genetskog programiranja u [12] i dao je dobre rezultate.

# 4 Eksperimentalni rezultati

## Tehnicke karakteristike i okruzenje

Projekat je implementiran u programskom jeziku Python. Od biblioteka koriscene su biblioteke *Numpy* (za generisanje pseudoslucajnih brojeva i nekoliko matematickih funkcija), *Pandas* (za ucitavanje i rad sa podacima), modul *metrics* i jos jedan metod iz *Scikit-learn* biblioteke (za evaluaciju kvaliteta klasifikovanja) i biblioteka *Matplotlib* za vizualizaciju.

Tokom razvoja koriscen je skup podataka *Spambase Data set* [1]. Pre formiranja populacije delili smo ga na trening i test podatke. Pre izvršavanja genetskog algoritma razmatrali bismo trening podatke. (to jest, prosečne vrednosti atributa svih instanci koje su spam iz tog podskupa podataka bismo iskoristili za izgradnju stabla odlucivanja)  
Na kraju, posle izvršavanja celog GP-a bismo vrsili evaluaciju na osnovu test podataka.

### Karakteristike okruzenja:

- Hardver: 2,4 GHz *Quad-Core Intel Core i5* (4 jezgara, u isto vreme je paralelno izvršavano vise skriptova sa razlicitim parametrima), RAM: 8 GB
- Operativni sistem: *Mac OS 11.5.2*
- Razvojno okruzenje: *PyCharm*
- Interpreter: *Python 3.9*

## 4.1 Dobijeni rezultati

U tabeli se mogu videti dobijeni eksperimentalni rezultati dobijeni za razlicite vrednosti parametara. Pokazuje se da je broj jedinki u populaciji najvazniji parametar. Sto je veka populacija, imacemo bolju klasifikaciju.

Najbolje bi bilo kada bismo uz veliki broj jedinki u populaciji imali i veliki broj iteracija. Medjutim, ako zbog vremenskih ili hardverskih ogranicenja moramo da ogranicimo vrednost jednog od parametara, za kvalitet klasifikacije se vise isplati da smanjimo broj iteracija. Kao sto se moze videti u tabeli, za cak vrlo mali broj iteracija na velikoj populaciji smo dobijali znatno bolje rezultate nego uz vise iteracija na manjoj populaciji.

Br. iteracija	Populacija	pm	Elitizam	Tacnost
1	5000	0.1	400	0.8298
50	100	0.05	400	0.6698
20	2000	0.05	400	0.7385

## 4.2 Poredjenje sa postojećim rezultatima

Iako postoji rad [9] koji se bavi poredjenjem različitih klasifikatora spama na istom skupu podataka koji je korišćen i u ovom radu, koriscene su drugacije mere kvaliteta u evaluaciji. (izuzev za najbolji klasifikator, slucajnu sumu, za koju su dodate dve mere koje su koriscene u ovom radu)

Razmatracemo rezultate iz [12]. // U ovom radu je radjeno na dva skupa podataka. Mi cemo uporedjivati sa vrednostima dobijenim na skupu podataka na kome su generalno dobijeni losiji rezultati.

U tabeli su dati rezultati za GP algoritme i za jos dva tipa klasifikatora.

Klasifikator	Tacnost	F-mera
GP	0.848	0.848
Unapredjeni GP (ansambl)	0.941	0.941
SVM	0.938	0.938
Nasumicna suma	0.939	0.939

Mozemo videti da je po meri tacnosti nas pristup blizak tacnosti GP algoritma. Trebalo bi jos pomenuti da su u ovom radu, za razliku od naseg, na efektniji nacin birani atributi koji ucestvuju u klasifikaciji. Kada bismo imali bolje gene, imali bismo i bolju klasifikaciju. U ovom radu se takodje mera kvaliteta tacnost koristi kao fitnes funkcija.

## 5 Zakljucak

U ovom radu smo jednim drugacijim pristupom pokusali da implementiramo algoritam klasifikacije spama. Fokus je bio na samom principu genetskog programiranja, dok se u kvalitet preprocesiranja podataka nije duboko ulazilo. Dobili smo rezultat koji po tacnosti nije blizu pristupima poput SVM ili slucajne sume, ali jeste blizak rezultatu dobijenom standardnim GP pristupom iz literature.

Ovo nas moze ohrabriti da daljim radom i poboljšavanjem pokusamo da dobijemo jos bolje rezultate. Narocito ako posvetimo vise paznje izboru najboljeg podskupa atributa (eng. *feature selection*) i izbora nacinu na koji da ispitujemo vrednost tog atributa u stablu.

Jos jedan nacin da unapredimo nas algoritam je da pronadjemo pogodniji izraz za fitnes funkciju. U kome bi, pored tacnosti takodje figurisale i druge bitne mere kvaliteta klasifikacije. U ovom slucaju, na primer, to bi mogao da bude broj lazno pozitivnih u imeniocu izraza za racunanje fitnes funkcije.

## Literatura

[1] UCI machine learning repository – Spambase Data Set, 1999. <http://archive.ics.uci.edu/ml/datasets/Spambase>.

- [2] Michael Affenzeller, Stephan Winkler, Stefan Wagner, and Andreas Beham. *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*. Chapman Hall/CRC, 1st edition, 2009.
- [3] Qurrat Ul Ain, Bing Xue, Harith Al-Sahaf, and Mengjie Zhang. Genetic programming for feature selection and feature construction in skin cancer image classification. In *Pacific rim international conference on artificial intelligence*, pages 732–745. Springer, 2018.
- [4] Mamoun Alazab and Roderic Broadhurst. Spam and criminal activity. *Trends and issues in crime and criminal justice*, (526):1–20, 2016.
- [5] Sung-Hyuk Cha and Charles Tappert. Constructing Binary Decision Trees using Genetic Algorithms. page 3, 2008.
- [6] V Christina, S Karpagavalli, and G Suganya. A study on email spam filtering techniques. *International Journal of Computer Applications*, 12(1):0975–8887, 2010.
- [7] R FORSYTH. Beagle—a darwinian approach to pattern recognition. *Kybernetes*, Vol. 10(No. 3):pp. 159–166, 1981.
- [8] John R Koza and John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.
- [9] R. Kumar, G. Poonkuzhali, and Sudhakar Pandiarajan. Comparative study on email spam classifier using data mining techniques. *Lecture Notes in Engineering and Computer Science*, 2195:539–544, 03 2012.
- [10] Michael O’Neill, Leonardo Vanneschi, Steven Gustafson, Anna Isabel Esparcia Alcázar, Ivanoe De Falco, Antonio Della Cioppa, and Ernesto Tarantino. *Genetic Programming: 11th European Conference, EuroGP 2008, Naples, Italy, March 26-28, 2008, Proceedings*, volume 4971. Springer, 2008.
- [11] S. F. Smith. *A Learning System Based on Genetic Adaptive Algorithms*. PhD thesis, University of Pittsburgh, 1980.
- [12] Shrawan Trivedi and Shubhamoy Dey. An enhanced genetic programming approach for detecting unsolicited emails. pages 1153–1160, 12 2013.
- [13] Bo Yu and Zong-ben Xu. A comparative study for content-based dynamic spam classification using four machine learning algorithms. *Knowledge-Based Systems*, 21(4):355–362, 2008.