Sinteza
Međunarodna naučna konferencija Univerziteta Singidunum

Impact of Internet on Business Activities in Serbia and Worldwide
Uticaj Interneta na poslovanje u Srbiji i svetu

DOI: 10.15308/SINTEZA-2014-345-350

REALIZACIJA SOFTVERSKOG SISTEMA ZA VIZUELNU SIMULACIJU ALGORITAMA MAŠINSKOG UČENJA

Marko Marković[1], Olivera Nikolić[1], Boško Nikolić[2]
[1] Univerzitet Singidunum, Poslovni fakultet Valjevo, Srbija
[2]Elektrotehnički fakultet Beograd

Abstract:
Zbog porasta važnosti algoritama mašinskog učenja, ukazala se potreba za razvijanjem alata koji bi pomogli u njihovom razumevanju obzirom da ovakvi algoritmi često mogu biti komplikovani za teorijsko objašnjavanje. Zbog toga se kao odličan alat za njihovo objašnjavanje pokazalo upravo korišćenje vizuelnih softverskih simulatora. U ovom radu predstavljena je tehnička realizacija jednog takvog simulatora razvijenog na Poslovnom fakultetu Valjevo Univerziteta Singidunum. Simulator poseduje module za stabla odlučivanja, klasterovanje, Naive Bayes i perceptrone.

Key words:
mašinsko učenje,
softverske simulacije,
Java,
JUNG.

UVOD

Za potrebe edukacije mašinskog učenja, sve više se razvijaju softverske simulacije, obzirom da algoritmi koji se koriste u ovoj oblasti mogu biti nepogodni za teorijsko objašnjavanje. Upravo su se softverski simulatori pokazali kao odličan alat za objašnjavanje ovakvih koncepata koji često mogu biti i u velikoj meri apstraktni. U ovom radu predstavljena je tehnička realizacija jednog takvog simulatora razvijenog da pomogne izučavanje oblasti poslovnog odlučivanja na Poslovnom fakultetu Valjevo Univerziteta Singidunum. Simulator poseduje module za stabla odlučivanja, klasterovanje, Naive Bayes i perceptrone. U izboru ovih algoritama vođeno je računa o tome šta preporučuju vodeće strukovne organizacije, kao i šta se izučava na poznatim svetskim univerzitetima. Pomogle su i preporuke donete na međunarodnoj konferenciji o tehnikama Data Mining-a održanoj 2006. godine, čiji je organizator IEEE, a tom prilikom identifokovano je 10 najboljih algoritama u ovoj oblasti [1], među kojima se nalaze i algoritmi izabrani za ovaj simulator. Takođe, u uporednim iskustvima primećeno je da se ovi algoritmi najčešće pominju i obrađuju, kao i da poseduju veliku upotrebnu vrednost. Pogodni su za donošenje odluka na osnovu raznih tipova ulaznih podataka i veoma lako mogu da nađu odgovarajuću primenu.

Kako bi ovakav sistem za vizuelnu simulaciju mogao da se iskoristi na najbolji način, potrebno je obezbediti da bude dostupan na što većem broju računarskih sistema. Zato se kao pogodno rešenje nametnulo korišćenje Java platforme koja je nezavisna od platforme na kojoj se izvršava. Java je pogodna i zbog toga što ima ugrađen veliki skup mogućnosti, a pritom postoji mogućnost i za dodavanjem novog skupa funkcionalnosti pomoću dodatnih biblioteka.

Rad je organizovan prema sledećoj strukturi: druga sekcija prikazuje tehnologije koje su korišćene za realizaciju sistema, u trećoj sekciji se obrazlažu tehnički aspekti razvoja simulatora uz objašnjenje rada svakog modula. Sekcija četiri predstavlja zaključak rada.

TEHNOLOGIJE KORIŠĆENE ZA IZRADU SISTEMA

Realizovani softverski sistem zasnovan je na Java platformi. Java je programski jezik otvorenog koda i predstavlja globalni standard za razvoj aplikacija nezavisnih od platforme na kojoj se izvršavaju. Jedna od najvažnijih karakteristika jezika je arhitekturna neutralnost. Ona se postiže tako što java kompajler generiše bajtkod instrukcije koje nisu povezane sa arhitekturom odredišnog računara. Bajtkod se interpretira na svakoj mašini i "u letu" se prevodi na izvorni mašinski kod. [2] Naravno, interpretiranje bajtkoda je neminovno sporije od izvršavanja mašinskih instrukcija pri punoj brzini, ali danas, sa napretkom računarske tehnike, to više ne predstavlja problem, pa se razlike u brzini praktično i ne primećuju.

Java platforma obuhvata čitav niz grafičkih biblioteka koje omogućavaju dodatne funkcionalnosti i koje se koriste u različite svrhe. Za potrebe razvoja simulatora, vođeno je računa da se izabere pravi sistem koji će moći da omogući najveći skup mogućnosti i pouzdanost u radu. U praksi su se kao najkorisnija rešenja pokazali:

- AWT koji predstavlja osnovu Swing-a, ali mu nedostaju naprednije mogućnosti. Predstavlja stabilno rešenje koje se dokazalo u praksi.
- Swing je u početku bio relativno spor i problematičan, ali su sa novijim verzijama prevaziđeni problemi. Izuzetno se često koristi za razvoj aplikacija jer poseduje veliki broj komponenti za bogat korisnički interfejs.

- SWT je nastao kao odgovor kompanije IBM na Swing. Iako u odnosu na njega nudi neke dodatne komponente, nije se pokazao kao stabilan i popularan za rad.
- SwingX je baziran na Swing-u. Ima određen broj kvalitetnih komponenti, ali je zvanično još uvek u razvoju.
- JavaFX predstavlja obećavajući standard za razvoj desktop i veb aplikacija. Java FX je sledeći korak u evoluciji Java programa sa bogatim korisničkim interfejsima.

Za potrebe simulatora, biblioteka AWT je izabrana za iscrtavanje komponenti na najnižem nivou, počevši od pravljenja najosnovnijih oblika. Swing komponente su korišćene za standardne elemente korisničkog interfejsa (paneli, dugmad, meniji, polja za unos podataka itd.), uključujući i korišćenje menadžera za pravljenje njihovog rasporeda. U narednim odeljcima ove biblioteke su opisane, a njihove tehničke karakteristike su detaljnije predstavljene.

Java AWT

Java AWT (*Abstract Window Toolkit*) je biblioteka programskog jezika Java zadužena za osnovno programiranje grafičkog korisničkog interfejsa i predstavlja vezu ka matičnom korisničkom interfejsu sistema na kome se izvršava.

Korisnički intefejs je deo programa koji obezbeđuje interakciju sa korisnikom programa. Na najnižem nivou, operativni sistem prenosi informacije od miša i tastature do programa kao ulaz, i obezbeđuje piksele za izlaz programa. AWT je osmišljen tako da programeri ne moraju da brinu o praćenju miša ili čitanju sa tastature ili da brinu o ispisu na ekranu. AWT obezbeđuje dobro dizajnirani obejktno orijentisani intfejs ka servisima i resursima niskog nivoa [3].

Najvažnije karakteristike AWT-a su [4]:

- veliki broj komponenti za korisnički interfejs,
- robustan model za rukovanje događajima,
- alati za rad sa slikama i i grafičkim elementima (uključujući klase za rad sa oblicima, bojama, fontovima),
- menadžeri rasporeda za fleksibilan rad sa prozorima,
- klase za prenos podataka - za korišćenje clipboard-a na matičnoj platformi.

AWT komponente u velikoj meri zavise od platforme na kojoj se program izvršava. Na primer, pravljenjem neke komponente korisničkog interfejsa pomoću AWT-a, biće direktno pozvana odgovarajuća sistemska rutina koja će obezbediti zahtevanu komponentu. To znači da će program na Windows-u imati izgled kao izvorne Windows aplikacije ili da će imati izgled Macintosh aplikacije kada se pokrene na Mac-u. Naravno, ovakav pristup može biti problematičan ukoliko je potrebno obezbediti da interfejs izgleda identično na svim platformama.

Java Swing

Biblioteka Swing obuhvata veliki set komponenti za izradu grafičkih korisničkih interfejsa i dodavanje interaktivnosti Java aplikacijama. Swing obuhvata sve komponente koje bi mogle da zatrebaju u jednoj modernoj aplikaciji: tabele, liste, stabla, dugmad, natpise... Pored svih komponenti, obuhvata i odličnu podršku za rad sa tekstom, kao i integrisanu podršku za internacionalizaciju i dostupnost. Swing poseduje i mnoštvo opcija za podešavanje *izgleda i osećaja* (*look and feel*), a postoji i mogućnost pravljenja sopstvenog. Pored svega pobrojanog, postoji i podrška za *povuci i pusti* tehnike (*drag and drop*), rukovanje događajima, prilagođeno iscrtavanje i rukovanje prozorima. [5]

Swing je razvijen kako bi omogućio sofisticiraniji skup komponenti za razvijanje korisničkih interfejsa. Sve ove komponente nisu implementirane u kodu koji je specifičan za određenu platformu, već su kompletno napisane u Javi zbog čega su platformski nezavisne.

Od J2SE 1.2 Swing je u velikoj meri potisnuo AWT. Pored obezbeđivanja bogatog skupa ulazno-izlaznih komponenti, Swing prikazuje i sopstvene dodatke koji se isrtavaju koristeći Java 2D pomoću koga se pozivaju sistemske rutine niskog nivoa u lokalnom grafičkom podsistemu. Na taj način je izbegnuto oslanjanje na module visokog nivoa za korisnički intefejs u okviru operativnog sistema. Swing obezbeđuje mogućnost izbora *izgleda i osećaja* koji će se koristi - može se upotrebiti ili onaj koji je platformski vezan, ili međuplatformski *izgled i osećaj* koji izgleda isto na svim sistemima za prikaz prozora.

Naravno, i pored toga što je Swing stekao izuzetno veliku popularnost, ipak ne predstavlja zamenu za AWT, već njegovu nadogradnju, obzirom da je napravljen na njegovim bibliotekama.

Pre pojavljivanja Java 6 Update 12, mešanje Swing komponenti sa AWT dodacima je često dovodilo do nepredviđenih efekata, kada bi se AWT dodaci pojavljivali preko onih definisanih u Swing-u, bez obzira na navedeni *z-raspored*. Ovi problemi su nastajali zbog toga što su dva navedena sistema u suštini veoma različita, i pored toga što Swing pozajmljuje neke kontejnere višeg nivoa od AWT-a.

Obzirom da je Swing i izuzetno modularna platforma, omogućava i "priključivanje" dodatnih, namenski pravljenih implementacija za potrebe specifičnih interfejsa. Podrazumevane komponente se mogu modifikovati nasleđivanjem klasa, čime se mogu dodati određene nove funkcionalnosti ili izmeniti postojeće.

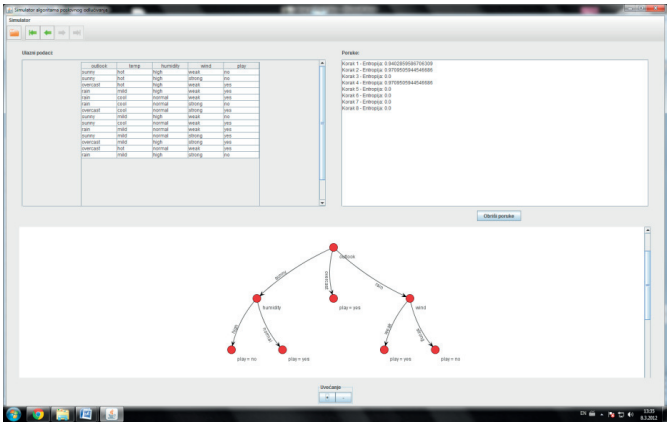
Biblioteka JUNG

Obzirom da je u određenim modulima potrebno iscrtavati stabla koja služe kao osnova za grafičko reprezentovnje rada algoritama, bilo je potrebno obezbediti način da se to obavi. Jedna mogućnost je ručno pravljenje mehanizama koji bi omogućili podršku za iscrtavanje, a druga da se upotrebi postojeća biblioteka koja može da pruži potrebne funkcionalnosti i čijim korišćenjem može da se uštedi vreme potrebno za razvoj.

Kao odlično rešenje nametnula se biblioteka JUNG [6] - *Java Universal Network/Graph Framework*. Ova biblioteka obezbeđuje proširiv jezik za modelovanje, analizu i vizualizaciju podataka koji mogu biti predstavljeni u obliku mreže ili grafa. Dizajnirana je tako da podrži varijacije prikaza određenih entiteta i njihovih veza, kao što su usmereni i neusmereni grafovi ili grafovi sa paralelnim granama. Zahvaljujući ovoj biblioteci, programeri ne moraju da brinu o grafičkoj implementaciji samih grafova ili stabala, obzirom da posao prikazivanja ovih elemenata preuzima JUNG.

Najvažnije karakteristike biblioteke JUNG su [7]:

- Podrška za razne reprezentacije entiteta i njihovih veza, uključujući usmerene i neusmerene grafove, multimodalne grafove, multi-grafove i hiper-grafove.
- Mehanizmi za anotaciju grafova, entiteta i veza pomoću metapodataka.
- Postojanje implementacije određenog broja algoritama iz teorije grafova i analize podataka.
- Alati za vizuelizaciju koji olakšavaju razvoj mehanizama za interaktivno pretraživanje i prikazivanje grafovskih podataka.



Sl. 1. Izgled modula za stabla odlučivanja.

Mehanizmi za filtriranje pomoću kojih se može doći do određenog podskupa grafovske mreže koja se prikazuje, kako bi se na njemu, na primer, primenio određeni algoritam.

IMPLEMENTACIJA NOVOG SIMULATORA

Sistem je podeljen na četiri logičke celine - modula. Svaki od modula zadužen je za simulaciju po jednog algoritma: stablo odlučivanja, klasterovanje, Naive Bayes i neuronske mreže.

Za simulaciju svakog algoritma, prilagođena je posebna radna površina. Pored obaveznih delova, kao što su grafički prikaz simulacije i okvir za ispis poruka, svaka radna površina može imati i neke neobavezne delove - polje za prikaz početnog skupa podataka, polja za unos dodatnih podataka itd.

Postoje i obavezna dugmad pomoću kojih se korisniku omogućava postupno kretanje kroz algoritam: napred, nazad, na početak i na kraj. Sl. 1 prikazuje izgled jednog ekrana simulatora - konkretno, radi se o modulu za stabla odlučivanja.

U narednim odeljcimas funkcionalnost svakog modula je detaljnije predstavljena.

Implementacija stabala odlučivanja

Prvi korak algoritma je izbor najdiskriminatornijeg atributa koji dovodi do što bolje klasifikacije primera. Idealno, ovaj atribut bi podelio skup na vrednosti koje su ili samo pozitivne ili samo negativne. Da bi se odredilo koji atribut je dobar, koriste se mere koja se nazivaju *entropija* i *informaciona dobit*. Entropija predstavlja meru neizvesnosti neke slučajne promenljive, a informaciona dobit je mera koju koristi *ID3* algoritam kako bi izabrao najbolji atribut prilikom pravljenja stabla odlučivanja. Za pravljenje stabla, biće izabran onaj atribut koji ima najveću informacionu dobit. [8]

Obzirom da algoritam ID3 u svojoj osnovi ima stablo kao strukturu podataka, veoma je važno predstaviti čvor tog stabla za šta se koristi klasa *TreeNode*. Neka od najvažnijih polja ove klase su:

- *entropy* čuva vrednost entropije određenog čvora. Ukoliko se radi o listu stabla, entropija će imati vrednost 0.
- *decompositionAttribute* predstavlja atribut po kom je izvršena podela stabla. Ovaj atribut bira algoritam stabla odlučivanja kao najbolji za podelu.
- *decompositionValue* predstavlja vrednost izabranog atributa po kojoj je izvršena podela ulaznog skupa podataka.
- *children* čuva reference ka svim čvorovima deci trenutnog čvora. ukoliko se radi o listu stabla, ovo polje neće imati vrednost.
- *vidljivo* je atribut koji se koristi kod grafičke prezentacije stabla. S obzirom na to da se zbog postupnog prolaska kroz algoritam svi čvorovi ne prikazuju odjednom, pomoću ovog atributa će se označiti koji će biti vidljivi, a koji ne. Na ovaj način će biblioteci JUNG biti saopšteno kako treba da prikaže stablo.
- *dobiti* čuva informacione dobiti atributa. Ove vrednosti će biti prikazane u kontekstnom meniju koji se pojavljuje kada se desnim tasterom miša klikne na neki čvor.

Klasa ID3 zadužena je za rukovanje samim stablom odlučivanja. U okviru nje se prolazi kroz proces podele ulaznog skupa podataka i pravljenja odredišnog stabla. Kao polje, u okviru nje se čuva *root* stabla. Sa ovim elementom povezani su svi sledeći čvorovi.

Da bi se napravljeno stablo moglo prikazati, mora se prilagoditi za korišćenje biblioteke JUNG. Za tu svrhu je napravljena klasa Stablo. U okviru ove klase, postoje sledeća polja:

- *trenutniPrikaz* čuva broj trenutnog koraka do kog je stigao algoritam. Početna vrednost -1 označava da algoritam još uvek nije počeo sa izvršavanjem.
- *stablo* je objekat tipa *DelegateTree<TreeNode, Veza>* i predstavlja stablo odlučivanja u formatu pogodnom za JUNG sistem.

- *listaCvorova* čuva listu referenci ka svim čvorovima koji postoje u stablu. Zahvaljujući ovoj listi, moguće je pristupiti svakom čvoru i po potrebi izmeniti neko od polja – npr. vidljivost tog čvora.

Takođe, ova klasa sadrži i mehanizam za kretanje kroz sam algoritam – to obavljaju metodi *kreni()*, *napred()*, *nazad()*, *naKraj()* i *naPocetak()*.

S obzirom da ovi metodi vraćaju objekat stabla – ili direktno, ili upakovano u objekat klase ObjekatPrikaza, postavljena ja osnova komunikacije sa bibliotekom JUNG koja će prihvatiti stablo, a potom i izvršiti njegovo predstavljanje.

Implementacija klasterovanja

Sl. 2. Izgled modula za klasterovanje.

Klasterovanje podrazumeva deljenje podataka u grupe koje imaju značenje i koje bi trebalo da odražavaju prirodnu strukturu podataka. Klasterovanje već dugo vremena ima važnu ulogu u mnogim poljima kao što su psihologija i druge društvene nauke, biologija, statistika i mašinsko učenje [9].

Postoji mnogo načina da se klasteri primene na praktične probleme. U ovom simulatoru je upotrebljen algoritam *k-means* koji je prvi put predložio Stuart Lloyd 1957. godine. K-means je jedan od najstarijih i najviše korišćenih algoritama klasterovanja.

Algoritam *k-means* je particionalna tehnika klasterovanja koja pokušava da nađe broj klastera koje je odredio korisnik, a koji su predstavljeni svojim centroidima [10].

Centroid (prosečna tačka) je obično aritmetička sredina grupe tačaka. U okviru klastera, centroid će veoma retko biti neka već postojeća tačka, osim ukoliko to nije izričit zahtev. Za merenje razdaljine između tačaka prilikom računanja pozicije centroida koristi se Euklidovo rastojanje koje u osnovi koristi Pitagorinu teoremu.

Osnova *k-means* algoritma je prikazana sledećim algoritmom [9]:

1. Izabrati K tačaka kao početne centroide.
2. Ponavljati
3. Formiranje K klastera dodeljivanjem svake tačke najbližem centroidu.
4. Ponovno računanje centroida svakog klastera.
5. Dok se centroidi menjaju.

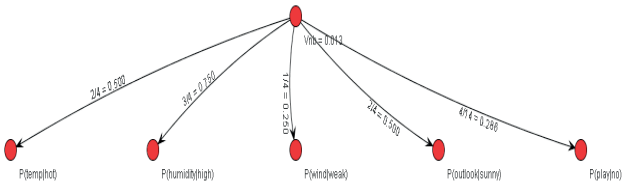
Osnovna klasa od koje modul klasterovanje počinje izvršavanje jeste *kmPanel*. U okviru ove klase poziva se metod *pokreniAlgoritam()*. Ovaj metod pokreće izvršenje algoritma. U svakom koraku se određuje koliko tačaka pripada kom klasteru i sračunava se tačna pozicija centroida. Listing 5 prikazuje klasu *Korak* koja čuva informacije o svakoj iteraciji algoritma. Najvažnija polja su:

- *centar* koji čuva poziciju centroida svakog klastera i
- *pripadnost* u okviru koje se čuvaju pozicije svih tačaka koje pripadaju određenom klasteru.

Ova klasa je zadužena i za prihvatanje početnih vrednosti koje korisnik unese kako bi rad algoritma prilagodio sopstvenim potrebama. Vrednosti se unose korišćenjem klizača, koji su pogodni zbog preglednijeg grafičkog predstavljanja. Bitno je naglasiti da je generisanje tačaka nasumično pomoću funkcije *random()*. Ali ono što je još važnije je to da je *seed* ove funkcije fiksiran. To praktično znači da će ona uvek, za iste ulazne parametre broja tačaka, klastera i raspršenost, davati potpuno iste pozicije tačaka, što je posebno pogodno u edukaciji, kada treba u potpunosti isto reprodukovati određeni primer. Sl. 2. prikazuje izgled modula za klasterovanje.

Implementacija algoritma Naive Bayes

Algoritam *Naive Bayes*, kao i većina sistema koji koriste rasuđivanje zasnovano na verovatnoći, imaju za temelj Bajesovu teoremu [11]. Bitno je pomenuti i poreklo naziva algoritima – iako bi neko iz naziva mogao da zaključi da se radi o „naivnom" algoritmu, zapravo se radi o tome da se podrazumeva da su atributi između sebe uslovno nezavisni [12]. I pored toga, u praksi, *Naive Bayes* modeli mogu raditi iznenađujuće dobro, u zavisnosti od prirode verovatnosnog modela, a do sada su svoju veliku upotrebnu vrednost dokazali u nekoliko veoma kompleksnih stvarnih situacija [13].



Sl. 3. Izgled stabla koje predstavlja algoritam Naive Bayes.

Izvršavanje ovog modula se kontroliše pomoću klase *nbPanel*. Nakon što se unese ulazni skup podataka, iz posebne tabele se mogu izabrati test primeri koji se nalaze u odgovarajućim padajućim listama. Pošto se i ovde, kao i u slučaju stabla odlučivanja, generiše struktura podataka stablo (Sl. 3.), potrebno je napraviti odgovarajuće čvorove. U tu svrhu se koristi klasa *CvorNB*. Najvažnija polja ove klase su:

- *brojUkupnih* predstavlja ukupan broj vrednosti koje su ušle u obračun.
- *brojUslovnih* predstavlja deo ukupnih vrednosti koje ispunjavaju određeni uslov. Količnik uslovnih i ukupnih vrednosti daće uslovnu verovatnoću određenog atributa.
- *visible* kao i kod stabla odlučivanja čuva oznaku da li trenutni čvor treba prikazati ili ne.

- *uslovnePozicije* predstavljaju brojeve redova iz tabele koji se koriste u obračunu. Pozadinska boja redova koji su pobrojani u ovom polju biće promenjena kako bi se naznačilo da su trenutno u upotrebi.

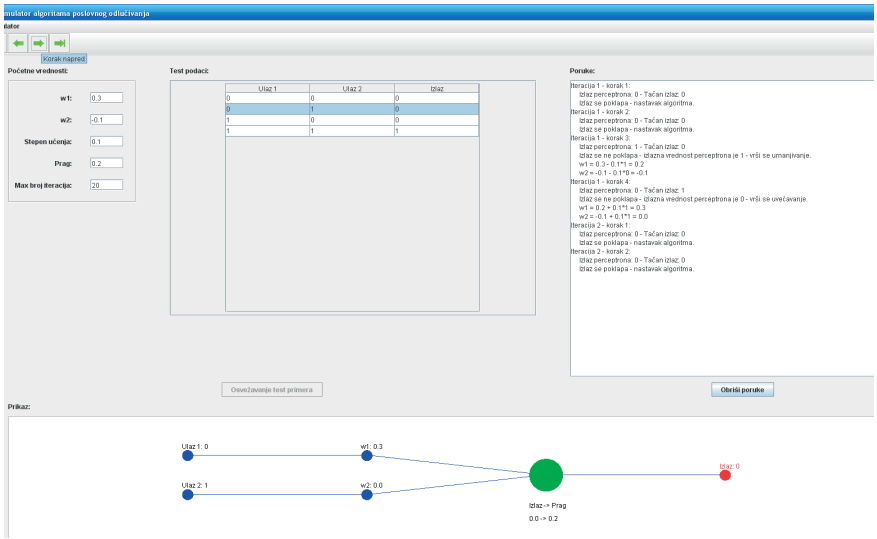
IMPLEMENTACIJA PERCEPTRONA

Neuronska mreža sa svim ulazima direktno povezanim sa izlazima naziva se jednoslojna neuronska mreža ili perceptron. Od svih neuronskih mreža, perceptroni su najbolje shvaćeni i najšire korišćeni. Termin perceptron je prvi put upotrebio Frenk Rozenblat 1958. godine. Njegova ideja je bila da napravi funkcionalni opis načina rada pravog neurona i da ga potom implementira kao softverski algoritam [14].

Neuronske mreže se sastoje od čvorova povezanih usmerenim vezama (Sl. 4.). Veza od jedinice *i* do jedinice *j* služi za prostiranje aktivacije a_i od *i* do *j*. Svaka veza ima i numeričku težinu $w_{i,j}$ koja joj je pridružena. Sve jedinice imaju po jedan ulaz $a_0 = 1$ sa odgovarajućom težinom $w_{0,j}$. Svaka jedinica *j* prvo izračunava ponderisani zbir svojih ulaza, a potom se na ovaj zbir primenjuje funkcija aktivacije *f* da bi se dobio izlaz. [11] Ova funkcija se još naziva i *threshold logic unit (TLU)*, a njen izlaz može biti 1 ili 0, u zavisnosti od toga da li je vrednost funkcije veća od definisanog praga (označava se sa □). Danas se reč perceptron najčešće koristi da označi jedan *TLU* [15].

Radom ovog modula se upravlja pomoću klase *perceptronPanel*. Izvršenje algoritma počinje pozivanjem metoda *pokreniAlgoritam()*. U promenljivu *suma* smešta se zbir proizvoda ulaznih vrednosti i težina. Na osnovu poređenja te *sume* sa pragom, određuje se izlazna vrednost. Pošto se radi o oštroj pragovskoj funkciji, izlazna vrednost može biti ili 0 ili 1.

Ukoliko se izlazna vrednost perceptrona ne poklapa sa traženom izlaznom vrednošću, potrebno je korigovati težine. Za obračun težina, ne koristi se tip podataka *double* zbog mogućeg gubitka preciznosti u radu sa malim brojevima. Koristi se tip *BigDecimal*, koji je pogodan zbog visoke preciznosti rada. Ukoliko je izlaz perceptrona 1, vrednosti težine će se smanjivati. U suprotnom, ukoliko je izlaz 0, težine će se uvećavati.



Sl. 5. Simulacija rada perceptrona.

Kako bi grafička prezentacija perceptrona imala iste proporcije na monitoru svake veličine, na osnovu dimenzija okvira se izračunava na koji način će biti iscrtan perceptron.

Izgled ekrana simulacije za jedan test primer perceptrona prikazan je na Sl. 5.

ZAKLJUČAK

Vizuelne softverske simulacije su se u nastavi pokazale kao odlično rešenje za prezentovanje određenih tehnika rada, ili algoritama u slučaju mašinskog učenja. Na taj način se, uz obavezno poštovanje određenih metodičkih pravila, mogu postići izuzetno dobri rezultati. Zahvaljujući razvoju softvera koji poštuje ta pravila, može se dobiti edukativni alat koji znatno može poboljšati kvalitet nastave.

Nakon proučavanja preporuka vodećih strukovnih organizacija i vodećih svetskih univerziteta, odlučeno je da se u okviru novog simulatora simulatora, čija je realizacija predstavljena u ovom radu, obrade sledeći algoritmi: stabla odlučivanja, klasterovanje, Naive Bayes i perceptroni.

Razvijeni simulator omogućava korisnicima da vide kako određeni algoritmi poslovnog odlučivanja rade. Moguće je zadati ulazni skup podataka i na njega primeniti izabrani algoritam, a potom, korak po korak, pratiti na koji način se izvršava. Tokom same simulacije, prikazuju se i detaljne informacije o tome u kojoj se trenutno fazi nalazi algoritam kako bi se lakše razumeo način njegovog rada.

Detalji tehničke realizacije predstavljenog sistema za vizuelnu simulaciju algoritama poslovnog odlučivanja, kao i rešenja problema koji su se pojavili tokom razvoja, predstavljeni su u ovom radu. Prikazano je i kako je ceo sistem organizovan po modulima na taj način da svaki obrađeni algoritam predstavlja zasebnu celinu.

Za razvoj samog softverskog sistema odabrana je Java platforma. Na ovaj način, softver može da se izvršava na velikom broju različitih sistema, a da se pritom zadrži isti izgled i osećaj prilikom korišćenja. Naravno, ne postoje ni troškovi licenciranja jer se radi o platformi otvorenog izvornog koda.

Za razvoj simulatora korišćene su i dodatne biblioteke. Za iscrtavanje grafičkih elemenata, počevši od najjednostavnijih oblika do složenijih struktura, korišćena je Java AWT. Komponente korisničkog interfejsa i njihov raspored su napravljeni pomoću biblioteke Java SWING. Za rad sa stablima korišćen je JUNG, pomoću koga je olakšan proces njihovog prikazivanja.

U radu je ukratko objašnjen i način rada svakog od algoritama, a potom su navedeni i detalji njihove tehničke realizacije. Naveden je model svakog algoritma, kao i opis kako je implementiran u okviru odgovarajućeg modula.

Ukoliko bi se posmatrali načini na koje bi sistem mogao da se poboljša,

došlo bi se do moguće nadogradnje sistema u smislu proširenja njegovih funkcionalnosti. Na primer, moguće je obraditi i neke od novijih varijanti prikazanih algoritama.

Ako bi se ukazala potreba, mogao bi se dodati i kompletan modul i obraditi još neki od algoritama mašinskog učenja. Na taj način bi bilo moguće posvetiti pažnju još nekom algoritmu koji nije ušao u izbor za ovu verziju simulatora.

Zahvalnice

Ovaj rad je delimično finansiran od strane Ministarstva prosvete, nauke i tehnološkog razvoja Republike Srbije (TR32054).

LITERATURA

[1] Wu X., Kumar V., Quinlan R., Ghosh J., Yang Q., Motoda H., McLachlan G., Ng A., Liu B., Yu P., Zhou Z., Steinbach M., Hand D., Steinber D., "Top 10 algorithms in data mining", Springer-Verlag, London, 2007.

[2] Horstmann C., Cornell G., "Java 2 – Osnove", CET, Beograd, 2007.

[3] http://www.javaworld.com/javaworld/jw-07-1996/jw-07-awt.html, datum pristupa: 11.12.2013.

[4] http://edn.embarcadero.com/article/26970, datum pristupa: 20.1.2014.

[5] http://docs.oracle.com/javase/tutorial/ui/overview/intro.html, datum pristupa: 11.12.2013.

[6] http://jung.sourceforge.net/, datum pristupa: 22.1.2014.

[7] O'Madadhain J., Fisher D., Smyth P., White S., Boey Y-B., "Analysis and Visualization of Network Data using JUNG", Journal of Statistical Software, Volume VV, Issue II, 2005.

[8] Mitchell T., "Machine Learning", McGraw Hill, Boston, 1997, str. 57.

[9] Tan Pang-Ning, Steinbach M., Kumar V., "Introduction to Data Mining", Addison-Wesley, Boston, 2006, str. 487.

[10] Norouzi M., Fleet D., "Cartesian k-means", pp.3017-3024, 2013 IEEE Conference on Computer Vision and Pattern Recognition, 2013.

[11] Russell S., Norvig P., "Veštačka inteligencija – savremeni pristup", CET, Beograd, 2011.

[12] Lowd D., Domingos P., "Naive Bayes Models for Probability Estimation", 22nd International Conference on Machine Learning, Bonn, Germany, 2005.

[13] http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Naive_Bayes_classifier.html, datum pristupa: 22.1.2014.

[14] Picton P., "Neural Networks", Palgrave, Hampshire, 2000, str. 29.

[15] Nilsson N., "Introduction to Machine Learning", Department of Computer Science, Stanford University, Stanford, CA, 1996, str. 40.

[16] Jantzen J., "Introduction to Perceptron Networks", Technical University of Denmark, Lyngby, 1998, str. 5.

REALISATION OF SOFTWARE SYSTEM FOR MACHINE LEARNING ALGORITHMS VISUAL SIMULATION

Abstract:
Due to the increased importance of the machine learning algorithms, there was a need for developing a tools that would assist in understanding of these algorithms, especially because they can often be difficult for a theoretical explanation. Therefore, visual software simulations had proven to be an excellent tool for their explaining. Technical realisation of such simulator developed at the Business Faculty of Valjevo, Singidunum University, has been presented in this paper. Simulator has modules for decission trees, clustering, Naive Bayes and perceptrons.

Key words:
machine learning,
software simulations,
Java,
JUNG.