Primena genetskog programiranja u klasifikaciji spama

Milena Stojić

26. septembar 2021.

Sažetak

Identifikacija neželjenih mejlova (eng. spam) je jedan od izazovnih problema u računarstvu. Razvijeno je puno algoritama klasifikacije koji sa velikom tačnošću vrše klasifikovanje mejlova. U ovom radu smo pokušali da uz drugačiju primenu metaheuristike genetskog programiranja (skraćeno GP) generišemo klasifikator spama.

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 napori 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 išla 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 Stenford koji se među prvima bavio istraživanjima na temu genetičkog programiranja i koji ima velike naučne doprinose u ovoj oblasti. Danas GP ima raznovrsne primene u najrazličitijim oblastima, od problema mašinskog učenja, kao što je, recimo, predviđanje cena akcija, do raznih primena u inženjerstvu. [10]

2.1 Opšti 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 čvora:

- **unutrašnji (neterminalni) čvorovi** - obično predstavljaju neke funkcije ili ope-

racije koje se primenjuju u toku izračunavanja

- **terminalni čvorovi (u listovima)** - obično predstavljaju vrednosti koje učestvuju u izračunavanjima. Deo terminalnih čvorova može biti namenjen ulaznim podacima (koji ce 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 fitnesu) Pri formiranju jedinke iz nove generacije obično se slučajno bira da li će se izvršiti mutiranje odabrane jedinke (sa veoma malom verovatnoćom, obično reda ~ 0.05) ili ukrštanje dve odabrane jedinke iz tekuće populacije. Određen broj jedinki sa najboljom vrednošću fitnes 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 nacine. Ukrštanje se radi razmenom podstabala između jedinki koje se ukrštaju.

3 Opis moje metode

Pošto u rešavanju 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 fitnes funkciju i na koji način je izračunavamo.

Jedinke će nam biti pojedinačni klasifikatori koji će biti stabla odlučivanja. U stablima odlučivanja neterminalni čvorovi su atributi, a u terminalnim čvorovima 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 čvor (i atribut u njemu) sve dok ne stignemo do terminalnog čvora. Instanci dodeljujemo onu klasu koja se nalazi u terminalnom čvoru u koji smo stigli.

3.1 Implementacija jedinki

3.1.1 Korišćeni atributi - geni

Atributi koje ćemo koristiti u neterminalnim čvorovima 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 računujaći ciljni atribut) reči, slova, kao i prosečna 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 reči 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 prosečna vrednost veća za mejlove koji su spam. Ideja je bila da se koriste atributi čije veće vrednosti su obicno karakteristicne za spam mejlove.

3.1.2 Sirova implementacija stabla odlucivanja

Stabla odlučivanja smo interno implementirali nizovima fiksne dužine. Definisali smo minimalnu i maksimalnu moguću dubinu stabla. Maksimalan broj čvorova u stablu dubine $d >= 1$ je

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

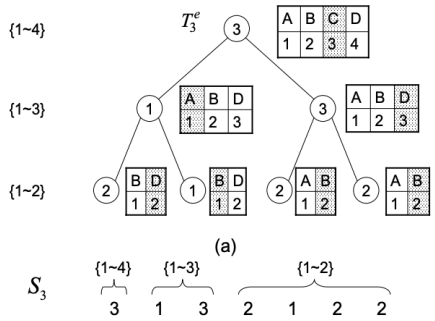
tako da su nam nizovi interno uvek dužine

$$2^{najveca\_dubina+1} - 1$$

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

Ako bismo jednostavno čuvali atribute u stablu, takva stabla bi dobro radila u klasifikaciji, ali ne bi bila pogodna za operatore ukrštanja. Na primer, dešavalo bi se da često pri ukrštanju 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 drugačije kodiramo atribute u stablu. Tako da i dalje imamo ispravna stabla odlučivanja, ali i da ukrštanjem takođe dobijamo ispravna stabla odlučivanja.

Da bismo prevazišli ovaj problem u kodiranju, iskorišćena je (samo) početna ideja iz jednog konferencijskog članka koji sam čitala radeći na drugom projektu. [5] Umesto da koristimo direktno atribute, koristićemo njihove indekse u *trenutnom nizu atributa*. Naime, u korenu ćemo čuvati indeks atributa u (celom) nizu atributa. U neterminalnom čvoru na nekoj dubini $d >= 1$ ćemo čuvati indeks atributa u nizu svih atributa iz koga su uklonjeni atributi u precima cvora. Ideja je jasnije prikazana na slici ispod.



Prikaz kodiranja stabla odlučivanja [5]

U čvoru 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 Omotač klasa jedinke

Objekat klase koja implementira jedinke je sačinjen od stabla odlučivanja i vrednosti fitnes funkcije. Za svaku jedinku znamo da izračunamo vrednost fitnes funkcije. Dva stabla se upoređuju po vrednosti fitnes funkcije.

3.2 Fitnes funkcija

Nama je cilj da ovim algoritmom dobijemo stablo koje će imati što bolju klasifikaciju. Tako da nam fitnes funkcija može biti neka od mera kvaliteta klasifikacije ili kombinacija više aspekata kvaliteta klasifikacije. (linearna kombinacija ili bilo kakva drugačija funkcija više merila klasifikacije)
Ovde će nam fitnes funkcija biti tačnost (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 slučajno generise ograničenim pristupom. To znači da dubina stabala može da varira između minimalne i maksimalne vrednosti. Pri samom kreiranju jedne instance stabla njen sadržaj se unapred slučajno generiše. Tako da je dovoljno da samo kreiramo željeni broj stabala i imamo spremnu početnu populaciju.

3.4 Selekcija

Selekcija jedinki za ukrštanje ili mutaciju je ruletska ili fitnes srazmerna selekcija. Biće primenjen i elitizam, jer ako dođemo do jako dobrih jedinki, ne želimo da ih izgubimo.

3.5 Operatori ukrštanja i mutacije

Ukrštanje je razmena podstabala između dve jedinke, kao u svakom GP algoritmu.
Operatorom mutacije mi menjamo slučajno izabrani čvor (terminalni ili neterminalni) slučajno izabranom vrednošću odgovarajućeg tipa.

3.6 Dodatno

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

4 Eksperimentalni rezultati

Tehničke karakteristike

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

Tokom razvoja korišćen 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 odlučivanja)
Na kraju, posle izvršavanja celog GP-a bismo vršili evaluaciju na osnovu test podataka.

Karakteristike okruženja:

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

4.1 Dobijeni rezultati

U tabeli se mogu videti eksperimentalni rezultati dobijeni za različite vrednosti parametara. Pokazuje se da je broj jedinki u populaci-

ji najvažniji parametar. Sto je veća populacija, imaćemo bolju klasifikaciju.

Najbolje bi bilo kada bismo uz veliki broj jedinki u populaciji imali i veliki broj iteracija. Međutim, ako zbog vremenskih ili hardverskih ograničenja moramo da ograničimo vrednost jednog od parametara, za kvalitet klasifikacije se više isplati da smanjimo broj iteracija. Kao što se može videti u tabeli, za čak vrlo mali broj iteracija na velikoj populaciji smo dobijali znatno bolje rezultate nego uz više iteracija na manjoj populaciji.

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

4.2 Poređenje sa postojećim rezultatima

Iako postoji rad [9] koji se bavi poređenjem različitih klasifikatora spama na istom skupu podataka koji je korišćen i u ovom radu, korišćene su drugačije mere kvaliteta u evaluaciji. (izuzev za najbolji klasifikator, slučajnu sumu, za koji je dodata mera tačnosti)

Razmatraćemo rezultate iz [12]. U ovom radu je rađeno na dva skupa podataka. Mi ćemo upoređivati sa vrednostima dobijenim na skupu podataka na kome su generalno dobijeni lošiji rezultati.

U tabeli su dati rezultati za GP algoritme i za još 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

Možemo videti da je po meri tačnosti naš pristup blizak tačnosti GP algoritma. Trebalo bi još pomenuti da su u ovom radu, za razliku od našeg, na efektniji način birani atributi koji učestvuju u klasifikaciji. Kada bismo imali bolje gene, imali bismo i bolju klasifikaciju. U ovom radu se takođe mera kvaliteta tačnost koristi kao fitnes funkcija.

5 Zaključak

U ovom radu smo jednim drugačijim pristupom pokušali 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 tačnosti nije blizu pristupima poput SVM ili slučajne šume, ali jeste blizak rezultatu dobijenom standardnim GP pristupom iz literature.

Ovo nas može ohrabriti da daljim radom i poboljšavanjem pokušamo da dobijemo još bolje rezultate. Naročito ako posvetimo vise pažnje izboru najboljeg podskupa atributa (eng. *feature selection*) i izbora načina na koji da ispitujemo vrednost tog atributa u stablu.

Još jedan način da unapredimo naš algoritam je da pronađemo pogodniji izraz za fitnes funkciju. U kome bi, pored tačnosti takođe figurisale i druge bitne mere kvaliteta klasifikacije. U ovom slučaju, na primer, to bi mogao da bude broj lažno pozitivnih u imeniocu izraza za računanje 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.