SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1667

Neizrazita logika u analizi vremenskih slijedova

Vinko Kolobara

Zagreb, lipanj 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 15. ožujka 2018.

DIPLOMSKI ZADATAK br. 1667

Pristupnik: **Vinko Kolobara (0036475679)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Neizrazita logika u analizi vremenskih sljedova**

Opis zadatka:

Opisati postupke neizrazitog zaključivanja temeljene na neizrazitoj logici, te primjenu evolucijskih postupaka u oblikovanju sustava neizrazitog zaključivanja. Ostvariti programski sustav neizrazitog zaključivanja i proširiti ga mogućnošću učenja neizrazitih pravila uz pomoć neuronske mreže. Ispitati različite algoritme učenja s obzirom na složenost i zadana ograničenja. Primijeniti razvijeni sustav na analizu i predviđanje vremenskih sljedova. Usporediti rezultate s postojećim metodama predviđanja vremenskih sljedova. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Zadatak uručen pristupniku: 16. ožujka 2018.
Rok za predaju rada: 29. lipnja 2018.

Mentor:
Prof. dr. sc. Domagoj Jakobović

Djelovođa:
Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za diplomski rad profila:
Prof. dr. sc. Siniša Srbljić

SADRŽAJ

1. Uvod

Vremenske slijedove možemo primijetiti u raznim oblicima, bila to vremenska prognoza, količina oborina, cijene dionica kroz vrijeme, uglavnom bilo što za što je važan faktor trenutak u kojem se nešto dogodilo. Analiza vremenskih slijedova je pogodna jer npr. možemo predvidjeti buduću vremensku prognozu, detektirati kvar nekog stroja prije nego se dogodi i tako spriječiti ogromne gubitke, pa i predvidjeti cijene dionica, ili samo odrediti u kojem trenutku ih je najbolje kupovati ili prodati.

Glavni cilj ovog rada je pokazati da je moguće nekim algoritmima strojnog učenja, točnije korištenjem genetskog algoritma i neizrazite logike, odrediti idealne trenutke za kupovinu ili prodaju neke dionice/kriptovalute/valutnog para. Sporedni ciljevi, koji svi u konačnici vode ka glavnom, su naravno i primjena iste vrste algoritma na jednostavnije klasifikacijske i regresijske probleme.

Neizrazita logika je izabrana jer se općenito standardni algoritmi za trgovanje oblikuju kao pravila koja signaliziraju prave trenutke kupovine i prodaje, a neizraziti sustavi zaključivanja izgledaju upravo tako. Također, i primjena na klasifikacijske probleme bi trebala biti jednostavna iz sličnih razloga. Jedino nešto složeniji pristup je za regresiju, jer se treba predvidjeti *točna* vrijednost u budućnosti, dok je za prethodna dva primjera dovoljno reći otprilike kakve su preporuke o kupovini ili prodaji dionica, ili odrediti klasu kojoj neki primjer pripada (a klasa uglavnom ima manji broj).

Zbog velikog broja pravila koja mogu nastati, za algoritam učenja je izabran genetski algoritam, koji je sposoban pronaći dobre optimume i u slučaju velikih prostora pretraživanja (uz ispravno odabrane genetske operatore i funkciju evaluacije).

U prvom dijelu rada bit će ukratko objašnjene sve komponente algoritma, tj. neizrazita logika, (financijski) vremenski slijedovi i genetski algoritam. Drugi dio rada je fokusiran na povezivanje svih komponenti u osnovne algoritme za probleme klasifikacije, regresije i analizu financijskih vremenskih slijedova, nakon čega će biti prikazani i objašnjeni rezultati navedenih algoritama.

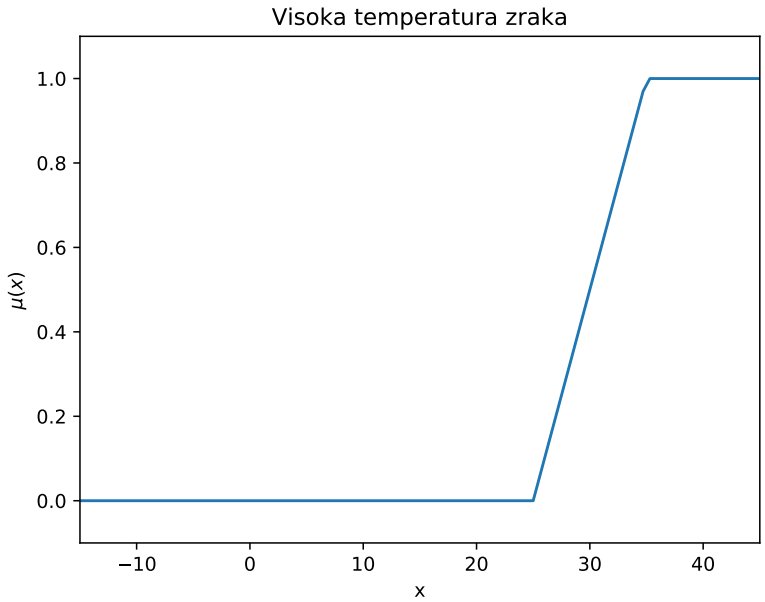
2. Neizrazita logika

2.1. Definicija

Pojam neizrazite logike se jednostavno može objasniti na primjeru sljedeće rečenice:

Jučer je bilo vruće vrijeme.

Postavlja se pitanje kako je definiran izraz *vruće*? Ako promatramo kroz okvir klasične logike, negdje moramo povući jasnu granicu pri kojoj temperaturi postaje vruće, npr. na $32°$C. Ali, ta definicija dovodi do zaključka da na temperaturi $31.99°$C nije vruće što ne bi trebalo biti točno.



Slika 2.1: Jezični izraz *vruće vrijeme*

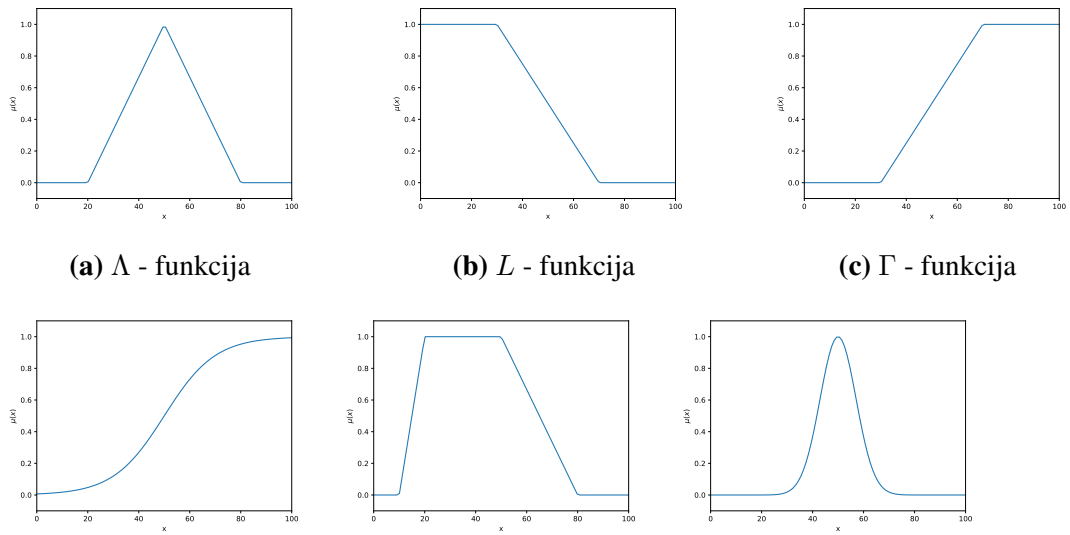
Taj problem se lako može riješiti uvođenjem neizrazite logike. Tako npr. izraz *vruće vrijeme* može biti definirano kao na slici. Time su riješeni problemi s tvrdim granicama gdje ih ne možemo jednostavno povući. Tu se javlja i pitanje kako definirati izraz, i po čemu je to gornja definicija ispravna, a neka druga ne, ali to nam nije toliko

ni bitno jer bi u većini slučajeva sve definicije otprilike bile slične.

2.1.1. Neizraziti skupovi

Za razliku od skupova u klasičnoj logici, gdje neki element skupa može samo pripadati ili ne pripadati skupu, neizrazita logika uvodi pojam *funkcije pripadnosti* (engl. *membership function*) i *stupnja pripadnosti* (engl. *membership degree*) kojima se opisuju elementi neizrazitog skupa. Svaki neizrazit skup je definiran nad klasičnim skupom, i to tako što se svakom elementu klasičnog skupa dodjeljuje funkcija koja određuje koji je stupanj pripadnosti tog elementa skupu. Funkcije pripadnosti mogu biti raznih oblika, a neki od najuobičajenijih su:

- Λ - funkcija
- L - funkcija
- Γ - funkcija
- σ - funkcija
- Π - funkcija
- $Gauss$ - funkcija



(a) Λ - funkcija **(b)** L - funkcija **(c)** Γ - funkcija

(d) σ - funkcija **(e)** Π - funkcija **(f)** $Gauss$ - funkcija

Slika 2.2: Neke od funkcija pripadnosti

Nakon definiranja funkcija pripadnosti treba odrediti i kako obavljati osnovne operacije iz klasične teorije skupova:

– unija

– presjek

– komplement

Sve tri operacije su u neizrazitoj logici generalizirane što omogućava definiciju operacija na različite načine.

U opisu svojstava pojedinih operacija se koriste sljedeće oznake:

– $s()$ - označava

– $t()$ - označava

– $c()$ - označava

– a, b, c, d - označavaju vrijednosti funkcije pripadnosti, tj. realne vrijednosti iz intervala $[0, 1]$

Operacija **unije** se u neizrazitoj logici naziva i *s-norma* i može biti definirana na više načina dok god su zadovoljena osnovna svojstva [17]:

1. $s(a, b) = s(b, a)$
2. $s(a, s(b, c)) = s(s(a, b), c)$
3. $a \leq c \wedge b \leq d \implies s(a, b) \leq s(c, d)$
4. $s(a, 1) = 1, s(0, a) = a$

Na sličan način se definira i **presjek** (*t-norma*), svojstva koja mora zadovoljavati su [17]:

1. $t(a, b) = t(b, a)$
2. $t(a, t(b, c)) = t(t(a, b), c)$
3. $a \leq c \wedge b \leq d \implies t(a, b) \leq t(c, d)$
4. $t(a, 1) = a, t(0, a) = 0$

Komplement mora zadovoljavati sljedeća svojstva [17]:

1. $c(0) = 1, c(1) = 0$
2. $a < b \implies c(a) > c(b)$
3. $c(c(a)) = a$

Za s-norme, t-norme i komplement vrijede svi zakoni klasične teorije skupova, osim zakona kontradikcije i zakona isključenja trećega [17]

Neki od osnovnih parova s i t-normi (uz Zadeh-ov komplement) su prikazani u tablici 2.1

Tablica 2.1: Parovi s-normi i t-normi

t-norma	s-norma
$min(a,b)$	$max(a,b)$
$\frac{ab}{a+b-ab}$	$\frac{a+b-2ab}{1-ab}$
ab	$a+b-ab$
$\frac{ab}{2-(a+b-ab)}$	$\frac{a+b}{1+ab}$
$max(0,a+b-1)$	$min(1,a+b)$

2.1.2. Jezične varijable

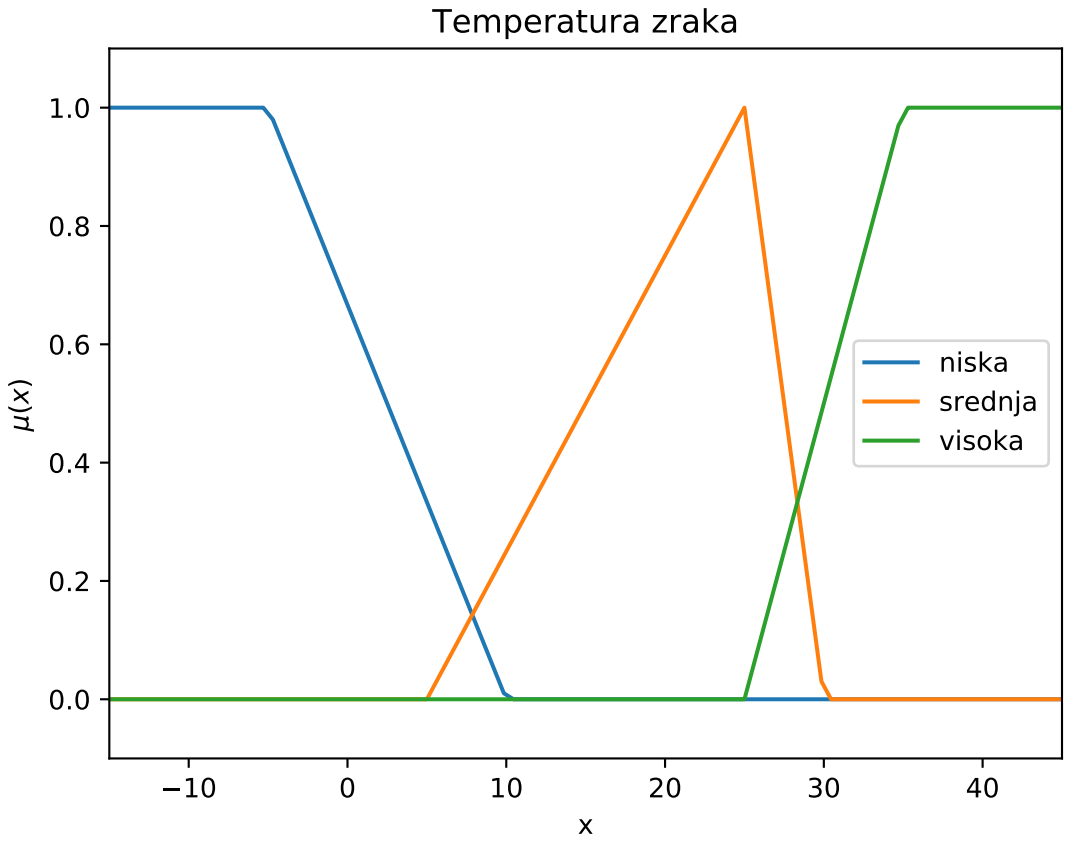
Jezične varijable su osnova neizrazitog zaključivanja. Svaka jezična varijabla sastoji se od domene, gramatike, skupa jezičnih izraza te naziva.

Domena jezične varijable označava područje nad kojim su izrazi jezične varijable definirani. Gramatika definira na koje sve načine je moguće definirati i koristiti jezične izraze. Jezični izraz označava jednu od mogućih vrijednosti koje jezični izraz poprima.

Prethodni pojmovi su lako objašnjivi kroz primjer jezične varijable *temperatura zraka*. Domena te jezične varijable može biti npr. $[-10, 45]°$C. Jezični izrazi koji bi možda odgovarali jezičnoj varijabli *temperatura zraka* mogu biti: *niska*, *srednja*, *visoka*. Definicije jezičnih izraza vidljive su iz slike. Dodatno, jezični izrazi mogu biti generirani i korištenjem gramatike koja, uz osnovne izraze, može upotrebom nekih modifikatora generirati i složenije izraze kao što su npr. *vrlo visoka*, *ne niska*. . . Naravno, kao i sve dosad, i jezični modifikatori mogu biti proizvoljno definirani.

2.2. Sustavi zaključivanja

Sustavi zaključivanja neizrazite logike sastoje se od skupa *ako-onda pravila*. U nastavku će biti objašnjeno osnovno o jednom pravilu, kao i donošenje ukupnog zaključka temeljem više pojedinačnih pravila.



Slika 2.3: Primjer jezične varijable temperatura zraka i pripadajućih jezičnih izraza

2.2.1. Ako-onda pravila

Svako ako-onda pravilo sastoji se od dva dijela: antecedenta i konsekvensa. Antecedent se odnosi na uvjete koji moraju biti zadovoljeni kako bi pravilo uopće moglo *okinuti*, tj. *ako* dio. Konsekvens se odnosi na zaključak koji se povezuje implikacijom sa antecedentom, tj. *onda* dio. Primjer ako-onda pravila može biti sljedeći:
Ako temperatura $=$ vruća $\wedge$ vlažnost $=$ visoka $\implies$ ne izlazi iz kuće

Antecedent se uglavnom sastoji od jezičnih izraza kojima ulazne varijable trebaju pripadati kako bi pravilo bilo aktivirano.

Konsekvens je nešto drugačiji, moguće su razne varijante konsekvensa ovisno o problemu koji rješavamo. Neki od tipova konsekvensa su

- Konsekvens u obliku neizrazitog skupa
- Konsekvens konstantne vrijednosti
- Konsekvens linearnih kombinacija

Konsekvens u obliku neizrazitog skupa je sličan dijelovima antecedenta, i to oblika AKO $\cdots$ ONDA $izlazna = A_1$ gdje je A_1 neki jezični izraz koji je definiran za izlazne varijable.

Konsekvens konstantne vrijednosti može se koristiti za klasifikacijske probleme, i oblika je AKO $\cdots$ ONDA $klasa = 1$. Dakle, u dijelu konsekvensa se nalazi samo jedna vrijednost, i to označava klasu kojoj pripada to pravilo.

Konsekvens linearnih kombinacija se uglavnom koristi za regresijske probleme. Jedan od oblika je taj gdje se u konsekvensu računa linearna kombinacija ulaznih varijabli. Drugi oblik može biti taj gdje se koriste proizvoljni jezični izrazi i računa se njihova linearna kombinacija.

I implikacija se može definirati na više načina, a uobičajeni operatori implikacije su:

- Mamdani - $min(a, b)$
- Lukasiewicz - $min(1, 1 - a + b)$
- Kleene-Diens - $max(1 - a, b)$

Operatori implikacije mogu biti lokalni i globalni što za pojedinačno pravilo nije tako važno, ali u sustavima zaključivanja dovodi do značajnih razlika.

Članovi antecedenta jednog pravila su povezani sa operatorom $\wedge$, pa se aktivacija antecedenta računa pomoću neke t-norme, a konačni zaključak se donosi izračunom implikacije toga s konsekvensom.

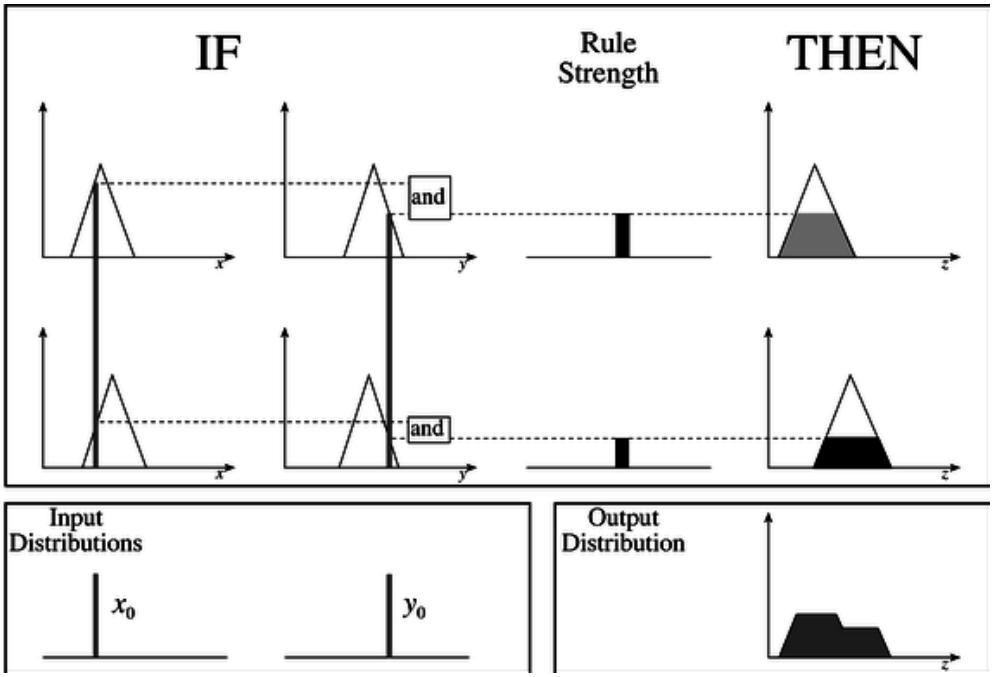
Tako dobijemo zaključak koji može biti vrijednost, ali i možemo ponovno dobiti neizrazit skup u kojem slučaju moramo napraviti dekodiranje neizrazitosti.

Dekodiranje neizrazitosti

Defazifikacija (engl. *Defuzzification*) označava pretvaranje neizrazitog zaključka u jednu vrijednost. Moguće je više metoda dekodiranja neizrazitosti, neke od najčešće korištenih su:

- težište (engl. *CoA - Center of Area*)
- srednji maksimum (engl. *MoM - Mean of Maximum*)
- polovište (engl. *BoA - Bisector of Area*)

U ovom radu u većini slučajeva korištena je metoda težišta kao, osim u posebnim slučajevima gdje to nije bilo moguće ili praktično.



Slika 2.4: Primjer zaključivanja temeljem više pravila, preuzeto sa `http://www.cs.princeton.edu/courses/archive/fall07/cos436/HIDDEN/Knapp/gfuzzy/fuzzy001.gif`

2.2.2. Zaključivanje temeljem više pravila

U osnovi su moguća dva načina zaključivanja temeljem više pravila:

- zaključci svih pravila zasebno se konačno spoje u konačni zaključak
- sva pravila se prethodno spoje u jednu relaciju temeljem koje se dobije odmah konačni zaključak

U ovom radu je korišten prvi način. Nakon izračuna zaključka jednog pravila kao što je opisano u prethodnom poglavlju (bez dekodiranja neizrazitosti), imamo $n_{pravila}$ zaključaka. Ovisno o tipu operatora implikacije (globalni / lokalni) imamo dva pristupa kombiniranju zaključaka.

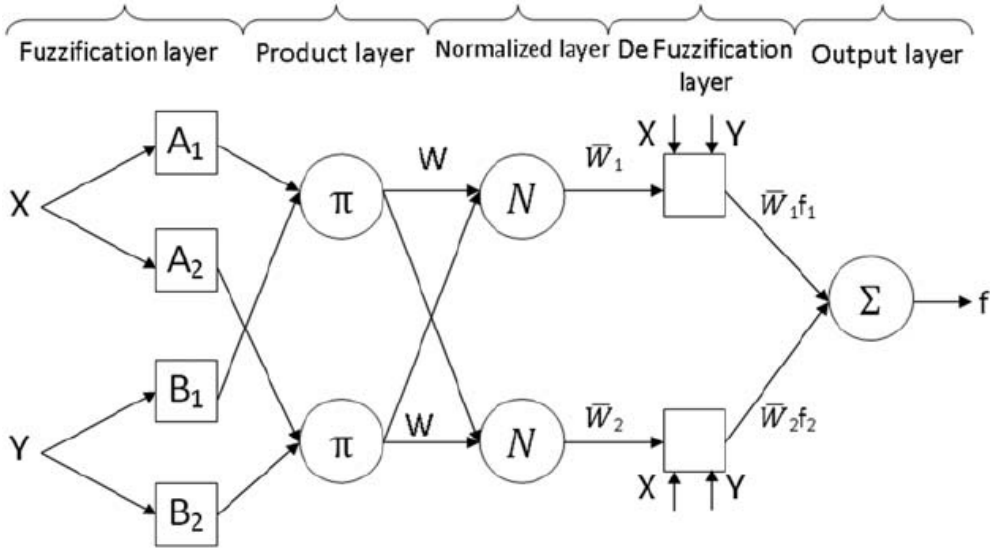
Tako, u slučaju globalnog operatora implikacije, zaključke kombiniramo korištenjem t-norme, dok u slučaju lokalnog operatora implikacije zaključke kombiniramo korištenjem neke s-norme.

Primjer zaključivanja temeljem više pravila vidi se na 2.4.

Nakon što izračunamo konačan zaključak, možemo pristupiti dekodiranju neizrazitosti i dobivanju konačnog rješenja.

2.3. ANFIS

ANFIS (engl. *Adaptive Neuro Fuzzy Inference System*) uz neizrazitu logiku koristi i (kao što i ime kaže) neuronsku mrežu. ANFIS omogućuje izravno učenje ako-onda pravila, ali i jezičnih izraza, tako da prethodno nije potrebno preveliko ekspertno znanje kao što je to slučaj u ručnom dizajniranju sustava zaključivanja.



Slika 2.5: Osnovni izgled ANFIS-a[13]

Iz slike 2.5 su vidljivi osnovni slojevi ANFIS-a:

- Sloj kodiranja neizrazitosti (engl. *Fuzzification layer*)
- Sloj t-norme (engl. *Product layer*)
- Sloj normalizacije (engl. *Normalized layer*)
- Sloj dekodiranja neizrazitosti (engl. *Defuzzification layer*)
- Izlazni sloj (engl. *Output layer*)

Sloj kodiranja neizrazitsoti zadužen je za učenje jezičnih izraza pojedinih pravila, ali i kodiranja ulaznih varijabli. Svaka ulazna varijabla vezana je na onoliko neurona tog sloja koliko ima pravila.

Nakon sloja kodiranja neizrazitosti, na red dolazi **sloj t-norme** koji računa aktivaciju antecedenta pojedinih pravila. Taj sloj sadrži onoliko neurona koliko ima pravila.

Sljedeći sloj, **sloj normalizacije**, normalizira aktivacije pojedinačnih pravila u odnosu na sva ostala. Na taj način je donošenje konačnog zaključka lako izvedivo korištenjem normaliziranih aktivacija kao težina u težinskoj sumi.

Sloj dekodiranja neizrazitosti računa linearnu kombinaciju ulaznih vrijednosti i taj rezultat, zajedno s normaliziranom aktivacijom šalje na zadnji sloj.

Zadnji sloj, **izlazni sloj**, računa težinsku sumu pojedinačnih zaključaka i time donosi konačan zaključak.

2.3.1. Algoritam učenja ANFIS-a

Kao i u drugim vrstama neuronskih mreža, moguće je koristiti backpropagation algoritam za pronalaženje optimalnih parametara mreže. Detaljniji izvodi i objašnjenja toga su u [17].

Ali, moguće je i optimirati parametre neuronskih mreža drugim algoritmima, kao npr. genetskim algoritmima, što je i rađeno u ovom radu. Tako možemo zanemariti da se raditi o neuronskoj mreži, genetskom algoritmu predati listu težina i pri evaluaciji rješenja te težine ponovno predati mreži, izračunati rezultat i ovisno o tome izračunati dobrotu rješenja zanemarujući u potpunosti informacije o gradijentima.

3. Vremenski slijedovi

Vremenski slijedovi se mogu definirati kao *uređeni slijed vrijednosti neke varijable*. Pojavljuju se u raznim oblicima, tako vremenski slijed može biti temperatura zraka grada Zagreba od 1980.g. do danas, cijena dionica Agrokora...

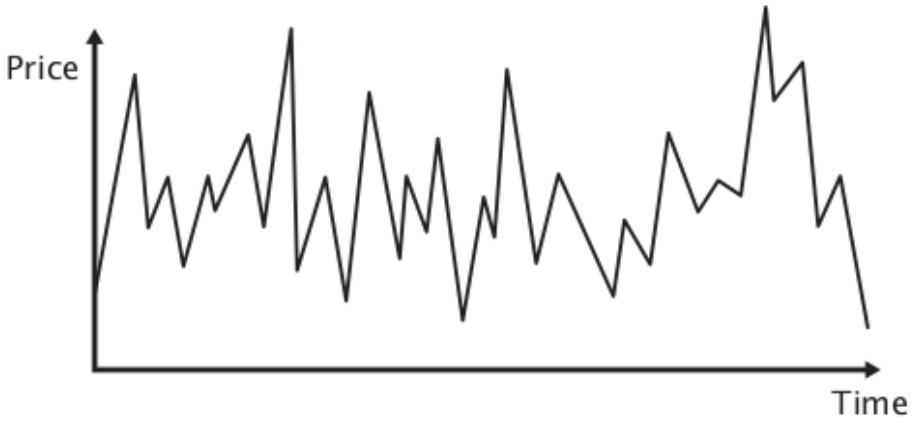
Prethodni primjeri su primjer univarijatnih vremenskih slijedova, ali su isto tako mogući i multivarijatni vremenski slijedovi.

3.1. Analiza vremenskih slijedova

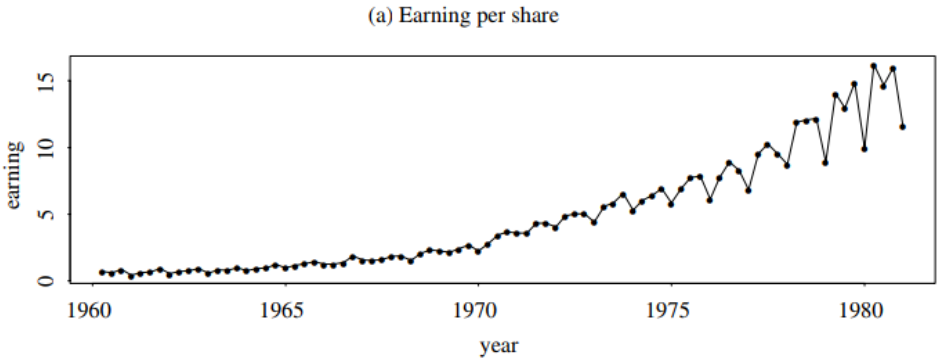
Analiza vremenskih slijedova se može odnositi u osnovi na dvije stvari:

- razumijevanje procesa kojima je nastao vremenski slijed
- predviđanje budućih vrijednosti, izvođenje novih značajki...

U ovom radu se sinonimom analize vremenskih slijedova smatra druga definicija. Svaki univarijatni vremenski slijed može biti obilježen s nekoliko osnovnih svojstava. **Stacionarnost** vremenskog slijeda govori o tome da se srednja vrijednost, varijanca i autokorelacija ne mijenjaju kroz vrijeme. To znači da se ne pojavljuje nikakav trend u podatcima. Primjer stacionarnog i nestacionarnog slijeda vidljiv je na slici 3.1.



Slika 3.1: primjer stacionarnog vremenskog slijeda [2]



Slika 3.2: Primjer sezonalnog vremenskog slijeda [2]

Sezonalnost označava periodične promjene nekong vremenskog slijeda. Tako npr. vremenski slijed koji prikazuje temperaturu zraka u Zagrebu, manifestira sezonalnost tako što se svake godine očekuje otprilike ista temperatura u istom mjesecu. Takav slijed je sezonalan na razini godine. Primjer sezonalnosti vidljiv je na slici 3.2.

Neke od uobičajenih metoda analize bit će objašnjene u sljedećem poglavlju.

3.1.1. Standardne metode analize vremenskih slijedova

Autoregresivni modeli (AR) su inačice obične linearne regresije gdje se trenutna vrijednost računa pomoću prošlih vrijednosti slijeda.

Osnovna formula modela izgleda ovako: $X_t = w_1 \cdot X_{t-1} + \cdots + w_n \cdot X_{t-n} + N_t + \delta$ gdje je X vremenski slijed, w težine koje se dodjeljuju pojedinim točkama u prošlosti, $\delta = \mu \cdot (1 - \sum_{i=1}^{n} w_i)$, a N_t označava šum. Vrijednost n se naziva i **redom** modela.

Modeli s pokretnim prosjecima (MA) su također inačice linearne regresije uz bitnu razliku u odnosu na autoregresivne modele. Ta razlika je što se u ovom slučaju ne koriste izravno prethodne vrijednosti slijeda, već samo prosjek, a koristi se u većoj mjeri šum. Formula modela je: $X_t = \mu + N_t + w_1 \cdot N_{t-1} + \cdots + w_n \cdot N_{t-n}$ gdje je X vremenski slijed, w težine koje se dodjeljuju točkama u prošlosti vrijednosti šuma (razlika u odnosu na autoregresivne modele), N šum, a μ označava srednju vrijednost slijeda.

Kombiniranjem prethodna dva modela dobije se najčešće korišteni model, takozvani **ARMA modeli** čiji i sam naziv dolazi iz činjenice da su koristeni autoregresivni (AR) i modeli s pokretnim prosjecima (MA (engl. *Moving average*)). Osnovna formula takvih modela je suma prethodna dva modela:

$X_t = \mu + N_t + w_1 \cdot N_{t-1} + \cdots + w_n \cdot N_{t-n} + \phi_1 \cdot X_{t-1} + \cdots + \phi_q \cdot X_{t-q} + \delta$

Kako bi ARMA model bio uspješan, bitno je da je vremenski slijed stacionaran. U

slučaju nestacionarnih slijedova, preporučeno je integrirati nestacionarni slijed nekoliko puta kako bi se dobio stacionaran. Time dobivamo varijatnu ARMA modela koja se naziva ARIMA (I - integriran). Osim osnovne formule, moguće je proširit model i sezonalnim izrazima za AR i MA modele.

3.2. Financijski vremenski slijedovi

Financijski vremenski slijedovi su nešto složeniji od običnih vremenskih slijedova. Po svojoj prirodi su jako nepredvidivi (za razliku od npr. predviđanja temperature zraka). Zbog toga se razvila cijela znanost oko pronalaženja uzoraka u financijskim vremenskim slijedovima, određivanja tehničkih indikatora koji mogu pomoći pri analizi financijskih vremenskih slijedova i još mnogo uspješnih, ali i neuspješnih teorija.

3.2.1. Analiza financijskih vremenskih slijedova

Uglavnom se za analizu financijskih vremenskih slijedova koriste OHLC (open, high, low, close) grafovi koji po određenom intervalu prikazuju cijenu otvaranja burze, najvišu i najnižu te cijenu pri zatvaranju burze. Nad takvim grafovima postoje različite metode utvrđivanja smjera kretanja cijene, je li u nekom trenutku precijenjeno ili podcijenjeno tržiste i sl...

Uz metode temeljene na grafovima, javlja se i druga skupina metoda koje pokušavaju temeljem cijene (i informacijama o prošlim cijenama) izračunati i neke tehničke indikatore koji bi isto tako trebali pokazati koji je smjer kretanja neke dionice/kriptovalute/valutnog para.

Neki od osnovnih indikatora su:

- RSI
- Stohastički oscilator
- MACD

RSI (engl. *Relative Strength Index*) mjeri brzinu i promjenu cijene. Vrijednost je uvijek u intervalu $[0, 100]$. Interpertacija ovog indikatora može biti takva da se za visoke vrijednosti (> 70) može smatrati prekupljenim (engl. *overbought*), dok za niske vrijednosti (< 30) može smatrati preprodanim (engl. *oversold*). Naravno, ovisno o potrebama i ponašanju indikatora, ove granice je poželjno pomjeriti.

Stohastički oscilator pokazuje momentum cijene. Kao i kod prethodnog indikatora, vrijednost je uvijek u intervalu $[0, 100]$. Moguća interpretacija je da u slučaju da

se tržište kreće pozitivno, cijene zatvaranja (engl. *close*) će biti bliže najvišoj cijeni (engl. *high*), dok će u slučaju negativnog kretanja tržišta cijena zatvaranja biti bliža najnižoj cijeni (engl. *low*).

MACD (engl. *Moving Average Convergence Divergence*) je indikator koji pokazuje odnose između dva pokretna prosjeka cijene. Eksponencijalni pokretni prosjeci se uglavnom računaju na nešto duljem i nešto kraćem pomičnom prozoru. Kraći pomični prozor pokazuje kratkotrajni trend, a dulji pokazuje dugotrajni trend. Računa se tako što se dulji prosjek oduzme od kraćeg i tako dobivena linija se uspoređuje sa još kraćim prosjekom koji se naziva signalna linija (engl. *signal line*). Vrijednosti koje se uglavnom koriste su 26, 12 i 9 dana za dulji, kratki prosjek i signalnu liniju redom. U slučaju križanja MACD-a sa signalnom linijom donose se odluke o kupovini ili prodaji.

Uz prethodne indikatore bitno je spomenuti i **volumen** (engl. *volume*). On označava količinu trgovanja u određenom vremenskom periodu.

Zbog prirode financijskih vremenskih slijedova (izražena stohastičnost, naziva *financijski* u sebi koji implicira da netko mora izgubiti kako bi netko drugi dobio) teško je razviti i koristiti ustaljene *zakone* trgovanja pa i razviti algoritam. Stoga se često algoritmi dizajniraju tako da netko uoči određenu neefikasnost u financijskom vremenskom slijedu i to iskoristi u svoju korist. Ali, nakon nekog vremena sve više ljudi primjećuje isto to i tako se ta neefikasnost ukloni i potrebno je tražiti nove načine. Zato se često događa da isti algoritam ne funkcionira nakon nekog vremenskog perioda i potrebno ga je mijenjati (ponekad i sitno, ali promjena je promjena). Zaključak toga je da su financijski vremenski slijedovi nestabilni, i algoritam koji bi trebao biti uspješan se mora stalno dorađivati (ili algoritam može i sam uvidjeti da mu ne ide najbolje pa se sam popraviti - algoritmi strojnog učenja).

4. Genetski algoritam

Genetski algoritam pripada obitelji evolucijskih algoritama koji su inspirirani evolucijom. U genetskom algoritmu upravlja se, ne jednim rješenjem, nego cijelom populacijom rješenja. Raznim genetskim operatorima primijenjenim na jedinke populacije se kroz više iteracija (generacija) dolazi do što boljih rješenja.

Postoje dvije osnovne inačice genetskog algoritma: generacijski i eliminacijski. **Generacijski** genetski algoritam između iteracija zamijeni cijelu populaciju novom populacijom, dok **eliminacijski** u svakoj iteraciji zamijeni samo jednu od lošijih jedinki. U ovom radu korišten je eliminacijski genetski algoritam.

Osnovni genetski operatori su selekcija, križanje i mutacija.

4.1. Selekcija

Selekcija se koristi za odabir jedinki koje idu u sljedeću generaciju, jedinki koje se koriste u križanju, ali i u odabiru jedinki koje se eliminiraju ako se koristi eliminacijski algoritam. Operator selekcije je važan što možemo kontrolirati (selekcijski pritisak) koliko velik utjecaj želimo dobrih rješenja u odnosu na loša.

4.2. Križanje

Genetski operator križanja zadužen je za fino pretraživanje prostora stanja i to tako što kombinira dvije jedinke (roditelje) i koristeći značajke obje dobija treću koja je nerijetko bolja, pa i jako bolja od roditelja.

4.3. Mutacija

Mutacija je sličnija nasumičnom pretraživanju od križanja i omogućuje široko pretraživanje prostora stanja, i često nam baš mutacija dozvoljava bijeg iz raznih lokalnih

optimuma.

5. Učenje neizrazitih pravila za predviđanje vremenskih slijedova

Za učenje neizrazitih pravila korišten je genetski algoritam. Kako bi genetski algoritam bio primjenjiv, potrebno je definirati genotipe, ali i odrediti jezične izraze za svaku značajku koja se pojavljuje u skupu koji se predviđa.

Sustav neizrazitog zaključivanja korišten je za klasifikacijske, regresijske probleme, ali i na probleme *predviđanja* financijskih vremenskih slijedova. U slučaju financijskih vremenskih slijedova neće se predviđati točna cijena (jer je to praktično nemoguće zbog same prirode financijskih slijedova i financijskih tržišta), nego će se pokušati pronaći optimalni trenutci za kupovanje ili prodavanje određene dionice/kriptovalute...

U radu su kao genotipi genetskog algoritma korišteni nizovi realnih brojeva, ali samo zato što je u korištenom programskom okviru puno veća podrška bila za taj oblik genotipa, nego za nizove cijelih brojeva. U ovom slučaju pri dekodiranju genotipa, uvijek se realni brojevi pretvaraju u cijele prije evaluacije rješenja.

5.1. Automatsko određivanje jezičnih izraza

Jedan od najvećih problema neizrazitih sustava zaključivanja je potrebno domensko znanje pri definiranju jezičnih izraza. Zato je bitno istražiti mogućnosti automatskog određivanja jezičnih izraza.

U literaturi [4] se pojavljuju dva osnovna načina automatskog određivanja jezičnih izraza:

- maksimiziranje entropije
- trapezoidalni izrazi

Maksimiziranje entropije je metoda koja omogućava određivanje bitnih značajki skupa nad kojim se određuju jezični izrazi. Ta metoda je slična onome što bi netko s ekspertnim znanjem zaključio jer izravno iz podataka vidi što je bitno.

U ovom radu je korištena metoda generiranja trapezoidalnih jezičnih izraza pa će ona biti detaljno objašnjena.

5.1.1. Generiranje trapezoidalnih izraza

Ova metoda dijeli ulazni skup na n jednakih trapezoidalnih područja i to sljedećim koracima:

- Definiranje univerzalnog skupa
- Izgradnja funkcija pripadnosti

Univerzalni skup se određuje tako što prvo izracunamo standardnu devijaciju σ iz podataka, i kao univerzalni skup definiramo interval $U = [D_{min} - \sigma, D_{max} + \sigma]$ gdje D_{min} predstavlja minimalnu vrijednost koja se pojavljuje u skupu, a D_{max} predstavlja maksimalnu vrijednost koja se pojavljuje u skupu. Dodavanje σ u izračun omogućava da i dosad neviđene vrijednosti u većini slučajeva ulaze u to područje. Ovisno o skupu i očekivanim vrijednostima, umjesto σ može se koristiti i 2σ ili neki drugi faktor.

Za određivanje **funkcija pripadnosti** potrebno je particionirati skup podataka na n jednakih dijelova. To se radi tako što prvo izračunamo raspon podataka $R = U_{max} - U_{min}$ gdje su U_{max} i U_{min} gornja i donja granica univerzalnog skupa. Tako dobiveni raspon možemo iskoristiti za izračun udaljenosti karakterističnih točaka trapezoidalne (Π) funkcije pripadnosti $S = \frac{R}{2n+1}$. Trapezoidalna funkcija pripadnosti koristi 4 parametra - α, β, γ, δ, koji predstavljaju karakteristične točke funkcije. Tako dobivenim vrijednostima možemo izgenerirati n jezičnih izraza s pripadajućim funkcijama pripadnosti.

Primjer

Tablica 5.1: Primjer vremenskog slijeda za generiranje izraza

Vrijednost	1	4	3	7	8	2	5	5	9	2	5

Rad algoritma generiranja trapezoidalnih izraza može se pokazati na primjeru vremenskog slijeda iz tablice 5.1. Krenimo redom, kako bi dobili ispravan univerzalni skup, potrebno je izračunati standardnu devijaciju podataka. Jednostavnim računom dobijamo da je $\sigma = 2.58$. Time možemo definirati univerzalni skup nad intervalom $[1 - 2.58, 9 + 2.58] \rightarrow [-1.58, 11.58]$.

Nakon toga počinjemo s određivanjem funkcija pripadnosti. Raspon podataka dobijemo jednostavno sa $R = U_{max} - U_{min} = 11.58 - (-1.58) = 13.16$. Recimo da želimo podatke particionirati na $n = 7$ jezičnih izraza, time ćemo dobiti $S = \frac{R}{2n+1} = \frac{13.16}{15} = 0.878$. Zatim, počevši od U_{min} kao parametra α trapezoidalne funkcije računamo ostale parametre nakon čega parametar γ prethodno određene funkcije koristimo kao parametar α sljedeće. Rezultati algoritma vidljivi su u tablici 5.2.

Tablica 5.2: Izračunate funkcije pripadnosti

Naziv izraza	Trapezoidalna funkcija
L_0	$\Pi(-1.58, -0.70, 0.18, 1.05)$
L_1	$\Pi(-0.18, 1.05, 1.93, 2.81)$
L_2	$\Pi(1.93, 2.81, 3.69, 4.57)$
L_3	$\Pi(3.69, 4.57, 5.44, 6.32)$
L_4	$\Pi(5.44, 6.32, 7.20, 8.08)$
L_5	$\Pi(7.20, 8.08, 8.96, 9.83)$
L_6	$\Pi(8.96, 9.83, 10.71, 11.58)$

5.2. Klasifikacija

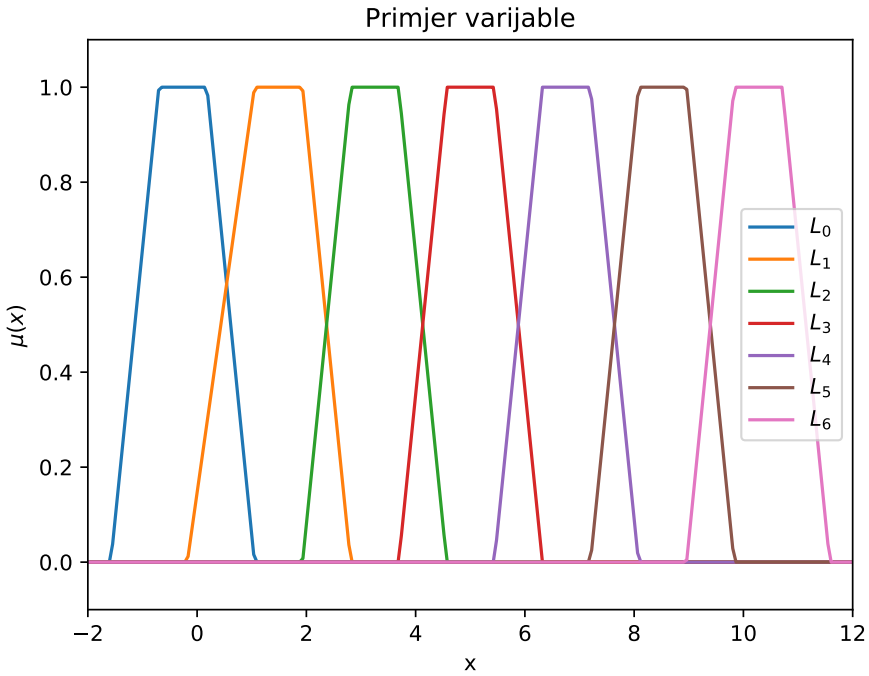
5.2.1. Genotip

Genotip klasifikacijskih problema se sastoji od dva niza realnih brojeva. Prvi niz predstavlja antecedent i sastoji se od $n_{pravila} \cdot n_{varijable}$ vrijednosti. Za svako pravilo govori koja varijabla bi trebala pripadati kojem izrazu kako bi antecedent bio zadovoljen. Drugi niz predstavlja konsekvens i sastoji se od $n_{pravila}$ vrijednosti. Svaka vrijednosti određuje kojoj klasi pripada koje pravilo. Primjer genotipa vidljiv je na 5.2.

Može se primijetiti da se u isto vrijeme koristi cijeli sustav zaključivanja, a zaključivanje se radi tako što se uzima ono pravilo čiji antecedent najviše *okida* (ovdje su moguće kombinacije s glasanjem, težinskim sumama...).

5.3. Regresija (prvi način)

Za regresiju se koristi kooperativna koevolucija s onoliko podpopulacija koliko pravila treba odrediti. Evaluira se svako rješenje zasebno, a na kraju svake generacije



Slika 5.1: Izračunate funkcije pripadnosti

$$\begin{bmatrix} 2 & 4 \\ 3 & 1 \\ 5 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$$

(a) Antecedent dio **(b)** Konsekvens dio

Slika 5.2: Primjer genotipa za klasifikaciju sa 3 pravila sa po 2 jezicne varijable i 3 klase

se i računa koevoluirano rješenje tako što se kombiniraju proporcionalnom selekcijom odabrana rješenja iz svake podpopulacije (proporcionalna selekcija je odabrana jer se često dogodi da podpopulacije evoluiraju u slična rješenja pa kombiniranjem sličnih pravila ne dobijemo bolji sustav zaključivanja).

5.3.1. Genotip

Genotip jednog rješenja se sastoji od $2 \cdot n_{varijable}$ vrijednosti koje predstavljaju jedno cijelo pravilo. Prvih $n_{varijable}$ vrijednosti određuju antecedent, dok drugih $n_{varijable}$ vrijednosti određuju težine kojim će se vršiti linearna kombinacija. Primjer genotipe vidljiv je na 5.3.

Zaključak jednog pravila se donosi tako što se aktivacija antecedenta pomnoži sa linearnom kombinacijom svih ulaznih vrijednosti.

Konačni zaključak koevoluiranog sustava se donosi tako što se računa težinski prosjek zaključaka pojedinačnih pravila pri čemu je faktor težine aktivacija antecedenta pojedinog pravila.

$$\begin{bmatrix} 2 & 4 & 1 & 2.3 \end{bmatrix}$$

Slika 5.3: Primjer genotipa za prvi način regresije sa 2 jezicne varijable

5.4. Regresija (drugi način)

Drugi način regresije koristi verziju algoritma spomenutog u [4]

5.4.1. Osnovni algoritam

Osnovna verzija algoritma sastoji se od niza koraka:

- Generacija jezičnih izraza
- Dodjeljivanje točaka jezičnim izrazima
- Određivanje pravila
- Predviđanje

Generacija jezičnih izraza se radi na prethodno objašnjeni način generiranja trapezoidalnih funkcija. Polazni slijed se *particionira* u n trapezoidalnih funkcija.

Nakon toga dolazi do **dodjeljivanja točaka jezičnim izrazima**. Kako se trapezoidalne funkcije mogu preklapati, određivanje se radi na način da se vidi koja trapezoidalna funkcija za određenu vrijednost daje najviši stupanj pripadnosti i onda se toj vrijednosti dodijeli ta funkcija.

Određivanje pravila se radi tako što se za svaku točku (tj. za svaki dodijeljeni jezični izraz) zapisuje koji mu je sljedeći jezični izraz. Tako će svaki jezični izraz koji se pojavljuje sebi imati dodijeljenu listu jezičnih izraza koji su dosad dolazili poslije njega.

Nakon određivanja pravila, **predviđanje** se radi vrlo jednostavno. Uzmemo trenutnu točku, dodijelimo joj jezični izraz, pogledamo u listu pridijeljenih jezičnih iz-

raza tom jezičnom izrazu i izračunamo srednju vrijednost (ili težinsku sumu) tih izraza i time smo dobili sljedeću vrijednost.

Pretpostavimo da su prva 2 koraka algoritma riješena i imamo ovakav niz:

Tablica 5.3: Primjer vremenskog slijeda

Vrijednost	Jezični izraz
2	L1
3	L2
4	L2
1	L1
4.5	L3
5	L3
4.5	L2
6	L3
1	L1
4.8	L3

Tablica 5.4: Jezični izrazi

Naziv	Trapezoidalna funkcija
L_1	$(0, 1, 2, 3)$
L_2	$(2, 3, 4, 5)$
L_3	$(4, 5, 6, 7)$

Pravila dobivena nakon trećeg koraka bi izgledala ovako:
$L_1 \rightarrow L_2, L_3, L_3$
$L_2 \rightarrow L_2, L_1, L_3$
$L_3 \rightarrow L_3, L_2, L_1$

U ovom trenutku možemo krenuti na predviđanje vremenskog slijeda, radi jednostavnosti, pokušat ćemo iz pravila predvidjeti polazni vremenski slijed sa jednostavnim prosjekom, ali i težinskim prosjekom gdje je težina ovisna o frekvenciji pojavljivanja jezičnog izraza u listi pravila za pojedini izraz.

Iz tablice 5.5 se može vidjeti da u ovom jednostavnom primjeru ovakva metoda i nije najpouzdanija, ali korištenjem više jezičnih izraza, čak i ova jednostavna metoda

Tablica 5.5: Predviđanje

Vrijednost	Jezični izraz	Predviđanje prosjekom	Predviđanje težinskim prosjekom
2	L1	4.5	4.83
3	L2	3.5	3.5
4	L2	3.5	3.5
1	L1	4.5	4.83
4.5	L3	3.5	3.5
5	L3	3.5	3.5
4.5	L2	3.5	3.5
6	L3	3.5	3.5
1	L1	4.5	4.83
4.8	L3	3.5	3.5

može dovesti do kvalitetnih rezultata.

Navedeni algoritam je moguće proširiti tako što pri kreiranju pravila umjesto samo u trenutni trenutak, gledamo i u prošlost. Tako bi ovaj algoritam drugog reda koristio informaciju i samo jednog prošlog trenutka uz trenutni. Pravila za tako definiran algoritam bi izgledala ovako:
$(L1, L2) \rightarrow L2, L3$
$(L2, L2) \rightarrow L1$
$(L1, L3) \rightarrow L3, L3, L1$
$(L3, L3) \rightarrow L2$
$(L2, L3) \rightarrow L3, L1$

Predviđanje se radi na isti način kao i kod algoritma prvog reda.

Još jedno moguće proširenje je korištenje genetskog algoritma za određivanje optimalnih težina kojim se računa težinska suma pri predviđanju. Time se ne bi oslanjali na frekvencije pojavljivanja, nego i na samu prirodu podataka.

5.4.2. Genotip

Genotip takvog rješenja bi se sastojao od dva niza realnih vrijednosti. Prvi niz predstavlja antecedente za sva pravila, tj. kojim jezičnim izrazima treba pripadati zadnjih k (ovisno o redu prethodnog algoritma) vrijednosti kako bi se pravilo aktiviralo. Drugi niz predstavlja težine koje su uz jezične izraze na desnoj strani kojima se radi težinska

suma kako bi se došlo do ispravnog zaključka (predviđanja).

5.5. Financijski vremenski slijedovi

Financijski vremenski slijedovi su nešto specijaliziraniji od prethodnih primjera. U ovom slučaju se ne predviđa nikakva vrijednosti već se radi vrsta podržanog učenja gdje se dobrota jedinke ocjenjuje u tome koliko joj je *novca* ostalo na kraju simulacije nad skupom za učenje.

5.5.1. Genotip

Genotip rješenja se sastoji od dva niza realnih vrijednosti. Kao i u problemu klasifikacije, prvi predstavlja antecedente za sva pravila, dok drugi predstavlja konsekvense svih pravila. Prvi niz se sastoji od $n_{varijabli} \cdot n_{pravila}$ vrijednosti. Drugi niz se sastoji od $2 \cdot n_{pravila}$ vrijednosti koje označavaju za svako pravilo treba li u tom trenutku kupiti dionicu ili prodati.

5.5.2. Priprema podataka

Financijski slijedovi uglavnom dolaze u OHLC (open, high, low, close) obliku. Izravno korištenje informacije o cijeni ne bi dovelo do dobrih rezultata s neizrazitom logikom. Razlog tome je vrlo jednostavan: ako algoritmu damo cijenu na nekom vremenskom intervalu i nad time definiramo jezične izraze, konačna pravila će biti oblika - ako $cijena = niska$ onda $kupi$; ako $cijena = visoka$ onda $prodaj$. Time ne bi ništa postigli jer kad bi znali u svakom trenutku da je cijena visoka ili niska, ne bi nam ovakvi *pametni* algoritmi niti trebali.

Zbog toga se umjesto informacija o cijeni koriste neki od tehničkih indikatora objašnjenih ranije. Značajke koje se koriste su:

- Volumen trgovanja
- Relativni indeks snage (RSI)
- Stohastički oscilator
- MACD
- Promjena vrijednosti u postotcima

6. Rezultati

Za sve algoritme je korišten isti način generacije neizrazitih izraza, i to prethodno opisani trapezoidalni način 5.1.1 gdje svaka jezična varijabla bude podijeljena na 7 jezičnih izraza koji se mogu interpretirati kao gradacija jezične varijable od jako jako niskih vrijednosti, do jako jako visokih vrijednosti.

6.1. Klasifikacija

Algoritam klasifikacije isproban je na skupu *iris*, preuzet sa [6] gdje je cilj ispravno klasificirati vrstu cvijeta iris po njegovim dimenzijama. Postoje 3 klase i svaka je jednako zastupljena u početnom skupu - po 50 primjera. Iako ovaj skup nije primjer vremenskog slijeda, i na njemu je moguće prikazati mogućnosti ovakvog pristupa problemima klasifikacije.

Pri učenju je korišteno 70% skupa, dok je ostalih 30% korišteno kao ispitni skup.

Kao funkcija pogreške se koristi pogreška klasifikacije, tj. postotak pogrešno klasificiranih primjera, rezultati su vidljivi na slici 6.1. Korišteni parametri su vidljivi u tablici 6.1.

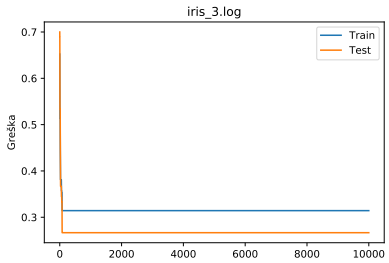
Tablica 6.1: Korišteni parametri

Naziv parametra	Vrijednost
vjerojatnost mutacije	0.3
veličina populacije	100

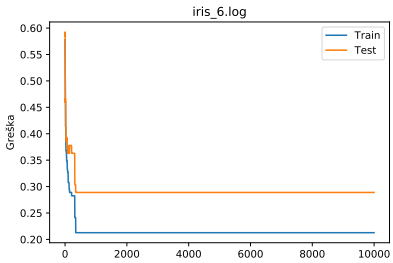
6.2. Regresija

Algoritam regresije je ispitan na skupu koji prikazuje broj zaklanih svinja u Alabami.

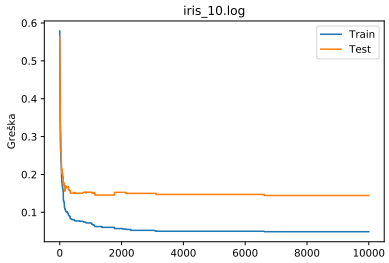
Pri učenju je korišteno 70% skupa, dok je ostalih 30% korišteno kao ispitni skup.



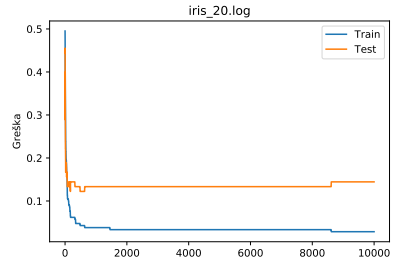
(a) Rezultati sa 3 pravila



(b) Rezultati sa 6 pravila



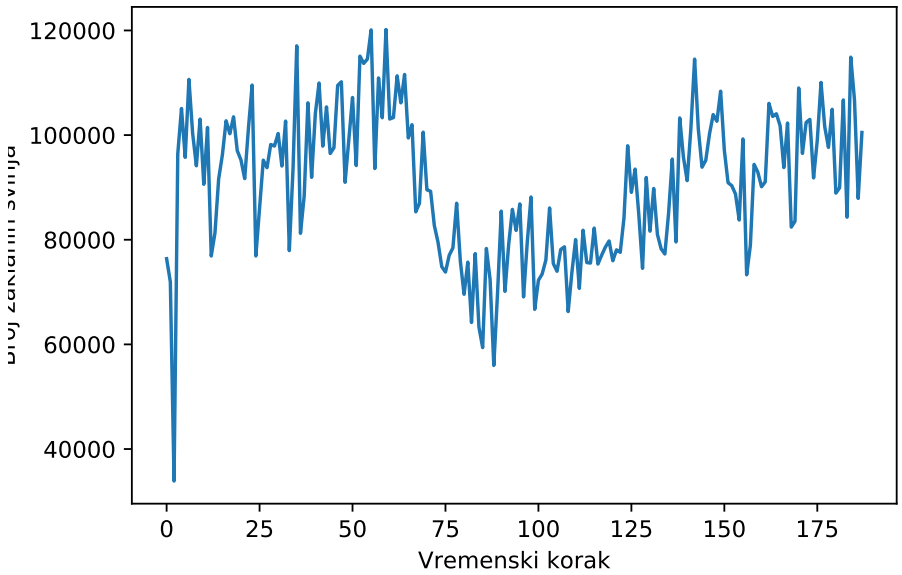
(c) Rezultati sa 10 pravila



(d) Rezultati sa 20 pravila

Slika 6.1: Rezultati klasifikacije

Kao funkcija pogreške se koristi *MAPE*.



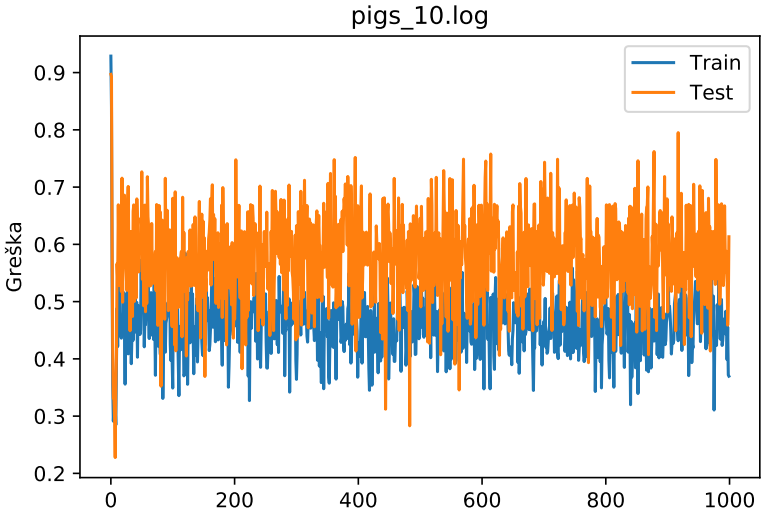
Slika 6.2: Broj zaklanih svinja po mjesecima u Alabami

6.2.1. Prvi način regresije

Korišteni parametri su vidljivi u tablici 6.2

Tablica 6.2: Korišteni parametri

Naziv parametra	Vrijednost
vjerojatnost mutacije	0.3
broj podpopulacija	10, 20
veličina podpopulacije	50



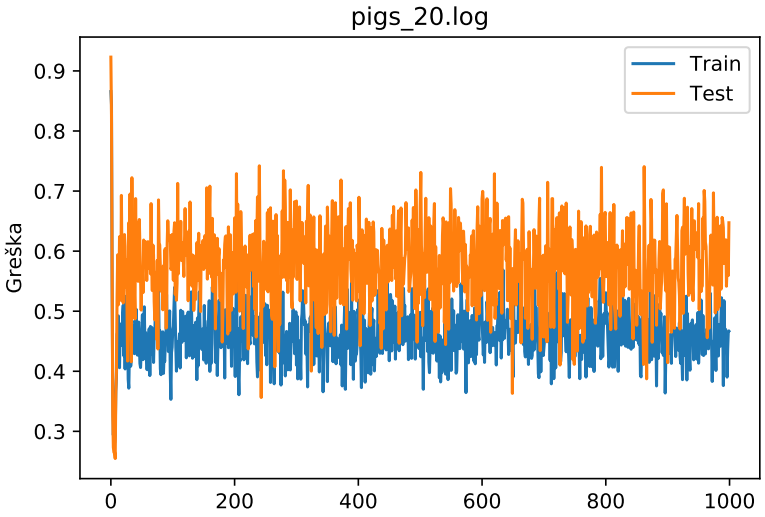
Slika 6.3: Rezultati sa 10 podpopulacija

Koveoluirano rješenje se odabire proporcionalnom selekcijom u podpopulacijama pa se stoga kroz generacije ne pokazuje neko poboljšanje, nego je ponašanje stohastično i bolja rješenja se pojavljuju povremeno. Ovakav način se nije pokazao dobrim za stabilno učenje pravila jer se često dogodi da podpopulacije evoluiraju u slična pravila čija kombinacija ne dovodi do boljih rješenja, a jedno pravilo samo po sebi nije sposobno dovoljno dobro predviđati buduće vrijednosti.

6.2.2. Drugi način regresije

Korišteni parametri su vidljivi na tablici 6.3

Prvo je pokrenut početni algoritam (prvog reda) koji određuje pravila, a zatim se optimiraju težine koje sudjeluju u linearnoj kombinaciji pojedinih pravila. Dobivena



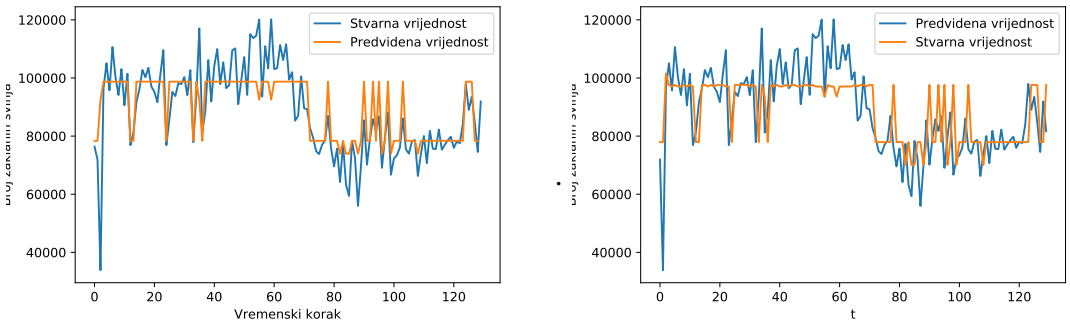
Slika 6.4: Rezultati sa 20 podpopulacija

Tablica 6.3: Korišteni parametri

Naziv parametra	Vrijednost
vjerojatnost mutacije	0.5
veličina populacije	50

pravila i rezultati početnog algoritma su vidljivi na slici 6.5.

Zanimljivost korištenog skupa i generacije jezične varijable koja označava broj zaklanih svinja, je ta što se u skupu ne pojavljuje jezični izraz L_1. Razlog tome je vrijednost koja značajno odstupa od ostatka skupa koja se vidi na slici 6.2.



(a) Rezultati regresije početnog algoritma **(b)** Rezultati regresije nakon učenja težina

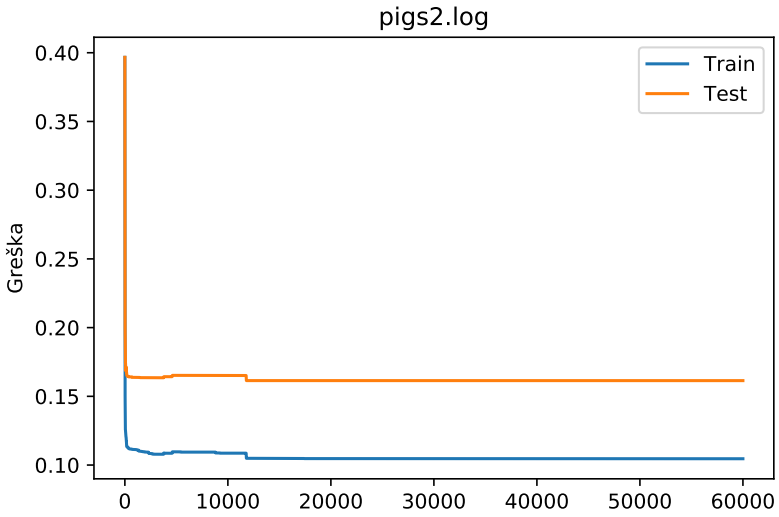
Slika 6.5: Rezultati drugog načina regresije

Tablica 6.4: Izračunate funkcije pripadnosti za jezičnu varijablu broja zaklanih svinja

Naziv izraza	Trapezoidalna funkcija
L_0	$\Pi(18890.16, 26641.94, 34393.72, 42145.50)$
L_1	$\Pi(34393.72, 42145.50, 49897.28, 57649.05)$
L_2	$\Pi(49897.28, 57649.05, 65400.83, 73152.61)$
L_3	$\Pi(65400.83, 73152.61, 80904.39, 88656.17)$
L_4	$\Pi(80904.39, 88656.17, 96407.95, 104159.72)$
L_5	$\Pi(96407.95, 104159.72, 111911.50, 119663.28)$
L_6	$\Pi(111911.50, 119663.28, 127415.06, 135166.84)$

Tablica 6.5: Početna pravila

Antecedent	Konsekvens
L_6	L_4, L_3, L_5
L_5	L_4, L_3, L_5, L_6
L_3	L_2, L_3, L_0, L_4
L_4	L_6, L_5, L_4, L_3, L_2
L_2	L_2, L_3
L_0	L_4



Slika 6.6: Rezultati algoritma učenja regresije nad početnim pravilima po generacijama

Tablica 6.6: Usporedba rezultata regresije (*MAPE*)

Početni algoritam	Učenje težina
10.15	9.9

Naučene težine ovim algoritmom, dovode do nešto boljih rješenja od običnog prosjeka ili težinske sume korištenjem frekvencije pojavljivanja. Korišten je algoritam prvog reda, pa je teško doći do boljih rezultata od prikazanih.

6.3. Financijski vremenski slijedovi

Zbog specifičnosti financijskih vremenskih slijedova, korišten je poseban način simulacije kako bi se omogućili što točniji rezultati. Korišteni skupovi u ovom radu su prikupljeni sa `https://cryptowat.ch/` i odnose se na cijene kriptovaluta u odnosu na američki dolar.

Tijekom učenja se koristi validacija s k preklopa (engl. *k-fold*) gdje je k postavljen na 5 i to tako što se ulazni skup podijeli na 5 dijelova pa se tijekom učenja svakih 100 generacija promijeni ispitni *fold*. Ovakvim načinom se može dobiti točnija procjena generalizirane pogreške, a i omogućeno je učenje nad različitim podskupovima ulaznog skupa.

Dobrota jedinke se dobija pojednostavljenom simulacijom trgovanja. Na početku simulacije jedinka dobije određeni iznos *novca* s kojim može baratati (desetorostruki prosjek na skupu). U svakom trenutku može ili kupiti ili prodati samo 1 količinu. Ako prodaje, ne mora nužno posjedovati, već može *posuditi* pa kasnije vratiti po nešto skupljoj cijeni (1.5 puta većoj cijeni nego što je). I konačno, sve što na kraju ostane ili posuđeno ili kupljeno se razriješi u *novac*. Nakon toga, dobrota se računa kao omjer preostalog iznosa nakon simulacije i početnog iznosa:

$fit = \frac{balance_{end}}{balance_{start}}$

Optimalni parametri 6.8 dobiveni su pokretanjem algoritma na istom skupu sa različitim kombinacijama vrijednosti parametara što se može vidjeti na slici 6.7. Za razliku od uobičajenih algoritama, gdje se za validaciju koristi samo informacija iz ispitnog skupa, ovdje je korištena geometrijska sredina skupa za učenje i ispitnog skupa, jer je očekivano ako se pojavi isti uzorak nekad u budućnosti, da naučeni model može iznadprosječno *zaraditi*.

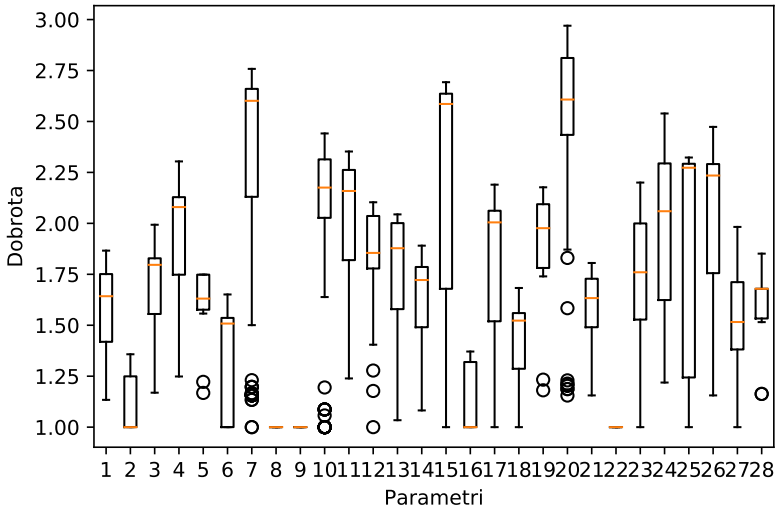
Nakon odabira optimalnih parametara, algoritam je pokrenut i na ostalim skupo-

Tablica 6.7: Odabir parametara

Naziv parametra	Isprobane vrijednosti
vjerojatnost mutacije	$(0.3, 0.5, 0.9)$
granica prihvaćanja pravila	$(0.7, 0.8, 0.9)$
veličina populacije	$(30, 100, 250)$

Tablica 6.8: Korišteni parametri

Naziv parametra	Vrijednost
vjerojatnost mutacije	0.9
broj pravila	10
granica prihvaćanja pravila	0.9
veličina populacije	100

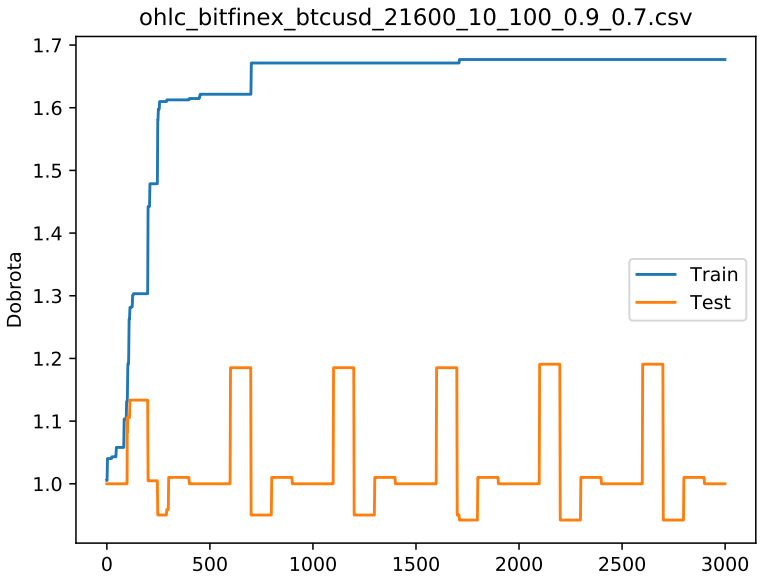


Slika 6.7: Boxplot za odabir parametara, indeksi objašnjeni na 6.9

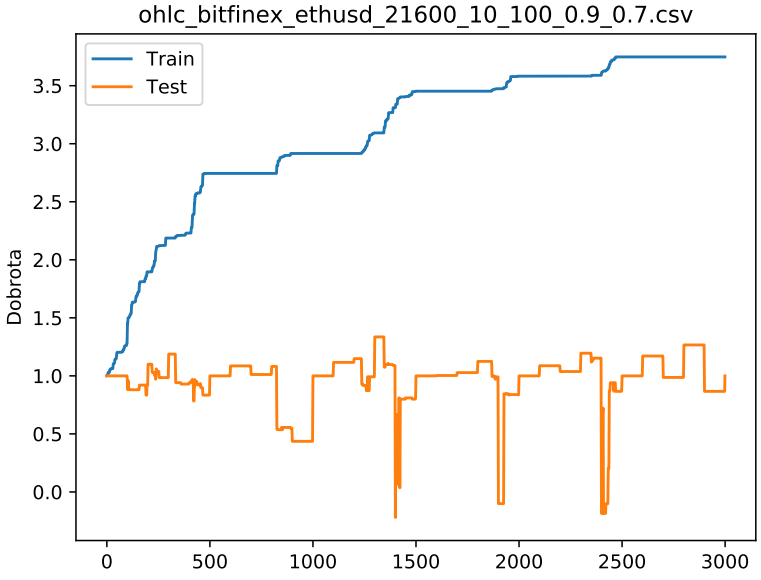
vima čiji su rezultati prikazani u nastavku.

Tablica 6.9: Objašnjenja indeksa pri odabiru parametara 6.7

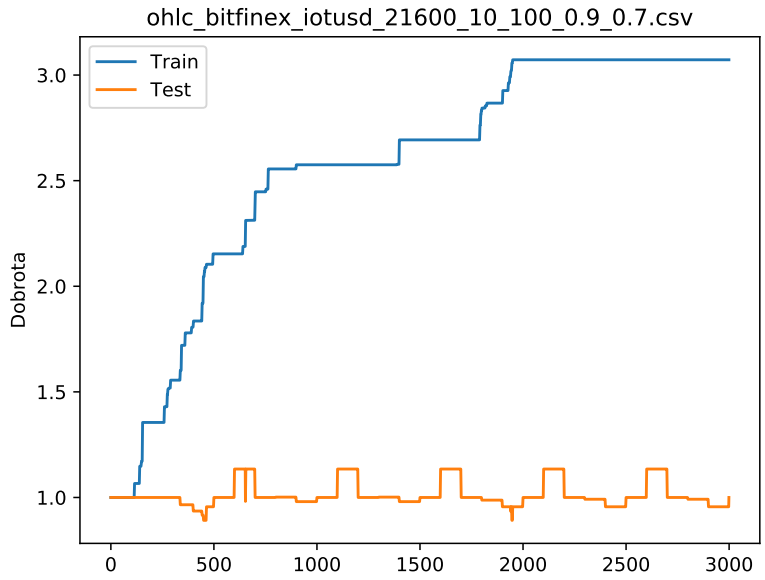
Indeks	Veličina populacije	Vjerojatnost mutacije	Granica prihvaćanja pravila
1	250	0.2	0.7
2	100	0.5	0.9
3	100	0.5	0.7
4	250	0.9	0.7
5	250	0.5	0.9
6	100	0.2	0.7
7	30	0.9	0.7
8	100	0.2	0.9
9	250	0.2	0.9
10	30	0.5	0.7
11	250	0.5	0.8
12	100	0.9	0.7
13	100	0.2	0.8
14	30	0.9	0.9
15	30	0.9	0.8
16	250	0.2	0.8
17	100	0.9	0.9
18	30	0.2	0.8
19	250	0.5	0.7
20	100	0.9	0.7
21	100	0.5	0.8
22	30	0.2	0.9
23	30	0.5	0.9
24	250	0.9	0.8
25	30	0.5	0.8
26	100	0.9	0.8
27	30	0.2	0.7
28	250	0.9	0.9



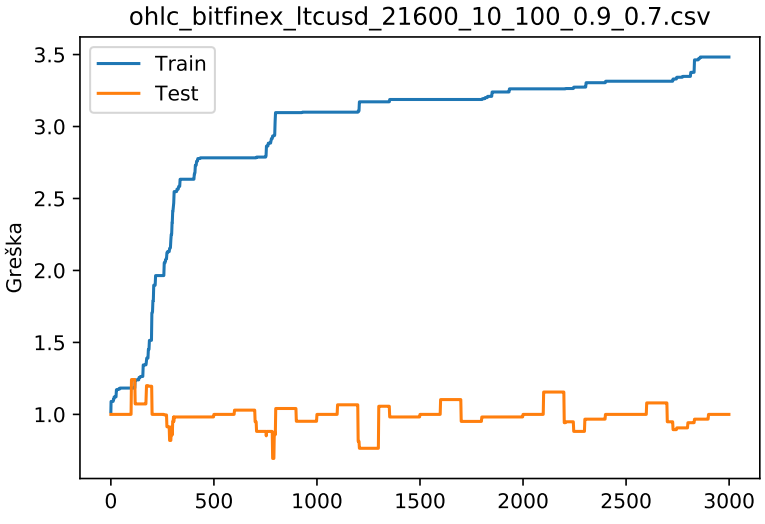
Slika 6.8: Skup BTCUSD 6h



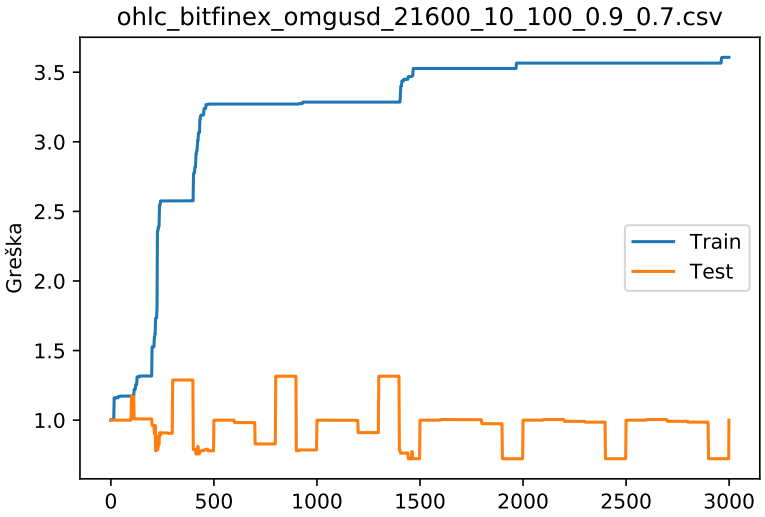
Slika 6.9: Skup ETHUSD 6h



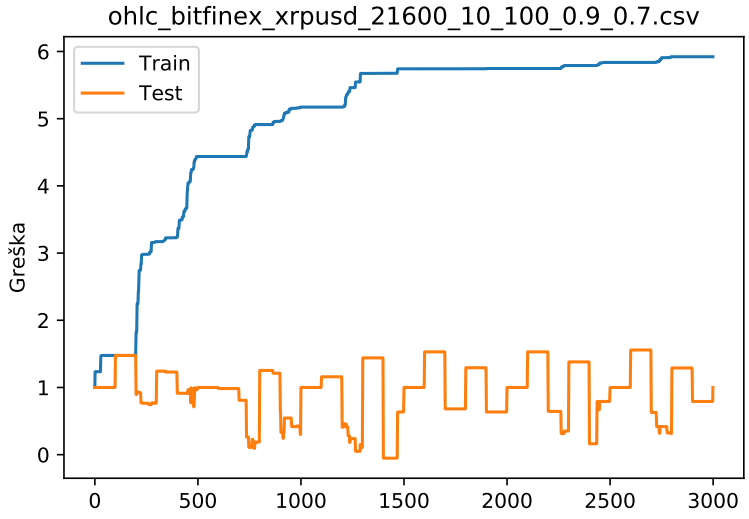
Slika 6.10: Skup IOTUSD 6h



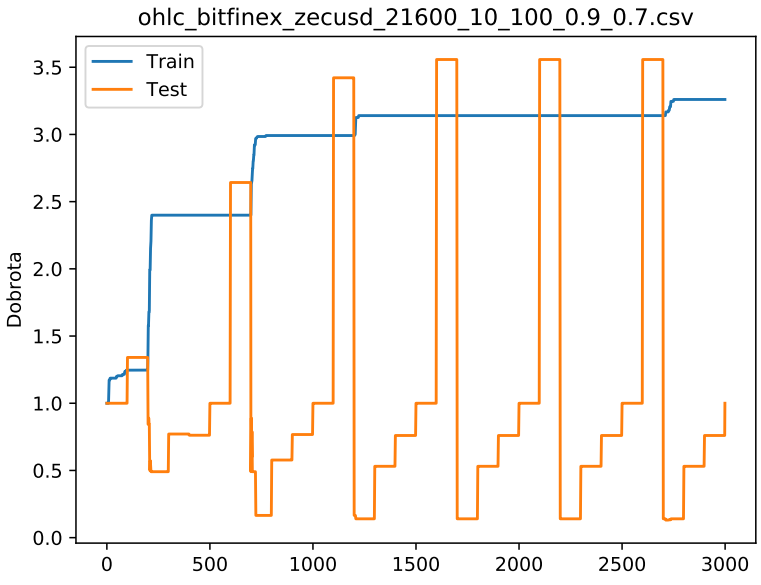
Slika 6.11: Skup LTCUSD 6h



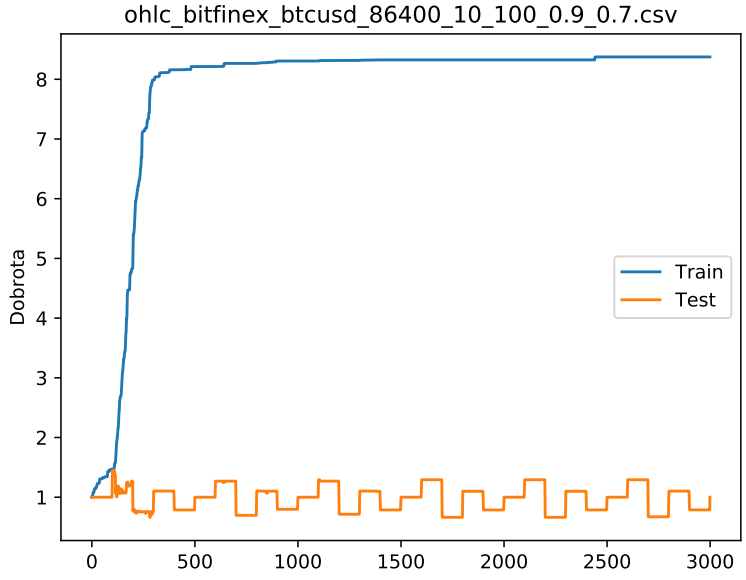
Slika 6.12: Skup OMGUSD 6h



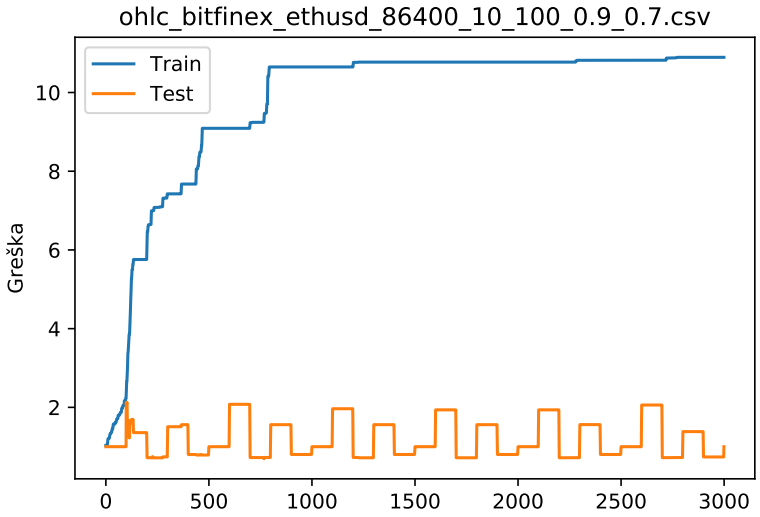
Slika 6.13: Skup XRPUSD 6h



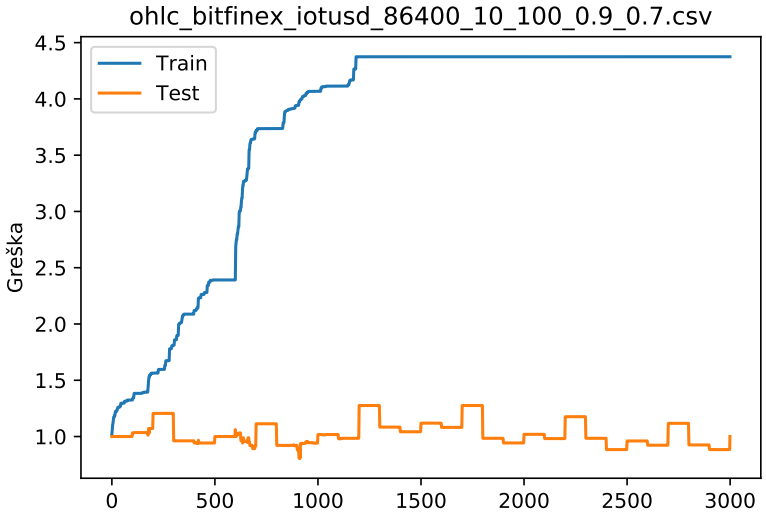
Slika 6.14: Skup ZECUSD 6h



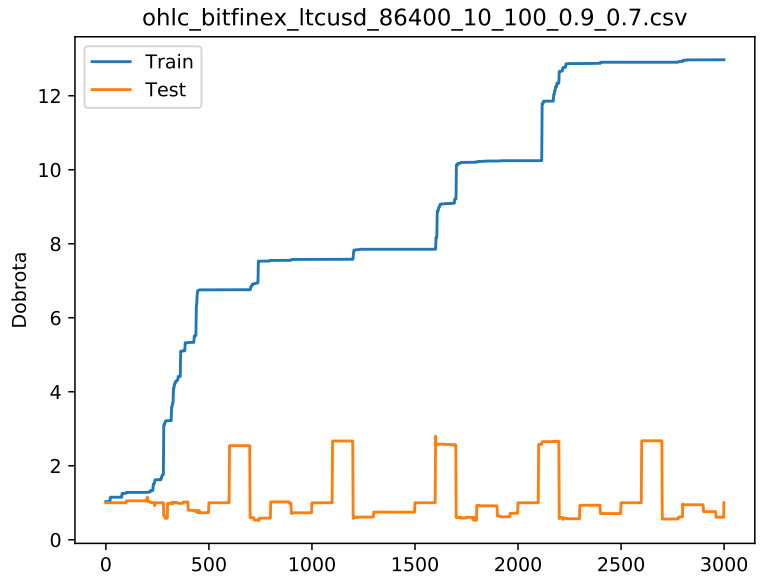
Slika 6.15: Skup BTCUSD 1d



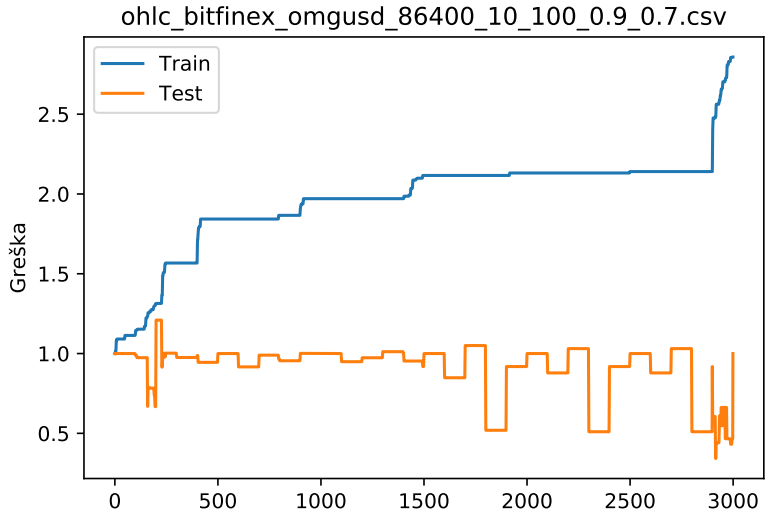
Slika 6.16: Skup ETHUSD 1d



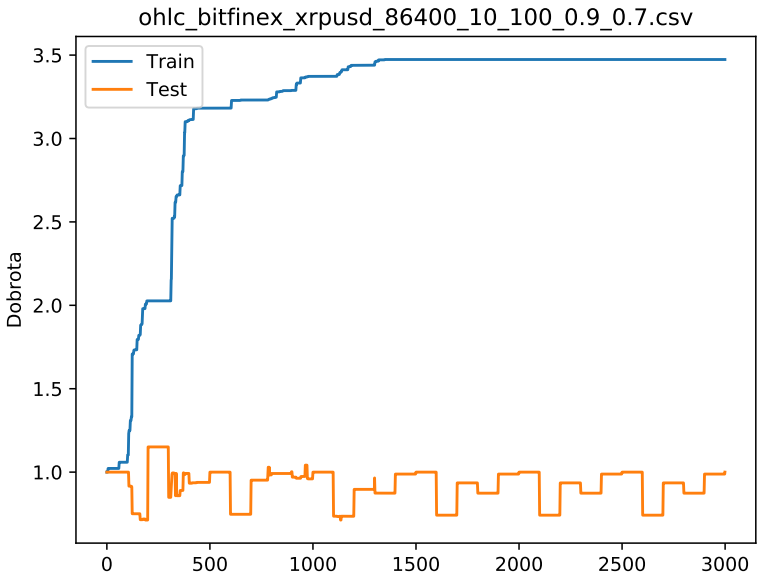
Slika 6.17: Skup IOTUSD 1d



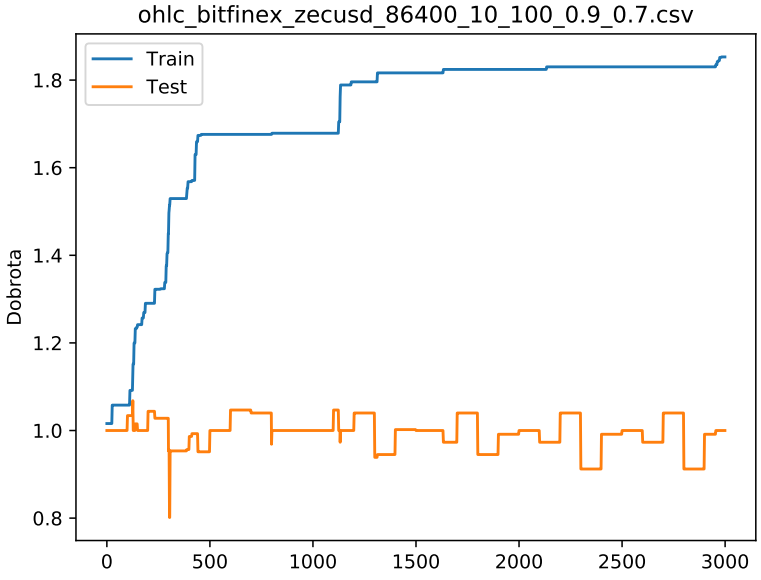
Slika 6.18: Skup LTCUSD 1d



Slika 6.19: Skup OMGUSD 1d



Slika 6.20: Skup XRPUSD 1d



Slika 6.21: Skup ZECUSD 1d

Iako se na skupovima za učenje, rješenja pokazuju dobrima, ispitni skupovi su druga priča. Ovisno o podskupu odabranom za ispitivanje, pojavljuju se različiti rezultati, ponekad i bolji od onih na skupu za učenje, ali uglavnom su jako blizu početne vrijednosti, pa i nešto lošiji. Vjerojatno je najveći uzrok tome pogrešno odabrano vrijeme za dohvat podataka, ali naravno i težina problema. U slučaju pojavljivanja sličnih uzoraka kao u skupovima za učenje u budućnosti, naučeni sustav zaključivanja bi doveo do nevjerojatnih rezultata. Na pripremljenim skupovima se pojavljivanje sličnih uzoraka nije dogodilo tako često pa su tako i rezultati očekivani.

7. Zaključak

U radu je pokazano da se korištenjem neizrazite logike mogu predviđati i vremenski slijedovi, ali i da je neizrazita logika primjenjiva i na probleme klasifikacije.

Primjena na klasifikacijske probleme je moguća u slučajevima problema s manje značajki, no dodavanjem sve više značajki, automatski raste prostor pretraživanja, i to značajno, čime se znatno otežava posao algoritma.

Za regresiju je isprobano više algoritama, i pokazano je da algoritam koji poboljšava težine dobivene nekim jednostavnijim algoritmom dolazi do nešto boljih rezultata. Algoritam koji koevolucijom spaja pojedinačna pravila u skupni sustav se nije uopće pokazao. Moguća poboljšanja tog algoritma su npr. uvođenje više pravila u svaku podpopulaciju i onda će spajanje najboljih rješenja iz podpopulacija dovesti do spajanja skupova pravila što bi trebalo dovesti do nešto boljih rezultata.

Ako uzmemo u obzir financijske vremenske slijedove, moguće je učiti neizrazita pravila i tako dobiti sustav koji je relativno sposoban samostalno trgovati. Zbog posebnosti korištenih skupova (nabavljeni su u vrijeme kada je tržište kriptovaluta bilo izrazito volatilno), ali i malog broja podataka, dobiveni sustavi nisu dovoljno stabilni za svakodnevno korištenje. Moguće poboljšanje algoritama je, uz korištenje većih skupova, i učenje u isto vrijeme nad više različitih skupova čime bi se razlike (koje su ponekad bile i drastične) između skupa za treniranje i ispitnog skupa smanjile. Razlog tome je što bi algoritam učenja vidio različite oblike skupova (rastuće, padajuće, stacionarne), i bio u mogućnosti naći neku zlatnu sredinu gdje dobiveni sustav može brinuti sam za sebe.

LITERATURA

[1] Am Abbasov i MH Mamedova. Application of fuzzy time series to population forecasting. 2003. URL http://papers.cumincad.org/data/works/att/50b1.content.pdf.

[2] David Aronson. *Evidence-Based Technical Analysis: Applying the Scientific Method and Statistical Inference to Trading Signals*. Wiley, 2006.

[3] Peter J. Brockwell i Davis Richard A. *Introduction to Time Series and Forecasting*. Springer, 2010.

[4] Ching-Hsue Cheng, Jing-Rong Chang, i Che-An Yeh. Entropy-based and trapezoid fuzzification-based fuzzy time series approaches for forecasting IT project cost. doi: 10.1016/j.techfore.2005.07.004. URL https://ac.els-cdn.com/S0040162505001125/1-s2.0-S0040162505001125-main.pdf?{_}tid=ee20de6b-7315-41b7-90ac-f4e81117a838{&}acdnat=1523209136{_}26432f6b747d67ab9c926b8c9f55bff3.

[5] Paulo Cortez, Miguel Rocha, i Jose Neves. Genetic and Evolutionary Algorithms for Time Series Forecasting. stranice 393–402. 2001. doi: 10.1007/3-540-45517-5_44. URL http://link.springer.com/10.1007/3-540-45517-5{_}44.

[6] Dua Dheeru i Efi Karra Taniskidou. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[7] Erol Egrioglu, Ufuk Yolcu, Cagdas Hakan Aladag, i Cem Kocak. An ARMA Type Fuzzy Time Series Forecasting Method Based on Particle Swarm Optimization. *Mathematical Problems in Engineering*, 2013:1–12, jul 2013. ISSN 1024-123X. doi: 10.1155/2013/935815. URL http://www.hindawi.com/journals/mpe/2013/935815/.

[8] Bindu Garg, M.M. Sufyan Beg, i A.Q. Ansari. Fuzzy time series model to forecast rice production. U *2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, stranice 1–8. IEEE, jul 2013. ISBN 978-1-4799-0022-0. doi: 10.1109/FUZZ-IEEE.2013.6622509. URL http://ieeexplore.ieee.org/document/6622509/.

[9] Leonid Hulianytskyi i Anna Pavlenko. DEVELOPMENT AND ANALYSIS OF GENETIC ALGORITHM FOR TIME SERIES FORECASTING PROBLEM. *International Journal*, 4(1), 2015. URL http://www.foibg.com/ijima/vol04/ijima04-01-p02.pdf.

[10] Sheng-Tun Li, Yi-Chung Cheng, i Su-Yu Lin. A FCM-based deterministic forecasting model for fuzzy time series. *Computers & Mathematics with Applications*, 56(12):3052–3063, dec 2008. ISSN 08981221. doi: 10.1016/j.camwa.2008.07.033. URL http://linkinghub.elsevier.com/retrieve/pii/S0898122108004409.

[11] Cheng-Jian Lin, Chun-Cheng Peng, i Chi-Yung Lee. Identification and Prediction Using Neuro-Fuzzy Networks with Symbiotic Adaptive Particle Swarm Optimization. *Informatica*, 35:113–122, 2011.

[12] Sibarama Panigrahi, Yasobanta Karali, i H S Behera. Time Series Forecasting using Evolutionary Neural Network. *International Journal of Computer Applications*, 75(10):975–8887, 2013. URL http://research.ijcaonline.org/volume75/number10/pxc3890553.pdf.

[13] Keyvan Rahmani-Monfared, Alireza Fathi, Ahmad Mozaffari, i Sayed Mahmood Rabiee. Application of self-learning evolutionary algorithm for optimal design of a porous pmma scaffold fabricated by laser drilling process. 227:211–224, 08 2013.

[14] Jens Rúni Poulsen. Fuzzy Time Series Forecasting -Developing a new forecasting model based on high order fuzzy time series. 2009. URL https://pdfs.semanticscholar.org/debc/b4b93f5bc796767d709f588242b6503e38f9.pdf.

[15] Sayan Saha, Timothy Breen, i Justin Zeth. Comparative Studies of Evolutionary Algorithms in Time-series Forecasting. URL http://homepages.rpi.edu/{~}sahas3/Course{_}Projects/Comp{_}Opt{_}Report.pdf.

[16] R.S. Tsay. *Analysis of Financial Time Series*. CourseSmart. Wiley, 2010. ISBN 9781118017098. URL https://books.google.hr/books?id=OKUGARAXKMwC.

[17] Marko Čupić, Dalbelo Bojana Bašić, i Marin Golub. Neizrazito, evolucijsko i neuroračunarstvo. 2013. URL http://java.zemris.fer.hr/nastava/nenr/knjiga-0.1.2013-08-12.pdf.

Neizrazita logika u analizi vremenskih slijedova

Sažetak

Uobičajeni algoritmi analize financijskih vremenskih slijedova uglavnom su temeljeni na pravilima koja signaliziraju optimalno vrijeme ulaska ili izlaska iz pozicije u određenom slijedu. Zbog toga je korišten neizraziti sustav zaključivanja koji također omogućuje definiranje pravila pa i učenje istih uz neki algoritam strojnog učenja, tj. genetski algoritam. Uz primjenu na financijske vremenske slijedove, ispitana je sposobnost učenja neizrazitih pravila i na problemima klasifikacije i regresije. Za svaki od navedenih problema potrebno je definirati poseban genotip i evaluacijsku funkciju, a u slučaju regresije i poseban način kooperativne koevolucije. Analizira se uspješnost navedenih algoritama u usporedbi s već postojećim algoritmima specijaliziranih za te probleme.

Ključne riječi: neizrazita logika, vremenski slijedovi, financijski vremenski slijedovi, strojno učenje, genetski algoritam, evolucijski algoritmi, koevolucija

Fuzzy Logic for the Analysis of Time Series

Abstract

The usual algorithms for the analysis of financial time series are mostly based on rules which signal the optimal time of entry or exit from a position in a specific financial series. Because of that, fuzzy inference system is used which allows for a similar definition of rules, and learning the same rules with a machine learning algorithm, genetic algorithm in this case. Beside the application on the financial time series, the ability of learning fuzzy rules for classification and regression is also tested. For each of the aforementioned problems, a specific genotype and evaluation function, or in the case of regression, even a special way of cooperative coevolution must be defined. The effectivenes of the algorithms is compared to the existing algorithms tailored for the problems.

Keywords: fuzzy logic, time series, financial time series, machine learning, genetic algorithm, evolutionary algorithms, coevolution