Univerzitet u Beogradu

Matematički fakultet

Radomir Đoković

Predviđanje polariteta iskaznih promenljivih u zadovoljavajućim valuacijama iskaznih formula metodama klasifikacije

— Master rad —

Mentor: dr Mladen Nikolić

Beograd

2015.

Sadržaj

1. Uvod

SAT problem je problem nalaženja valuacije koja zadovoljava formulu u konjuktivnoj normalnoj formi (KNF). SAT rešavači su programi koji rašavaju SAT problem. Kao ulaz prihvataju formulu u KNF i cilj im je da nađu valuaciju koja je zadovoljava.

SAT problem je NP-kompletan. Vremenska složenost poznatih algoritama za njegovo rešavanje je eksponencijalna i zato u slučaju ulaza većih dimenzija često nije moguće rešiti problem u realnom vremenu. Ipak, SAT problem ima razne primene, na primer, pravljenje rasporeda [7], verifikacija hardvera[5] i mnoge druge, te je često veoma bitno rešiti određene instance SAT problema u realnom vremenu. Zbog toga su performanse SAT rešavača od velikog značaja i veliki broj naučnih radova je napisan na temu poboljšanja njihovih performansi.

Razvoj SAT rešavača je bio intenzivan u prethodne dve decenije, što je rezultovalo velikim brojem SAT rešavaca koji se mogu podeliti u više familija - pre svega na potpune, zasnovane na DPLL proceduri i nepotpune, među kojima su najkorišćeniji stohastički SAT rešavači. Stohastički SAT resavači rade tako što na slučajan način biraju početnu valuaciju i onda u njenoj okolini traže valuaciju koja će zadovoljiti formulu. U svakom koraku oni biraju promenljivu kojoj će promeniti polaritet. Tu promenljivu mogu izabrati deterministički tako da maksimizuju neku unapred izabranu funkciju, ili na slučajan način. Zbog mogućnosti izbora promenljive na slučajan način zovu se stohastički. Jedan od takvih modernijih stohastičkih SAT rešavača je ProbSAT [4] .

Performanse stohastičkih SAT rešavača u velikoj meri zavise od toga koliko je polazna valuacija blizu zadovoljavajućoj valuaciji. U slučaju da je početna valuacija blizu valuacije koja zadovoljava formulu, program će vrlo brzo doći do rešenja. Izbor polazne valuacije je često slučajan ili se vodi jednostavnim heuristikama, ali se može lako izmeniti. Zbog toga je od značaja doći do načina biranja polazne valuacije koja će biti pogodnija od slučajnog biranja početne valuacije i tako poboljšati performase rešavanja SAT problema.

Formule u okviru istih familija imaju neku zajedničku strukturu. Jedna mogućnost popravljanja performansi stohastičkih SAT rešavača je da se metodama mašinskog učenja "nauče" neke osobine tih familija i da se na osnovu toga predviđa početna valuacija za svaku instancu ponaosob.

U neobjavljenom radu Brajana Silvertrona [1] je modifikovan SAT rešavač, tako da na osnovu modela dobijenog metodom logističke regresije približno predviđa zadovoljavajuću valuaciju i nju koristi kao početnu u pretrazi. Ovaj model je u stanju da promenljive klasifikuje kao tačne ili netačne u zavisnosti od vrednosti atributa tih promenljivih u ulaznoj formuli.

Silvertron je u svom radu, kako navodi, dobio pozitivne rezultate i poboljšao performanse SAT rešavača. U ovom radu će njegova metodologija biti još jednom eksperimentalno evaluirana čime će se videti da li su prethodno dobijeni rezulati validni samo za rešavač i familije instanci koje je on koristio u svom istraživanju.

Za klasifikaciju se koriste isti atributi kao i atributi u Silvertornovom radu. U ovom radu je korišćena eksperimentalna postavka koja nalikuje onoj iz Silvertornovog rada u meri u kojoj je to iz rada moguće reprodukovati. Doprinosi ovog rada su sledeći:

- Kreirana je biblioteka za računanje atributa promenjivih u formulama. Ona se koristi pri kreiranju trening skupa, proceni preciznosti klasifikacije i pri predviđanju početne valuacije.
- Logističkom regresijom je kreiran je klasifikacioni model za predviđanje polariteta promenljivih u zadovoljavajućim formulama i testirana je njegova preciznost.
- Modifikovan ja SAT rešavač ProbSAT tako da može da pomoću već dobijenog logističkog modela i kreirane biblioteke za računanje atributa izračuna početnu valuaciju za rešavanje formule.
- Sprovedeni su eksperimenti na nekoliko familija iskaznih formula kojima je izvršena provera upotrebljivosti ovog pristupa upoređivanjem performansi originalnog i modifikovanog SAT rešavača.

Struktura ovog rada je sledeća. U drugoj glavi biće opisani SAT problem, njegova primena, SAT rešavači, vrste SAT rešavača, neki algoritmi i rešavač ProbSAT. Mašinsko učenje, vrste mašinskog učenja, klasifikacija i njene primene, Bernulijeva raspodela i određivanje modela logističke regresije biće diskutovani u trećoj glavi. U četvrtoj glavi će biti opisani metodologija predviđanja polariteta promenljivih u zadovoljavajućoj valuaciji, eksperimenti vezani za treniranje modela i proveru performansi modifikovanog SAT resavača. U petoj glavi će biti izvedeni zaključci i biće predlozene ideje za dalji rad.

2. SAT problem i stohastički SAT rešavači

2.1. SAT problem i njegove primene

Iskazna logika je podoblast matematičke logike, koja se bavi istinitošću iskaznih formula. Prikaz osnovnih pojmova iskazne logike dat je u skladu sa knjigom Matematička logika u računarstvu [10]. Iskazne formule se takođe nazivaju i logički iskazi. *Logičke formule* se grade pomoću logičkih promenljivih i logičjih veznika. *Logičke promenljive* mogu biti tačne ili netačne. One predstavljaju određene iskaze. Logičke promenljive su elementarne logičke formule. *Logički veznici* spajaju logičke promenjlive u logičke formule. Oni mogu biti unarni i binarni. Unarni je ¬ (ne). Binarni su ∨ (ili), ∧ (i), ⇒ (sledi) i ⇔ (ekvivalentno).

Sintaksu iskazne logike predstavljaju pravila za izgradnju iskaznih formula od drugih iskaznih formula i logičkih veznika. Ona isključivo govori o formulama kao nizovima simbola i ne uzima u obzir njihovo potencijalno značenje.

Definicija. 2.1 Alfabet Σ je unija sledeca četiri skupa:

1. Prebrojivog skupa iskaznih slova P;
2. Skupa logičkih veznika {¬, ∧, ∨, ⇒, ⇔} pri čemu je ¬ unarni veznik, a ∧, ∨, ⇒ i ⇔ su binarni veznici;
3. Skupa logičkih konstanti {⊤, ⊥};
4. Skupa pomoćnih simbola {(,)}.

Definicija. 2.2 *Skup iskaznih formula* (ili jezik iskazne logike) nad skupom P je najmanji podskup skupa svih reči nad Σ takav da važi:

- Iskazna slova (iz skupa P) i logičke konstante su iskazne formule;
- Ako su A i B iskazne formule, onda su i (¬A), (A ∧ B), (A ∨ B), (A ⇒ B) i (A ⇔ B) iskazne formule.

Semantika iskazne logike je aspekt iskazne logike koji se bavi istinitošću konstruisanih iskaza.

Ekvivalencija i implikacija se mogu predstaviti pomocu negacije, konjunkcije i disjunkcije, tako da su od veznika negacija, konjunkcija i disjunkcija dovoljne za predstavljanje svih logičkih formula.

Atomičke iskazne formule su iskazne promenljive i logičke konstante. *Literali* su atomičke iskazne formule i njihove negacije. Literal koji je negiran se naziva *negativni literal*. Inače *literal* je *pozitivan*. *Klauza* je jedan literal ili disjunkcija više literala.

Definicija. 2.3 Klauza iskazne formule koja sadrži najviše jedan pozitivan literal je *Hornova klauza*.

Definicija. 2.4 Ako u Hornovoj klauzi postoji literal koji je pozitivan, taj literal se naziva *posledica*.

Definicija. 2.5 Ako u formuli postoji bar jedna klauza koja sadrži promenljive x i y, tada kažemo da su x i y susedi.

Definicija. 2.6 Funkcije v koje slikaju iz skupa P u {0, 1} zovemo *valuacije*. Skup {0, 1} zovemo domenom valuacije.

Definicija. 2.7 Svaka valuacija v određuje funkciju I_v koja je *interpretacija* za valuaciju v i koja preslikava skup iskaznih formula u skup {0, 1}. *Interpretaciju* I_v definišemo na sledeći način:

- $I_v(p) = v(p)$, za svaki element p skupa P;
- $I_v(\top) = 1$ i $I_v(\bot) = 0$;
- $I_v(\neg A) = 1$ ako je $I_v(A) = 0$ i $I_v(\neg A) = 0$ ako je $I_v(A) = 1$;
- $I_v(A \wedge B) = 1$ ako je $I_v(A) = 1$ i $I_v(B) = 1$; $I_v(A \wedge B) = 0$ inače;
- $I_v(A \vee B) = 0$ ako je $I_v(A) = 0$ i $I_v(B) = 0$; $I_v(A \vee B) = 1$ inače;
- $I_v(A \Rightarrow B) = 0$ ako je $I_v(A) = 1$ i $I_v(B) = 0$; $I_v(A \Rightarrow B) = 1$ inače;
- $I_v(A \Leftrightarrow B) = 1$ ako je $I_v(A) = I_v(B)$; $I_v(A \Leftrightarrow B) = 0$ inače.

Definicija. 2.8 Ako postoji valuacija v nad P takva da je $I_v(F) = 1$, onda se kaže da je *iskazna formula zadovoljiva*. Inače je nezadovoljiva. Iskazna formula F je tautologija

ako za svaku valuaciju v važi da je $I_v(F) = 1$. U suprotnom, ako postoji valuacija v takva da važi $I_v(F) = 0$, tada se kaže da je iskazna formula poreciva.

Definicija. 2.9 Iskazna formula koja je konjukcija klauza je u *konjuktivnoj normalnoj formi* - KNF.

To su formule oblika: $C_1 \wedge C_2 \ldots \wedge C_n$, pri čemu su C_i klauze, to jeste oblika: $A_1 \vee A_2 \vee \ldots \vee A_n$, gde su A_i literali.

Definicija. 2.10 *SAT problem* je problem provere zadovoljivosti iskaznih formula za koji je na ulazu data iskazna formula u KNF. Potrebno je odediti da li je data formula zadovoljiva i ako jeste naći valuaciju koja je zadovoljava.

SAT rešavač je program koji služi za rešavanje SAT problema. Sat rešavači se dele u dve grupe - potpuni SAT rešavači i nepotpuni SAT rešavači. Potpuni SAT rešavači uvek nalaze zadovoljavajuću valuaciju, ili ustanovljavaju da ona ne postoji. Zasnovani su na DPLL proceduri. Nepotpuni SAT rešavači zasnovani su na lokalnoj pretrazi. Oni služe da bi odredili valuaciju koja zadovoljava iskaznu formulu, ali ne mogu da ustanove nezadovoljivost.

O potpunim i nepotpunim SAT rešacačima će biti više rečeno u glavama 2.2 i 2.3. Njihov opis će biti dat u skladu sa tekstom Karle Gomez i koautora [11].

SAT problem je NP-kompletan. Zato ga za mnoge njegove instance nije moguće rešiti tako što ćemo ispitati sve slučajeve, već moramo koristiti razne heuristike. SAT problem je jedan od NP-kompletnih problema koji su najviše izučavani. Jedan od razloga za to je njegova široka primena, počevši od verifikacije hardvera, pa do planiranja rasporeda.

2.2 Potpuni SAT rešavači

Potpuni SAT rešavači kao ulaz primaju formulu u KNF obliku i onda proveravaju da li je ona zadovoljiva ili ne. U slučaju da je zadovoljiva nalaze valuaciju koja je zadovoljava. Osnova potpunih sat rešavača je DPLL procedura [12, 13] koja vrši bektreking nad prostorom valuacija u potrazi za valuacijom koja bi zadovoljila datu formulu. Osnovna prednost koja daje efikasnost varijantama DPLL procedure je odsecanje stabla u pretrazi pri određivanju valuacije. To se radi kada se naiđe na klauzu sa samo jednim literalom i onda se zna koja vrednost tog literala mora biti u

dubljim pozivima. Celo podstablo pretrage sa suprotnom vrednošću literala se odseca, jer će za sve valuacije u tom podstablu ta klauza biti netačna, pa ni jedna od njih neće zadovoljiti formulu.

DPLL-rekurzivno(F,v) skicira rad osnovne DPLL procedure na KNF formulama. Osnovna ideja je da se nađe promenljiva p kojoj nije odredjena vrednost i rekurzivno se poziva funkciju za $F_{|p}$ (u F se p menja sa $\top$) i $F_{|\neg p}$ (u F se p menja sa $\bot$). Odluka da li će p biti tačno ili netačno naziva se grananje. Dubina rekurzije je uvek manja ili jednaka broju promenljivih. Da bi se ubrzao agloritam, kada god klauza sadrži jedan literal, tada se promenljivoj iz tog literala određuje polaritet bez grananja, već u skladu sa polaritetom tog literala.

Algoritam : DPLL-recursive(F,v)

```
Ulaz: Formula F u KNF obliku i
        v - lista literala kojima smo odredili polaritet, inicijalno prazan

        Izlaz: NEZADOVOLJAVA - u slučaju da je formula nezadovoljiva,
        inače vraća ZADOVOLJIVA i ispisuje valuaciju koja zadovoljava formulu.

Begin
        (F,v) = PropagacijaJedinicnihLiterala (F,v);

        if ( F sadrži praznu klauzu)
                return NEZADOVOLJIVA;

        if ( u F nema više klauza)
        Begin
                ispiši v;
                return ZADOVOLJIVA;
        End

         p = literal čiji polaritet nije određen valuacijom  v;
        if(DPLL-recursivno (F|p, [p|v]) ==  ZADOVOLJIVA)
                return ZADOVOLJIVA;

        return DPLL-recursivno (F|¬p, [¬p|v]);
End
```

```
funkcija PropagacijaJedinicnihLiterala(F,v)
Begin
        While (F nema praznih klauza k u F postoji klauza sa samo jednim literalom p)
        Begin
                v = [p | v];
                F = F|p ;
        End
        return(F,v);
End
```

Tokom godina razvijena su mnoga poboljšanja ovog algoritma, koja koriste razne metode iz različitih oblasti matematike i računarstva. Često možemo kombinovati više poboljšanja i tako dobiti bolji rešavač.

Neka od najvažnijih poboljšanja su sledeća:

Heuristika za odabir promenljive i njene vrednosti – ovo je važan izbor u dizajnu SAT rešavača, koji se često razlikuje od rešavača do rešavača i može imati jako veliki uticaj nanjihovu efikasnost. Neki rešavači na slučajan način biraju promenljivu, ali većina njih to radi tako što maksimizuje neku funkciju. Ta funkcija zavisi od raznih karakteristika te promenljive i klauza u kojima se ona pojavljuje.

Učenje klauza - ovo je jedna od modifikacija koje su donele najveća poboljšanja u prethodnih dvadeset godina. Osnovna ideja je da se prilikom ustanovljavanja konflikta (situacija u kojoj je neka klauza netačna u tekućoj valuaciji) utvrde dublji razlozi tog konflikta i da se zapamte u vidu klauze koja će u budućnosti onemogućiti ponovno pojavljivanje konflikta iz istog ili sličnih razloga.

Postoji još dosta različitih poboljšanja, ali se nećemo dublje baviti njima.

2.3 Nepotpuni SAT rešavači

Nepotpuni SAT rešavači ne garantuju da će ili naći valuaciju koja zadovoljava datu formula ili dokazati da je formula nezadovoljiva. Takvi metodi se obično pokreću sa unapred fiksiranim vremenskim ograničenjem posle koga možemo, a i ne moramo dobiti rešenje. Za razliku od potpunih SAT rešavača koji su bazirani na bektrekingu, nepotpuni SAT rešavači rade na principu stohastičke lokalne pretrage.

Prva primena lokalne pretrage bila je na problem raspoređivanja N kraljica na šahovsku tablu veličine N za veliko N, koristeći GSAT (greedy local search). U početku je vladalo mišljenje da je ovaj metod dao poboljšanje jer je primenjen na lak problem, kao i da lokalna pretraga neće biti uspešna kod drugih primera. Ipak eksperimenti su pokazali suprotno. GSAT je često radio brže od sistematičnih potpunih pretraga.

GSAT je zasnovan na slučejnoj lokalnoj pretrazi. U nastavku je data osnovna GSAT procedura, na koju su kasnije dodavana razna poboljšanja. Algoritam prvo na slučajan način definiše valuaciju. Zatim se menja polaritet jedne promenljive najviše MAX-PROMENA puta. Može se i ranije prestati sa menjanjem polariteta ako formula bude zadovoljena. Ona se bira tako da broj zadovoljenih klauza posle toga bude maksimalan. Ako se dostigne MAX-PROMENA pre nego što se nađe tražena valuacija, vraća se na slučajno generisanje valuacije. To se radi najviše MAX-POKUŠAJA puta.

Algoritam: GSAT(F)

Ulaz: Formula F u KNF

Parametri: MAX-PROMENA, MAX-POKUŠAJA

Izlaz: Valuacija koja zadovoljava formulu, ili NENAĐENA

```
Begin
        for i = 1 to MAX-POKUŠAJA
        Begin
                v - slučajno generisana valuacija;

                for j = 1 to MAX-PROMENA
                Begin
                        if(s zadovoljava F)
                                return v;
                        p =  promenljiva takva da ako joj se promeni polaritet,
                                broj zadovoljenih klauza ce biti maksimalan;
                        Zameni polaritet p u v;
                End
        End
        return NENAĐENA;
End
```

Eksperimenti su pokazali da GSAT gubi dosta vremena da dođe do lokalnog minimuma i da mu vrlo često to čak i ne uspeva. Zbog toga su se mnogi istraživači bavili time kako da ubrzaju ovaj proces. Jedno od uspešnih ubrzanja traženja lokalnog minimuma je Walksat.

Walksat poboljšava lokalnu pretragu GSAT-a tako što promenljivu kojoj će promeniti polaritet uvek bira iz neke nasumično izabrane nezadovoljene klauze. Prvo na slučajan način bira nezadovoljenu klauzu C. Kada se nekoj promenjivoj iz literala koji pripada klauzi C menja polaritet, bitno je da to ne učini nezadovoljenom neku već zadovoljenu klauzu. Ako postoji takva promenljiva, onda se njoj menja polaritet. U suprotnom sa verovatnoćom P se bira slučajan literal iz klauze, a sa verovatnoćom 1-P bira se litaral koji će minimizovati vrednost funkcije BrojPokvarenih. Vrednost funkcije BrojPokvarenih je broj klauza koje su bile zadovoljene, a posle promene polariteta promenljive više nisu zadovoljene. U ovom algoritimu pored parametara MAX-PROMENA i MAX-POKUŠAJA imamo i parametar p iz [0, 1] koji predstavlja verovatnoću sa kojom se bira slučajni literal iz klauze. U nastavku je dat pseudokod algoritma.

Algoritam: Walksat(F)

Ulaz: Formula u KNF

Parametri: MAX-PROMENA, MAX-POKUŠAJA i p

Izlaz: Valuacija koja zadovoljava F, ili NENAĐENA

```
Begin
    for i = 1 to MAX-POKUŠAJA
    Begin
        v - slučajno generisana valuacija;
        for j = 1 to MAX-PROMENA
        Begin
            if(v zadovoljava F)
                return v;
            Slučajno se bira nezadovoljena klauza C iz F;
            if (postoji promenljiva x iz C takva da BrojPokvarenih (F,v,x) = 0 )
                p = x;
            else
                sa verovatnoćom P, p je slučajno izabrana promenljiva iz C,
                sa verovatnoćom 1-P, p je promenljiva iz C koja minimizuje
                BrojPokvarenih (F,v,x);
```

```
                    Zameni polaritet p u v;
                End
            End
            return NENAĐENA;
End
```

2.4. Primena SAT rešavača

U današnje vreme SAT rešavači imaju veoma široku primenu. Koriste se u raznim oblastima. U nastavku će biti opisano nekoliko primera primene SAT rešavača.

Provera ekvivalencije kombinatornih kola [5]. Osnovni problem je proveriti funkcijonalnu ekvivalenciju dva kola. Neka su C_A i C_B dva kola i neka oba imaju ulaz x_1, ... x_n i oba imaju izlaz dimenzije m. C_A ima izlaz y_1, ... ,y_m ,a C_B ima izlaz w_1, ... , w_m. Treba proveriti da li za iste ulaze kola C_A i C_B daju iste izlaze. Vrednosti na izlazima kombinatornih kola se mogu predstaviti odgovarajućim formulama Y_i i W_i za $1 \leq i \leq m$. Provera funkcionalne korektnosti se vrši tako sto se pokuša ustanoviti nezadovoljivost formula $\neg(Y_i \Leftrightarrow W_i)$. Ukoliko se pronađe valuacija koja zadovoljava neku od ovih formula, ona predstavlja ulaz za koji kola nisu ekvivalentna.

Ograničena provera modela (OPM, eng. bounded model checking) [6]. Trenutno najrasprostranjenija primena SAT rešavača je provera ograničenih modela. Provera modela je automatizovana tehnika za proveru da li data implementacija sistema zadovoljava date uslove, specifikovane u nekoj logici. Odgovor može ili biti “da” u slučaju da data implementacija zadovoljava date uslove, ili “ne” kada će biti vraćena i sekvenca operacija koja je navela sistem na grešku. Ranije su pri traženju greške bili korišćeni binarni dijagrami odlučivanja (BDO, eng. binary decision diagrams), sistemi koji su proveravali sva moguća stanja u kojima bi se sistem mogao naći. Ipak memorijska i vremenska složenost takve provere su jako velike.

Kao zamena za BDO 1998. godine predstavljena je OPM. OPM je parametrizovana parametrom n koji označava najveću dubinu pretrage, to jest proverava se da li će sekvence dužine ne veće n dovesti do greške u sistemu.

Model OPM pretpostavlja da se svako stanje sistema može označiti kao povoljno(stanje koje ispunjava željene osobine), ili nepovoljno(stanje koje ne ispunjava željene osobine). Pretpostavlja se da se svako stanja sistema može predstaviti kao konačan vektor logičkih promenljivih S. Sigurnosna formula je predstavljena sa P(S) koja je tačna za povoljna stanja s, a netačna za nepovoljna stanja S. Za inicijalna stanja sistema se koristi formula I(S) koja je tačna samo ako je stanje S inicijalno. Prelazi

između stanja u sistemu modelovani su formulom $T(S_1, S_2)$ koja je tačna samo ako je u sistemu moguće preći iz stanja S_1 u stanje S_2. U slučaju da je sledeća formula zadovoljena za neke S_0 ... S_n sistem nije ispravan jer se može od početnog stanja S_0 doći do nekog nepovoljnog stanja. Pri traženju takvih S_0 ... S_n se koriste SAT rešavači.

$$I(S_0) \wedge (T(S_0, S_1) \wedge T(S_1, S_2) \wedge ... \wedge T(S_{n-1}, S_n)) \wedge (\neg P(S_0) \vee \neg P(S_1) \vee ... \vee \neg P(S_n))$$

Pravljenje rasporeda časova [7]. Pri kreiranju rasporeda časova često je bitno ispuniti kriterijume kao što su:

- korektnost rasporeda - svaki čas se mora naći na rasporedu, grupa ne sme imati više časova istovremeno, profesor ne sme imati više časova istovremeno, ne mogu se u istoj učionici istovremeno održati više časova
- Neželjeni i željeni časovi rada – profesorima je dozvoljeno da navedu neke termine kada su zauzeti i kada ne mogu imati časove, nekim profesorima je dozvoljeno da navedu vreme kada bi im najviše odgovaralo da imaju časove...
- Broj radnih dana – neki profesor može tražiti da mu svi časovi budu smešteni u određenom broju radnih dana.
- Trajanje radnog dana – broj časova grupe tokom radnog dana može biti ograničen, broj časova profesora tokom radnog dana može biti ograničen.
- I još mnogi drugi.

Na osnovu zadatih uslova može se kreirati model u obliku formule u KNF, gde je za nalaženje željenog rasporeda dovoljno naći valuaciju koja zadovoljava dobijenu formulu, to jest rešiti SAT problem.

2.5. ProbSAT

U okviru ovog rada će biti modifikovan SAT rešavač ProbSAT[1]. Zato je ovo poglavlje posvećeno ProbSAT-u i njegovom načinu rada. Prob-SAT spada u stohastičke SAT rešavače.

Rad rešavača ProbSAT se ugrubo može skicirati na sledeći način.

1. Na slučajan način se bira počenta valuacija v.
2. Ako valuacija v zadovoljava formulu dolazi se do rešenja i prelazi na korak 4.

3. Bira se promenljiva kojoj će se promeniti vrednost. Menja se vrednost toj promenjivoj i prelazi na korak 2.

4. Ispisuje se valuacija koja je nađena i izlazi se iz programa.

Heuristički metod za biranje promenljive je najčešće zasnovan na nekom probabilističkom modelu po kome se funkcijom izaberiVar slučajno bira promenljiva kojoj će biti zamenjena vrednost. Različite promenljive u datom trenutku biramo sa različitim verovatnoćama. Te verovatnoće najčešće zavise od vrednosti kao što su: starost - koliko dugo promenljiva nije menjala vrednost, broj zadovoljenih i nezadovoljenih klauza u kojima se pojavljuje promenljiva itd.

Kao najuspešniji način odabira promenjive se pokazalo biranje promenjive koja se nalazi u klauzi koja nije zadovoljena. Najbitnija stvar pri odabiru promenljive je broj zadovoljenih klauza pri određenoj valuaciji. Recimo, može se uzeti u obzir sledeći skor koji opisuje poželjnost promene vrednosti promenljive:

$$skor(x) = BD(x, a) - BP(x,a)$$

pri čemu je BP(x, a) broj pokvarenih klauza pri promeni vrednosti promenljive x u valuaciji v, to jest broj klauza koje su pre promene polariteta promenjive x bile zadovoljene, a promenom polariteta promenjive x postaju nezadovoljene. BD(x, a) je broj dodatih klauza pri promeni vrednosti promenljive x u valuaciji v, to jest broj klauza koje su pre promene polariteta promenjive x bile nezadovoljene, a promenom polariteta promenjive x postaju zadovoljene. Primetimo da pri ovakvom odabiru promenljive BD uvek mora biti bar 1.

ProbSAT rešavač odluku kojoj promenljivoj će promeniti polaritet bazira na vrednostima BD i BP, ali ne obavezno BD - BP. U eksperimentima je pokazano da vrednost BD pri odabiru promenljive nema veliki uticaj na performanse rešavača, pa se često može i zanemariti i odluku o odabiru promenljive možemo bazirati na BP.

ProbSAT rešavač radi po sledećem algoritmu:

Algoritam: ProbSAT(F)

Ulaz: Formula F u KNF

Parametri: MAX-PROMENA, MAX-POKUŠAJA

Izlaz: Valuacija koja zadovoljava formulu, ili NENAĐENA

```
Begin
    for i = 1 to MAX-POKUŠAJA
    Begin
        v - slučajno generisana valuacija;
        for j = 1 to MAX-PROMENA
        Begin
            if(s zadovoljava F)
                return v;
            Slučajno izaberemo nezadovoljenu klauzu C iz F;
            Za svaku promenljivu x iz klauze C izračunati f(x, v);
            Promenljiva p iz klauze C se uzima slučajnim izborom  sa
            verovatnoćom:
            f(x, s) / ∑_{z∈C} f(z, s) , za promenljivu x;

            Zameni polaritet p u v;
        End
    End
    return NENAĐENA;
End
```

Naveden je uopšten algoritam za ProbSAT rešavače, tako da još treba odrediti funkciju f. Najčešće se za f koriste eksponencijalna i polinomijalna funkcija.

Za eksponencijalnu funkciju se koristi:

$$f(x, a) = \frac{(C1)^{BD(x,a)}}{(C2)^{BI(x,a)}}$$

Eksperimentalno treba izabrati konstante C1 i C2. Algoritam koji je zasnovan na toj funkciji naziva se eksponencijalni ProbSAT algoritam.

Kao što je već navedeno BD ima mali uticaj na performanse rešavača, pa se često može zanemariti i onda funkcija glasi:

$$f(x, a) = (C2)^{-BI(x,a)}$$

Za polinomijalnu funkciju koristi se sledeća funkcija:

$$f(x, a) = \frac{(BD(x,a))^{C1}}{(eps + BI(x,a))^{C2}}$$

Pri čemu je eps neki mali broj

I ovde eksperimentalno, posmatrajući performase rešavača, treba odrediti konstante C1 i C2. Algoritam koji je zasnovan na toj funkciji je polinomijalni ProbSAT algoritam. Kao kod eksponenicjalne funkcije i ovde se može zanemaiti BD i tada se dobija sledeća funkcija:

f(x, a) = $(eps + BI(x,a))^{-C2}$

U zavisnosti od familije SAT formula, varijante ProbSAT rešavača mogu nekad imati bolje performanse kada koristi eksponencijalnu, a nekad polinomijalnu funkciju.

3. Mašinsko učenje i klasifikacija

U poslednje vreme u računarstvu se sve češće koriste velike količina podataka. Radi efikasnije obrade, potrebno je o tim podacima donositi izvesne zaključke. Na primer, internet pretraživači (Google, Bing, Yandrex...) svakodnevno prihvataju milione upita različitih kategorija. Ti upiti traže veoma raznovrsne internet stranice. Neki od njih se odnose na naučne časopise, neki na hotele i slično. Zbog jako velikog broja upita i internet stranica, nemoguće ih je razvrstati "ručno". Korišćenje programa koji vrši razdvajanje po određenim kriterijumima, u velikoj meri olakšava pretragu.

Za ovakav i druge slične probleme, kada imamo veliki broj podataka o kojima treba donositi potrebne zaključke, koristi se mašinsko učenje. *Mašinsko učenje* predstavlja skup metoda koji u podacima mogu automatski da uoče neki šablon, koji se dalje koristi da bi se napravila određena predviđanja vezana za druge podatke slične prirode. Najčešće se u tu svrhu koriste razni probabilistički modeli i druga znanja iz teorije verovatnoće. Pregled pojmova vezanih za masinsko ucenje dat je u skladu sa knjogom Machine learning - a probabilistic perspective [2].

Mašinsko učenje se deli na nadgledano i nenadgledano učenje.

Glavna razlika je u tome što se u *nadgledanom mašinskom* učenju podrazumeva da je za svaku instancu data vrednost neke relevantne funkcije $D = \{(x_i, y_i)\}_{i=1...N}$. Cilj je predvideti vrednosti funkcije y za neko novo x. Nadgledano mašinsko učenje će detaljnije biti predstavljeno u glavi 3.1.

Nenadgledano mašinsko učenje donosi zaključke samo na osnovu podataka x. Program kao ulaz ima samo $D = \{x_i\}_{i=1...N}$. Cilj je da se naprave odeđeni zaključci o podacima, to jest da se otkriju neki zanimljivi šabloni.

3.1. Nadgledano mašinsko učenje

Cilj nadgledanog mašinskog učenja je da predvidi funkciju koja pridružuje vrednost y objektu x. Funkcija koja pridružuje vrednost y objektu x naziva se *ciljna funkcija*. Za to se koristi skup D, koji se naziva trening skup. On sadrži parove ulaza i izlaza.

$$D = \{(x_i, y_i)\}_{i=1...N}$$

Obično su x_i vektori brojeva koji određuju neke karakteristike. Na primer to mogu biti težina i visina ljudi. Te veličine se nazivaju atributi. Ipak, nekada x_i mogu biti složene strukture ili objekti, kao što su na primer slika, rečenica, DNK, graf i slično.

Izlaz kod nadgledanog učenja je najčešće element iz konačnog skupa (u slučaju klasifikacije) ili broj (u slučaju regresije).

Vrsta mašinskog učenja koja za izlaz vraća broj zove se regresija. Ona se koristi kada je potrebno predvideti određenu numeričku vrednost na osnovu postojećih podataka. Na primer koristi se ako je potrebno predvideti starost osobe na osnovu fotografije lica, cenu stana na osnovu kvadature i udaljenosti od centra grada, kretanje cena akcija, kao i u slučaju puno drugih različitih primera.

3.2 Klasifikacija

Klasifikacija je vrsta problema nadgledanog mašinskog učenja. Ciljna funkcija klasifikacije za svoju vrednost ima element iz konačnog skupa. y_i je iz skupa {1, 2... C}. Algoritmi klasifikacije za učenje koriste skup $D = \{(x_i, y_i)\}_{i=1...N}$.

Na osnovu skupa D treba da konstruiše preslikavanje iz domena X u skup {1, 2... C}. Klasifikacija u stvari treba da podeli elemente, na osnovu nekih njihovih svojstava, u C klasa. Specijalan primer klasifikacije za C=2 se zove *binarna klasifikacija*. Tada se najčešće elementi kodomena obeležavaju sa {0, 1}. U slučaju da je C > 2 u pitanju je *višeklasna klasifikacija*.

U nekim problemima, potrebno je da klasifikacija osim toga što će predvideti da će element pripadati nekoj klasi, predvidi i verovatnoću sa kojom će element pripadati toj klasi. Verovatnoća da će x pripadati klasi označenoj sa y, ocenjena na osnovu datog trening skupa D obeležava se sa p(y | x, D). U opštem slučaju p je vektor dužine C.

Za binarnu klasifikaciju p je jedan broj p(y = 1 | x, D), jer važi

$$p(y = 0 \mid x, D) = 1 - p(y = 1 \mid x, D).$$

Na osnovu vektora verovatnoća uvek je moguće odrediti najverovatniju predikciju, to jest izabrati kao predikciju za x klasu za koju je naveća verovatnoća da joj x pripada.

$$\hat{y} = \underset{c}{argmax}\, p(y = c|x, D)$$

Može se desiti da verovatnoća za klasu za koju je najveća, ipak nije dovoljno velika i da program treba da odgovori da nije siguran kojoj klasi pripada x. To može biti veoma bitno kod nekih primera kao što su finansije, medicina i slično.

Klasifikacija je verovatno najkorišćenija vrsta mašinskog učenja. Koristi se da bi se rešili neki veoma interesantni i često veoma teški zadaci. Već su navedeni neki problemi klasifikacije. U nastavku su navedeni neki interesantni primeri primene klasifikacije.

Klasifikacija dokumenata i određivanje neželjene pošte

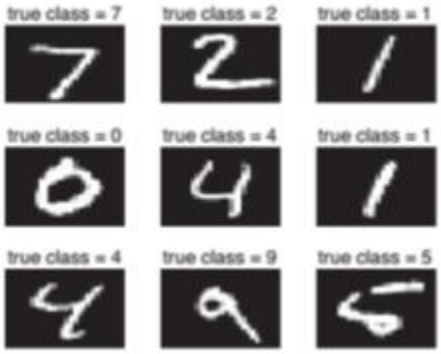
U klasifikaciji dokumenata, glavni cilj je rasporediti dokumente kao što su web stranice ili e-mail poruke u C klasa, to jest izračunati p(y = c | x, D). x je prezentacija određenog teksta. Specijalni slučaj klasifikacije dokumenata je određivanje neželjene pošte. Potrebno je da se e-mail poruke podele u dve klase. Prva klasa je y = 1, za neželjene poruke. Druga klasa je y = 0, za poruke koje nisu neželjene. Većina klasifikatora podrazumeva da vektor x ima fiksnu dužinu. Najčešći način da se predstavi dokument kao vektor atributa je skup neželjenih reči. Klasifikator može primetiti reči koje su najčešće u porukama neželjene pošte i na osnovu njih doneti određene zaključke. Te reči su najčešće "kupi", "jeftino", "vijagra" i slične.

Klasifikacija slike

Direktna klasifikacija slike, za koju čovek nije posebno obradio podatke, je svakako veći problem. Kritierijmi po kojima slike mogu biti klasifikovane mogu biti razni. Na primer da li slika prikazuje zatvoreni ili otvoreni prostor, da li je horizontalna ili vertikalna, da li je na njoj određeni predmet ili osoba...

Klasifikacija rukom pisanih slova ili brojeva

U specijalnom sličaju može se vršiti *klasifikacija rukom pisanih slova ili brojeva*. Na primer pošta može klasifikovati pošiljke na osnovu ispisanih poštanskih kodova. Mešoviti standardni skup podataka koji se koristi u ovoj oblasti poznat je kao MNIST koji je standardizovan od strane američkog “Nacionalnog instituta za standarde i tehnologiju“.



Prvih 9 instanci iz MNIST skupa u sivim tonovima

Pronalaženje predmeta u okviru slike

Složeniji problem je *pronalaženje predmeta u okviru slike*. Važan specijalni slučaj je pronalaženje lica, kao što je ilustrovano na sledećoj slici. Ovaj sistem je ugrađen u većinu modernih digitalnih foto aparata. Lokacije otkrivenih lica se mogu koristiti za različite primene, na primer auto-focus, automatsko zamagljivanje u Google-ovom StreetView sistemu i sl. Kada sistem pronađe lice, može nastaviti ka prepoznavanju, to jest ka proceni indentiteta osoba. U tom slučaju broj mogućih klasa, često je veoma veliki.

Primer detekcije lica: (a) ulaz (b) izlaz sa otktivenih 5 lica u različitim položajima

3.3 Bernulijeva raspodela

U teoriji verovatnoće, Bernulijeva raspodela predstavlja poseban slučaj raspodele sa dva moguća ishoda. Verovatnoća ishoda 1 je p, a verovatnoća ishoda 0 je 1 – p za

$0 \leq p \leq 1$.

Očekivanje slučajne promenljive sa Bernulijevom raspodelom je E(X) = p, a disperzija slučajne promenljive sa Bernulijevom raspodelom jednaka je D(X) = p (1-p)

Bernulijeva raspodela je pogodna za modelovanje binarne klasifikacije tako što bi se parametar p predivdeo na osnovu nekih atributa instance.

3.4 Logistička regresija

Logistički model je metod za rešavanje problema klasifikacije. On služi rešavanju problema klasifikacije tako što ne samo da određuje kojoj klasi pripada određeni element x, već i koja je verovatnoća da x pripada toj klasi. To se radi tako što se pravi probabilistički model za p(y | x). Određuje se verovatnoća da element x pripada klasi y. Logistički model se odnosi na binarnu klasifikaciju, to jest $y \in \{0, 1\}$.

U modelu su verovatnoće p(y | x) određene Bernulijevom raspodelom:

$$p(y \mid x, w) = Ber(y \mid \mu(x))$$

gde je:

$$\mu(x) = E[y \mid x] = p(y = 1|x)$$

μ(x) se računa na osnovu linearne kombinacije ulaza x. Da bi se veovatnoća skalirala na interval [0, 1] koristi se funkcija σ. Funkcija σ se naziva *sigmoidna funkcija.* Naziv logistička funkcija je takođe u upotrebi. Definiše se na sledeći način:

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

Očekivanje Bernulijeve raspodele μ(x) se modeluje kao:

$\mu(x) = \sigma(w^T x)$

Stoga se uslovna verovatnoća modeluje kao:

$p(y|x,w) = Ber(y|\ \sigma(w^T x))$

Ovom relacijom je definisan logistički model.

3.4.1 Ocena parametara logističkog modela

U ovoj sekciji akcenat će biti na ocenjivanju parametara w logističkog modela. Funkcija E(w), koja određuje grešku pri izboru vrednosti parametara w u odnosu na trening skup, zapravo je negativna vrednost logaritma funckije verodostojnosti. Data je na sledeći način:

$$E(w) = -\sum_{i=1}^{N} \log[\mu_i^{-I(y_i=1)} \cdot (1-\mu_i)^{I(y_i=0)}] =$$

$$= -\sum_{i=1}^{N} [y_i \cdot log\ (\mu_i) + (1-y_i) \cdot log\ (1-\mu_i)]$$

U nekim slučajevima pogodnije je izabrati da važi y ∈ {-1, 1} , umesto y ∈ {0, 1}. Tada se E(w) računa na sledeći način:

$$E(w) = \sum_{i=1}^{N} \log[1 + e^{-y_i w^T x_i}]$$

Parametri w se biraju tako da vrednost E(w) bude minimalna, a da se to radi nekim algoritmom optimizacije. Ti algoritmi koriste gradijent i Hesijan (matricu drugih parcijalnih izvoda funkcije).

$$g = \frac{d}{dw} f(w) = \sum_i (\mu_i - y_i) \cdot x_i = X^T \cdot (\mu - y)$$

$$H = \frac{d}{dw} g(w)^T = \sum_i (\nabla_w \mu_i) \cdot x_i^T = \sum_i \mu_i \cdot (1 - \mu_i) \cdot x_i \cdot x_i^T = X^T \cdot S \cdot X$$

Gde se S definiše kao dijagonalna matrica:

$S = \text{diag}(\mu_i \cdot (1-\mu_i))$

Matrica H je pozitivno definitna. Zato je funkcija E(w) konveksna i ima jedan globalni minimum.

Najjednostavniji algoritam za određivanje minimuma funkcija E(w) je gradijentni spust (eng. gradient decent), gde je η_k veličina koraka. On je dat sledećom formulom:

$$\theta_{k+1} = \theta_k - \eta_k g_k$$

Glavni problem kod gradijentnog spusta je kako odrediti veličinu koraka. To se ispostavlja kao netrivijalan zadatak. Može se η_k postaviti na neku konstantu, međutim ako je η_k isuviše malo, niz može vrlo sporo konvergirati, a ako je η_k isuviše veliko niz može divergirati.

Zbog svega toga treba izabrati bolji metod za biranje veličine koraka, koji će nam garantovati konvergenciju bez obzira na polaznu poziciju. Po Tejlorovoj formuli važi:

$$f(\theta + \eta d) \approx f(\theta) + \eta\, g^T d$$

pri čemu je d pravac pomeranja.

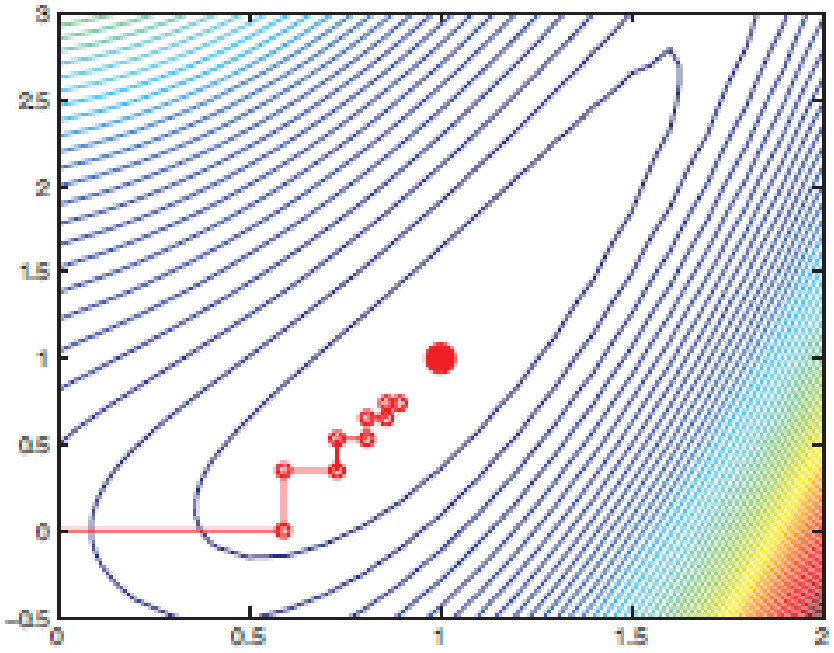
Ako se za η izabere dovoljno mala vrednost, onda važi:

$$f(\theta + \eta d) < f(\theta)$$

Da bi se što brže približili globalnom minimumu, bira se η koje će minimizovati:

$$\varphi(\eta) = f(\theta_k + \eta\, d_k)$$

To se zove *linearna minimizacija*, ili *linearna pretraga*. Postoji puno metoda koje rešavaju taj problem. Na sledećoj slici se može videti primer korišćenja ovog metoda.



Može se primetiti da pri spuštanju vektor uvek menja pravac, tako da je normalan na predhodni vektor. To se može objasniti tako što se uvek bira η_k tako da minimizije funkciju φ(η). Kada se minimizuje funkcija φ(η) tada važi:

$$\varphi'(\eta) = 0$$
$$\varphi'(\eta) = d^T g$$

gde je $g = f'(\theta + \eta\, d)$ gradijent na kraju koraka (početak sledećeg koraka).

Postoje dve mogućnosti. Može biti da je g = 0, što bi značilo da je minimum nađen, ili da su d i g normalni što znači da će gradijent u sledećem koraku biti normalan na gradijent u predhodnom koraku. To objašnjava cik-cak kretanje.

Sledeća heuristika na vrlo jednostavan način smanjuje efekat tog cik-cak kretanja, da bi se brže došlo do globalnog minimum:

$\theta_{k+1} = \theta_k - \eta_k g_k + \mu_k (\theta_k - \theta_{k-1})$,

gde važi $0 \leq \mu_k \leq 1$.

Njutnov metod se može koristiti za minimizaciju bilo koje konvekse funkcije, pa tako i funkcije E(w). Dat je sledećim algoritmom:

Inicijalizuj θ_0;
k= 0;
while $|\theta_i - \theta_{i-1}| > \varepsilon$ do
Begin
Izračunati $g_k = \nabla f(\theta_k)$;
Izračunati $H_k = \nabla^2 f(\theta_k)$;
Rešiti $H_k d_k = -g_k$, po d_k;
Koristeći linearnu pretragu naći η_k na pravcu d_k;
$\theta_{k+1} = \theta_k + \eta_k d_k$;
k++;
End

Primenom Njutnovog metoda na funkciju E(w) dobija se metod *IRLS*. Korak u njutnovom metodu za računanje k+1-og elementa glasi:

$$
\begin{aligned}
w_{k+1} &= w_k - H^{-1} g_k \\
&= w_k - (X^T S_k X)^{-1} X^T (y - \mu_k) \\
&= (X^T S_k X)^{-1} [(X^T S_k X) w_k + X^T (y - \mu_k)] \\
&= (X^T S_k X)^{-1} X^T [S_k X w_k + y - \mu_k] \\
&= (X^T S_k X)^{-1} X^T S_k z_k \qquad (*)
\end{aligned}
$$

Pri čemu se z_k definiše kao:

$z_k = X w_k + S_k^{-1} (y - \mu_k)$

Data jednakost (*) je primer težinskog problema minimizacije kvadrata za:

$$\sum_{i=1}^{N} S_{ki} (Z_{ki} - W^T X_i)^2$$

Algoritam IRLS je dat sledećim pseudokodom:

$w = 0_D$;
$w_0 = \log(\frac{y}{1-y})$
i=0;
repeat
    i++;
    $\eta_i = w_0 + w^T x_i$;
    $\mu_i = sigm(\eta_i)$;
    $s_i = \mu_i(1 - \mu_i)$;
    $z_i = \eta_i + (y_i - \mu_i) / s_i$
    $S = diag(s_{1:N})$;
    $w = (X^T S X)^{-1} X^T Sz$;
while $|w_i - w_{i-1}| < \varepsilon$;

4. Predviđanje polariteta promenljivih u zadovoljavajućoj valuaciji

Stohastički SAT rešavači rade tako što na slučajan način biraju početnu valuaciju i onda u njenoj okolini traže valuaciju koja će zadovoljiti formulu. Osnovna ideja rada je modifikovati SAT rešavač, tako da on više ne kreće od slučajno izabrane valuacije, već približno predviđa zadovoljavajuću, pomoću metoda klasifikacije. Rad se bavi modifikovanjem stohastičkih SAT rešavača, jer se oni mogu prirodno izmeniti tako da na drugačiji način biraju početnu valuaciju. Predviđanje zadovoljavajuće valuacije se vrši tako što se promenljive klasifikuju u tačne i netačne. Takvom klasifikacijom se teško može predvideti zadovoljavajuća valuacija. Ipak, tako dobijena početna valuacija bi mogla biti "bliže" tačnoj i time skratiti vreme rešavanja date instance SAT problema. Za računanje početne valuacije (predviđanje zadovoljavajuće valuacije) se koristi logistički model. Treniranje (kreiranje logističkog modela) se vrši nad jednostavnijim fromulama za rešavanje. One se mogu lako rešiti pomoću originalnog SAT rešavača i tako se može kreirati trening skup. Modifikovan SAT rešavač se kasnije primenjuje na složenije formule. Eksperimenti u ovom radu su urađeni po ugledu na neobjavljeni rad Brajana Silvertorna [1].

U ovoj glavi opisujemo metodologiju za predviđanje polariteta promenljivih u zadovoljavajućoj valuaciji i evaluiramo je na nekoliko familija iskaznih formula. Time vršimo i proveru navoda iz Silvertornovog neobjavljenog rada.

4.1. Pregled metodologije

Metodologija se sastoji od tri osnovne faze. U prvoj fazi se kreira trening skup. U drugoj fazi se vrši treniranje nad dobijenim trening skupom i dobija se klasifikacioni model. U trećoj fazi se vrši pokretanje stohastičkog SAT rešavača koji je u stanju da pomoću dobijenog klasifikacionog modela predvidi početnu valuaciju.

U prvoj fazi metodologije, kao što je već navedeno, se kreira trening skup. Prvo se izabrani stohastički SAT rešavač pokreće nad svim formulama i pamte se nađene valuacije koje zadovoljavaju formule. Zatim se izdvajaju formule koje se koriste za kreiranje trening skupa, kao i formule izabrane za testiranje. Za kreiranje trening skupa biraju se formule koje se procene kao lakše za rešavanje na osnovu dužine rešavanja od strane izabranog SAT rešavača. Preostale formule su formule za testiranje. Radi daljeg testiranje čuvaju se i performanse rešavanja formula. Računaju se atributi promenljivih u svim formulama. Trening skup i test skup se kreiraju tako da jednu

instancu čini vektor atributa promenljive i njena vrednost u valuaciji koju je našao SAT rešavač. U nastavku je sumiran prvi korak metodologije.

1. Pokretanje stohastičkog SAT rešavača nad svim formulama.

2. Izdvajanje formula za kreiranje trening skupa i formula za testiranje.

3. Računanje atributa za sve promenljive svih formula.

U drugoj fazi se kreira klasifikacioni model, to jest vrši se treniranje nad dobijenim trening skupom. Za metodu klasifikacije izabrana je logistička regresija. U ovoj fazi se biraju alati koji će biti korišćeni za računanje parametara logističkog modela. Parametri modela se računaju zasebno za svaku familiju formula i čuvaju se. Nakon toga se, nad test skupom, dobijenim u prvoj fazi, vrši procena preciznosti klasifikacije. Procena preciznosti klasifikacije vrši se kao dijagnostička mera, kako bi se videlo za koje familije je došlo do uspešnog učenja. U nastavku je sumiran drugi korak metodologije.

1. Računaju se parametri logističkog modela na osnovu trening skupa.

2. Vrši se procena preciznosti klasifikacije na osnvu test skupa i konstatuje se za koje familije je došlo do uspesnog učenja .

U trećoj fazi se vrši pokretanje stohastičkog SAT rešavača koji je u stanju da pomoću logističkog modela izračuna početnu valuaciju pri rešavanju SAT problema. Stohastički SAT resavač je izmenjen tako da može da kao argument komandne linije primi putanju fajla u kome se nalaze parametri logističkog modela. U slučaju da putanja takvog fajla nije navedena SAT rešavač će se ponašati kao i pre promena, to jest na slučajan način će izabrati početnu valuaciju. Ako je putanja sa parametrima logističkog modela navedena, učitaće parametre iz datog fajla koje će iskoristiti pri računanju početne valuacije. SAT resavač računa atribute svake promenljive. Za svaku promenljivu primenjuje logistički model za predviđanje na osnovu njenih atributa. Zbog bolje procene efikasnosti, u toku pokretanja SAT rešavača, meri se vreme potrebno za računanje atributa. Nakon toga se meri efikasnost modifikovanog SAT rešavača i poredi se sa efikasnošću originalnog SAT rešavača pre načinjenih promena. U nastavku sumiran treći korak metodologije.

1. Za svaku formulu iz skupa formula koje je potrebno rešiti:
   - Računaju se atributi za svaku od promenljivih
   - Na osnovu atributa i logističkog modela, vrši se predviđanje njihovog polariteta
   - SAT rešavač se pokreće polazeći od predviđene valuacije

2. Vrši se procena efikasnosti SAT rešavača

4.2. Atributi iskaznih promenljivih

Predviđanje polariteta promenljivih u valuaciji je napravljeno po ugledu na Silvertornov neobjavljeni rad. Problem predikcije polariteta promenljivih se može posmatrati kao problem binarne klasifikacije, gde se promenljive klasifikuju u dva skupa tačno i netačno. Kao što je već navedeno, za predikciju polariteta promenljivih koristi se logistički model. Za korišćenje logističkog modela za klasifikaciju potrebno raspolagati atributima instanci. Atributi su navedeni u tabeli koja sledi.

Tabela atributa

clause_length_mu	Aritmetička sredina broja literala u klauzama u kojima se promenljiva pojavljuje.
clause_length_sigma	Standardna devijacija broja literala u klauzama u kojima se promenljiva pojavljuje.
clause_polarity_mu	Aritmetička sredina količnika broja pozitivnih literala u klauzi i broja literala u klauzi. Uzimaju se u obzir samo klauze u kojima se pojavljuje promenljiva.
clause_prolarity_sigma	Standardna devijacija količnika broja pozitivnih literala u klauzi i broja literala u klauzi. Uzimaju se u obzir samo klauze u kojima se pojavljuje promenljiva.
Clauses	Količnik broja klauza u kojima se promenljiva pojavljuje i ukupnog broja promenljivih.
Consequents	Količnik broja klauza gde se promenljiva pojavljuje kao posledica i broja Hornovih klauza u kojima se pojavljuje promenljiva.
horn_clauses	Količnik broja Hornovih klauza i ukupnog broja klauza u kojima se pojavljuje promenljiva.
Polarity	Količnik broja pozitivnih pojavljivanja promenljive i ukupnog broja pojavljivanja promenljive.
Neighbors	Količnik broja promenljivih koje su susedi sa tom promenljivom i ukupnog broja promenljivih.

Iz gore navedenih 9 atributa je dobijeno još 18 atributa, to jest iz svakog atributa su dobijena još po 2, tako sto su za svaki atribut X izracunati X_sigma i X_mu, gde je X_mu je aritmetička sredina atributa X za sve susedne promenljive, dok je X_sigma njihova disperzija.

4.3. Izbor familija formula

Ova glava opisuje familije formula nad kojima su vršeni eksperimenti. Korišćeno je sledećih osam familija formula.

- BMS_k3 - slučajno generisane instance 3-SAT problema
- CBS - slučajno generisane instance 3-SAT problema
- RTI - slučajno generisane instance 3-SAT problema
- UF - slučajno generisane instance 3-SAT problema
- FLAT - instance problema bojenja grafova kodiranog u vidu SAT problema
- SW - instance problema bojenja grafova kodiranog u vidu SAT problema, pri čemu se struktura grafova razlikuje u odnosu na FLAT.
- QCP - kombinatorne instance
- II - Instance koje kodiraju induktivno zaključivanje

Korišćeno je više familija 3-SAT problema, ali se one razlikuju po parametrima pomoću kojih su generisane.

Familije BMS_k3, CBS, II, RTI, QCP su korišćene u Silvertornovom radu. Ostale familije su dodate iz kolekcije SATLIB.

Detalji o familijama formula su prikazani u sledećoj tabeli. Prva kolona predstavlja ime familije, druga broj formula u familiji. U trećoj koloni se nalazi najmanji broj promenljivih koji ima neka formula iz te familije, a u četvrtoj najveći broj promenljivih. Peta kolona sadrži najmanji broj klauza neke formule iz familije, a šesta najveći broj klauza.

Familija	Broj formula	Min promenljivih	Max promenljivih	Min klauza	Max klauza
BMS_k3	505	100	100	254	318
CBS	2005	100	100	403	449
FLAT	1705	90	600	300	2237
II	60	66	759	186	20862
QCP	2080	8	4964	24	56238
RTI	505	100	100	429	429
SW	505	500	500	3100	3100
UF	3705	20	250	91	1065

Pored navedenih familija u tabeli isprobano je još oko 15 familija iz nešto novijih korpusa. Ipak pokazalo se da su preteške za rešavanje pomoću rešavača ProbSAT i zato su izostavljene.

Prilikom rešavanja formula sa vremenskim ograničenjem od 60 sekundi, uočen je jedan neobičan fenomen, a to je da praktično sve formule bivaju rešene ili za manje od jedne sekunde ili ne bivaju rešene u datom vremenskom periodu.

4.4. Kreiranje i evalucija logističkih modela

Logistički model za neku familiju formula se kreira na osnovu njenog trening skupa. Za kreiranje logističkog modela su korišćeni Matlab i SLEP biblioteka [9].

Pri kreiranju trening skupa je potrebno rešiti formule iz kojih se on kreira. Zato se treniranje vrši nad formulama lakšim za rešavanje, tako da ih već postojeći SAT rešavač može rešiti. Zatim se dobijeni logistički model koristi za poboljšanje performansi rešavanja formula težih za rašavanje. Trening skupovi familija su kreirani nad 80% lakših formula, a test skupovi nad preostalih 20%. Ipak formule iz test skupa sadrže više promenljivih nego formule iz trening skupa, tako da to nije tačan odnos veličina trening skupa na test skup. On može biti i 60%:40%.

Napisane su Matlab skripte koje koriste SLEP biblioteku u kojima je implementiran trening logističke regresije. One primenjuju dobijeni logistički model na test skup i računaju preciznost.

Za procenu efikasnosti je korišćena preciznost. Ipak preciznost ne oslikava dovoljno dobro efikasnost klasifikacije. Na primer, nije isto kada neki metod klasifikacije ima preciznost od 99% pri binarnoj klasifikaciji pri kojoj 50% instanci pripada klasi 0 a 50%

klasi 1 i kada 99% instanci pripada klasi 0 a 1% klasi 1. U tom slučaju ta preciznost bi mogla biti postignuta trivijalnom klasifikacijom svih instanci u većinsku klasu.

Performanse klasifikatora su prikazane u sledećoj tabeli. Prva vrsta sadrži ime familije, druga preciznost klasifikacije, treća udeo najčešće klase u predviđanju (100.0% znači da je sve klasifikovano u jednu klasu i da stoga nista nije naučeno) i četvrta udeo najčešće klase u test skupu (ako je preciznost veća od ovoga, onda je nešto naučeno).

Ime familije	Preciznost	Udeo najčešće klase u predviđanju	Udeo najčešće klase u test skupu
BMS_k3	51,7%	96,5%	50,1%
FLAT	66,7%	100,0%	66,7%
CBS	64,5%	64,9%	50,0%
II	81,7%	53,6%	63,0%
QCP	83,8%	100,0%	83,8%
RTI	65,0%	54,6%	50,4%
UF	65,3%	53,5%	50,2%
SW	80,0%	100,0%	80,0%

Iz tabele se može videti da je treniranje bilo neuspešno za familije FLAT, QCP, BMS_k3 i SW. U slučaju ovih familija je svim ili skoro svim promenljivim pridružen isti polaritet. Ipak i za te familije će, uniformnosti radi, klasifikator biti korišćen u evaluaciji rešavača. Za ostale familije, učenje je bilo u određenoj meri uspešno, mada ni u jednom slučaju izrazito. S druge strane, najmerodavnija evaluacija kvaliteta učenja je pokretanjem SAT rešavača koji koristi naučeni model. Sledeća tabela prikazuje familije na kojima je treniranje bilo uspešno i broj bitnih (nenula) parametara dobijenih pri kreiranju logističkog modela za datu familiju.

Ime familije	broj bitnih parametara
CBS	19
II	22
RTI	19
UF	18

Iz tabele se vidi da je u svim modelima bitno preko pola atributa, odnosno da ne postoji mala grupa atributa na osnovu koje je moguće vršiti efikasno predviđanje.

Preciznost klasifikacije u eksperimentima opisanim u Silvertrornovom radu je prikazana u sledećoj tabeli.

Ime familije	Preciznost
QCP	83
BMS_k3	59
II	80
RTI	67

Može se primetiti da je za iste familije dobijena slična preciznost, ali ne uvek jednaka. Uzrok za dobijanje različite preciznosti kod nekih familija može biti i u različitom kreiranju trening i test skupa.

4.5 Evaluacija modifikovanog SAT rešavača

Ova glava opisuje izmene nad SAT rešavačem ProbSAT [4]. Performanse modifikovanog ProbSAT rešavača će biti upoređene sa originalnim rešavačem.

ProbSAT je izabran za eksperimente kao jedan od modernijih stohastičkih SAT rešavača. ProbSAT je modifikovan tako da može da učita logistički model i pomoću tog modela izabere početnu valuaciju. Nakon toga su testirane i upoređene performance originalnog ProbSAT rešavača i modifikovanog ProbSAT rešavača.

Performanse originalnog ProbSAT-a i modifikovanog SAT rešavača su prikazane u sledeće dve tabele. Pošo je vreme rešavanja formula vrlo često manje od jedne skunde, kao mera efikasnosti korišćen je i broj koraka stohastičkog SAT rešavača. Prva kolona predstavlja ime familije formula, druga broj formula u test skupu, a treća prosečno vreme računanja atributa za tu familiju. U četvrtoj koloni je prikazano prosečno vreme rašavanja formule modifikovanim ProbSAT rešavačem (u to vreme nije uračunato vreme računanja atributa), a u petoj prosečno vreme rašavanja formule originalnim ProbSAT rešavačem. Šesta kolona predstavlja prosečan broj koraka pri rešavanju formule modifikovanim ProbSAT rešavačem, a sedma prosečan broj koraka pri rešavanju formule originalnim ProbSAT-om. Osma kolona predstavlja aritmetičku sredinu količnika broja koraka pri rešavanju formule modifikovanim ProbSAT rešavačam i broja koraka pri rešavanju formule originalnim ProbSAT-om.

familija	Br. formula	T atributa	T mod.	T orig.	Br. koraka mod.	Br. koraka orig.	Odnos mod./orig.
BMS_k3	101	0,003	0,070	0,092	323670	435666	0,99
CBS	401	0,005	0,001	0,000	6642	11265	0,75
FLAT	341	0,057	0,131	0,097	464433	598778	1,67
II	12	0,563	33,424	6,734	138747298	66617504	2.08
RTI	101	0,004	0,001	0,001	7291	15117	0,64
SW	101	0,098	3,151	2,662	17255441	15045069	2,97
UF	741	0,010	0,046	0,014	158431	64036	1,32
QCP	416	4.329	11.454	10.937	27515151	31167703	4.61

U sledećoj tabeli prva kolona predstavlja ime familije nad kojom je vršeno testiranje. U drugoj je dat broj formula u familiji koje je rešio samo modifikovani ProbSAT rešavač, a u trećoj broj formula koje je rešio samo originalni ProbSAT. Broj formula koje nije rešio ni jedan rešavač je predstavljen u četvrtoj koloni. U petoj koloni je predstavljen broj formula u familiji nad kojima je modifikovani ProbSAT rešavač imao bolje performanse, to jest ili ih je rešio u manjem broju koraka ili ih je samo on rešio. U šestoj koloni je broj formula nad kojima se originalni ProbSAT bolje pokazao.

familija	Samo mod.	Samo orig.	nijedan	Bolji mod.	Bolji orig.
BMS_k3	0	0	0	73	28
CBS	0	0	0	309	92
FLAT	0	0	0	223	118
II	1	5	3	1	5
RTI	0	0	0	80	21
SW	0	0	2	48	49
UF	0	0	0	522	219

Na nekim familijama formula modifikovani ProbSAT rešavač ima bolje performanse od originalnog ProbSAT-a, dok na drugim ima lošije. Familije formula kod kojih je došlo do poboljšanja se ne poklapaju jasno sa familijama na kojima je učenje bilo uspešno. Ne može se uočiti zakonitost u tome nad kojim familijama je došlo do poboljšanja, pa je racionalno pretpostaviti da su poboljšanja i pogoršanja performansi slučajna. Najverovatniji razlog za ovaj ishod je to što je polaritet promenljivih predviđan nezavisno za svaku promenljivu, dok između promenljivih postoji velika zavisnost.

Ipak, u Silvertornovom radu se navodi da su promene nad stohastičkim SAT rešavačem dovele do poboljšanja, što pokazuje sledeća tabela. Prva kolona predstavlja familiju nad kojom je vršen eksperiment, a druga aritmetiču sredinu količnika broja koraka pri rešavanju formule modifikovanim SAT rešavačam i broja koraka pri rešavanju formule originalnim SAT rešavačem.

Familija	Odnos mod./org.
QCP	0.99
BMS	0.97
II	1.33
RTI	0.87
CBS-M403-B10	0.76
CBS-M449-B90	0.93

Razlozi za odstupanje razultata u ovom i u Silvertornovom radu, mogu biti različiti. On ne navodi detalje eksperimentalne postavke, kao što su način kreiranja trening skupa (da li je isto koristio po jednu valuaciju koja zadovoljava formulu), ograničenje za vreme rešavanja formula i slično. Jasno, različita eksperimentalna postavka može dovesti do nepodudaranja rezultata. Takođe je moguće da je Silvertorn prikazao statistiku na povoljnom skupu instanci, to jest prikazao samo familije formula na kojima je došlo do poboljšanja. Pored toga, u Silvertornovom radu je korisćen drugi SAT rešavač i moguće je da su rezultati specifični za njega, iako to nije previše verovatno uzimajući u obzir ukupan broj korišćenih formula.

5. Zaključci i pravci daljeg rada

SAT rešavači imaju veoma široku primenu u rešavanju mnogih praktičnih problema, na mnogim poljima. Pravljenje rasporeda i verifikacija hardvera su klasični primeri. Zbog toga je često jako bitno rešiti SAT problem u realnom vremenu. Razne optimizacije i poboljšanja performansi SAT rešavača su veoma bitne. Veliki broj naučnih radova je napisan na tu temu. Jedan, mada neobjavljen, je rad Brajana Silvertorna [1]. U tom radu je opisan pokušaj poboljšanja performansi stohastičkog SAT rešavača, tako što će se na bolji način izabrati početa valuacija.

U cilju ponovne evaluacije mogućnosti poboljšanja performansi stohastičkog SAT rešavača metodama opisanim u Silvertrornovom radu, ponovljeni su eksperimenti iz tog rada. Kreiran je logistički klasifikacioni model za klasifikaciju promenljivih u formulama u tačne i netačne. Izmenjen je stohastički SAT rešavač ProbSAT tako da pomoću dobijenog logističkog modela može klasifikovati promenljive u tačne i netačne i tako dobiti početnu valuaciju pri rešavanju ulazne instance SAT problema. Testirane su performanse modifikovanog SAT rešavača i upoređene su sa performansama polaznog rešavača.

Rezultati dobijeni pri poređenju performansi polaznog i modifikovanog SAT rešavača, za razliku od rezultata prikazanih u Silvertrornovom radu, nisu tako povoljni po modifikovani SAT rešavač. On nad formulama nekih familija radi bolje, dok nad formulama drugih familija radi lošije od polaznog. Ne može se uočiti zakonitost nad familijama i predvideti nad formulama kojih familija će modifikovani SAT rešavač raditi bolje od polaznog, a nad kojima lošije. Iako neka zakonitost, koja se trenutno ne uočava, može postojati, može se konstatovati da performanse polaznog rešavača nisu poboljšane i da metodologija nije dovoljno upotrebljiva.

Razlozi za dobijanje lošijih rezultata u odnosu na Silvertornov rad mogu biti različiti. U svom radu on ne navodi detalje eksperimentalne postavke, te je možda u ovom radu korišćena nešto drugačija eksperimantalna postavka, što je moglo dovesti do razlike u rezultatima. Postoji mogućnost i da je Silvertrorn u svom radu prikazao rezultate na povoljnom skupu instanci, to jest prikazao samo familije formula na kojima su izmene dovele do poboljšanja i tako došao do zaključka da su izmene poboljšale performanse SAT rešavača. Kako se radi o ličnoj beleški autora, koja nije objavljena, ne dovodimo u pitanje akademsku čestitost autora, već samo ostavljamo mogućnost da rezultati nisu potpuni. Treći razlog je taj što je Silvertorn modifikovao neki drugi SAT rešavač, pa su rezultati igrom slučaja mogli biti drugačiji, što smatramo manje verovatnim zbog broja formula korišćenih u eksperimentima.

Razlozi zašto se učenje polariteta promenljivih pokazalo kao neuspešno mogu biti različiti. Najveći problem pri učenju polariteta promenljivih verovatno predstavlja njihova velika međuzavisnost, nametnuta klauzama ulazne formule, koja je u korišćenom pristupu zanemarena, iako može biti veoma važna.

Unapređivanje učenja tačnih valuacija i poboljšanju performansi stohastičkog SAT rešavača može se kretati u više pravaca. Navešćemo nekoliko ideja za dalji rad.

Modeli mašinskog učenja se formulišu nad nekim skupom atributa. Dodavanje novih, ili izbor drugih atributa, potencijalno može dovesti do preciznijeg učenja polaznih valuacija i time poboljšati performanse stohastičkog SAT rešavača.

Izbor metode, takođe može biti bitan za uspešnost klasifikacije. Korišćenje neuronskih mreža, ili neke druge metode klasifikacije bi možda moglo dovesti do poboljšanja performansi. Ipak, konstrukcija boljih atributa bi verovatno imala dosta veći uticij na kvalitet predviđanja, imajući u vidu da je polazni skup atributa relativno siromašan (konstruisan je nad 9 osnovnih atributa od kojih neki predstavljau proseke i standardne devijacije istih veličine).

Međuzavisnosti između promenljivih, koje su izražene klauzama formule, imaju jako bitnu ulogu pri rešavanju SAT problema. Ipak, one su pri klasifikaciji u ovom radu potpuno zanemarene. Uzimanje u obzir međusobnih zavisnosti promenljivih, recimo pomoću metoda uslovnih slučajnih polja (eng. conditional random fields) [8], bi možda moglo poboljšati kvalitet predviđanja početne valuacije i time poboljšati performanse stohastičkog SAT rešavača.

Metoda uslovnih slučajnih polja je u stanju da na osnovu grafa koji izražava zavisnosti promenljivih koriguje nezavisna predviđanja za pojedinačne promenljive. Taj graf treba da bude težinski graf, izabran tako da parovima promenljivih koje u zadovoljavajućim valuacijama česće imaju jednaku vrednost, odgovaraju grane sa većim težinama. Izbor težina bi se zasnivao na nekim brzo izračunljivim statistikama parova promenljivih, ali bi tačan način računanja tih težina bio predmet istraživačkog rada.

Literatura

[1] Bryan Silverthorn and Risto Miikkulainen, Polarity Prediction as Supervised Learning, Department of Computer Science The University of Texas at Austin.

[2] Kevin P. Murphy, Machine Learning A Probabilistic Perspective, The MIT Press, 2012.

[3] Carla P. Gomes, Henry Kautz, Ashish Sabharwal, and Bart Selman, Satisfiability Solvers, Elsevier, 2008.

[4] Adrian Balint and Uwe Schoning, Choosing Probability Distributions for Stochastic Local Search and the Role of Make versus Break, SAT, 2012.

[5] Joao Marques-Silva, Practical Applications of Boolean Satisfiability, WODES, 2008.

[6] Koen Claessen, Niklas Een, Mary Sheeran and Niklas Sorensson, SAT-solving in practice, WODES, 2008.

[7] Filip Marić, Timetabling Based on SAT Encoding: a Case Study, Faculty of Mathematics, University of Belgrade, Serbia, 2008.

[8] Philipp Krähenbühl, Vladlen Koltun, Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials, NIPS, 2011.

[9] Jun Liu, Shuiwang Ji, Jieping Ye, SLEP: Sparse Learning with Efficient Projections, Arizona State University, 2009.

[10] Predrag Janičić, Matematička logika u računarstvu, Matematički fakultet, 2005.

[11] Carla Gomes, Henry Kautz, Ashish Sabharwal, Bart Selman, Satisfiability solvers, Handbook of Knowledge Representation, Elsevier, 2008.

[12] Martin Davis, Hilary Putnam, A computing procedure for quantification theory, CACM, 1960.

[13] Martin Davis, George Logemann, Donald Loveland, A machine program for theorem proving, CACM, 1961.