# Neizrazito, evolucijsko i neuroračunarstvo.

Marko Čupić Bojana Dalbelo Bašić Marin Golub

12. kolovoza 2013.

# Sadržaj

# Popis slika

# Popis tablica

# Predgovor

Ovaj dokument predstavlja radnu verziju knjige za kolegij Neizrazito, evolucijsko i neuroračunarstvo: *Neizrazito, evolucijsko i neuroračunarstvo*. Molimo sve pogreške, komentare, nejasnoće te sugestije dojaviti na Marko.Cupic@fer.hr, Bojana.Dalbelo@fer.hr ili Marin.Golub@fer.hr.



Verzija dokumenta: 0.1.2013-08-12.

# Dio I

# Neizrazita logika

# Poglavlje 1

# Neizraziti skupovi

Neizrazita logika, neizraziti skupovi, neizrazite relacije, neizraziti brojevi, neizrazita aritmetika, sustavi neizrazitog upravljanja... U ovom tekstu puno ćemo govoriti o *neizrazitosti* – idemo se stoga najprije upoznati s tim pojmom. Izraz *neizrazit* hrvatski je prijevod za englesku riječ *fuzzy*. Taj se pojam odnosi na jednu vrstu neodređenosti, i povezan je s još dvije vrlo slične engleske riječi: *vague* i *uncertainty*. Evo izvatka iz riječnika *Collins English Dictionary*.

*fuzzy*

1. of, resembling, or covered with fuzz;
2. indistinct; unclear or distorted;
3. not clearly thought out or expressed;
4. (of the hair) tightly curled or very wavy;
5. Maths. of or relating to a form of set theory in which set membership depends on a likelihood function: fuzzy set; fuzzy logic.

*vague*

1. (of statements, meaning, etc.) not explicit; imprecise: vague promises;
2. not clearly perceptible or discernible; indistinct: a vague idea; a vague shape;
3. not clearly or definitely established or known: a vague rumour;
4. (of a person or his expression) demonstrating lack of precision or clear thinking; absent-minded; [C16: via French from Latin vagus wandering, of obscure origin].

*uncertainty*

1. also called: uncertainness; the state or condition of being uncertain;

2. an uncertain matter, contingency, etc.

Povijesni razvoj neizrazite logike može se pratiti od dvadesetih godina dvadesetog kada je Jan Łukasiewicz, poljski logičar i filozof koji je proučavao analitičku filozofiju i matematičku logiku postavio temelje viševrijednosne logike – logike kod koje ne postoji samo istina i laž već postoje različite razine istinitosti propozicija. Porodice viševrijednosnih logika označavaju se s $\mathrm{L}_n$, pa tako imamo:

- $\mathrm{L}_2$: vrijednosti istinitosti mogu biti $\{0, 1\} \Rightarrow$ klasična logika,
- $\mathrm{L}_3$: vrijednosti istinitosti mogu biti $\{0, 1/2, 1\}$,
- ...
- $\mathrm{L}_n$: vrijednosti istinitosti mogu biti $\{0, 1/n-1, 2/n-1, \ldots., n-2/n-1, 1\}$,
- $\mathrm{L}_\infty$: vrijednosti istinitosti mogu biti iz intervala $[0, 1] \subset \mathbb{Q}$ te
- $L_1$: vrijednosti istinitosti mogu biti iz intervala $[0, 1] \subset \mathbb{R}$.

Izraz *vagueness* uveo je Bertrand Russell, engleski filozof, logičar, matematičar, povijesničar, i društveni kritičar, između ostaloga poznat i po Russellovom paradoksu koji pokazuje mane klasične logike. Evo o čemu se radi: prema klasičnoj teoriji skupova, svaka kolekcija koju je moguće definirati je skup. Neka je tada s $\mathcal{R}$ označen skup svih skupova koji nemaju samog sebe za člana. Evo paradoksa: ako bi $\mathcal{R}$ imao samog sebe za člana, tada bi to bilo u suprotnosti s definicijom tog skupa, pa je to nemoguće; s druge pak strane, ako taj skup ne sadrži samog sebe kao člana, onda je po definiciji element skupa $\mathcal{R}$ čime imamo kontradikciju. Ovaj problem može se iskazati i na konkretnom primjeru:

> U nekom selu postoji brijač koji brije isključivo one ljude koji se ne briju sami; tko brije brijača?

Ako se on sam ne brije, onda se po definiciji brije upravo kod sebe samoga, pa slijedi da on brije samog sebe. S druge pak strane, ako se sam brije, onda se ne brije kod sebe samoga, jer on brije samo one ljude koji se ne briju sami – i eto paradoksa. No Bertranda Russella spomenuli smo zbog pojma *vagueness*: evo paradoksa poznatog pod nazivom *Sorites' Paradox* koji ovo ilustrira.

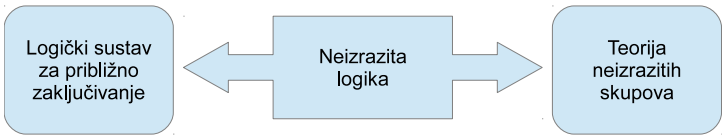
> Svi ćemo se složiti da 1,000,000 zrna pijeska čini gomilu. Ako uklonimo jedno zrno pijeska, i dalje imamo gomilu. Koliko zrna pijeska trebamo ukloniti da ono što nam ostane više ne bude gomila?

1937. godine Max Black, englesko-američki filozof koji je ostavio značajan trag na području analitičke filozofije objavio je članak: "Vagueness: An Exercise in Logical Analysis", Philosophy of Science, Vol.4, 1937., koji međutim u to vrijeme nije zadobio pažnju šire znanstvene javnosti. Čini se da svijet u to vrijeme još nije bio spreman za ove ideje. Tridesetak godina kasnije, 1965. poljski Lotfi A. Zadeh, iransko-ruski matematičar, inženjer elektrotehnike, znanstenik koji je proučavao područja računarstva i umjetne inteligencije objavio je članak: "Fuzzy sets", Information and Control, Vol.8, No.4, pp. 338-353, June 1965. i tim je člankom pokrenuo i popularizirao istraživanja iz područja neizrazite logike i neizrazitih skupova. Evo kako je Lotfi A. Zadeh opisao odnos klasične i neizrazite logike.

> Klasična logika je kao osoba koja dolazi na zabavu obučena u crno odijelo, bijelu uštirkanu košulju, sjajne cipele itd. Neizrazita logika je poput osobe koja je neformalno obučena, u traperice, majcu i tenesice. U prošlosti, takvo neformalno odijevanje bilo bi neprihvatljivo, danas je to obrnuto.

Izraz *neizrazita logika* danas se koristi za bilo koje od dva različita značenja, kako prikazuje slika 1.1. Sam Lotfi A. Zadeh ovo je 1994. godine opisao sljedećim riječima.

> Izraz neizrazita logika upotrebljava se u dva različita smisla. Uži smisao je logički sustav koji je proširenje viševrijednosne logike i služi kao logika približnog zaključivanja. U širem smislu to je sinonim za teoriju neizrazitih skupova, teoriju razreda s nejasnim granicama. Danas se izraz neizrazita logika pretežno upotrebljava u ovom širem smislu.




Slika 1.1: Značenja pojma *neizrazita logika*.

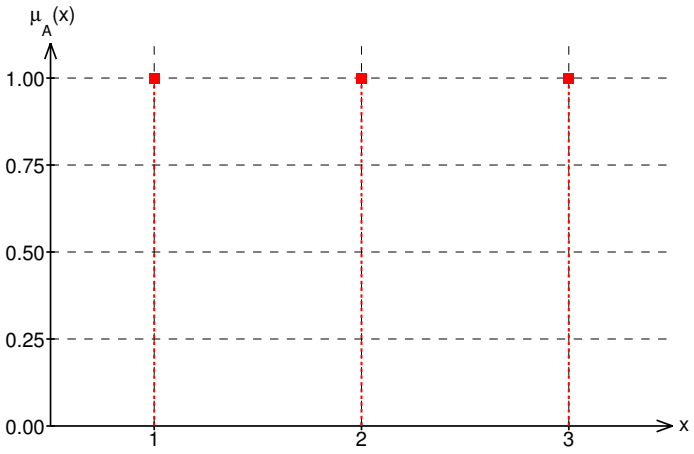
U nastavku ovog poglavlja osvrnut ćemo najprije na klasičnu teoriju skupova a potom definirati i pobliže obraditi pojam neizrazitog skupa i njegova svojstva.

$$A = \{1, 2, 3\}$$

(a) Navođenje elemenata skupa $A$

$$A = \{x | x \in U \text{ i } x > 0 \text{ i } x < 4\}$$

(b) Navođenje svojstava skupa $A$




(c) Karakteristična funkcija skupa $A$

Slika 1.2: Tri načina definiranja klasičnog skupa

## 1.1 Klasični skupovi

U klasičnoj teoriji skupova (engl. *crisp set theory*) neki element ili pripada skupu ili ne pripada skupu. Tako imamo primjerice skup cijelih brojeva, skup parnih cijelih brojeva, skup neparnih cijelih brojeva, skup prostih brojeva i sl. Na pitanje pripada li broj 3 u skup parnih cijelih brojeva postoje točno dva moguća odgovora: "da, pripada" ili "ne, ne pripada". Uzevši u obzir karakteristike brojeva koje su bitne za ovu klasifikaciju, a to je da su parni brojevi svi oni brojevi koji su djeljivi s "2", zaključujemo da je ispravan odgovor na postavljeno pitanje "ne, ne pripada", budući da 3 nije djeljiv s 2.

Općenito se klasični skupovi mogu zadavati na nekoliko načina (slika 1.2). Prvi korak je definirati univerzalni skup $U$[1], tj. domenu iz koje će se potom odabirati elementi koji će sačinjavati ostale skupove. Npr. neka je $U$ skup svih cijelih brojeva. U drugom koraku možemo definirati skupove koji nas zanimaju. Jedan je način nabrajanjem svih elemenata skupa. Primjerice, možemo reći da je skup $A$ definiran nad univerzalnim skupom $U$ kao skup brojeva $\{1, 2, 3\}$. Ovo je moguće ukoliko je broj elemenata koje skup sadrži konačan, a u praksi se dodaje i uvjet da taj broj treba biti relativno malen (jer teško da će netko moći nabrojati $10^{40}$ elemenata). Drugi način je definiranje uvjeta koje moraju zadovoljiti elementi da bi pripadali nekom skupu. Npr. skup $A$ možemo definirati na sljedeći način: to je skup cijelih brojeva koji su veći od 0 i manji od 4. Na ovaj način skup je jednoznačno

definiran, jer je jasno određena domena iz koje se biraju elementi (u primjeru: skup cijelih brojeva) te uvjet koji elementi moraju zadovoljiti da bi pripadali skupu (veći od 0 i manji od 4). Treći način za definiranje skupa je uporaba karakteristične funkcije $\mu_A : U \to \{0, 1\}$. Karakteristična funkcija $\mu_A$ definirana je na sljedeći način:

$$\mu_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases} .$$

Tako naš skup $A$ možemo definirati sljedećom karakterističnom funkcijom:

$$\mu_A(x) = \begin{cases} 1, & (x = 1) \vee (x = 2) \vee (x = 3) \\ 0, & (x \neq 1) \wedge (x \neq 2) \wedge (x \neq 3) \end{cases} .$$

Sva tri načina prikazana su na slici 1.2.

## 1.2 Neizraziti skupovi

Iako se obični skupovi u praksi koriste vrlo često, postoji mnoštvo primjera u kojima je vrlo teško utvrditi pripada li neki element skupu ili ne, odnosno primjera u kojima odluka nije binarna. Uzmimo za primjer univerzalni skup textitdani u 2001. godini i pokušajmo izgraditi skup *dani koji su bili sunčani*. Kako ćemo odrediti koji dani pripadaju u naš skup? Možemo pokušati ovako: u naš skup pripadaju svi oni dani kada na nebu nije bilo niti jednog oblaka. No što je s danima u kojima je na pet minuta u nekom kutku neba naišao neki mali oblačak? Pripadaju li oni u naš skup? A što je s danima u kojima se na deset minuta pojavio neki oblačak? Na petnaest minuta? Na dvadeset minuta?

Pogledajmo još jedan primjer. Zadan je univerzalni skup koji čine svi ljudi koji žive u gradu Zagrebu. Treba odrediti koji od tih ljudi pripadaju u *skup visokih ljudi*. Kako se ovo može napraviti? Kada bismo radili s klasičnim skupovima, morali bismo odrediti neku graničnu visinu, i sve ljude koji su visoki barem toliko uvrstiti u skup. Uzmimo za primjer da je granična visina 180 cm. Tada će u skup *visoki ljudi* ući svi oni koji su visoki 180 cm ili više. Međutim je li ovo dobar način za stvaranje skupa visokih ljudi? Pogledajmo jednu trivijalnu posljedicu ovakvog stvaranja skupa: Pero Perić koji je visok 180 cm pripada skupu *visoki ljudi* ali Ivica Ivić koji je visok 179.98 cm ne pripada. Hm, pa dobro, možemo pomaknuti granicu na 179.98 cm tako da i Ivica Ivić bude član visokih ljudi. Ali eto novog problema – Ante Antić koji je visok 179.97 cm ne pripada tom skupu iako nitko golim okom ne može

[1] Ispada da se dolazi do paradoksa "što je prije bilo: kokoš ili jaje". Naime, da bismo definirali neki skup, moramo najprije definirati univerzalni skup iz kojeg ćemo birati koje elemente želimo u našem skupu; no kako je i univerzalni skup opet skup, kako ćemo definirati njega?

uočiti da je on niži od Ivice Ivića koji je u skupu. Moramo li opet pomicati granicu?

Iz ova dva primjera jasno vidimo da se prilikom određivanja klasičnih skupova javlja jedan vrlo neugodan problem – kako odrediti što pripada takvom skupu, a da granica bude jasna, i što je još važnije, da ima veze sa zdravim razumom. Naime, kažemo li da je granica 180 cm, bili smo potpuno jasni, no vidjeli smo da to nikako nije dobar način, jer jednostavno nije smislen. Ovakve poteškoće jedan su od razloga nastanka teorije neizrazitih skupova, koju je razvio Lotfi A. Zadeh (prvi članak 1965. godine).

Vidjeli smo da se klasični skup može opisati karakterističnom funkcijom koja kao ulaz prima element koji ispitujemo, a kao izlaz daje vrijednost 0 ako element nije član skupa ili 1 ako element je član skupa; formalno, kažemo da je domena funkcije univerzalni skup a kodomena skup cijelih brojeva $\{0, 1\}$. Zadeh je ovaj koncept proširio na sljedeći način: neizraziti skup (engl. *fuzzy set*) opisivat će se *funkcijom pripadnosti* (engl. *membership function*) koja će kao ulaz primati element koji se ispituje, a kao izlaz davati realni broj u intervalu $[0, 1]$, čime će opisivati s kolikim stupnjem taj član pripada neizrazitom skupu. Osnovna je razlika između klasičnih i neizrazitih skupova u tome što kod klasičnog skupa element ili pripada ili ne pripada skupu, dok kod neizrazitog skupa element pripada skupu s određenim stupnjem, odnosno određenom mjerom. Iz ovoga se jasno vidi da su neizraziti skupovi proširenje klasičnih skupova, jer ako kodomenu funkcije pripadnosti neizrazitog skupa ograničimo isključivo na vrijednosti 0 i 1, dobiti ćemo upravo klasične skupove, odnosno funkcija pripadnosti degradirat će u karakterističnu funkciju.

**Definicija 1** (Neizrazit skup)**.** *Neka je $U$ univerzalni skup*[2]*. Neizraziti skup $A$ definiran nad univerzalnim skupom $U$ je skup uređenih parova $A = \{(x; \mu_A(x)) \mid x \in U,\ \mu_A(x) \in [0, 1]\}$, gdje je $\mu_A(x)$ funkcija pripadnosti (engl. membership function), i ona određuje stupanj pripadnosti elemenata $x \in U$ neizrazitom skupu $A$.*

I neizrazite skupove možemo definirati na nekoliko načina. Nakon što odaberemo univerzalni skup $U$, neizraziti skup $A$ možemo definirati na dva načina. Možemo koristiti nabrajanje elemenata, s time da uz svaki element obavezno pišemo i stupanj pripadnosti tog elementa (naravno, zbog jednostavnosti navodit ćemo samo elemente kojima je stupanj pripadnosti veći od nule, podrazumijevajući da je stupanj pripadnosti svih ostalih elemenata jednak nuli). Ovo se obično svodi na sljedeći zapis:

$$A = \left\{ \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \frac{\mu_A(x_3)}{x_3} + \ldots + \frac{\mu_A(x_n)}{x_n} \right\}$$

što treba pročitati na sljedeći način: element $x_1$ neizrazitom skupu $A$ pripada sa stupnjem pripadnosti $\mu_A(x_1)$, element $x_2$ neizrazitom skupu $A$ pripada

[2] Univerzalni skup je i dalje klasičan skup.

sa stupnjem pripadnosti $\mu_A(x_2)$, ..., element $x_n$ neizrazitom skupu $A$ pripada sa stupnjem pripadnosti $\mu_A(x_n)$. Uočite da se znak "+" ne odnosi na nikakvo zbrajanje već se jednostavno koristi kao sredstvo za nabrajanje elementa skupa $A$. Treba uočiti da kod nabrajanja elemenata ne moramo imati eksplicitno definiranu funkciju $\mu_A(x)$ jer možemo direktno napisati stupanj pripadnosti svakog elementa[3]. Drugi način za definiranje neizrazitih skupova je upravo kroz definiciju funkcije pripadnosti $\mu_A(x)$. U tom slučaju na raspolaganju nam stoje dvije vrste zapisa neizrazitog skupa. Ukoliko je neizraziti skup definiran nad diskretnim univerzalnim skupom, koristi se zapis sa sumom, koji se krati u oblik:

$$A = \sum_{x \in U} \frac{\mu_A(x)}{x}$$

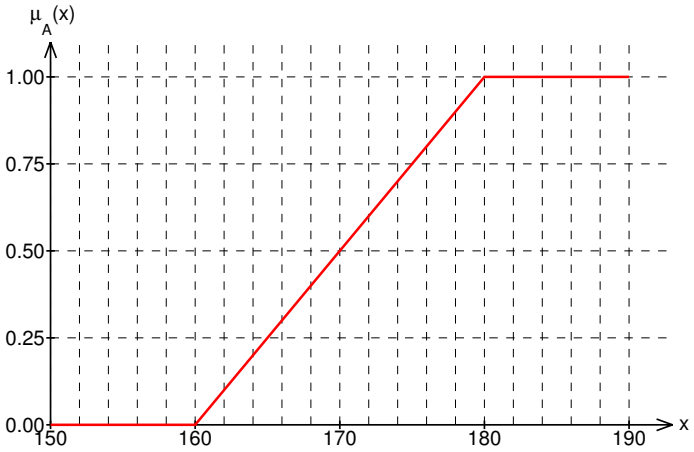
pri čemu znak sumacije treba shvatiti kao znak nabrajanja elemenata, a ne aritmetičke operacije zbrajanja. U skladu s time, ukoliko je univerzalni skup kontinuiran, tada znak sumacije jednostavno prelazi u znak integracije:

$$A = \int_{x \in U} \frac{\mu_A(x)}{dx}$$

opet uz opasku da znak integracije ne predstavlja operaciju integracije, već jednostavno nabrajanje.

Vratimo se definiciji skupa *visokih ljudi*. Kao univerzalni skup uzet ćemo skup pozitivnih realnih brojeva koji predstavljaju moguće visine ljudi. Kada smo skup *visokih ljudi* pokušavali definirati kao klasični skup, uzeli smo granicu od 180 cm i složili se s tvrdnjom da su svi ljudi visoki barem 180 cm visoki ljudi. Kod neizrazitih skupova za taj slučaj možemo reći da svi ljudi koji su barem visoki 180 cm pripadaju skupu *visoki ljudi* sa stupnjem pripadnosti 1. Problem kod klasičnog skupa bili su ljudi koji su niži od 180 cm, ali su blizu te visine. Za ljude koji su niži od 160 cm zasigurno nećemo reći da su visoki. Kod neizrazitih skupova za taj slučaj možemo reći da svi ljudi koji su niži od 160 cm pripadaju skupu *visoki ljudi* sa stupnjem pripadnosti 0, odnosno da mu ne pripadaju. Sada možemo iskoristiti punu snagu neizrazitih skupova i definirati da ljudi koji su visoki između 160 cm i 180 cm skupu visoki ljudi pripadaju sa stupnjem pripadnosti koji je veći od 0, i manji od 1, i to biliži vrijednosti 1 što je visina bliža visini od 180 cm. Na koji ćemo način odlučiti kako raspodijeliti stupnjeve pripadnosti, potpuno ovisi o našim željama. Možemo npr. linearno razvući stupanj pripadnosti od vrijednosti 0 za visinu 160 cm do vrijednosti 1 za visinu 180 cm. Tako bi osoba visoka 170 cm u skup visokih ljudi pripadala sa stupnjem pripadnosti 0.5, osoba visoka 175 cm sa stupnjem pripadnosti 0.75, osoba visoka 179.98 cm sa stupnjem pripadnosti 0.999 i sl. Ovakvu definiciju skupa visoki ljudi prikazuje slika 1.3.

[3]Zapravo, ako znamo stupanj pripadnosti svakog elementa, tehnički govoreći znamo i funkciju pripadnosti, iako je nigdje ne definiramo.




Slika 1.3: Funkcija pripadnosti skupa *visoki ljudi*, po dijelovima linearna.

Ovakav način definiranja skupa *visoki ljudi* elegantno je eliminirao problem stroge granice i svega što to povlači za sobom. No sada se javlja niz drugih pitanja. Npr. zašto smo odabrali baš brojeve 160 cm i 180 cm? Zašto smo uzeli takvu funkciju pripadnosti koja linearno raste od 0 za 160 cm do 1 za 180 cm? Jesmo li mogli ovo definirati nekako drugačije? Krenimo redom s odgovorima.

Još kod klasičnih skupova morali smo na neki način odrediti parametre unutar kojih ćemo definirati skup. Tamo je to bio jedan jedini parametar: granična visina. No ipak, već tamo smo mogli doći do spoznaje koja se ovdje samo potvrđuje: MI definiramo skup visoki ljudi onako kako to želimo, i u tome imamo potpunu slobodu. Hoćemo li uzeti graničnu visinu od 180 cm, ili od 170 cm, ili neku treću, ovisi isključivo o nama i našoj procjeni što bi bilo u datoj situaciji prihvatljivo. Prelaskom na neizrazite skupove jedino što se promijenilo je broj stupnjeva slobode koje smo dobili. Sada umjesto jednog parametra (granična visina) trebamo odrediti nekoliko parametara: gornja granična visina, donja granična visina, način raspodjele stupnja pripadnosti elemenata između tih granica i sl. A ako govorimo o nekom općem slučaju, tada uopće ne moramo imati ove parametre, već neke sasvim druge, jer je cilj odrediti funkciju pripadnosti, ma kakvog ona oblika bila. Sve ovo ovisi isključivo o nama; imamo potpunu slobodu da skup modeliramo onako kako mislimo da bi to bilo prikladno.

Ovdje se uslijed silne slobode koju smo dobili javlja i jedan možebitni problem. Pretpostavimo da smo mi izgradili sustav $A$ koji pojam visoki ljudi shvaća na način kako smo mi to zamislili; tim iz Japana izgradi svoj sustav $B$ koji pojam visoki ljudi shvaća na način kako su oni to zamislili (što će vrlo vjerojatno biti nešto drugačije od našeg shvaćanja, budući da

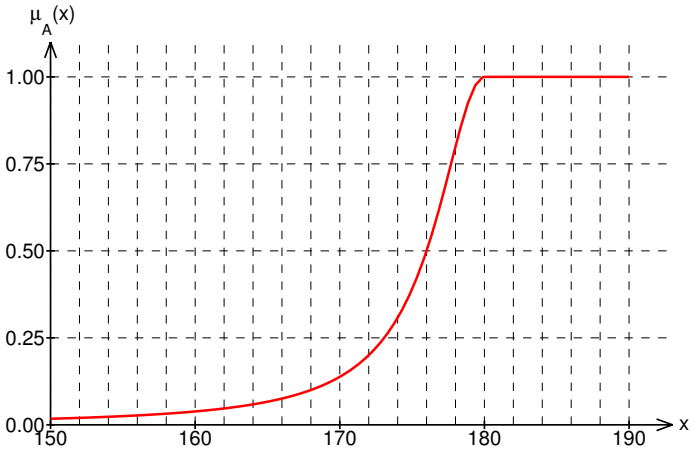
su Japanci inače niži rastom). Kako će ovi sustavi komunicirati? Malom analizom dolazimo do ekvivalentnog problema koji će pokazati da zapravo nema problema: sretnu se osoba iz Hrvatske i osoba iz Japana i svako od njih priča o visokim ljudima; Hrvat na hrvatskom, Japanac na japanskom; kako će oni komunicirati kada jedan drugog ne razumiju? Promatramo li osobu iz Hrvatske kao sustav $A$, sustav je napravljen tako da ga razumiju svi oni koji žive u njegovoj okolini i koji dijele zajedničko shvaćanje potrebnih koncepata; kod osobe iz Hrvatske to je govor hrvatskim jezikom. Isto vrijedi i za osobu iz Japana koja govori Japanski jer upravo u svojoj sredini to je jedini način da sustav obavlja svoj zadatak. Problem komunikacije rješava se na sasvim drugom nivou, npr. standardizacijom, ili pak učenjem. Npr. obje osobe mogu naučiti engleski, i sporazumijevati se engleskim jezikom, ili pak jedna osoba može naučiti jezik one druge, i koristi ga u komunikaciji, iako cijelo vrijeme ta ista osoba misli i radi na način na koji ona sama inherentno razumije. Ovo je upravo razlog zašto se ovakvim stvarima ne treba opterećivati. Kasnije ćemo još na puno mjesta dobiti određene stupnjeve slobode u nekim odabirima, i na svakom tom mjestu može se postaviti pitanje na koje smo ovdje pokušali odgovoriti. Ono što je bitno jest da smo mi dizajneri našeg sustava, i mi znamo kako želimo da taj sustav radi i na koji način treba shvaćati pojedine koncepte. Kada jednom dođe do potrebe za razmjenom informacija između više sustava, tada se obavezno uvodi neki od dobro definiranih, standardiziranih komunikacijskih protokola kako bi svim sudionicima u toj izmjeni informacija bilo jasno o čemu je riječ.

Vratimo se sada definiranju neizrazitih skupova. Stali smo na problemu kako odrediti funkciju pripadnosti koja nam najbolje odgovara. Problem možemo riješiti i bez određivanja donje granice. Npr. kao karakterističnu funkciju možemo uzeti:

$$\mu_A(x) = \begin{cases} 1, & x \geq 180, \\ \frac{1}{1+\left(\frac{x-180}{4}\right)^2}, & 0 \leq x < 180. \end{cases}$$

Tada ćemo dobiti skup visokih ljudi čiju funkciju pripadnosti prikazuje slika 1.4. Na ovaj način svi ljudi pripadaju skupu visokih ljudi, samo što se mijenja stupanj pripadnosti (koji je uvijek veći od nule).

Kako se problem odabira odgovarajuće funkcije pripadnosti javlja pri definiranju svakog neizrazitog skupa, postoji nekoliko uobičajenih funkcija koje se često koriste, i koje ćemo pogledati u nastavku. Pri tome još jednom treba naglasiti da je odabir funkcije pripadnosti u potpunosti subjektivan, i kontekstno ovisan, ili kako je to Zadeh rekao: "nisu važne točne vrijednosti funkcije pripadnosti u određenom kontekstu, već su važni međusobni odnosi funkcije pripadnosti neizrazitih skupova jednih prema drugima".




Slika 1.4: Funkcija pripadnosti skupa *visoki ljudi*, na drugačiji način.

## 1.3 Karakteristični oblici funkcija pripadnosti

U primjeni neizrazite logike u početnoj fazi obično definiramo osnovne neizrazite skupove s kojima ćemo dalje raditi. Skupove definiramo tako da modeliraju temeljne koncepte našeg sustava. Ovisno o tim konceptima, za modeliranje možemo koristiti različite oblike neizrazitih skupova.

### 1.3.1 Po dijelovima linearne funkcije

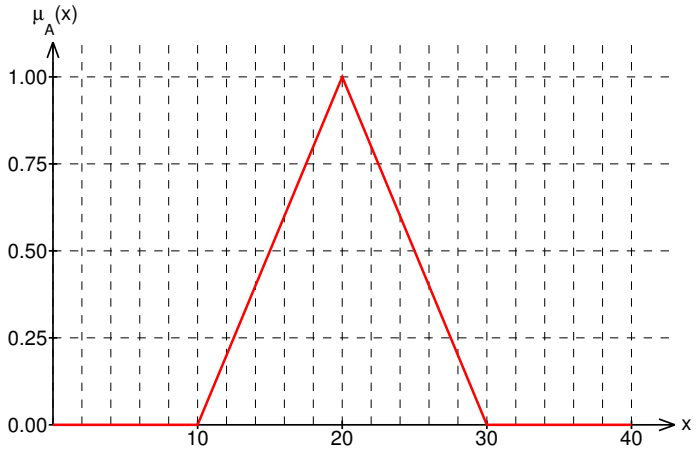
#### Λ-funkcija

Želimo li definirati koncepte poput "brojevi oko 5", "brzina oko 60 km/h" i sl., trebat ćemo neizraziti skup čija funkcija pripadnosti poprima vrijednost 1 upravo na centralnom elementu ("broj 5", "brzina 60 km/h" i sl.) a udaljavanjem od tog elementa opada (dakle za manje i veće brojeve od broja 5, za manje i veće brzine od 60 km/h, ...). Dobar kandidat za takvu funkciju pripadnosti jest Λ-funkcija (lambda).

Λ-funkcija prikazana je na slici 1.5 i definirana je s tri parametra:

$$\Lambda(x;\alpha,\beta,\gamma) = \begin{cases} 0, & x < \alpha, \\ \frac{x-\alpha}{\beta-\alpha}, & \alpha \leq x < \beta, \\ \frac{\gamma-x}{\gamma-\beta}, & \beta \leq x < \gamma, \\ 0, & x \geq \gamma. \end{cases} \tag{1.1}$$

Slika 1.5 prikazuje Λ-funkciju uz parametre: $\alpha = 10$, $\beta = 20$ te $\gamma = 30$.




Slika 1.5: Funkcije pripadnosti: Λ-funkcija.

### Γ-funkcija

Za koncepte poput "visoka temperatura", "visoki ljudi", "broj zrna koji je dovoljan da čini gomilu", "broj pušaka koji čini dobro naoružanu vojsku" i sl. Λ-funkcija više nije prikladna. U ovom slučaju trebamo funkciju pripadnosti koja ima vrijednost nula sve do nekog graničnog elementa a zatim počinje rasti, i konačno, za neki element poprima vrijednost 1, i tu vrijednost ima i za sve elemente koji ga slijede. Za ovakvu funkciju dobar je kandidat Γ-funkcija (gama).

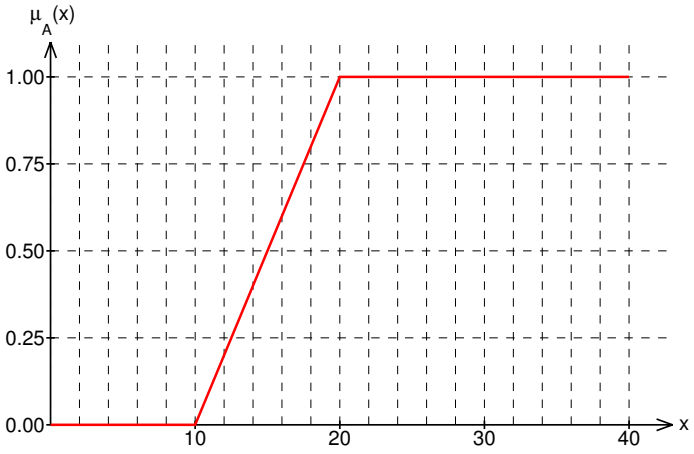
Γ-funkcija prikazana je na slici 1.6 i definirana je s dva parametra:

$$\Gamma(x;\alpha,\beta) = \begin{cases} 0, & x < \alpha, \\ \frac{x-\alpha}{\beta-\alpha}, & \alpha \leq x < \beta, \\ 1, & x \geq \beta. \end{cases} \tag{1.2}$$

Slika 1.6 prikazuje Γ-funkciju uz parametre: $\alpha = 10$ i $\beta = 20$.

### *L*-funkcija

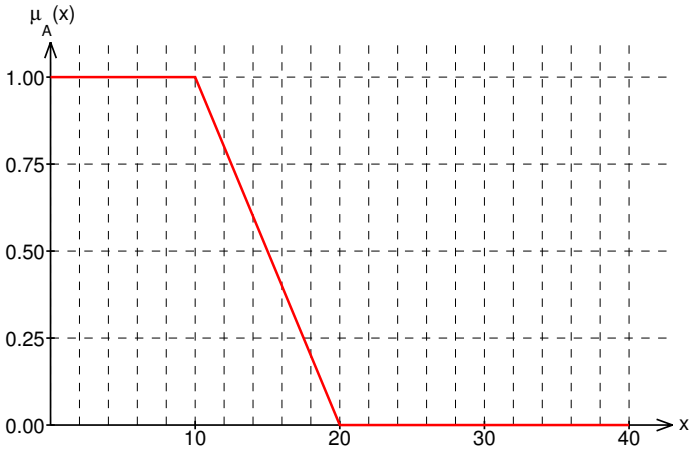
Želimo li definirati upravo suprotnu vrstu koncepata od prethodno navedenih, primjerice "niska temperatura", "niski ljudi", "broj zrna koji ne čini gomilu", "broj pušaka koji čini slabo naoružanu vojsku", "niska brzina okretanja kotača" i sl., trebat ćemo funkciju koja se ponaša suprotno od Γ-funkcije – dakle, funkciju pripadnosti koja za sve "male" elemente ima vrijednost 1, zatim počinje padati, i konačno od nekog elementa na dalje poprima vrijednost nulu. Dobar kandidat koji zadovoljava ove zahtjeve jest *L*-funkcija.




Slika 1.6: Funkcije pripadnosti: Γ-funkcija.

$L$-funkcija prikazana je na slici 1.7 i definirana je s dva parametra:

$$L(x;\alpha,\beta) = \begin{cases} 1, & x < \alpha, \\ \frac{\beta - x}{\beta - \alpha}, & \alpha \leq x < \beta, \\ 0, & x \geq \beta. \end{cases} \tag{1.3}$$




Slika 1.7: Funkcije pripadnosti: $L$-funkcija.

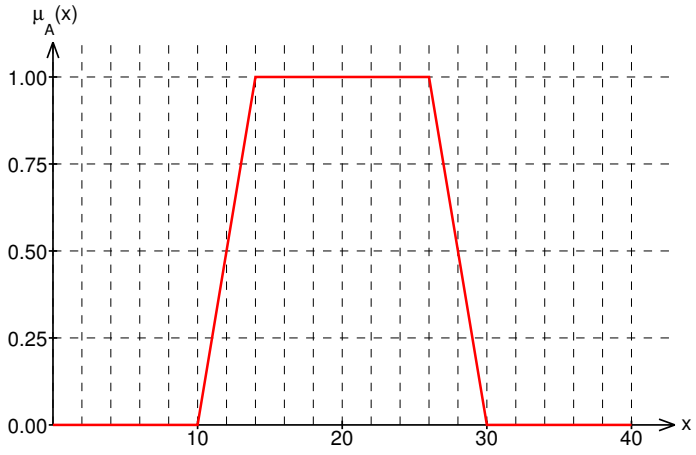
Slika 1.7 prikazuje $L$-funkciju uz parametre: $\alpha = 10$ i $\beta = 20$.

### Π-funkcija

Konačno, ne smijemo zaboraviti na još jedan tip koncepata: "broj zrna koji čini srednje veliku gomilu", "broj pušaka koji čini srednje naoružanu vojsku", "broj kapi kiše koji dovoljno navlaži tlo" i slični koncepti. Zajedničko ovoj vrsti koncepata je sličnost s konceptima koje opisujemo Λ-funkcijom – međutim uz jednu bitnu razliku. Kod ovih koncepata ne možemo pronaći jedan centralni element koji najbolje opisuje koncept, već postoji čitav interval elemenata koji dobro opisuju koncept. To znači da funkcija pripadnosti neće poprimiti vrijednost 1 za samo jedan element, već će funkcija poprimati vrijednost 1 na određenom centralnom intervalu, a tek će na krajevima tog intervala postupno padati prema nuli. Dobar predstavnik ovog tipa funkcije pripadnosti jest Π-funkcija (pi).

Π-funkcija prikazana je na slici 1.8 i definirana je s četiri parametra:

$$L(x;\alpha,\beta,\gamma,\delta) = \begin{cases} 0, & x < \alpha, \\ \frac{x-\alpha}{\beta-\alpha}, & \alpha \le x < \beta, \\ 1, & \beta \le x < \gamma, \\ \frac{\delta-x}{\delta-\gamma}, & \gamma \le x < \delta, \\ 0, & x \ge \delta. \end{cases} \tag{1.4}$$




Slika 1.8: Funkcije pripadnosti: Π-funkcija.

Slika 1.8 prikazuje Π-funkciju uz parametre: $\alpha = 10$, $\beta = 14$, $\gamma = 26$ te $\delta = 30$.

Sve prethodno navedene funkcije pripadaju klasi po dijelovima linearnim funkcijama. Osnovna prednost ovakvih definicija je u lakoći i brzini izračunavanja. Naime, ne trebamo nikakav složeni matematički aparat kao niti jake računske sklopove za realizaciju takvih funkcija. Naravno, osim dobrih svojstava ove funkcije imaju i neka svojstva ih čine manje pogodnim za

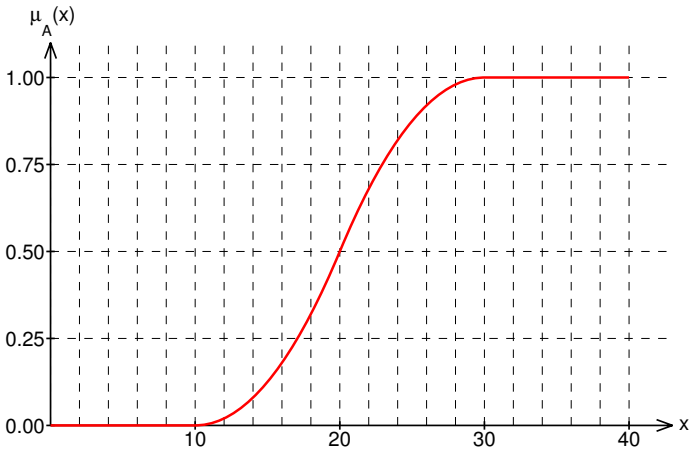
određene implementacije. Npr. svojstvo derivabilnosti u svim točkama ove funkcije ne zadovoljavaju. Upravo na "prijelomnim" točkama derivacija nije jednoznačna. Zbog toga je razvijena nova klasa funkcija pripadnosti, tzv. glatke funkcije, koje se također definiraju po dijelovima, ali u svakoj točci jesu derivabilne. Sve funkcije koje smo opisali u ovom poglavlju imaju svoj ekvivalent u klasi glatkih funkcija, i koriste se za opisivanje istih koncepata kao i po dijelovima linearne funkcije. Zbog toga ćemo u nastavku samo dati kratak pregled glatkih funkcija bez dodatnih objašnjenja, jer za njih vrijedi sve što je već rečeno i kod po dijelovima linearnih funkcija.

### 1.3.2 Glatke funkcije

#### *S*-funkcija

*S*-funkcija prikazana je na slici 1.9 i definirana je s tri parametra:

$$S(x;\alpha,\beta,\gamma) = \begin{cases} 0, & x < \alpha, \\ 2 \cdot \left(\frac{x-\alpha}{\gamma-\alpha}\right)^2, & \alpha \leq x < \beta, \\ 1 - 2 \cdot \left(\frac{x-\gamma}{\gamma-\alpha}\right)^2, & \beta \leq x < \gamma, \\ 0, & x \geq \gamma. \end{cases} \tag{1.5}$$




Slika 1.9: Funkcije pripadnosti: *S*-funkcija.

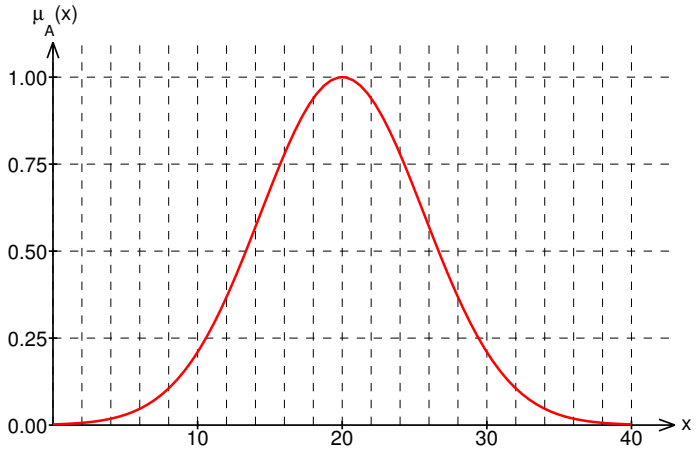
Parametar $\alpha$ određuje vrijednost do koje je iznos funkcija pripadnosti jednak 0 a parametar $\gamma$ određuje vrijednost od koje je iznos funkcija pripadnosti jednak 1. Parametar $\beta$ treba biti postavljen na vrijednost $\frac{\alpha+\gamma}{2}$ – to je točka infleksije funkcije pripadnosti.

Slika 1.9 prikazuje *S*-funkciju uz parametre: $\alpha = 10$, $\beta = 20$ te $\gamma = 30$.

### *Gaussova* funkcija pripadnosti

*Gaussova* funkcija prikazana je na slici 1.10 i definirana je s dva parametra:

$$Gauss(x;\mu,\sigma) = e^{-\left(\frac{x-\mu}{\sigma}\right)^2}. \tag{1.6}$$




Slika 1.10: Funkcije pripadnosti: *Gaussova* funkcija.

Slika 1.10 prikazuje *Gaussovu* funkciju uz parametre: $\mu = 20$ te $\sigma = 8$.

### *Sigmoidalna* funkcija

*Sigmoidalna* funkcija funkcija prikazana je na slici 1.11 i definirana je s dva parametra:

$$Sigm(x;a,c) = \frac{1}{1+e^{-a(x-c)}}. \tag{1.7}$$

Slika 1.11 prikazuje *sigmoidalnu* funkciju uz parametre: $a = 0.4$ te $b = 20$.

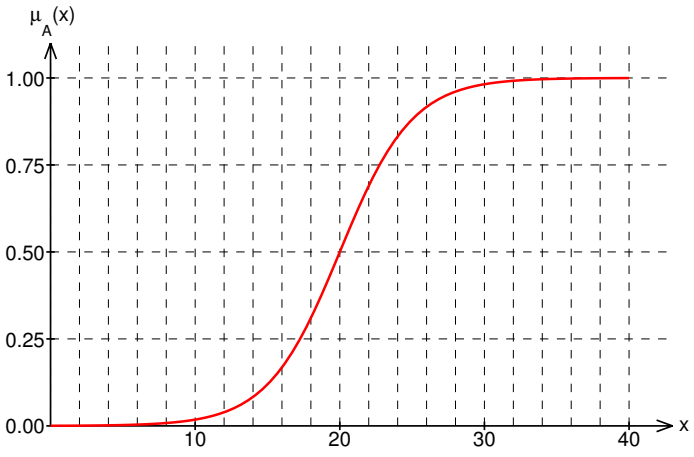
### $\pi$-funkcija

$\pi$-funkcija prikazana je na slici 1.12 i definirana je s dva parametra:

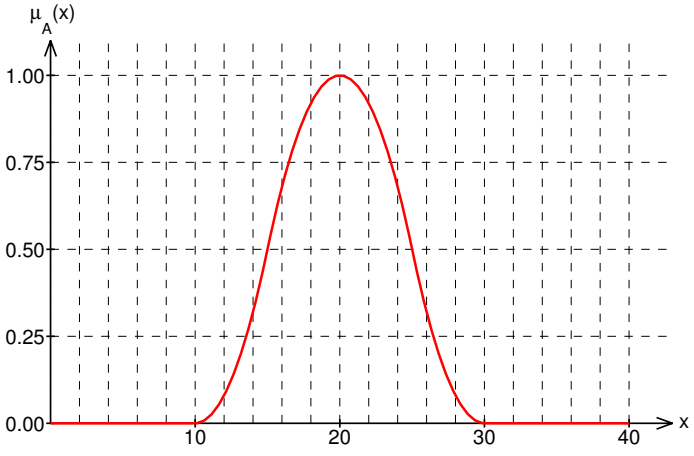
$$\pi(x;\beta,\gamma) = \begin{cases} S(x;\gamma-\beta,\gamma-\beta/2,\gamma), & x < \gamma, \\ 1-S(x;\gamma,\gamma+\beta/2,\gamma+\beta), & x \geq \gamma. \end{cases} \tag{1.8}$$

Parametar $\gamma$ kod ove krivulje predstavlja točku oko koje je krivulja ("zvono") centrirano. Parametar $\beta$ određuje širinu zvona. Funkcija za sve vrijednosti manje od $\gamma - \beta$ te veće od $\gamma + \beta$ poprima vrijednost 0.

Slika 1.12 prikazuje $\pi$-funkciju uz parametre: $\beta = 10$ te $\gamma = 20$.




Slika 1.11: Funkcije pripadnosti: *Sigmoidalna* funkcija.

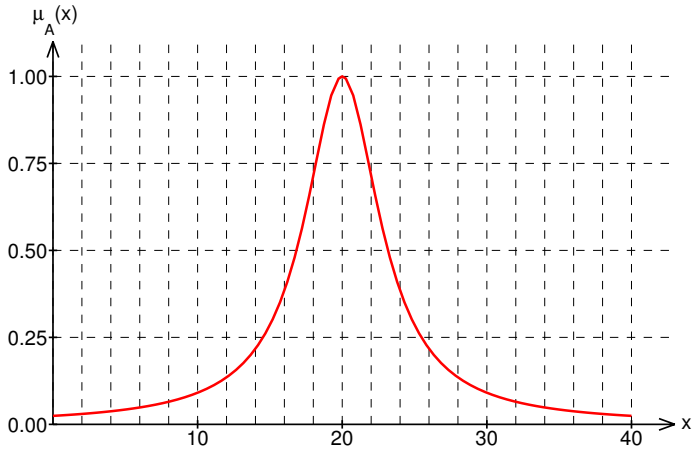



Slika 1.12: Funkcije pripadnosti: $\pi$-funkcija.

***Kvadratna zvonolika* funkcija**

*Kvadratna zvonolika* funkcija (u literaturi se može naći pod nazivom *exponential-like membership function*) prikazana je na slici 1.13 i definirana je s dva parametra:

$$Zvon(x;\mu,k) = \frac{1}{1+k\cdot(x-\mu)^2}. \tag{1.9}$$




Slika 1.13: Funkcije pripadnosti: *Kvadratna zvonolika* funkcija.

Parametar $\mu$ kod ove krivulje predstavlja točku oko koje je krivulja ("zvono") centrirano. Parametar $k$ određuje širinu zvona. Funkcija nigdje ne poprima vrijednost 0.

Slika 1.13 prikazuje *kvadratnu zvonoliku* funkciju uz parametre: $\mu = 20$ te $k = 0.1$.

## 1.4 Svojstva neizrazitih skupova

U nastavku ćemo se upoznati s osnovnim svojstvima neizrazitih skupova.

**Definicija 2** (Jednakost neizrazitih skupova)**.** *Neka su $A$ i $B$ dva neizrazita skupa definirana nad univerzalnim skupom $U$, i neka je $\mu_A, \mu_B : U \to [0,1]$. Neizraziti skupovi $A$ i $B$ su jednaki, $A = B$, ako je $\mu_A(x) = \mu_B(x)$, $\forall x \in U$.*

**Definicija 3** (Podskup neizrazitog skupa)**.** *Neka su $A$ i $B$ dva neizrazita skupa definirana nad univerzalnim skupom $U$, i neka je $\mu_A, \mu_B : U \to [0,1]$. Neizraziti skup $A$ je podskup od neizrazitog skupa $B$, tj. $A \subseteq B$, ako je $\mu_A(x) \leq \mu_B(x)$, $\forall x \in U$.*

**Definicija 4** (Jezgra neizrazitog skupa)**.** *Jezgra neizrazitog skupa $A$ (engl.* core*) je podskup univerzalnog skupa (dakle klasični skup) sa svojstvom $\mu_A(x) = 1$, tj. $core(A) = \{x \in U \mid \mu_A(x) = 1\}$.*

**Definicija 5** (Potpora neizrazitog skupa). *Potpora neizrazitog skupa $A$ (engl.* support*) je podskup univerzalnog skupa (dakle klasični skup) sa svojstvom $\mu_A(x) > 0$, tj. $support(A) = \{x \in U \mid \mu_A(x) > 0\}$.*

**Definicija 6** (Visina neizrazitog skupa). *Visina neizrazitog skupa $A$ je maksimalna vrijednost funkcije pripadnosti $\mu_A(x)$, tj. $hgt(A) = \max_{x \in U} \mu_A(x)$.*

**Definicija 7** (Univerzalni neizraziti skup). *Univerzalni neizraziti skup $X$ je skup sa svojstvom $\mu_X(x) = 1$, $\forall x \in U$.*

**Definicija 8** (Prazan skup). *Prazan skup $\emptyset$ je skup sa svojstvom $\mu_\emptyset(x) = 0$, $\forall x \in U$.*

Ako je $hgt(A) = 1$, tada se kaže da je neizraziti skup $A$ *normalan.*

**Definicija 9** (Jednoelementni neizraziti skup). *Jednoelementni neizraziti skup $S$ definiran nad univerzalnim skupom $U$ je neizraziti skup čija funkcija pripadnosti poprima vrijednost različitu od nule za samo jedan element iz univerzalnog skupa. Drugim riječima, to je neizraziti skup čija je potpora skup kardinaliteta 1.*

---

**Primjer:** 1

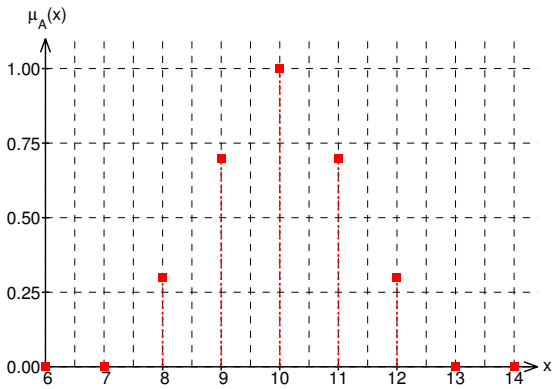
Treba definirati neizraziti skup "brojevi oko 10" i potom odrediti jezgru, potporu i visinu tog skupa. Je li to normalan skup?

**Rješenje:**

Odaberimo za univerzalni skup skup cijelih brojeva. Skup "brojevi oko 10" možemo definirati proizvoljno; odabrat ćemo sljedeću definiciju:

$$A = \textit{brojevi oko } 10 = \frac{0.3}{8} + \frac{0.7}{9} + \frac{1}{10} + \frac{0.7}{11} + \frac{0.3}{12}.$$

Slika u nastavku prikazuje funkciju pripadnosti ovako definiranog skupa.



Jezgru ovog skupa čine svi elementi za koje je stupanj pripadnosti jednak 1 (definicija 4). To je u ovom slučaju samo broj 10, pa je jezgra ovog skupa skup $core(A) = \{10\}$. Uočite da je ovo pravi skup, a ne neizraziti skup!

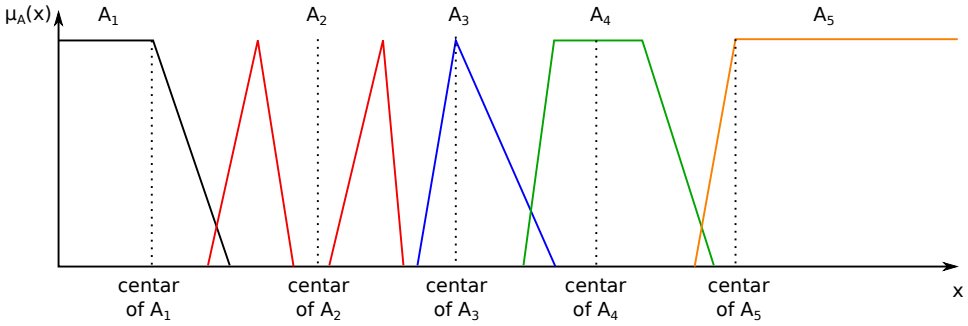
Potporu ovog skupa čine svi elementi za koje je stupanj pripadnosti veći od 0 (definicija 5). To su u našem slučaju brojevi 8, 9, 10, 11 i 12 pa je potpora ovog skupa skup $supp(A) = \{8, 9, 10, 11, 12\}$. Uočite da je ovo također pravi skup, a ne neizraziti skup!

Visina ovog skupa, prema definiciji 6, je 1, jer je maksimum funkcije pripadnosti upravo 1 (postiže se za $x = 10$). Skup je normalan jer mu je visina jednaka 1.

---

**Definicija 10** (Centar neizrazitog skupa)**.** *Ako je srednja vrijednost svih točaka u kojima funkcija pripadnosti neizrazitog skupa $A$ poprima maksimalnu vrijednost konačna, tada je ta vrijednost centar. Ako je ta srednja vrijednost jednaka pozitivnoj (negativnoj) beskonačnosti, tada se kao centar uzima najmanja (najveća) točka u kojoj funkcija pripadnosti poprima maksimalnu vrijednost.*

Ilustracija centara različitih neizrazitih skupova prikazana je na slici 1.14. Uz pretpostavku da je domena nad kojom su skupovi definirani od $-\infty$ do $+\infty$, neizraziti skupovi $A_1$ i $A_5$ nemaju konačnu srednju vrijednost elemenata za koje im je funkcija pripadnosti jednaka visini (u prikazanom slučaju visina je 1). Stoga je centar od $A_1$ najveći element $x$ za koji je $\mu_{A_1}(x) = hgt(A_1)$ dok je centar od $A_5$ najmanji element $x$ za koji je $\mu_{A_5}(x) = hgt(A_5)$.




Slika 1.14: Primjeri centara neizrazitih skupova.

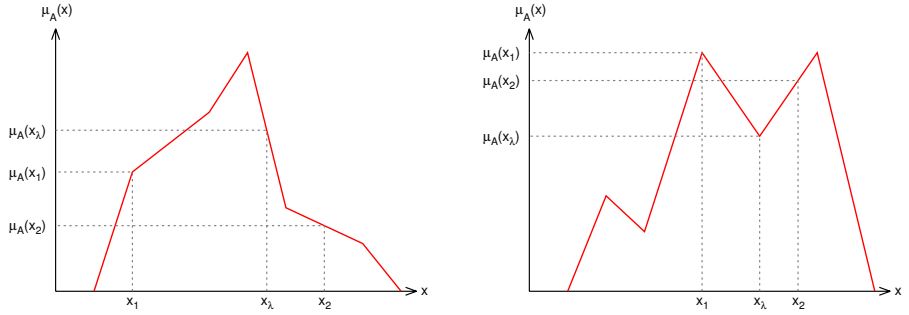
Centar od $A_3$ je očekivan – postoji samo jedan element za koji je funkcija pripadnosti maksimalna i taj element je ujedno i centar. Kako neizraziti skup $A_4$ ima više elemenata za koje funkcija pripadnosti poprima maksimalnu vrijednost, centar je njihova aritmetička sredina. Međutim, moguće je da centar neizrazitog skupa bude i element koji ne pripada neizrazitom skupu! Taj je primjer upravo ilustriran neizrazitim skupom $A_2$ čija funkcija pripadnosti najprije raste do 1 pa pada na nulu, pa je neko vrijeme nula i onda opet raste na 1 i pada na nulu. Postoje dva elementa za koje funkcija pripadnosti poprima maksimalnu vrijednost: njihova aritmetička sredina nalazi se točno između gdje su elementi koji ne pripadaju neizrazitom skupu. Ovih

problema neće biti ako je neizraziti skup neizrazito konveksan – upoznajmo se stoga i s tim pojmom.

**Definicija 11** (Neizrazito konveksan skup)**.** *Neizraziti skup $A$ definiran nad univerzalnim skupom $U$ je neizrazito konveksan ako vrijedi:*

$$\mu_A(\lambda \cdot x_1 + (1 - \lambda \cdot x_2)) \geq \min(\mu_A(x_1), \mu_A(x_2)).$$

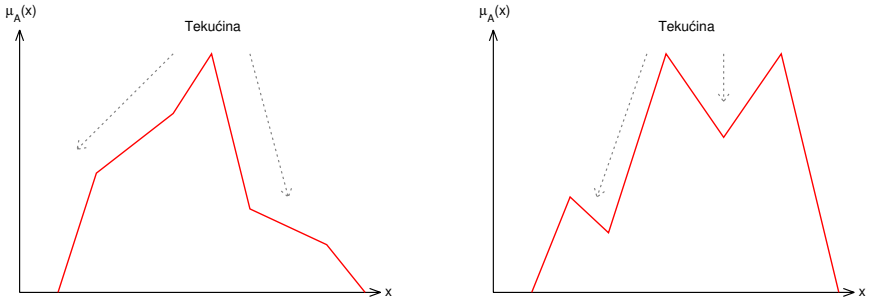
Prethodna definicija nam zapravo kaže da je neizraziti skup neizrazito konveksan ukoliko je, nakon što fiksiramo dvije točke ($x_1$ i $x_2$), stupanj pripadnosti svake točke ($x_\lambda$) između te dvije fiksirane točke veći od manjeg od stupnjeva pripadnosti točaka $x_1$ i $x_2$. Slika 1.15 jasno ilustrira ovo pravilo. Treba uočiti da neizrazita konveksnost nije isto što i konveksnost poligona u geometrijskom smislu i to ne treba miješati.




(a) Neizrazito konveksan skup (b) Skup koji nije neizrazito konveksan

Slika 1.15: Pojam neizrazito konveksnog skupa.

Postoji jednostavan način kako se pogledom na grafički prikaz funkcije pripadnosti može utvrditi je li skup neizrazito konveksan ili nije: princip je prikazan na slici 1.16.




(a) Neizrazito konveksan skup (b) Skup koji nije neizrazito konveksan

Slika 1.16: Ocjena neizrazite konveksnosti skupa.

Zamislimo da smo na dijagram koji prikazuje funkciju pripadnosti skup izlili kantu vode. Neizraziti skup je neizrazito konveksan ukoliko ne zadržava vodu, odnosno ako voda može iscuriti u potpunosti. Neizraziti skup nije neizrazito konveksan ukoliko postoje žljebovi u kojima bi se voda zadržavala.

**Definicija 12** (Kardinalni i relativni kardinalni broj). *Neka je $A$ konačan neizraziti skup definiran nad univerzalnim skupom $U$. Kardinalni broj skupa $A$ je: $|A| = \sum_{x \in U} \mu_A(x)$. Relativni kardinalni broj skupa $A$ je $\|A\| = \frac{|A|}{|U|}$.*

---

**Primjer:** 2

Dajte neku definiciju neizrazitog skupa "brojevi oko 5" definiranog nad univerzalnim skupom $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Odredite njegov kardinalni i relativni kardinalni broj. Koliki bi bio njegov relativni kardinalni broj kada bismo za univerzalni skup uzeli skup cijelih brojeva?

**Rješenje:**

Neizraziti skup "brojevi oko 5" možemo definirati kako nam god to odgovara. Npr. definirajmo taj skup ovako:

$$BO5 = \frac{0.1}{3} + \frac{0.6}{4} + \frac{1}{5} + \frac{0.6}{6} + \frac{0.1}{7}.$$

Kardinalni broj skupa $BO5$ iznosi $|BO5| = 0.1 + 0.6 + 1.0 + 0.6 + 0.1 = 2.4$ (suma stupnjeva pripadnosti svih elemenata skupa $BO5$, definicija 12). Za relativni kardinalni broj od $BO5$ treba nam još i kardinalni broj od univerzalnog skupa: $|U| = 1+1+1+1+1+1+1+1+1+1 = 10$ (suma stupnjeva pripadnosti svih elemenata univerzalnom skupa, definicija 12; prema definiciji, svaki element univerzalnom skupu pripada sa stupnjem pripadnosti 1, definicija 7). Relativni kardinalni broj neizrazitog skupa $BO5$ je tada $\|BO5\| = |BO5|/|U| = 2.4/10 = 0.24$ (definicija 12). Ukoliko bismo za univerzalni skup uzeli skup cijelih brojeva, tada bi relativni kardinalni broj skupa $BO5$ iznosio 0, budući da je skup cijelih brojeva beskonačan, pa mu je kardinalni broj $\infty$.

---

## 1.5 Teorem predstavljanja. $\alpha$-presjeci.

U prethodnim poglavljima podsjetili smo se teorije klasičnih skupova i uveli smo jednu potpuno novu teoriju – teoriju neizrazitih skupova. Međutim, već smo rekli da su klasični skupovi i neizraziti skupovi čvrsto povezani. Prvi argument ove tvrdnje bio je da su neizraziti skupovi zapravo proširenje klasičnih skupova. Sada ćemo se upoznati s još jednom vezom koju iznosi teorem predstavljanja.

**Definicija 13** (Produkt $\alpha A$). *Neka je $\alpha \in [0, 1]$ i neka je $A$ običan skup. Produkt $\alpha A$ je neizraziti skup u kojem svaki element ima stupanj pripadnosti jednak $\alpha$.*

**Definicija 14** (Teorem predstavljanja). *Svaki se neizraziti skup $A$ može prikazati kao $A = \sum_\alpha \alpha A_\alpha$ ili $A = \int_{[0,1]} \alpha A_\alpha$.*

**Definicija 15** ($\alpha$-presjek). *$\alpha$-presjek neizrazitog skupa $A$ je klasični skup $A_\alpha$ koji sadrži sve one elemente iz $A$ koji neizrazitom skupu $A$ pripadaju barem sa stupnjem pripadnosti $\alpha$.*

Teorem predstavljanja često se koristi za proširivanje klasičnih matematičkih koncepata na neizrazite. Ovaj nam teorem zapravo govori da svaki neizraziti skup možemo razložiti na niz klasičnih skupova, pri čemu za svaki klasični skup pamtimo još i njegov stupanj pripadnosti. Ovo će nam vrlo jasno demonstrirati sljedeći primjer.

---

**Primjer:** 3

Zadan je neizraziti skup "brojevi oko 5, ili malo veći":

$$A = \frac{0.1}{3} + \frac{0.7}{4} + \frac{1}{5} + \frac{0.9}{6} + \frac{0.7}{7} + \frac{0.3}{8} + \frac{0.1}{9}$$

nad univerzalnim skupom $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Pronađite $\alpha$-presjeke $A_0$, $A_{0.1}$, $A_{0.3}$, $A_{0.7}$, $A_{0.9}$, $A_{1.0}$ i pokažite kako se iz tih $\alpha$-presjeka može ponovno rekonstruirati neizraziti skup $A$.

**Rješenje:**

Definicija 15 govori nam kako izračunati $\alpha$-presjeke. Za $\alpha = 0$ dobivamo $\alpha$-presjek $A_0$ (tj. 0-presjek). To je klasični skup koji sadrži sve one elemente koje neizraziti skup $A$ sadrži sa stupnjem pripadnosti od barem 0 ($\alpha = 0$). To je očito cijeli univerzalni skup. Dakle $A_0 = U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

Za $\alpha = 0.1$ dobivamo $\alpha$-presjek $A_{0.1}$ (tj. 0.1-presjek). To je klasični skup koji sadrži sve one elemente koje neizraziti skup $A$ sadrži sa stupnjem pripadnosti od barem 0.1 ($\alpha = 0.1$). To je skup $A_{0.1} = \{3, 4, 5, 6, 7, 8, 9\}$. Analogno se računaju i ostali $\alpha$-presjeci. Imamo redom: $A_{0.3} = \{4, 5, 6, 7, 8\}$, $A_{0.7} = \{4, 5, 6, 7\}$, $A_{0.9} = \{5, 6\}$ te $A_{1.0} = \{5\}$.

Uočite odmah da vrijedi $A_\phi = A_{0.1}$ $\forall \phi \in (0, 0.1]$, $A_\phi = A_{0.3}$ $\forall \phi \in (0.1, 0.3]$, $A_\phi = A_{0.7}$ $\forall \phi \in (0.3, 0.7]$, $A_\phi = A_{0.9}$ $\forall \phi \in (0.7, 0.9]$ te $A_\phi = A_{1.0}$ $\forall \phi \in (0.9, 1.0]$. Ovo je posljedica toga što je neizraziti skup u našem primjeru diskretan i funkcija pripadnosti ne poprima sve vrijednosti iz intervala $[0, 1]$ već samo neke. Iz ovoga slijedi da je u našem slučaju dovoljno izračunati samo one $\alpha$-presjek za vrijednosti $\alpha$ koje funkcija pripadnosti doista poprima, jer će se ostali $\alpha$-presjeci poklapati s već izračunatima. Budući da naša funkcija pripadnosti poprima samo vrijednosti iz skupa $\{0, 0.1, 0.3, 0.7, 0.9, 1.0\}$, dovoljno je izračunati $\alpha$-presjeke $A_0$, $A_{0.1}$, $A_{0.3}$, $A_{0.7}$, $A_{0.9}$, $A_{1.0}$. Skup $A$ iz $\alpha$-presjeka možemo rekonstruirati na način kako to govore definicija 14 i definicija 13. Definicija 13 nam daje naputak kako iz jednog $\alpha$-presjeka dobiti odgovarajući neizraziti skup:

$$\begin{aligned}
0A_0 &= \frac{0}{1} + \frac{0}{2} + \frac{0}{3} + \frac{0}{4} + \frac{0}{5} + \frac{0}{6} + \frac{0}{7} + \frac{0}{8} + \frac{0}{9} + \frac{0}{10} \\
0.1A_{0.1} &= \frac{0.1}{3} + \frac{0.1}{4} + \frac{0.1}{5} + \frac{0.1}{6} + \frac{0.1}{7} + \frac{0.1}{8} + \frac{0.1}{9} \\
0.3A_{0.3} &= \frac{0.3}{4} + \frac{0.3}{5} + \frac{0.3}{6} + \frac{0.3}{7} + \frac{0.3}{8} \\
0.7A_{0.7} &= \frac{0.7}{4} + \frac{0.7}{5} + \frac{0.7}{6} + \frac{0.7}{7} \\
0.9A_{0.9} &= \frac{0.9}{5} + \frac{0.9}{6} \\
1.0A_{1.0} &= \frac{1.0}{5}
\end{aligned}$$

Definicija 14 sada nam govori kako iz neizrazitih skupova dobivenih iz $\alpha$-presjeka rekonstruirati polazni neizraziti skup. Potrebno je jednostavno uzeti sve one elemente koji su sadržani u nekom od $\alpha$-presjeka, i to s onim stupnjem pripadnosti koji je najveći (ukoliko se isti element nalazi u više $\alpha$-presjeka s različitim stupnjevima pripadnosti). Npr. element 5 nalazi se u $0A_0$ sa stupnjem pripadnosti 0, u $0.1A_{0.1}$ sa stupnjem pripadnosti 0.1, u $0.3A_{0.3}$ sa stupnjem pripadnosti 0.3, u $0.7A_{0.7}$ sa stupnjem pripadnosti 0.7, u $0.9A_{0.9}$ sa stupnjem pripadnosti 0.9 te u $1.0A_{1.0}$ sa

stupnjem pripadnosti 1.0. U neizrazitom skupu $A$ element 5 će se nalaziti sa stupnjem pripadnosti koji je najveći od nabrojanih, a to je 1.0. Analognim razmatranjem za sve elemente dobije se:

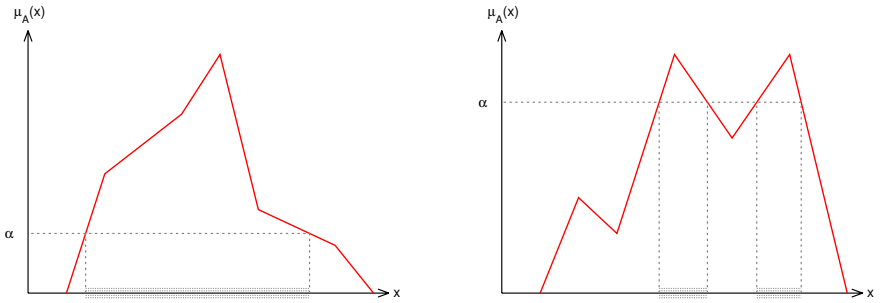
$$A = \sum_{\alpha} \alpha A_\alpha = \frac{0.1}{3} + \frac{0.7}{4} + \frac{1}{5} + \frac{0.9}{6} + \frac{0.7}{7} + \frac{0.3}{8} + \frac{0.1}{9}.$$

---

$\alpha$-presjeci mogu nam poslužiti i za još jedan način definiranja neizrazite konveksnosti skupa. Naime, vrijedi sljedeća definicija.

**Definicija 16** (Neizrazita konveksnost skupa)**.** *Neka je $A$ neizraziti skup definiran nad univerzalnim skupom $U$. Neizraziti skup $A$ je neizrazito konveksan* akko *su svi njegovi $\alpha$-presjeci konveksni.*

Ovo jasno demonstrira slika 1.17. Na slici 1.17a prikazan je neizrazito konveksan skup i jedan njegov $\alpha$-presjek; sa slike se vidi da je taj $\alpha$-presjek konveksan (povezan), i to svojstvo vrijedi za svaki $\alpha$-presjek tog skupa. Na slici 1.17b prikazan je neizraziti skup koji nije neizrazito konveksan. Za taj skup postoje $\alpha$-presjeci koji nisu konveksni (povezani); tako je na slici istaknut jedan $\alpha$-presjek koji je jedan uniji dvaju istaknutih konveksnih skupova, i on sam nije konveksan.




(a) Neizrazito konveksan skup – konveksan $\alpha$ presjek

(b) Skup koji nije neizrazito konveksan – $\alpha$ presjek nije konveksan

Slika 1.17: Neizrazita konveksnost preko $\alpha$-presjeka.

# Poglavlje 2

# Operacije nad neizrazitim skupovima

Klasična teorija skupova poznaje čitav niz operacija koje je moguće provoditi nad skupovima – unije, presjeci i komplementi samo su neke od njih. U ovom poglavlju pogledat ćemo kako se te operacije definiraju nad neuizrazitim skupovima.

## 2.1 Standardne operacije

U nastavku slijede definicije operacija unije, presjeka i komplementa za neizrazite skupove.

**Definicija 17** (Unija neizrazitih skupova)**.** *Neka su $A$ i $B$ neizraziti skupovi definirani nad univerzalnim skupom $U$. Unija skupova $A$ i $B$ je neizraziti skup $A \cup B$, sa svojstvom: $\forall x \in U\ \mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x))$, odnosno*

$$A \cup B = \{(x; \mu_{A\cup B}(x)) \mid x \in U, \mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x))\}.$$

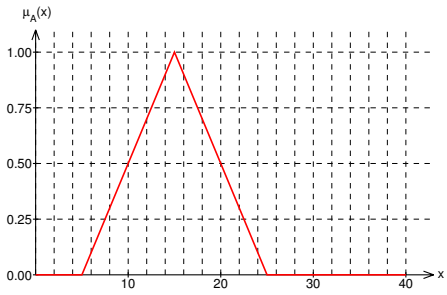
**Definicija 18** (Presjek neizrazitih skupova)**.** *Neka su $A$ i $B$ neizraziti skupovi definirani nad univerzalnim skupom $U$. Presjek skupova $A$ i $B$ je neizraziti skup $A \cap B$, sa svojstvom: $\forall x \in U\ \mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x))$, odnosno*

$$A \cap B = \{(x; \mu_{A\cap B}(x)) \mid x \in U, \mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x))\}.$$

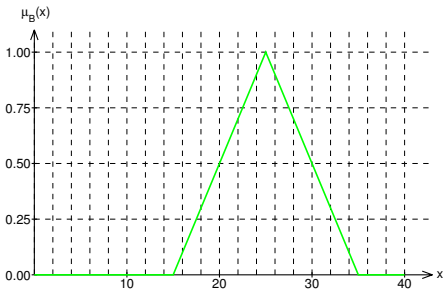
**Definicija 19** (Komplement neizrazitog skupa)**.** *Neka je $A$ neizraziti skup definiran nad univerzalnim skupom $U$. Komplement skupa $A$ je neizraziti skup $\neg A$, sa svojstvom: $\forall x \in U\ \mu_{\neg A}(x) = 1 - \mu_A(x)$, odnosno*

$$\neg A = \{(x; \mu_{\neg A}(x)) \mid x \in U, \mu_{\neg A}(x) = 1 - \mu_A(x)\}.$$

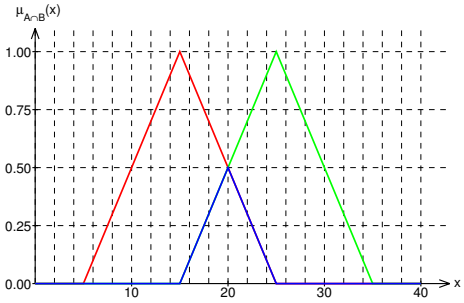
*Komplement skupa $A$ još će se označavati i s $A^C$.*



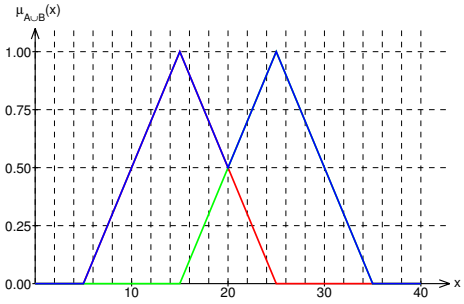

(a) Neizraziti skup $A$



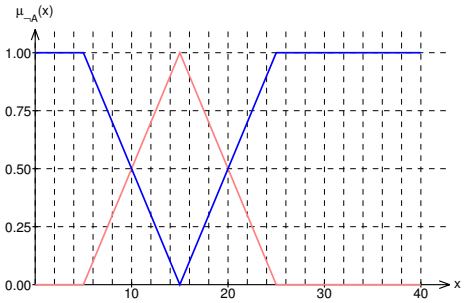

(b) Neizraziti skup $B$




(c) Neizraziti skup $A \cap B$




(d) Neizraziti skup $A \cup B$




(e) Neizraziti skup $\neg A$

Slika 2.1: Standardne definicije presjeka, unije i komplementa neizrazitih skupova. Rezultat operacija prikazan je plavom bojom.

Ilustracije ovih operacija dane su na slici 2.1. Ove definicije dao je u svojim radovima Zadeh, a još ih zovemo i *min-max* definicije (definicija 17, definicija 18, definicija 19). Tablica 2.1 navodi svojstva *min-max* operatora.

Tablica 2.1: Svojstva *min-max* operatora

| | |
|---|---|
| $A \cup B = B \cup A$ | Komutativnost |
| $A \cap B = B \cap A$ | Komutativnost |
| $A \cup (B \cup C) = (A \cup B) \cup C$ | Asocijativnost |
| $A \cap (B \cap C) = (A \cap B) \cap C$ | Asocijativnost |
| $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ | Distributivnost |
| $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ | Distributivnost |
| $A \cup A = A$ | Idempotencija |
| $A \cap A = A$ | Idempotencija |
| $(A \cup B)^C = A^C \cap B^C$ | DeMorganovi zakoni |
| $(A \cap B)^C = A^C \cup B^C$ | DeMorganovi zakoni |
| $(A^C)^C = A$ | Involutivnost |
| $A \cup \emptyset = A$, $A \cup U = U$ | Rubni uvjeti |
| $A \cap \emptyset = \emptyset$, $A \cap U = A$ | Rubni uvjeti |
| Ako je $A \subseteq B$ i $B \subseteq C$ tada je $A \subseteq C$ | Tranzitivnost |
| $A \cap A^C = \emptyset$ | Zakon kontradikcije – NE VRIJEDI |
| $A \cup A^C = U$ | Zakon isključenja trećega – NE VRIJEDI |

## 2.2 *s*-norme i *t*-norme

Osim *min-max* operatora postoji još mnoštvo načina za definiranje ovih istih operacija. Štoviše, napravljena je generalizacija ovih operacija i tako su izvedene *t*-norme (za operacije presjeka), *s*-norme (za operacije unije; još se nazivaju i *t*-konorme) te generalizirane operacije komplementa. U nastavku ćemo dati njihove definicije.

**Definicija 20** (*t*-norma)**.** *t-norma označava klasu binarnih funkcija* $t : [0,1] \times [0,1] \to [0,1]$ *sa svojstvima T1-T4.*

| | | |
|---|---|---|
| *T1* | $t(a,b) = t(b,a)$ | *komutativnost* |
| *T2* | $t(a,t(b,c)) = t(t(a,b),c)$ | *asocijativnost* |
| *T3* | $a \leq c \wedge b \leq d \Rightarrow t(a,b) \leq t(c,d)$ | *monotonost* |
| *T4* | $t(a,1) = a$, $t(0,a) = 0$ | *rubni uvjeti* |

**Definicija 21** (*s*-norma)**.** *s-norma označava klasu binarnih funkcija* $s : [0,1] \times [0,1] \to [0,1]$ *sa svojstvima S1-S4.*

| | | |
|---|---|---|
| *S1* | $s(a,b) = s(b,a)$ | *komutativnost* |
| *S2* | $s(a,s(b,c)) = s(s(a,b),c)$ | *asocijativnost* |
| *S3* | $a \leq c \wedge b \leq d \Rightarrow s(a,b) \leq s(c,d)$ | *monotonost* |
| *S4* | $s(a,1) = 1,\ s(0,a) = a$ | *rubni uvjeti* |

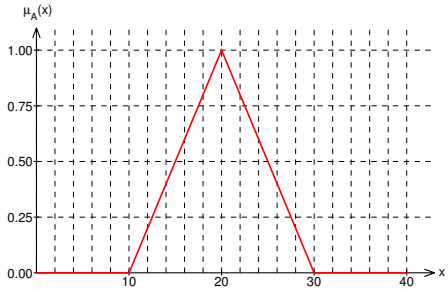
**Definicija 22** (Generalizirana operacija komplementa)**.** *Generalizirana operacija komplementa označava klasu unarnih funkcija* $c : [0,1] \to [0,1]$ *sa svojstvima C1-C3.*

| | | |
|---|---|---|
| *C1* | $c(0) = 1,\ c(1) = 0$ | *rubni uvjeti* |
| *C2* | $a < b \to c(a) > c(b)$ | |
| *C3* | $c(c(a)) = a$ | *involutivnost* |

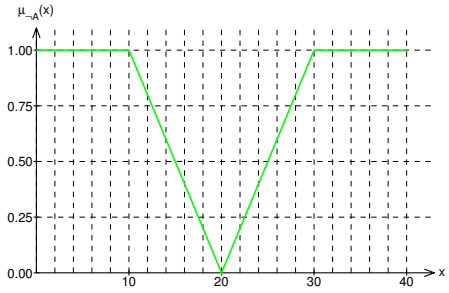
Za $s$-norme i $t$-norme te generaliziranu operaciju komplementa vrijede DeMorganovi zakoni, odnosno:

$$s(a,b) = c(t(c(a),c(b))),$$
$$t(a,b) = c(s(c(a),c(b))).$$

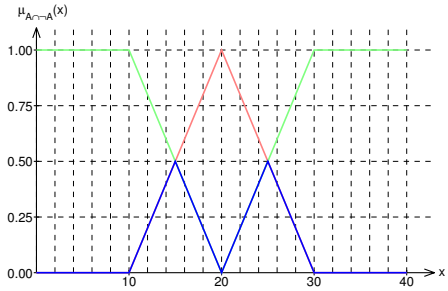
Osim DeMorganovovih zakona za koje smo upravo naveli da vrijede, može se pokazati da vrijede i svi ostali zakoni klasične teorije skupova, osim zakona kontradikcije $A \cap A^C = \emptyset$ i zakona isključenja trećega $A \cup A^C = U$ (vidi tablica 2.1). Da ova dva zakona ne vrijede, lako se možemo uvjeriti ako pokušamo izračunati te dvije operacije za neki konkretan skup, što je prikazano na slici 2.2. Slike 2.2a i 2.2b prikazuju skupove $A$ i $A^C$, dok su na slikama 2.2c i 2.2d prikazani presjek i unija. Jasno se vidi da presjek nije prazan skup, i unija nije univerzalni skup.



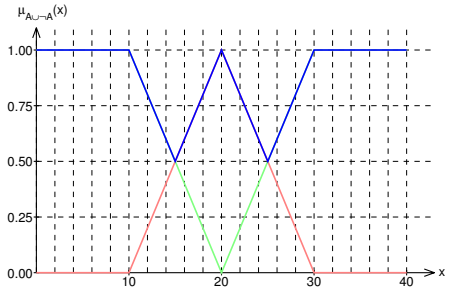

(a) Neizraziti skup $A$.




(b) Neizraziti skup $A^C$.




(c) Zakon kontradikcije – ne vrijedi.




(d) Zakon isključenja trećega – ne vrijedi.

Slika 2.2: Dva zakona koja ne vrijede.

---

**Primjer:** 4

Pokažite da je Zadehova definicija unije ispravno definirana $s$-norma. Pokažite da uz Zadehove definicije unije, presjeka i komplementa vrijede DeMorganovi zakoni.

**Rješenje:**

Zadeh je uniju definirao kao maksimum. Treba provjeriti da za takvu definiciju vrijede zahtjevi S1-s4.

Zahtjev S1 traži da vrijedi $s(a,b) = s(b,a)$. Provjerimo je li to istina za operaciju maksimum. $\max(a,b) = \max(b,a)$ ... vrijedi!

Zahtjev S2 traži da vrijedi $s(a, s(b,c)) = s(s(a,b),c)$. Provjerimo je li to istina za operaciju maksimum. $\max(a, \max(b,c)) = \max(\max(a,b),c)$ ... vrijedi!

Zahtjev S3 traži da vrijedi $a \leq c \wedge b \leq d \rightarrow s(a,b) \leq s(c,d)$. Provjerimo je li to istina za operaciju maksimum, tj. vrijedi li $\max(a,b) \leq \max(c,d)$. Odgovor ćemo potražiti analizom po slučajevima. Neka je npr. $a < b$ i $c < d$. Tada je $\max(a,b) = b$, $\max(c,d) = d$; kako S3 kaže da je $b < d$, tada doista vrijedi $max(a,b) = b < max(c,d) = d$. Slično se može ispitati i za ostale kombinacije te pokazati da zahtjev S3 vrijedi!

Zahtjev S4 traži da vrijedi $s(a,1) = 1$ i $s(0,a) = a$. Provjerimo je li to istina za operaciju maksimum, tj. vrijedi li $\max(a,1) = 1$ i $\max(0,a) = a$. Budući da se vrijednosti za $a$ kreću u intervalu $[0,1]$, tada doista vrijedi da je $\max(a,1) = 1$ i da je $\max(0,a) = a$.

Pokažimo sada još i da vrijede DeMorganovi zakoni. Dokazati ćemo samo jedan, jer se drugi dokazuje ekvivalentno (a ako vrijedi jedan, onda nam $s$-norme i $t$-norme te generalizacija operacije komplementa osiguravaju da vrijedi i drugi). Npr. pogledajmo zakon $s(a,b) = c(t(c(a), c(b)))$. U slučaju operacije max to bi zahtjevalo da vrijedi: $\max(a,b) = 1 - (\min(1-a, 1-b))$. Opet ćemo napraviti analizu po slučajevima.

Neka je $a < b$. Tada je $\max(a,b) = b$. Tada je i $1 - a > 1 - b$ pa je $\min(1-a, 1-b) = 1 - b$.

Uvrštavanjem slijedi:

$$b = 1 - (1 - b) = 1 - 1 + b = b.$$

Neka je $a > b$. Tada je $\max(a, b) = a$. Tada je i $1 - a < 1 - b$ pa je $\min(1 - a, 1 - b) = 1 - a$. Uvrštavanjem slijedi:

$$a = 1 - (1 - a) = 1 - 1 + a = a.$$

Dakle, DeMorganovi zakoni vrijede.

---

Zadehove definicije unije i presjeka definirane su imajući u vidu neizrazite skupove. Međutim, specijalni slučaj neizrazitih skupova su upravo klasični skupovi kod kojih elementi ili pripadaju skupu ili ne pripadaju. Zbog toga definicije unije i presjeka koje se koriste u neizrazitoj logici za specijalne slučajeve neizrazitih skupova (kada se skupovi mogu poistovjetiti sa klasičnima) moraju odgovarati definicijama unije i presjeka iz klasične teorije skupova. Ovo ilustrira sljedeći primjer.

---

**Primjer:** 5

Pokažite da se Zadehove definicije unije i presjeka za slučaj kada se koriste neizraziti skupovi s funkcijom pripadnosti ograničenom na vrijednosti $\{0, 1\}$ ponašaju jednako kao i definicije unije i presjeka kod klasičnih skupova.

**Rješenje:**

Operacije unije i presjeka u klasičnoj se teoriji skupova definiraju na sljedeći način. Element pripada uniju dvaju skupova ukoliko pripada barem jednom od skupova. Element pripada presjeku dvaju skupova ukoliko pripada u oba skupa. Ukoliko se poslužimo karakterističnom funkcijom, operacije unije i presjeka možemo prikazati i tablično.

| $\mu_A(x)$ | $\mu_B(x)$ | $\mu_{A\cap B}(x)$ | $\mu_{A\cup B}(x)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Kada se funkcija pripadnosti neizrazitog skupa ograniči na elemente $\{0, 1\}$, rezultat mora biti jednak. Provjerimo to.

| $\mu_A(x)$ | $\mu_B(x)$ | $\mu_{A\cap B}(x)$ | $\mu_{A\cup B}(x)$ |
|---|---|---|---|
| 0 | 0 | $\min(0,0) = 0$ | $\max(0,0) = 0$ |
| 0 | 1 | $\min(0,1) = 0$ | $\max(0,1) = 1$ |
| 1 | 0 | $\min(1,0) = 0$ | $\max(1,0) = 1$ |
| 1 | 1 | $\min(1,1) = 1$ | $\max(1,1) = 1$ |

Vidimo da se rezultati u tablici doista poklapaju s tablicom u kojoj smo prikazali klasične operacije unije i presjeka, što jasno govori da se za specijalni slučaj neizrazitih skupova koji odgovaraju klasičnim skupovima definicije unije i presjeka iz teorije neizrazitih skupova poklapaju s operacijama unije i presjeka iz klasične teorije skupova.

---

Osim prethodno opisanih Zadehovih $s$-normi i $t$-normi, postoji još mnoštvo normi koje su u uporabi, a neke od njih navedene su u nastavku (tablica 2.2).

Pri tome je zanimljivo istaknuti da se različite $t$-norme, baš kao i različite $s$-norme mogu poredati po veličini. Naime, vrijedi sljedeći odnos:

$$t_d \leq t_o \leq t_E \leq t_a \leq t_H \leq t_{\min},$$

Tablica 2.2: Česte neparametrizirane $t$-norme i $s$-norme

| | | |
|---|---|---|
| Zadeh Minimum ($t_{\min}$), Zadeh Maksimum ($t_{\min}$) | $\min(a,b)$ | $\max(a,b)$ |
| Hamacherov produkt ($t_H$), Hamacherova suma ($s_H$) | $\frac{ab}{a+b-ab}$ | $\frac{a+b-2ab}{1-ab}$ |
| Algebarski produkt ($t_a$), Algebarska suma ($s_H$) | $ab$ | $a+b-ab$ |
| Einsteinov produkt ($t_E$), Einsteinova suma ($s_E$) | $\frac{ab}{2-(a+b-ab)}$ | $\frac{a+b}{1+ab}$ |
| Ograničeni produkt ($t_o$), Ograničena suma ($s_o$) | $\max(0,a+b-1)$ | $\min(1,a+b)$ |
| Drastični produkt ($t_d$), Drastična suma ($t_s$) | $\begin{cases}\min(a,b), & \max(a,b)=1\\ 0, & \text{inače}\end{cases}$ | $\begin{cases}\max(a,b), & \min(a,b)=0\\ 1, & \text{inače}\end{cases}$ |

$$s_{\max} \leq s_H \leq s_a \leq s_E \leq s_o \leq s_d.$$

Iz ovoga je vidljivo da je funkcija *minimum* gornja ograda za sve $t$-norme a funkcija *maksimum* donja ograda za sve $s$-norme.

**Primjer:** 6

Pokažite da se sve neparametrizirane norme koje smo definirali (tablica 2.2) u slučaju kada se koriste neizraziti skupovi s funkcijom pripadnosti ograničenom na vrijednosti $\{0,1\}$ ponašaju jednako kao i definicije unije i presjeka kod klasičnih skupova.

**Rješenje:**

Za Zadehove operacije tvrdnju smo već provjerili. Provjerimo za ostale norme.

*Hamacherov produkt – Hamacherova suma*

| $\mu_A(x)$ | $\mu_B(x)$ | $\mu_{A\cap B}(x)$ | $\mu_{A\cup B}(x)$ |
|---|---|---|---|
| 0 | 0 | $\frac{0\cdot 0}{0+0-0\cdot 0}=0$ | $\frac{0+0-2\cdot 0\cdot 0}{1-0\cdot 0}=0$ |
| 0 | 1 | $\frac{0\cdot 1}{0+1-0\cdot 1}=0$ | $\frac{0+1-2\cdot 0\cdot 1}{1-0\cdot 1}=1$ |
| 1 | 0 | $\frac{1\cdot 0}{1+0-1\cdot 0}=0$ | $\frac{1+0-2\cdot 1\cdot 0}{1-1\cdot 0}=1$ |
| 1 | 1 | $\frac{1\cdot 1}{1+1-1\cdot 1}=1$ | $\frac{1+1-2\cdot 1\cdot 1}{1-1\cdot 1}=1$ |

Kod Hamacherovog produkta treba uočiti slučaj $t_H(0,0)$ koji direktnim uvrštavanjem u izraz daje razlomak $\frac{0}{0}$ pa nije jasno koju to vrijednost poprima. Međutim, može se pogledati slučaj $t_H(0,\delta)$ za $\delta > 0$:

$$t_H(0,\delta) = \frac{0\cdot\delta}{0+\delta-0\cdot\delta} = \frac{0}{\delta} = 0.$$

Vidimo da je za sve pozitivne iznose od $\delta$ vrijednost jednaka 0, pa se definira da i u limesu kad $\delta \to 0$ iznos ostaje 0. Također, ako dopustimo da oba parametra budu različita od 0 i da se mjenjaju na isti način (neka je $a = b = \xi$), tada imamo da je $t_H(a,b) = t_H(\xi,\xi)$ jednak:

$$t_H(\xi,\xi) = \frac{\xi \cdot \xi}{\xi + \xi - \xi \cdot \xi} = \frac{1}{\frac{2}{\xi} - 1}$$

iz čega se vidi da izraz u limesu teži prema nuli:

$$\lim_{\xi \to 0} = \frac{1}{\frac{2}{\xi} - 1} = 0.$$

Na sličan se način može pokazati da Hamacherova suma za slučaj $s_H(1,1)$ poprima vrijednost 1.

*Algebarski produkt – Algebarska suma*

| $\mu_A(x)$ | $\mu_B(x)$ | $\mu_{A\cap B}(x)$ | $\mu_{A\cup B}(x)$ |
|---|---|---|---|
| 0 | 0 | $0 \cdot 0 = 0$ | $0 + 0 - 0 \cdot 0 = 0$ |
| 0 | 1 | $0 \cdot 1 = 0$ | $0 + 1 - 0 \cdot 1 = 1$ |
| 1 | 0 | $1 \cdot 0 = 0$ | $1 + 0 - 1 \cdot 0 = 1$ |
| 1 | 1 | $1 \cdot 1 = 1$ | $1 + 1 - 1 \cdot 1 = 1$ |

*Einsteinov produkt – Einsteinova suma*

| $\mu_A(x)$ | $\mu_B(x)$ | $\mu_{A\cap B}(x)$ | $\mu_{A\cup B}(x)$ |
|---|---|---|---|
| 0 | 0 | $\frac{0\cdot 0}{2-(0+0-0\cdot 0)} = 0$ | $\frac{0+0}{1+0\cdot 0} = 0$ |
| 0 | 1 | $\frac{0\cdot 1}{2-(0+1-0\cdot 1)} = 0$ | $\frac{0+1}{1+0\cdot 1} = 1$ |
| 1 | 0 | $\frac{1\cdot 0}{2-(1+0-1\cdot 0)} = 0$ | $\frac{1+0}{1+1\cdot 0} = 1$ |
| 1 | 1 | $\frac{1\cdot 1}{2-(1+1-1\cdot 1)} = 1$ | $\frac{1+1}{1+1\cdot 1} = 1$ |

*Ograničen produkt – Ograničena suma*

| $\mu_A(x)$ | $\mu_B(x)$ | $\mu_{A\cap B}(x)$ | $\mu_{A\cup B}(x)$ |
|---|---|---|---|
| 0 | 0 | $\max(0, 0+0-1) = 0$ | $\min(1, 0+0) = 0$ |
| 0 | 1 | $\max(0, 0+1-1) = 0$ | $\min(1, 0+1) = 1$ |
| 1 | 0 | $\max(0, 1+0-1) = 0$ | $\min(1, 1+0) = 1$ |
| 1 | 1 | $\max(0, 1+1-1) = 1$ | $\min(1, 1+1) = 1$ |

*Drastični produkt – Drastična suma*

| $\mu_A(x)$ | $\mu_B(x)$ | $\mu_{A\cap B}(x)$ | $\mu_{A\cup B}(x)$ |
|---|---|---|---|
| 0 | 0 | $\begin{cases} \min(0,0), & \max(0,0) = 1 \\ 0, & \text{inače} \end{cases} = 0$ | $\begin{cases} \max(0,0), & \min(0,0) = 0 \\ 1, & \text{inače} \end{cases} = 0$ |
| 0 | 1 | $\begin{cases} \min(0,1), & \max(0,1) = 1 \\ 0, & \text{inače} \end{cases} = 0$ | $\begin{cases} \max(0,1), & \min(0,1) = 0 \\ 1, & \text{inače} \end{cases} = 1$ |
| 1 | 0 | $\begin{cases} \min(1,0), & \max(1,0) = 1 \\ 0, & \text{inače} \end{cases} = 0$ | $\begin{cases} \max(1,0), & \min(1,0) = 0 \\ 1, & \text{inače} \end{cases} = 1$ |
| 1 | 1 | $\begin{cases} \min(1,1), & \max(1,1) = 1 \\ 0, & \text{inače} \end{cases} = 1$ | $\begin{cases} \max(1,1), & \min(1,1) = 0 \\ 1, & \text{inače} \end{cases} = 1$ |

Ovime smo dokazali da vrijede rubni uvjeti za sve navedene neparametrizirane norme.

## 2.3 Parametrizirane $s$-norme i $t$-norme

Uz prethodno prikazane norme, razvijena je još jedna porodica normi – parametrizirane $s$-norme i $t$-norme, koje se od prethodno navedenih razlikuju po tome što sve nude jedan dodatni parametar kojim se može fino podešavati djelovanje same norme. Na ovaj način omogućava se izgradnja sustava koji svoj rad zasnivaju na neizrazitoj logici, uz mogućnost naknadnog finog podešavanja rada sustava bez potrebe za kompletnim redizajnom i uporabom neke druge norme. Neke uobičajene parametrizirane $s$-norme i $t$-norme navedene su u nastavku (tablica 2.3). Lako se može provjeriti da i ove parametrizirane norme zadovoljavaju sve zahtjeve koji su postavljeni generalizacijom (S1-S4 za $s$-norme te T1-T4 za $t$-norme). Uz ove parametarske norme parametrizirani su i odgovarajući komplementi. Tipične parametrizirane komplemente prikazuje tablica 2.4.

Tablica 2.3: Česte parametrizirane $t$-norme i $s$-norme

| $t$-norma | $s$-norma |
|---|---|
| Hamacherova norma, $\nu \geq 0$ | |
| $\frac{ab}{\nu+(1-\nu)(a+b-ab)}$ | $\frac{a+b-(2-\nu)ab}{1-(1-\nu)ab}$ |
| Yagerova norma, $q \geq 0$ | |
| $1-\min(1, \sqrt[q]{(1-a)^q+(1-b)^q})$ | $\min(1, \sqrt[q]{a^q+b^q})$ |
| Frankova norma, $s>0$, $s\neq 1$ | |
| $\log_s\left(1+\frac{(s^a-1)(s^b-1)}{s-1}\right)$ | $1-\log_s\left(1+\frac{(s^{1-a}-1)(s^{1-b}-1)}{s-1}\right)$ |
| Dubois-Prade norma, $\alpha\in[0,1]$ | |
| $\frac{ab}{\max(a,b,\alpha)}$ | $\frac{a+b-ab-\min(a,b,1-\alpha)}{\max(1-a,1-b,\alpha)}$ |

Tablica 2.4: Parametrizirane operacije komplementa

| Naziv komplementa | Izraz |
|---|---|
| Sugenov komplement | $\frac{1-a}{1+\alpha\cdot a}$ |
| Yagerov komplement | $\sqrt[\beta]{1-a^\beta}$ |

Sada se može jasno uočiti da su neparametrizirane norme samo specijalni slučajevi parametriziranih. Tablica 2.5 prikazuje neparametrizirane norme, kao i koje parametrizirane norme za koju vrijednost parametra prelaze u odgovarajuće neparametrizirane inačice.

Tablica 2.5: Odnos parametriziranih i neparametriziranih normi

| Naziv norme | Parametrizirana norma |
|---|---|
| Hamacherov produkt ($t_H$) / Hamacherova suma ($s_H$) | $H(0)$ |
| Algebarski produkt ($t_a$) / Algebarska suma ($s_a$) | $H(1)$, $DP(1)$ |
| Einsteinov produkt ($t_E$) / Einsteinova suma ($s_E$) | $H(2)$ |
| Ograničena razlika ($t_o$) / Ograničena suma ($s_o$) | $Y(1)$ |
| Drastična razlika ($t_d$) / Drastična suma ($s_d$) | $H(\infty)$, $Y(0)$ |

**Primjer:** 7

Pokažite da se ograničena suma odnosno ograničena razlika doista dobiju iz Yagerove norme za vrijednost parametra 1.

**Rješenje:**

Pogledajmo najprije Yagerovu $s$-normu: $\min(1, \sqrt[q]{a^q + b^q})$. Za vrijednost parametra $q = 1$ imamo: $\min(1, \sqrt[1]{a^1 + b^1}) = \min(1, a + b)$ što je doista ograničena suma.

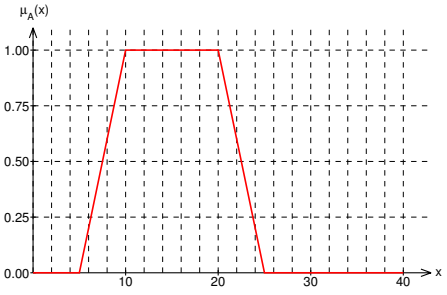
Yagerova $t$-norma $1 - \min(1, \sqrt[q]{(1-a)^q + (1-b)^q})$ za vrijednost parametra $q = 1$ prelazi u: $1 - \min(1, \sqrt[1]{(1-a)^1 + (1-b)^1}) = 1 - \min(1, 1 - a + 1 - b) = 1 - \min(1, 2 - a - b)$. Da bismo dobili oblik u kojem je zapisan ograničeni produkt, treba se prisjetiti veze između minimuma i maksimuma kada se vrijednosti nalaze u intervalu $[0, 1]$. Vrijedi: $\min(a, b) = 1 - \max(1-a, 1-b)$ pa uvrštavanjem slijedi: $1 - \min(1, 2-a-b) = 1 - (1 - \max(1-1, 1-(2-a-b))) = \max(0, a+b-1)$ što je upravo formula ograničenog produkta.

Primjer djelovanja parametriziranih Hamacherovih $t$-normi i $s$-normi prikazan je na slici 2.3. Iz tog primjera je jasno vidljivo ponašanje pri rubnim uvjetima, kao i kakav utjecaj imaju vrijednosti parametra na konačni rezultat.

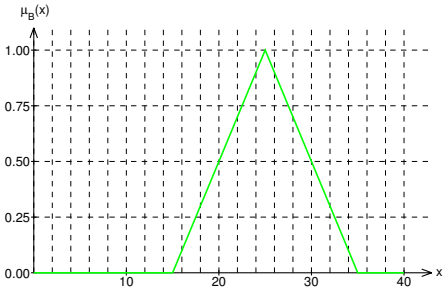
Iz prethodnog razmatranja vidimo da nam na raspolaganju stoji velik broj različitih normi. Međutim, kada se odlučimo za realizaciju konkretnog sustava morati ćemo odabrati jednu od tih normi i s njom raditi do kraja. Na temelju kojih kriterija možemo obaviti taj odabir? Nekoliko prijedloga dano je u nastavku.

**Aksiomi koje operator zadovoljava.** Primjer aksioma Bellman-Giertz i Hamacher. Ako je sve drugo isto, bolji je onaj operator čiji su aksiomi manje ograničavajući.

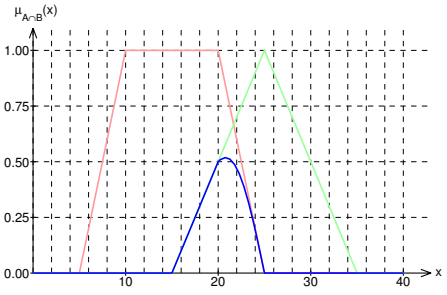
**Empirijska provjera.** Teorija neizrazitih skupova koristi se za modeliranje realnih situacija i modela; stoga nije samo dovoljno da operator zadovoljava određene aksiome već mora adekvatno modelirati stvarne situacije.



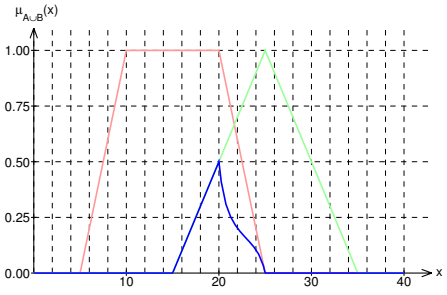

(a) Neizraziti skup $A$



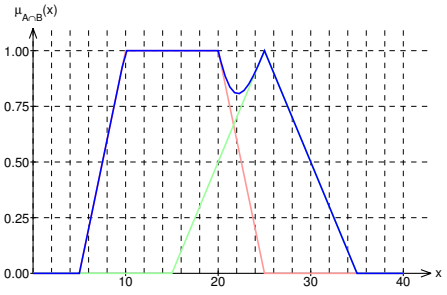

(b) Neizraziti skup $B$



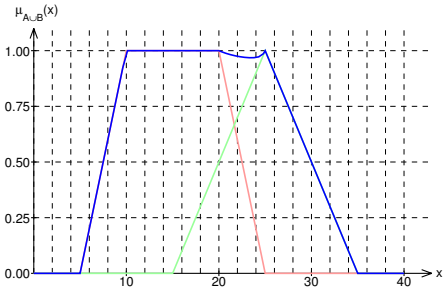

(c) Neizraziti skup $A \cap B$ uz $\nu = 0.1$




(d) Neizraziti skup $A \cap B$ uz $\nu = 10$




(e) Neizraziti skup $A \cup B$ uz $\nu = 0.1$




(f) Neizraziti skup $A \cap B$ uz $\nu = 10$

Slika 2.3: Unija i presjek neizrazitih skupova uporabom Hamacherovih parametriziranih normi. Rezultat operacija prikazan je plavom bojom. Vrijednost parametra navedena je uz svaku sliku zasebno.

**Prilagodljivost.** Izbor operatora ovisi o kontekstu i semantičkoj interpretaciji. Ako je potrebno modelirati jednim operatorom više stvarnih situacija tada se to postiže odabirom odgovarajućeg parametra parametriziranih operatora (Yager, Hamacher, itd.). *Min-max* operatori nisu prihvatljivi u smislu prilagodljivosti, ali oni imaju druge prednosti poput numeričke efikasnosti.

**Numerička efikasnost.** *Min-max* operatori numerički su efikasni (za razliku od primjerice Hamacherovog i $\gamma$ operatora) što može biti jako važno kod rješavanja većih realnih problema.

**Kompezatornost.** Logički "i" ne dozvoljava kompenzaciju; mala vrijednost stupnja pripadnosti jednom skupu ne može se kompenzirati povećanjem stupnja pripadnosti drugom skupu.

Kompenzacija u kontekstu operatora združivanja znači sljedeće: za danu vrijednost funkcije pripadnosti elementa $x$ združenom neizrazitom skupu

$$\mu_{Aggr}(x) = f(\mu_A(x), \mu_B(x)) = k,$$

$f$ je kompenzatorni operator ako se vrijednost $k$ može dobiti za različite $\mu_A(x)$ mijenjanjem $\mu_B(x)$. Primjerice, *min* nije kompenzatoran ali produkt i $\gamma$-operator jesu).

**Rang kompenzacije** Za kompenzatorne operatore vrijedi da su bolji što imaju veći rang kompenzacije. Primjer: konveksna kombinacija *min* i *max* omogućava kompenzaciju između *min* i *max*, dok produkt operator dozvoljava kompenzaciju na $(0, 1)$.

**Ponašanje kod združivanja** Ovisnost o broju skupova koji se združuju. Primjer: kod produkt operatora svaki novi dodani skup utječe na vrijednost funkcije pripadnosti u združenom skupu, kod *min* operatora to nije slučaj.

**Skala vrijednosti funkcije pripadnosti** Skale općenito mogu biti: nominalna, uređajna (ordinalna), intervalna te racionalna. Operator koji zahtijeva slabiju skalu je poželjniji. Na primjer, *min* operator dozvoljava ordinalnu skalu dok produkt ne dozvoljava.

Prilikom kombiniranja dvaju neizrazitih skupova do sada smo razmotrili dva općenita načina njihovog kombiniranja. Jedan je uporabom $t$-normi. Rezultat združivanja dvaju neizrazitih skupova uporabom $t$-normi ograničen je na interval $[0, \min(\mu_A(x), \mu_B(x))]$ (jer je minimum gornja ograda svih $t$-normi). Drugi način združivanja je uporabom $s$-normi. Rezultat združivanja dvaju neizrazitih skupova uporabom $s$-normi ograničen je na interval $[\max(\mu_A(x), \mu_B(x)), 1]$ (jer je maksimum donja ograda svih $s$-normi). Rezultat združivanja koji bi pripadao u međuprostor, odnosno u interval

$[\min(\mu_A(x), \mu_B(x)), \max(\mu_A(x), \mu_B(x))]$, nije moguće dobiti niti $t$-normama niti $s$-normama. Taj prostor pokrivaju *operatori uprosječivanja*.

U kontekstu donošenja odluka (npr. kod višekriterijskog programiranja) vrlo važnu ulogu imaju operatori uprosječivanja (engl. *averaging operators*). Ovi operatori realiziraju ideju "trgovanja" (engl. *trade-off*) između konfliktnih ciljeva kada je dozvoljena kompenzacija. Tada rezultat leži između najoptimističnije donje (*min*) i najpesimističnije gornje (*max*) granice. Primjeri ovih operatora su dobro poznati:

- aritmetička sredina,
- geometrijska sredina,
- harmonijska sredina,
- neizraziti "i",
- neizraziti "ili" te
- kompenzatorni "i" (tj. $\gamma$-operator).

Tablica 2.6 prikazuje definicije ovih operatora.

Tablica 2.6: Definicije operatora uprosječivanja

| Naziv operatora | Definicija |
|---|---|
| aritmetička sredina | $\bar{x} = \frac{x_1+x_2}{2}$ |
| geometrijska sredina | $G = \sqrt{x_1^2 + x_2^2}$ |
| harmonijska sredina | $H = \frac{2}{\frac{1}{x_1}+\frac{1}{x_2}}$ |
| neizraziti "i" | |
| $\mu_{A \text{ i } B}(x) = \gamma \min(\mu_A(x), \mu_B(x)) + (1-\gamma)\frac{\mu_A(x)+\mu_B(x)}{2}$, za $\gamma \in [0,1]$ | |
| neizraziti "ili" | |
| $\mu_{A \text{ ili } B}(x) = \gamma \max(\mu_A(x), \mu_B(x)) + (1-\gamma)\frac{\mu_A(x)+\mu_B(x)}{2}$, za $\gamma \in [0,1]$ | |
| kompenzatorni "i", odnosno $\gamma$-operator, $0 \leq \gamma \leq 1$ | |
| $\mu_\gamma(x) = \left(\prod_{i=1}^{m} \mu_i(x)\right)^{1-\gamma} \left(1 - \prod_{i=1}^{m}(1-\mu_i(x))\right)^{\gamma}$ | |

Uočite da operator neizraziti "i" temeljem parametra $\gamma$ interpolira rezultat između Zadehovog minimuma i aritmetičke sredine funkcija pripadnosti dok operator neizraziti "ili" temeljem parametra $\gamma$ interpolira rezultat između Zadehovog maksimuma i aritmetičke sredine funkcija pripadnosti. S operatorima neizraziti "i" i neizraziti "ili" postižu se vrlo dobri empirijski rezultati. Za $\gamma = 1$ $\mu_{A \text{ i } B}(x)$ odgovara Zadehovom *min* operatoru, a $\mu_{A \text{ ili } B}(x)$ Zadehovom *max* operatoru. Za $\gamma < 1$ operator uprosječivanja je više optimističan nego minimum operator koji je najoptimističniji u slučaju $t$-normi. Bitno je napomenuti da ovi operatori nisu asocijativni.

**Primjer:** 8

Pokažite da operator neizraziti "i" nije asocijativan.

**Rješenje:**

Pretpostavimo sljedeće: $\gamma = 0.6$, $a = 0.2$, $b = 0.4$ te $c = 0.8$. Računamo redom:

$$a \text{ i } b = 0.6 \cdot \min(a, b) + 0.4 \cdot (a + b)/2 = 0.24$$

$$b \text{ i } c = 0.6 \cdot \min(a, b) + 0.4 \cdot (a + b)/2 = 0.44$$

$$(a \text{ i } b) \text{ i } c = 0.6 \cdot \min(0.24, c) + 0.4 \cdot (0.24 + c)/2 = 0.312$$

$$a \text{ i } (b \text{ i } c) = 0.6 \cdot \min(a, 0.44) + 0.4 \cdot (a + 0.44)/2 = 0.248.$$

Očito je da redoslijed primjene operatora utječe na konačni rezultat, pa operator nije asocijativan.

# Poglavlje 3

# Jezične varijable

Neizrazita logika vrlo je pogodna za približno zaključivanje, kada imamo neprecizno ili nejasno definirane podatke. Osnovni pojam koji se koristi u približnom zaključivanju jest *jezična varijabla*. Uvođenjem jezične varijable omogućujemo približan i sustavan opis i analizu sustava koji su inače nejasni ili prekomplicirani da bi se na njih primijenile konvencionalne matematičke metode. Formalna definicija jezične varijable dana je u nastavku (definicija 23).

**Definicija 23** (Jezična varijabla)**.** *Jezična varijabla je uređena petorka* $(X, T_X, U, G, M_X)$ *gdje je:*

$X$ *naziv jezične varijable,*

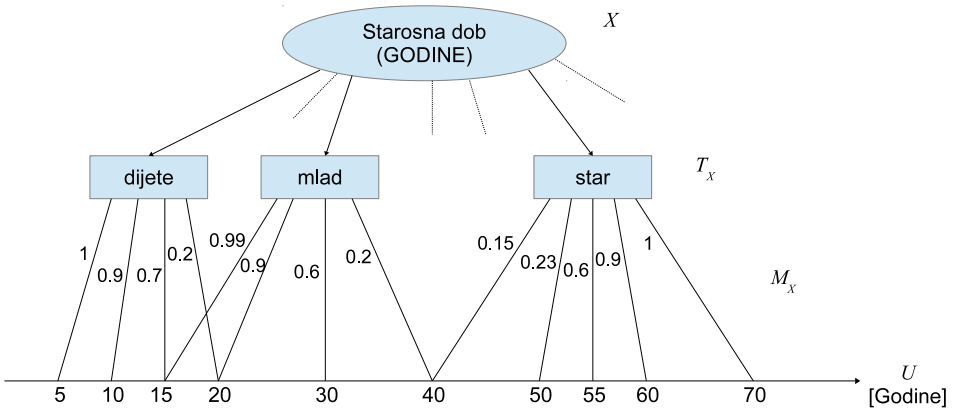
$T_X$ *skup jezičnih vrijednosti koje jezična varijabla može poprimiti (engl.* term set*),*

$U$ *stvarna fizička domena u kojoj elementi iz* $T_X$ *poprimaju numeričke vrijednosti (engl.* universe of disclosure*),*

$G$ *gramatika (odnosno skup sintaksnih pravila) koja generira skup* $T_X$ *iz osnovnih termina; to je kontekstno ovisna gramatika te*

$M_X$ *semantička funkcija koja daje značenje lingvističkim izrazima. To je funkcija koja svakom elementu iz* $T_X$ *pridružuje neizraziti podskup od* $U$ *(engl.* meaning*).*

Pogledajmo to na jednostavnom primjeru jezične varijable "starosna dob", prikazane na slici 3.1. Jezična variabla starosna dob definirana je nad skupom cijelih nenegativnih brojeva koje tumačimo kao starost osobe. Ta jezična varijable ima kao skup osnovnih termina vrijednosti {"dijete", "mlad", "star"} (i još druge koje nisu prikazane na slici). Za svaki od tih termina definiran je neizraziti skup koji terminu daje značenje; tako je terminu "dijete" pridružen neizraziti skup za koji sa slike vidimo da je $\mu_{\text{dijete}}(5) = 1$,




Slika 3.1: Jezična varijabla "starosna dob".

$\mu_{\text{dijete}}(10) = 0.9$, $\mu_{\text{dijete}}(15) = 0.7$, i tako dalje (zbog preglednosti na slici nisu napisane sve vrijednosti). Na sličan način je i svakom drugom osnovnom terminu pridružen jedan neizraziti skup koji mu daje značenje.

Uz definiranu jezičnu varijablu možemo zamisliti i da je dana gramatika koja bi omogućila definiranje izvedenih vrijednosti, poput "dijete ili mlad", "ne start", "dijete ili vrlo mlad" i slično, gdje su operacije "i", "ili" i "ne" klasične neizrazite operacije a "vrlo" jezični modifikator (obradit ćemo ih uskoro). Svakom od tih izraza može se opet pridružiti jedan neizraziti skup koji mu definira značenje; taj se skup međutim ne zadaje unaprijed već ga se može izračunati. Primjerice, pogledajmo izraz "dijete ili mlad". U okviru definicije jezične varijable definiran je neizraziti skup koji je pridružen terminu "dijete" te neizraziti skup koji je pridružen terminu "mlad". Izrazu "dijete ili mlad" možemo pridružiti neizraziti skup koji je unija tih dviju neizrazitih skupova, što je moguće napraviti automatski tijekom tumačenja izraza koji je generiran pridruženom gramatikom i semantičkom funkcijom.

Uz jezične varijable vezan je i pojam jezičnih modifikatora. Prilikom definiranja jezične varijable (npr. "starosna dob") definira se skup osnovnih termina (npr. "mlad", "star") i skup jezičnih modifikatora koji mogu djelovati na te osnovne izraze i graditi njihove modifikacije (npr. modifikator "vrlo" kako bismo mogli izgraditi izraz "vrlo mlad"). Uobičajeno se definiraju tri osnovna modifikatora: koncentracija, dilatacija te kontrastna intenzifikacija.

**Definicija 24** (Koncentracija, dilatacija, kontrastna intenzifikacija)**.** *Neka je $A$ neizrazit skup definiran nad univerzalnim skupom $U$, te neka je $\mu_A(x)$ funkcija pripadnosti skupa $A$. Koncentracija od $A$ je također neizraziti skup $con(A)$, sa svojstvom: $\mu_{con(A)}(x) = \mu_A(x)^2$. Dilatacija od $A$ je također neizraziti skup $dil(A)$ sa svojstvom: $\mu_{dil(A)}(x) = \sqrt{\mu_A(x)}$. Kontrastna in-*

*tenzifikacija od $A$ je također neizraziti skup $int(A)$ sa svojstvom:*

$$\mu_{int(A)}(x) = \begin{cases} 2 \cdot \mu_A(x)^2 & 0 \leq \mu_A(x) \leq 0.5 \\ 1 - 2 \cdot (1 - \mu_A(x))^2 & 0.5 < \mu_A(x) \leq 1. \end{cases}$$

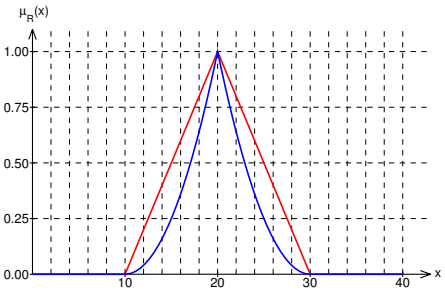
Djelovanje ovih modifikatora prikazuju slike 3.2a, 3.2b i 3.2c, pri čemu je funkcija pripadnosti skupa $A$ nacrtana crvenom bojom, a funkcija pripadnosti modificiranih skupova plavom bojom.

Pokušajmo sada pronaći jezične izraze kojima bismo mogli opisati učinke pojedinih modifikatora. Npr. pogledajmo što prikazuje slika 3.2a. Skup $A$ (crvena boja) definiran je nad univerzalnim skupom realnih brojeva $[0, 40]$. Kada bismo morali pogađati koji koncept predstavlja taj skup, mogli bismo npr. reći da on predstavlja koncept "brojevi oko 20", jer mu broj 20 pripada sa stupnjem pripadnosti 1, dok stupnjevi pripadnosti brojeva lijevo i desno opadaju sa udaljavanjem od tog broja. Tako npr. broj 15 pripada skupu "brojevi oko 20" sa stupnjem pripadnosti 0.5. Ako bismo sada morali pogađati što predstavlja koncentracija skupa $A$ (plava boja), što bismo odgovorili? Broj 20 i ovom skupu pripada sa stupnjem pripadnosti 1, što opet upućuje na koncept "brojevi oko 20". Međutim, kako smo rekli da baš skup $A$ opisuje koncept "brojevi oko 20", ne možemo to isto reći i za njegovu koncentraciju jer ta dva skupa nisu jednaka. Pogledajmo primjerice broj 15. On koncentraciji skupa $A$ pripada sa stupnjem pripadnosti od 0.25, što je manji stupanj od onoga kojim taj isti broj pripada skupu $A$ (0.5). Stoga možemo zaključiti da ovaj skup predstavlja koncept koji puno drastičnije shvaća pojam "oko" odnosno ne dozvoljava veliko udaljavanje. Koju riječ bismo mogli upotrijebiti za ovo pooštrenje koncepta "brojevi oko 20"? Prirodno se nameće riječ: *vrlo*. Koncentracija skupa $A$ može se shvatiti kao koncept "brojevi oko 20, koji su *vrlo* blizu oko 20". Slično se razmatranje može provesti i za ostale modifikatore. Štoviše, uobičajeno je poistovjećivanje prikazano tablicom .

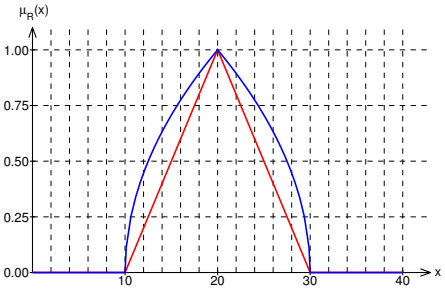
Tablica 3.1: Veza jezičnih modifikatora i jezičnih izraza

$$con(A) \equiv vrlo\ A\ (very\ A)$$
$$dil(A) \equiv neznatno\ A\ (slightly\ A)$$
$$A^{1.25} \equiv više\ A\ (more\ A)$$
$$A^{0.75} \equiv manje\ A\ (less\ A)$$
$$int(više\ A \wedge ne\ vrloA) \equiv manje\ ili\ više\ A\ (more\ or\ less\ A)$$

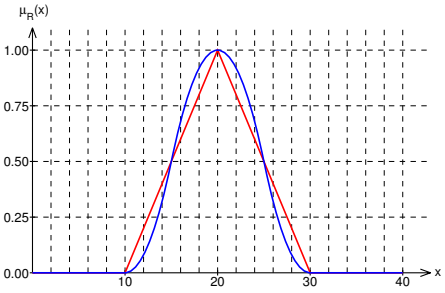
I odmah da odgovorim na pitanje koje se jasno postavlja: mora li se pojam *vrlo* vezati baš uz kvadriranje? Ne mora! Kako ćemo shvaćati značenje izraza *vrlo* $A$, potpuno je subjektivno. Možemo npr. uz izraz *vrlo* $A$ vezati potenciranje stupnjeva pripadnosti na 3.14-tu potenciju, ako smatramo da je baš to ono što će nam omogućiti da ostvarimo sustav koji gradimo, ili slično. Bitno je jedino da uz izraz *vrlo* povežemo takvu operaciju koja će biti smislena. Npr. ako broj 15 skupu "brojevi oko 20" pripada sa stupnjem




(a) Koncentracija neizrazitog skupa $A$




(b) Dilatacija neizrazitog skupa $A$




(c) Kontrastna intenzifikacija neizrazitog skupa $A$

Slika 3.2: Jezični modifikatori: koncentracija, dilatacija, kontrastna intenzifikacija. Rezultat operacija prikazan je plavom bojom.

pripadnosti 0.5, tada je za izgradnju koncepta "brojevi VRLO blizu oko 20" pogodna svaka operacija koja će tom istom broju 15 pridružiti stupanj pripadnosti koji je manji od 0.5 (jer se inače dovodimo u sukob sa zdravim razumom). Slično vrijedi i za ostale jezične izraze. Npr. *više A* se može računati kao $A$ na 1.38997-u potenciju i sl.

U ovom poglavlju koristili smo još nešto što je potrebno malo pobliže definirati. Kada smo pokušavali koncentraciji dodijeliti nekakav jezični izraz (uspoređujući skup $A$ i njegovu koncentraciju), igrali smo se pogađanja i pokušavali smo otkriti što bi to zapravo moglo biti. Ovaj proces "pogađenja" zove se *jezična aproksimacija*. Jezična aproksimacija je proces pronalaženja jezičnog izraza čije je značenje (funkcija pripadnosti) jednako ili najbliže moguće značenju (ili funkciji pripadnosti) složenog izraza. Jezičnu aproksimaciju provodimo uvijek kada kao rezultat djelovanja sustava (odnosno funkcije) na ulaznu vrijednost dobijemo neku izlaznu vrijednost koju zatim želimo "shvatiti", odnosno na neki pročitati što to zapravo jest.

Sada kada smo naučili što je jezična varijabla i što su jezični modifikatori, pogledajmo jedan konkretan primjer definiranja jezične varijable "starosna dob".

---

**Primjer:** 9

Prikazati jedan poptun primjer definicije jezične varijable "starosna dob".

**Rješenje:**

Jezična varijabla je uređena petorka (definicija 23). Stoga je potrebno ponuditi definicije svih pet komponenata.

$X$ je "starosna dob".

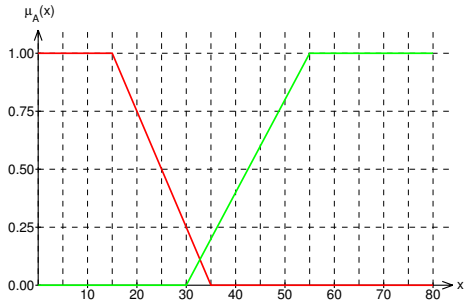
$T_X$ se sastoji od dva osnovna izraza: {*mlad*, *star*} i svih složenih izraza koji se mogu izvesti iz njih uporabom gramatike $G$ koju ćemo definirati u nastavku.

$U$ je skup cijelih nenegativnih brojeva; to je stvarna fizička domena u kojoj elementi iz $T_X$ poprimaju numeričke vrijednosti.

Gramatika $G$ je uređena četvorka $(V_T, V_N, P, S)$. Pri tome je $V_T$ skup završnih znakova (engl. *terminal*), $V_N$ skup nezavršnih znakova (engl. *non-terminal*), $P$ skup produkcijskih pravila te $S$ početni nezavršni znak. Definirajmo redom:
$V_T$={mlad, star, vrlo, i, ili, ne, (, )},
$V_N$={S, A, B, C, D, E},
$S = S$,
$P$ = {
A → **mlad** | **star** | **(**E**)**
B → **vrlo** B | A
C → **ne** C | B
D → D **i** C | C
E → E **ili** D | D
S → E
}.

$M$ definira neizrazite skupove (slika 3.3):

$$\mu_{\text{mlad}}(x) = \begin{cases} 1, & x \leq 15, \\ \frac{35-x}{35-15}, & 15 < x \leq 35, \\ 0, & x > 35. \end{cases} \qquad \mu_{\text{star}}(x) = \begin{cases} 0, & x \leq 30, \\ \frac{x-30}{55-30}, & 30 < x \leq 55, \\ 1, & x > 55. \end{cases}$$




Slika 3.3: Neizraziti skupovi "mlad" (crveno) i "star" (zeleno).

$M$ jezičnom izrazu *vrlo* pridružuje operaciju koncentracije (kvadriranje), jezičnom vezniku "i" pridružuje operaciju presjeka neizrazitih skupova, jezičnom vezniku "ili" pridružuje operaciju unije neizrazitih skupova te negaciji pridružuje operaciju komplementa neizrazitog skupa (presjek, uniju i komplement računa prema definiciji Zadeha; ZadehT, ZadehS, ZadehNot).
Ovime je u potpunosti definirana jezična varijabla "starosna dob".

---

**Primjer:** 10

Prethodni primjer definira jezičnu varijablu "starosna dob". Pokažite kako se može izvesti izraz "mlad ili vrlo star i ne vrlo vrlo star". Primjenom pravila koja propisuje $M$ izvedite neizraziti skup koji odgovara tom izrazu.

**Rješenje:**

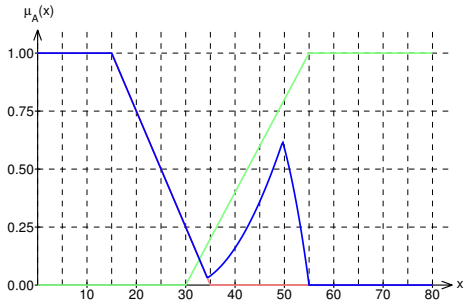
Svi izrazi izvode se pomoću gramatike $G$. Početni nezavršni znak smo definirali kao $S$, pa od tuda i krećemo, primjenjujući uzastopno pravila gramatike.

$$\begin{aligned} S &\to E \\ &\to E \textbf{ ili } D \\ &\to D \textbf{ ili } D \textbf{ i } C \\ &\to C \textbf{ ili } C \textbf{ i ne } C \\ &\to B \textbf{ ili } B \textbf{ i ne } B \\ &\to A \textbf{ ili vrlo } B \textbf{ i ne vrlo } B \\ &\to \textbf{mlad ili vrlo } A \textbf{ i ne vrlo vrlo } B \\ &\to \textbf{mlad ili vrlo star i ne vrlo vrlo } A \\ &\to \textbf{mlad ili vrlo star i ne vrlo vrlo star}. \end{aligned}$$

Da bismo izveli neizraziti skup koji odgovara ovom izrazu, potrebno je primjenjivati operacije onim redosljedom kako to diktira gramatika (jer gramatika određuje prioritete pojedinih operacija). Tako će se rezultantni neizraziti skup dobiti primjenom operacija:

*izraz = ZadehS( mlad, ZadehT( vrlo(star), ZadehNot( vrlo( vrlo(star) ) ) ) );*

Ovaj neizraziti skup ima funkciju pripadnosti koju prikazuje slika 3.4.




Slika 3.4: Funkcija pripadnosti izvedenog izraza (plavo, podebljano).

# Poglavlje 4

# Neizrazite relacije

## 4.1 Klasična relacija

Svima nam je poznata definicija klasične relacije: klasična relacija je podskup kartezijevog produkta $R \subset A \times B = \{(x, y) | x \in A \wedge y \in B\}$. Ako su skupovi $A$ i $B$ konačni, tada smo ovakve relacije mogli zapisivati i matrično, pri čemu je $i$-ti redak matrice odgovarao elementu $x_i \in A$, a $j$-ti stupac matrice odgovarao elementu $y_j \in B$. Na mjestu $R[i, j]$ nalazio se broj 1 ako su elementi $x_i$ i $y_j$ bili u relaciji ili 0 ako nisu bili.

Npr. neka su zadani univerzalni skupovi $A = \{1, 2, 3, 4\}$ i $B = \{2, 3, 4, 5, 6, 7\}$. Zadana je relacija $R$ za koju su elementi $x_i$ i $y_j$ u relaciji *akko* je $y_j$ prim-broj i to $x_i$-ti. Znači, uređeni par $(1, 2)$ je u relaciji, jer je broj 2 prvi prim broj. Uređeni par $(2, 3)$ je u relaciji jer je broj 3 drugi prim broj. Uređeni par $(1, 3)$ nije u relaciji jer broj 3 nije prvi po redu prim-broj. Zapišimo ovu relaciju kao skup uređenih parova te kao matricu.

Relaciju možemo zapisati kao skup uređenih parova na sljedeći način: $R = \{(1, 2), (2, 3), (3, 5), (4, 7)\}$. U matričnom obliku zapis relacije je:

$$R = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccc} 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}.$$

Relacija je zapisana kao matrica dimenzija $4 \times 6$ jer je kardinalni broj skupa $A$ jednak 4 a kardinalni broj skupa $B$ jednak 6.

## 4.2 Neizrazita relacija

Ako sada umjesto 0 ili 1 uvedemo stupanj s kojim uređeni par pripada relaciji (pa umjesto broja iz skupa $\{0, 1\}$ uzmemo broj iz intervala $[0, 1]$) dobili smo *neizrazitu relaciju*; dok kod klasične relacije uređeni par ili pripada relaciji

ili ne pripada, kod neizrazite relacije uređeni parovi relaciji mogu pripadati s određenom mjerom. Princip je isti kao i kod prelaska s klasičnih skupova na neizrazite skupove. Ako neizrazitu relaciju zapisujemo matrično, tada će u $i$-tom retku i $j$-tom stupcu pisati stupanj s kojim uređeni par $(x_i, y_j)$ pripada toj neizrazitoj relaciji.

Kao primjer možemo uzeti i klasičnu relaciju jednakosti $R_=$ te neizrazitu relaciju približne jednakosti $R_\approx$ nad $U \times U$, gdje je $U = \{1, 2, 3\}$.

$$R_= = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array}.$$

$$R_\approx = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 0.6 & 0.1 \\ 0.6 & 1 & 0.6 \\ 0.1 & 0.6 & 1 \end{bmatrix} \end{array}.$$

Općenito se $n$-arna neizrazita relacija definira na sljedeći način.

**Definicija 25** (Neizrazita relacija)**.** *Neka su $U_1$, $U_2$, ..., $U_n$ univerzalni skupovi. Neizrazita relacija $R$ nad $U_1 \times U_2 \times \ldots \times U_n$ definira se kao preslikavanje $\mu_R : U_1 \times U_2 \times \ldots \times U_n \to [0, 1]$.*

Binarne relacije (relacije koje su definirane nad $U_1 \times U_2$) su poopćenje pojma "funkcije"; dok funkcija svakom elementu domene pridružuje točno jedan element kodomene, relacija svakom elementu domene može pridružiti više elemenata kodomene. Sukladno tome, za binarne relacije se također definiraju pojmovi domene i kodomene.

**Definicija 26** (Domena binarne neizrazite relacije)**.** *Neka su $U_1$ i $U_2$ univerzalni skupovi i neka je $R$ neizrazita relacija definirana nad $U_1 \times U_2$. Domena (engl.* domain*) neizrazite relacije $R$ je neizraziti skup:*

$$dom(R) = \{(x; \mu_{dom(R)}(x)) | x \in U_1, \mu_{dom(R)}(x) = \max_{y \in U_2}(R(x, y))\}.$$

*Element $x$ pripada domeni relacije u mjeri koja odgovara najvećem stupnju s kojim je taj element uspostavio vezu s bilo kojim elementom kodomene.*

**Definicija 27** (Kodomena binarne neizrazite relacije)**.** *Neka su $U_1$ i $U_2$ univerzalni skupovi i neka je $R$ neizrazita relacija definirana nad $U_1 \times U_2$. Kodmena (engl.* range*) neizrazite relacije $R$ je neizraziti skup:*

$$ran(R) = \{(y; \mu_{ran(R)}(y)) | y \in U_2, \mu_{ran(R)}(y) = \max_{x \in U_1}(R(x, y))\}.$$

*Element $y$ pripada kodomeni relacije u mjeri koja odgovara najvećem stupnju s kojim je bilo koji element domene uspostavio vezu s tim elementom kodomene.*

**Definicija 28** (Visina binarne neizrazite relacije). *Neka su $U_1$ i $U_2$ univerzalni skupovi i neka je $R$ neizrazita relacija definirana nad $U_1 \times U_2$. Visina (engl.* height*) neizrazite relacije $R$ je broj:*

$$hgt(R) = \max_{x \in U_1} \max_{y \in U_2} (R(x, y)).$$

*Relaciju čija je visina jednaka* 1 *zovemo normalnom.*

Možda Vam se ova definicija visine već odnegdje čini poznata? To je stoga što je to upravo definicija visine neizrazitog skupa. Naime, iako relacije tipično zapisujemo matrično, neizrazita relacija nije ništa drugo doli običan neizraziti skup čija domena međutim nije skup "običnih" elemenata već skup uređenih parova! Stoga će se i sve operacije, poput unije, presjeka i komplementa direktno ponoviti.

Neizrazite relacije pojavljuju se u puno situacija. Za sada ćemo pogledati najjednostavniji primjer – izgradnju neizrazite relacije kartezijevnim produktom dvaju neizrazitih skupova.

**Definicija 29** (Kartezijev produkt neizrazitih skupova). *Neka je definiran neizraziti skup $A$ nad univerzalnim skupom $U_1$ te neizraziti skup $B$ nad univerzalnim skupom $U_2$. Kartezijev produkt $A \times B$ definira binarnu relaciju $R$ nad $U_1 \times U_2$ sa sljedećim svojstvom:*

$$\mu_R(x, y) = \min(\mu_A(x), \mu_B(y)) \quad \forall x \in A,\ \forall y \in B.$$

Ovako izgrađenu relaciju možemo shvatiti kao model jednostavnog sustava s jednim ulazom i jednim izlazom čija je prijenosna funkcija definirana pravilom "ako *ulaz* je $A$ tada *izlaz* je $B$". Više o modeliranju nelinearnih odnosa, neizrazitom zaključivanju i neizrazitom upravljanju govorit ćemo malo kasnije. Pogledajmo najprije uporabu kartezijevog produkta za izgradnju binarne relacije na jednostavnom primjeru.

---

**Primjer:** 11

Nad univerzalnim skupom $U = \{1, 2, 3, 4\}$ zadana su dva neizrazita: neizraziti skup $A$ predstavlja koncept "broj oko 1" dok neizraziti skup $B$ predstavlja koncept "broj oko 4".

$$A = \frac{1.0}{1} + \frac{0.7}{2} + \frac{0.3}{3} + \frac{0.1}{4} \qquad B = \frac{0.1}{1} + \frac{0.3}{2} + \frac{0.7}{3} + \frac{1.0}{4}.$$

Odredite relaciju koja se dobiva kartezijevim produktom $A$ i $B$.

**Rješenje:**

Kako su oba neizrazita skupa definirana nad univerzalnim skupom $U$, relacija će biti definirana nad univerzalnim skupom $U \times U$. Oba neizrazita skupa zapisat ćemo matrično i potom napraviti

"množenje" (umjesto umnoška, dakako, uzimat ćemo minimum).

$$\begin{aligned} R &= A^T \times B \\ &= \begin{bmatrix} 1.0 & 0.7 & 0.3 & 0.1 \end{bmatrix}^T \times \begin{bmatrix} 0.1 & 0.3 & 0.7 & 1.0 \end{bmatrix} \\ &= \begin{bmatrix} 1.0 \\ 0.7 \\ 0.3 \\ 0.1 \end{bmatrix} \times \begin{bmatrix} 0.1 & 0.3 & 0.7 & 1.0 \end{bmatrix} \\ &= \begin{bmatrix} 0.1 & 0.3 & 0.7 & 1.0 \\ 0.1 & 0.3 & 0.7 & 0.7 \\ 0.1 & 0.3 & 0.3 & 0.3 \\ 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix} \end{aligned}$$

## 4.3 Operacije nad neizrazitim relacijama

Definira se 5 osnovnih operacija s neizrazitim relacijama:

- unija,
- presjek,
- projekcija,
- cilindrično proširenje te
- kompozicija.

**Definicija 30** (Unija i presjek neizrazitih relacija)**.** *Neka su $U_1$, $U_2$, ..., $U_n$ univerzalni skupovi. Neka su definirane neizrazite relacije $A$ i $B$ nad $U_1 \times U_2 \times \ldots \times U_n$.*
*Unija neizrazitih relacija je opet neizrazita relacija nad $U_1 \times U_2 \times \ldots \times U_n$ sa svojstvom: $\mu_{A \cup B}(x_1, \ldots, x_n) = s\text{-}norma(\mu_A(x_1, \ldots, x_n), \mu_B(x_1, \ldots, x_n))$.*
*Presjek neizrazitih relacija je opet neizrazita relacija nad $U_1 \times U_2 \times \ldots \times U_n$ sa svojstvom: $\mu_{A \cap B}(x_1, \ldots, x_n) = t\text{-}norma(\mu_A(x_1, \ldots, x_n), \mu_B(x_1, \ldots, x_n))$.*

Najčešće se kao $s$-norma za uniju koristi Zadehov maksimum, a kao $t$-norma za presjek Zadehov minimum. Uočite da relacije između kojih se računa unija/presjek moraju biti definirane nad istim univerzalnim skupovima.

**Primjer:** 12

Zadane su relacije $A$ i $B$ nad univerzalnim skupom $\{1,2,3,4\} \times \{1,2,3\}$. Izračunajte njihovu uniju i presjek koristeći Zadehove definicije unije i presjeka.

$$A = \begin{bmatrix} 1.0 & 0.9 & 0.0 \\ 0.8 & 1.0 & 0.3 \\ 0.6 & 0.8 & 0.7 \\ 0.1 & 0.4 & 1.0 \end{bmatrix} \quad B = \begin{bmatrix} 0.6 & 0.1 & 0.1 \\ 0.5 & 0.7 & 0.4 \\ 0.8 & 0.9 & 0.6 \\ 0.2 & 0.3 & 0.0 \end{bmatrix}$$

**Rješenje:**

Unija se računa kao $(A \cup B)[i,j] = \max(A[i,j], B[i,j])$.

$$A \cup B = \begin{bmatrix} 1.0 & 0.9 & 0.0 \\ 0.8 & 1.0 & 0.3 \\ 0.6 & 0.8 & 0.7 \\ 0.1 & 0.4 & 1.0 \end{bmatrix} \cup \begin{bmatrix} 0.6 & 0.1 & 0.1 \\ 0.5 & 0.7 & 0.4 \\ 0.8 & 0.9 & 0.6 \\ 0.2 & 0.3 & 0.0 \end{bmatrix} = \begin{bmatrix} 1.0 & 0.9 & 0.1 \\ 0.8 & 1.0 & 0.4 \\ 0.8 & 0.9 & 0.7 \\ 0.2 & 0.4 & 1.0 \end{bmatrix}$$

Presjek se računa kao $(A \cup B)[i,j] = \min(A[i,j], B[i,j])$.

$$A \cup B = \begin{bmatrix} 1.0 & 0.9 & 0.0 \\ 0.8 & 1.0 & 0.3 \\ 0.6 & 0.8 & 0.7 \\ 0.1 & 0.4 & 1.0 \end{bmatrix} \cap \begin{bmatrix} 0.6 & 0.1 & 0.1 \\ 0.5 & 0.7 & 0.4 \\ 0.8 & 0.9 & 0.6 \\ 0.2 & 0.3 & 0.0 \end{bmatrix} = \begin{bmatrix} 0.6 & 0.1 & 0.0 \\ 0.5 & 0.7 & 0.3 \\ 0.6 & 0.8 & 0.6 \\ 0.1 & 0.3 & 0.0 \end{bmatrix}$$

---

**Definicija 31** (Projekcija neizrazite relacije). *Neka su $U_1$, ..., $U_k$, ..., $U_n$ univerzalni skupovi. Neka je definirana neizrazita relacija $A$ nad $U_1 \times \ldots \times U_k \times \ldots \times U_n$. Projekcija neizrazite relacije $A$ je opet neizrazita relacija ali nad $U_1 \times \ldots \times U_{k-1} \times U_{k+1} \times \ldots \times U_n$ sa svojstvom:*

$$\mu_{proj}(x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n) = \max_k(\mu_A(x_1, \ldots, x_{k-1}, x_k, x_{k+1}, \ldots, x_n)).$$

Postupkom projekcije smanjuje se red relacije; tako će se, primjerice, projekcijom ternarne relacije dobiti binarna relacija a projekcijom binarne relacije običan neizraziti skup.

**Definicija 32** (Cilindrično proširenje). *Neka su $U_1$, ..., $U_n$ te $U_{n+1}$ univerzalni skupovi. Neka je definirana neizrazita relacija $A$ nad $U_1 \times \ldots \times U_n$. Cilindrično proširenje neizrazite relacije $A$ je opet neizrazita relacija ali nad $U_1 \times \ldots \times U_n \times U_{n+1}$ sa svojstvom:*

$$\mu_{cil}(x_1, \ldots, x_n, x_{n+1}) = \mu_A(x_1, \ldots, x_n).$$

---

**Primjer:** 13

Zadana je relacija $A$ nad univerzalnim skupom $U_1 \times U_2 = \{1,2,3,4\} \times \{1,2,3\}$. Izračunajte njezine projekcije na $U_1$ i na $U_2$. Izračunajte cilindrična proširenja tih projekcija na univerzalni skup $U_1 \times U_2$.

$$A = \begin{bmatrix} 1.0 & 0.9 & 0.0 \\ 0.8 & 1.0 & 0.3 \\ 0.6 & 0.8 & 0.7 \\ 0.1 & 0.4 & 1.0 \end{bmatrix}$$

**Rješenje:**

Projekcija relacije $A$ na $U_1$ dobije se tako da se u svakom retku pronađe maksimum, a projekcija relacije $A$ na $U_2$ dobije se tako da se u svakom stupcu pronađe maksimum. Kako je $A$ binarna

relacije, rezultat projekcije relacije $A$ na $U_1$ je neizraziti skup definiran nad $U_1$. Element $x_i \in U_1$ tom neizrazitom skupu pripada s maksimalnom mjerom s kojom su parovi oblika $(x_i, y)\ \forall y \in U_2$ pripadali relaciji $A$. Rezultat projekcije relacije $A$ na $U_2$ je neizraziti skup definiran nad $U_2$. Element $y_j \in U_2$ tom neizrazitom skupu pripada s maksimalnom mjerom s kojom su parovi oblika $(x, y_j)\ \forall x \in U_1$ pripadali relaciji $A$.

$$A_{projU_1} = \begin{bmatrix} 1.0 \\ 1.0 \\ 0.8 \\ 1.0 \end{bmatrix} \quad A_{projU_2} = \begin{bmatrix} 1.0 & 1.0 & 1.0 \end{bmatrix}$$

Cilindrično proširenje relacije (projekcije) $A_{projU_1}$ na $U_1 \times U_2$ dobije se tako da se matrica $A_{projU_1}$ preoblikuje tako da ima onoliko stupaca koliko $U_2$ ima elemenata, i u prazna mjesta prekopira vrijednost iz prvog stupca. Ono što time radimo jest da za svaki element $x_i \in U_1$ pogledamo s kojom on mjerom pripada relaciji $A_{projU_1}$, i potom stvorimo za svaki $y_j \in U_2$ par oblika $(x_i, y_j)$ i postavimo da on cilindričnom proširenju pripada s istom mjerom s kojom je $x_i$ pripadao relaciji $A_{projU_1}$. Cilindrično proširenje projekcije $A_{projU_2}$ na $U_1 \times U_2$ dobije se tako da se matrica $A_{projU_2}$ preoblikuje tako da ima onoliko redaka koliko $U_1$ ima elemenata, i u prazna mjesta prekopira vrijednost iz prvog retka.

$$A_{cil\_projU_1 \to U_1 \times U_2} = \begin{bmatrix} 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \\ 0.8 & 0.8 & 0.8 \\ 1.0 & 1.0 & 1.0 \end{bmatrix} \quad A_{cil\_projU_2 \to U_1 \times U_2} = \begin{bmatrix} 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \end{bmatrix}$$

Zgodno je ovdje spomenuti i sljedeće pitanje. Relacija $A$ definirana na $U_1 \times U_2$ u matričnom prikazu ima $|U_1| \cdot |U_2|$ elemenata. Ako bismo takvu matricu prenosili komunikacijskim kanalom, trebali bismo prenijeti sve te elemente. Obje projekcije zajedno imaju samo $|U_1| + |U_2|$ elemenata, što je bitno manje i može se efikacije prenijeti. S obzirom da projekcije možemo cilindrično proširiti, postavlja se pitanje: može li se samo iz projekcija rekonstruirati originalna relacija? Odgovor na ovo pitanje dat ćemo uskoro.

---

**Definicija 33** (Kompozicija binarnih relacija). *Neka su $X$, $Y$ i $Z$ univerzalni skupovi. Neka je $A$ neizrazita relacija na $X \times Y$, i neka je $B$ neizrazita relacija na $Y \times Z$. Neka su odabrane jedna s-norma i jedna t-norma. Kompozicija binarnih relacija $A$ i $B$ s obzirom na odabranu s-normu i t-normu, oznaka $A \circ B$, je opet binarna relacija definirana nad $X \times Z$, sa svojstvom:*

$$\mu_{A \circ B}(x, z) = \textit{s-norma}(\textit{t-norma}(\mu_A(x, y), \mu_B(y, z))) \quad \forall x \in X, y \in Y, z \in Z.$$

*Najčešće korištene kompozicije su sup-min kompozicija, kod koje se kao s-norma koristi Zadehov maksimum (ili supremum) a kao t-norma Zadehov minimum pa vrijedi:*

$$\mu_{A \circ B}(x, z) = \sup_{y \in Y}(\min(\mu_A(x, y), \mu_B(y, z))) \quad \forall x \in X, z \in Z$$

*odnosno sup-produkt kompozicija, kod koje se kao s-norma koristi Zadehov maksimum (ili supremum) a kao t-norma algebarski produkt pa vrijedi:*

$$\mu_{A \circ B}(x, z) = \sup_{y \in Y}(\mu_A(x, y) \cdot \mu_B(y, z)) \quad \forall x \in X, z \in Z.$$

*Sup-min* kompozicija još se naziva i *max-min* kompozicija. Objašnjenje za ovakav način izračunavanja može se dobiti promatranjem što zapravo relacija dobivena kompozicijom predstavlja: komponirana relacija predstavlja direktnu vezu između parova $(x, z)$ ako smo početno znali veze između parova $(x, y)$ i $(y, z)$. Budući da treba izračunati s kojim stupnjem par $(x, z)$ pripada komponiranoj relaciji, kod *max-min* kompozicije primjenjuje se stara poslovica koja kaže da lanac puca na najslabijoj karici. Ovo možemo zamisliti na sljedeći način. Neizrazita relacija za svaki par $(\alpha, \beta)$ definira stupanj pripadnosti tog para toj relaciji. Možemo zamisliti da su element $\alpha$ i element $\beta$ povezani lancem čija je prekidna sila jednaka stupnju pripadnosti tog para relaciji. Ako se element $\alpha$ i element $\beta$ pokušaju razdvojiti, lanac koji ih povezuje održat će ih zajedno sve dok se ne primjeni sila koja je veća prekidnoj sili lanca koji ih povezuje.

Kod kompozicije imamo dvije relacije. U slučaju binarnih relacija, ako je prva relacija definirana nad $X \times Y$ a druga nad $Y \times Z$, kompozicija će biti definirana nad $X \times Z$; ta relacija uspostavlja veze između elemenata skupa $X$ i elemenata skupa $Z$. Primjenimo sada istu analogiju s prekidnim silama lanaca. Pitanje na koje treba odgovoriti jest: kolikom silom treba djelovati da bismo razdvojili konkretan element $x \in X$ od konkretnog elementa $z \in Z$? Iz relacije $A$ znamo da je $x$ povezan s nizom elemenata $y$. Iz relacije $B$ za svaki takav $y$ možemo utvrditi koliko je povezan sa $z$. Ako u skupu $Y$ imamo tri elementa $y_1$, $y_2$ i $y_3$, to znači da od konkretnog $x$ do konkretnog $z$ možemo doći preko tri puta $x \to y_1 \to z$, $x \to y_2 \to z$ te $x \to y_3 \to z$.

Svaki od tih puteva predstavlja serijski spoj dvaju lanaca: $x \to y_i$ i $y_i \to z$. Taj će spoj puknuti na prekidnoj sili slabijeg lanca. Zato se za sve parove $(x, y_i) - (y_i, z)$ traži minimum. Međutim, kako od $x$ do $z$ u ovom primjeru postoje tri paralelna puta (tri paralelna lanca), $x$ i $z$ će se držati zajedno tako dugo dok ne pukne i ona najjača veza: stoga se od svih pronađenih minimuma traži maksimum – taj maksimum odgovara jakosti veze između $x$ i $z$ i on se uzima kao stupanj pripadnosti para $(x, z)$ relaciji koja se dobije kompozicijom.

Osim *sup-min* kompozicije u praksi se koristi i *sup-produkt* kompozicija kod koje je operacija minimuma zamijenjena umnoškom.

---

**Primjer:** 14

Zadane su binarne relacije $A$ nad $X \times Y$, $B$ nad $Y \times Z$ i $C$ nad $Y$. Izračunajte *max-min* kompozicije relacija $A \circ B$ i $C \circ B$.

$$A = \begin{bmatrix} 0.1 & 0.7 & 0.5 & 0.1 \\ 0.5 & 1.0 & 0.9 & 0.4 \\ 0.2 & 0.1 & 0.6 & 0.9 \end{bmatrix} \quad B = \begin{bmatrix} 1.0 & 0.2 \\ 0.7 & 0.5 \\ 0.3 & 0.9 \\ 0.0 & 0.4 \end{bmatrix} \quad C = \begin{bmatrix} 0.7 & 0.9 & 1.0 & 0.3 \end{bmatrix}$$

**Rješenje:**

Izračun kompozicije može se zamisliti kao modificirano izvođenje množenja matrica. Matrice se množe tako da se pomnoži odgovarajući redak s odgovarajućim stupcem i rezultati se zbroje. Kompozicija se radi na sličan način: uzima se minimum elemenata (redak sa stupcem) koje bismo inače množili, a zbrajanje se zamijeni s uzimanjem maksimuma. Npr. ako množimo matrice $A$ i $B$, element u prvom retku i prvom stupcu umnoška dobio bi se tako da pomnožimo prvi redak matrice $A$ s prvim stupcem matrice $B$ i pri tome pozbrajamo umnoške:

$$0.1 \cdot 1.0 + 0.7 \cdot 0.7 + 0.5 \cdot 0.3 + 0.1 \cdot 0.0 = 0.74;$$

kompozicija će za prvi redak i prvi stupac računati:

$$\max(\min(0.1, 1.0), \min(0.7, 0.7), \min(0.5, 0.3), \min(0.1, 0.0)) = \max(0.1, 0.7, 0.3, 0.0) = 0.7.$$

Provedeno nad svim elementima, dobit će se rezultat prikazan u nastavku.

$$A \circ B = \begin{bmatrix} 0.1 & 0.7 & 0.5 & 0.1 \\ 0.5 & 1.0 & 0.9 & 0.4 \\ 0.2 & 0.1 & 0.6 & 0.9 \end{bmatrix} \circ \begin{bmatrix} 1.0 & 0.2 \\ 0.7 & 0.5 \\ 0.3 & 0.9 \\ 0.0 & 0.4 \end{bmatrix} = \begin{bmatrix} 0.7 & 0.5 \\ 0.7 & 0.9 \\ 0.3 & 0.6 \end{bmatrix}.$$

Izračunavanje $C \circ B$ može se na prvi pogled učiniti malo čudnim. Naime, $C$ je definiran samo nad $Y$, pa se možda ne vidi odmah što će biti rezultat. Da bi se razriješio problem, jednostavno treba zamisliti da je $C$ definiran nad $W \times Y$ gdje je $W$ jednočlani skup, pa je tada rezultat definiran nad $W \times Y \circ Y \times Z = W \times Z \to Z$ a postupak računanja je isti kao i u prethodnom slučaju.

$$C \circ B = \begin{bmatrix} 0.7 & 0.9 & 1.0 & 0.3 \end{bmatrix} \circ \begin{bmatrix} 1.0 & 0.2 \\ 0.7 & 0.5 \\ 0.3 & 0.9 \\ 0.0 & 0.4 \end{bmatrix} = \begin{bmatrix} 0.7 & 0.9 \end{bmatrix}.$$

---

Sljedeći jednostavan primjer pokazat će nam kako možemo kompoziciju koristiti za izvođenje jednostavnih zaključaka.

---

**Primjer:** 15

Zadana je neizrazita relacija $R$ na $U \times U$, $U = \{1, 2, 3\}$. Relacija opisuje koncept "približno za jedan veći". Definirajte dva neizrazita skupa $N_1$ i $N_2$ nad $U$, na sljedeći način. Skup $N_1$ neka opisuje koncept "točno broj 1", a skup $N_2$ neka opisuje koncept "približno broj 1". Uporabom neizrazite relacije $R$ izračunajte koji su brojevi "približno za jedan veći" od brojeva $N_1$ i $N_2$. Relacija $R$ je:

$$R = \begin{bmatrix} 0.3 & 1.0 & 0.3 \\ 0.1 & 0.3 & 1.0 \\ 0.0 & 0.1 & 0.3 \end{bmatrix}.$$

**Rješenje:**

Već smo pokazali kako treba čitati ovakve relacije. Svakom retku pridružen je jedan element iz prvog univerzalnog skupa ($U_1 \times U_2$ općenito), a svakom stupcu jedan element iz drugog univerzalnog skupa. Tako npr. relacija u drugom retku i trećem stupcu ima broj 1.0, što označava da je uređeni par $(2, 3)$ pripadnik relacije "približno za jedan veći" sa stupnjem 1.0 jer je 3 doista približno za jedan veće od 2.

Neizraziti skup koji opisuje koncept "točno broj 1" glasi:

$$N_1 = \frac{1.0}{1} + \frac{0.0}{2} + \frac{0.0}{3}$$

što matrično zapisujemo kao:

$$N_1 = \begin{bmatrix} 1.0 & 0.0 & 0.0 \end{bmatrix}.$$

Prilikom definiranja ovog skupa odabrali smo da broj 1 skupu pripada sa stupnjem pripadnosti 1.0 a brojevi 2 i 3 sa stupnjem pripadnosti 0.0. Neizraziti skup koji opisuje koncept "približno broj 1" glasi:

$$N_2 = \frac{1.0}{1} + \frac{0.4}{2} + \frac{0.1}{3}$$

što matrično zapisujemo kao:

$$N_2 = \begin{bmatrix} 1.0 & 0.4 & 0.1 \end{bmatrix}.$$

Zašto baš ovako – sjetite se, definicija je subjektivna; ako se ne slažete, slobodno redefinirajte ovaj koncept i izračunajte rezultat za vježbu. Ovako definirani skup nam govori da mu broj 1 pripada sa stupnjem pripadnosti 1.0, broj 2 sa stupnjem pripadnosti 0.4 a broj 3 sa stupnjem pripadnosti 0.1.

Sada kada imamo definirane ove skupove, i relaciju $R$, kako ćemo doći do željenog odgovora? Vrlo jednostavno – relacija $R$ nam govori u kakvom su odnosu elementi s $U_1$ i elementi s $U_2$. Ako već znamo elemente s $U_1$, tada će nam kompozicija neizrazitog skupa s $U_1$ i relacije $R$ dati upravo neizrazit skup na $U_2$. Idemo ovo izračunati.

$$N_1^* = N_1 \circ R = \begin{bmatrix} 1.0 & 0.0 & 0.0 \end{bmatrix} \circ \begin{bmatrix} 0.3 & 1.0 & 0.3 \\ 0.1 & 0.3 & 1.0 \\ 0.0 & 0.1 & 0.3 \end{bmatrix} = \begin{bmatrix} 0.3 & 1.0 & 0.3 \end{bmatrix}.$$

$$N_2^* = N_2 \circ R = \begin{bmatrix} 1.0 & 0.4 & 0.1 \end{bmatrix} \circ \begin{bmatrix} 0.3 & 1.0 & 0.3 \\ 0.1 & 0.3 & 1.0 \\ 0.0 & 0.1 & 0.3 \end{bmatrix} = \begin{bmatrix} 0.3 & 1.0 & 0.4 \end{bmatrix}.$$

Još je potrebno izvršiti jezičnu aproksimaciju kako bismo dodijelili neko značenje dobivenim rezultatima. Skup $N_1^*$ možemo pročitati kao "približno broj 2". I doista, "približno za jedan veći" broj od broja "točno broj 1" je "približno broj 2". Skup $N_2^*$ možemo pročitati kao "broj manje-više oko broja 2". Naime, ovakav rezultat je i bio za očekivati jer smo sada krenuli s brojem koji nije točno jedan, već je negdje oko 1 ("približno broj 1"). Tada će i rezultat operacije "približno za jedan veći" biti opet "približno broj 2" ali u nešto širim granicama oko broja 2, jer je i početni broj imao dosta široke granice oko broja 1. Stoga ćemo ovaj broj pročitati kao "broj manje-više oko broja 2".

---

Definirajmo još dva pojma vezana uz relacije: inverz relacije te potenciju relacije.

**Definicija 34** (Inverz relacije)**.** *Neka je zadana binarna relacije $R$ nad $U_1 \times U_2$. Relacija $R$ uspostavlja veze od elemenata univerzalnog skupa $U_1$ do elemenata univerzalnog skupa $U_2$. Inverz relacije $R$ je relacija $R^{-1}$ koja je definirana nad $U_2 \times U_1$ i koja uspostavlja veze od elemenata univerzalnog skupa $U_2$ do elemenata univerzalnog skupa $U_1$. Matrica koja odgovara relaciji $R^{-1}$ je transponirna matrica od $R$. Također, direktno slijedi da je $\left(R^{-1}\right)^{-1} = R$.*

**Definicija 35** (Potencije relacije)**.** *Neka je zadana binarna relacije $R$. Potencije relacije definiraju se uporabom kompozicije. Pri tome vrijedi: $R^i = R^{i-1} \circ R$. Primjerice, vrijedi $R^1 = R$, $R^2 = R^1 \circ R$, $R^3 = R^2 \circ R$ itd.*

## 4.4 Svojstva binarnih neizrazitih relacija

Počet ćemo s tri osnovna svojstva koja se definiraju za relacije: refleksivnost, simetričnost i tranzitivnost (te inačice).

**Definicija 36** (Refleksivnost neizrazite binarne relacije)**.** *Neka je $R$ binarna relacija definirana nad $U \times U$, pri čemu je $U$ univerzalni skup. $R$ je refleksivna ako je $\mu_R(x,x) = 1\ \forall x \in U$. Ako postoji barem jedan $x \in U$ za koji to ne vrijedi, tj. za koji je $\mu_R(x,x) \neq 1$, takva relacija je irefleksivna. Ako pak to ne vrijedi niti za jedan $x \in U$, drugim riječima ako je $\mu_R(x,x) \neq 1\ \forall x \in U$, takva relacija je antirefleksivna.*

Relacije od kojih očekujemo svojstvo refleksivnosti su relacije poput relacije ekvivalencije (očekujemo da je $x \equiv x$). S druge strane, od relacija strogog uređaja ovo svojstvo ne očekujemo (primjerice, ne očekujemo da vrijedi $x < x$ gdje je "$<$" operator stroge usporedbe).

**Definicija 37** (Simetričnost neizrazite binarne relacije)**.** *Neka je $R$ binarna relacija definirana nad $U \times U$, pri čemu je $U$ univerzalni skup. $R$ je simetrična ako je $\mu_R(x,y) = \mu_R(y,x)\ \forall x \in U, y \in U$. Ako postoji barem jedan $x$ i $y$ za koje ne vrijedi $\mu_R(x,y) = \mu_R(y,x)$, kažemo je da relacije asimetrična. Relaciju pak zovemo antisimetričnom ako za svaki $x$ i $y$ vrijedi $R(x,y) \neq R(y,x)$ ili $R(x,y) = 0 \wedge R(y,x) = 0$. Konačno, relacija $R$ je* perfektno *antisimetrična ako $\forall x \in U, \forall y \in U:\ R(x,y) > 0 \Rightarrow R(y,x) = 0$.*

I opet, simetričnost je svojstvo koje očekujemo od relacija poput relacije ekvivalencije: ako je $x \equiv y$ u nekoj mjeri, tada očekujemo i da je $y \equiv x$ u toj istoj mjeri. Kod relacija strogog uređaja ovo nikako ne očekujemo; primjerice, ako pogledamo relaciju "$<$" definiranu nad cijelim brojevima, očekujemo da joj par $(1, 100)$ pripada s visokim stupnjem (jer je 1 doista manje od 100 i to značajno), ali nikako ne očekujemo da joj s tim istim stupnjem pripada i par $(100, 1)$.

Kod neizrazitih relacija ne definira se klasična tranzitivnost, već nešto modificirani oblik.

**Definicija 38** (Max-min tranzitivnost neizrazite binarne relacije)**.** *Neka je $R$ binarna relacija definirana nad $U \times U$, pri čemu je $U$ univerzalni skup. Relacija $R$ je* max-min *tranzitivna ako*

$$\mu_R(x,z) \geq \max_{y \in U} \min(\mu_R(x,y), \mu_R(y,z))\ \forall x \in U, \forall z \in U.$$

Prethodnu definiciju moguće je poopćiti tako da se gledaju proizvoljne $s$-norme i $t$-norme. Sljedeća definicija prikazuje kako.

**Definicija 39** ($s$-$t$ tranzitivnost neizrazite binarne relacije)**.** *Neka je $R$ binarna relacija definirana nad $U \times U$, pri čemu je $U$ univerzalni skup. Neka je odabrana neka $s$-norma i neka $t$-norma. Relacija $R$ je tranzitivna s obzirom na odabrane $s$- i $t$-normu ako*

$$\mu_R(x,z) \geq s\text{-}norma_{y \in U}\, t\text{-}norma(\mu_R(x,y), \mu_R(y,z))\ \forall x \in U, \forall z \in U.$$

*U tom smislu,* max-min *tranzitivnost samo je poseban slučaj općenite tranzitivnosti, pri čemu se kao $s$-norma koristi operator maksimuma a kao $t$-norma operator minimuma.*

Ovdje je potrebno napomenuti još jedno svojstvo vezano uz kompoziciju relacije same sa sobom: nakon konačno mnogo koraka komponiranja relacije same sa sobom dobiva se relacija koja je max-min tranzitivna. Ovo svojstvo koristi se kada se pokušava izgraditi relacija ekvivalencije iz relacije koja to nije.

Pogledajmo sada neke tipične neizrazite relacije i njihove klasične pandane.

### 4.4.1 Relacija ekvivalencije i neizrazita relacija ekvivalencije

*Klasična relacija ekvivalencije* definirana nad kartezijevim produktom $U \times U$ klasičnog skupa $U$ je svaka klasična relacija koja je refleksivna, simetrična i tranzitivna. Za svaki element $u \in U$ može se izgraditi skup $P_u$ koji je podskup od $U$ i koji sadrži sve elemente iz $U$ koji su u relaciji ekvivalencije s $u$:

$$P_u = \{v \mid R(u, v) = 1 \wedge y \in U\}.$$

Element $u$ bit će u tom skupu, i niti jedan element tog skupa neće biti u relaciji niti s jednim drugom elementom iz $U$ koji nije u $P_u$. Ovaj skup naziva se razred ekvivalencije relacije ekvivalencije s obzirom na element $u$. Dva razreda ekvivalencije ili su jednaki ili disjunktni. Skup svih porodica razreda ekvivalencije čini particiju skupa $U$ (disjunktni su a unija im je jednaka čitavom skupu $U$) i obično se označava s $U/R$.

*Neizrazita relacija ekvivalencije* (ili relacija sličnosti) je neizrazita relacija koja je refleksivna, simetrična i tranzitivna (najčešće se misli na *max-min* tranzitivnost, ali može biti bilo koji oblik). $\alpha$-presjeci relacije sličnosti su relacije ekvivalencije. Binarne relacije koje su simetrične i tranzitivne ali nisu refleksivne nazivaju se *kvazi-relacije ekvivalentnosti.*

Relacija sličnosti ne generira direktno particiju skupa $U$ već omogućava podesivo generiranje particija ovisno o tome koliku minimalnu mjeru sličnosti tražimo da bismo elemente grupirali zajedno. Pogledajmo to na sljedećem primjeru.

---

**Primjer:** 16

U autosalonu posjetitelje su pitali da procjene od četiri ponuđena automobila koji je automobil koliko sličan drugim automobilima (na skali od 0 do 1 pri čemu 0 znači da uopće nisu slični a 1 da su jako slični). Temeljem te ankete dobiveni su rezultati koji su prikazani relacijom sličnosti $R$ izgrađenom nad univerzalnim skupom $U = \{A_1, A_2, A_3, A_4\}$ gdje su $A_1, \ldots, A_4$ automobili. Kako sve skup automobila možemo grupirati prema sličnosti?

$$R = \begin{bmatrix} 1.0 & 0.0 & 0.8 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.6 \\ 0.8 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.6 & 0.0 & 1.0 \end{bmatrix}.$$

**Rješenje:**

Jedna mogućnost jest zahtjevati da dva automobila pripadaju u isti skup samo ako im je mjera sličnosti 1. U tom slučaju potrebno je napraviti $\alpha$-presjek relacije sličnosti za $\alpha = 1$:

$$R_{\alpha=1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Relacija $R_{\alpha=1}$ je klasična relacija, i ona jest relacija ekvivalencije. Razredi ekvivalencije koje definira ova relacija su:

$$\begin{aligned} U_{A_1} &= \{A_1\} \\ U_{A_2} &= \{A_2\} \\ U_{A_3} &= \{A_3\} \\ U_{A_4} &= \{A_4\} \end{aligned}$$

Dobili smo četiri porodice razreda ekvivalencije i $U/R_{\alpha=1}$ možemo zapisati kao $\{A_1\} \cup \{A_2\} \cup \{A_3\} \cup \{A_4\}$. Kako vidimo, ništa se nije grupiralo niti s čime.

Pokušajmo međutim biti manje zahtjevni. Dopustimo da u isti skup sličnih automobila upadnu svi oni automobili koji su barem slični s mjerom 0.8. Računamo stoga $\alpha$-presjek relacije sličnosti za $\alpha = 0.8$:

$$R_{\alpha=0.8} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Relacija $R_{\alpha=0.8}$ je klasična relacija, i ona je relacija ekvivalencije. Razredi ekvivalencije koje definira ova relacija su:

$$\begin{aligned} U_{A_1} &= \{A_1, A_3\} \\ U_{A_2} &= \{A_2\} \\ U_{A_3} &= \{A_3, A_1\} \\ U_{A_4} &= \{A_4\} \end{aligned}$$

Ovaj puta smo dobili samo tri porodice razreda ekvivalencije i $U/R_{\alpha=0.8}$ možemo zapisati kao $\{A_1, A_3\} \cup \{A_2\} \cup \{A_4\}$. Automobil $A_1$ grupiran je zajedno s automobilom $A_3$, dok su preostali automobili ostali u zasebnim grupama.

Smanjimo kriterije još malo. Dopustimo da u isti skup sličnih automobila upadnu svi oni automobili koji su barem slični s mjerom 0.7. Računamo stoga $\alpha$-presjek relacije sličnosti za $\alpha = 0.7$:

$$R_{\alpha=0.7} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}.$$

Relacija $R_{\alpha=0.7}$ je klasična relacija, i ona je relacija ekvivalencije. Razredi ekvivalencije koje definira ova relacija su:

$$\begin{aligned} U_{A_1} &= \{A_1, A_3\} \\ U_{A_2} &= \{A_2, A_4\} \\ U_{A_3} &= \{A_3, A_1\} \\ U_{A_4} &= \{A_4, A_2\} \end{aligned}$$

Ovaj puta smo dobili samo dvije porodice razreda ekvivalencije i $U/R_{\alpha=0.7}$ možemo zapisati kao $\{A_1, A_3\} \cup \{A_2, A_4\}$. Automobili $A_1$ i $A_3$ upali su u jednu grupu a automobili $A_2$ i $A_4$ u drugu. U praksi, ovakvo podesivo grupiranje često može biti vrlo koristan alat pri različitim analizama i kao pomoć u postupcima donošenja odluka.

---

Pogledajmo još jedan primjer.

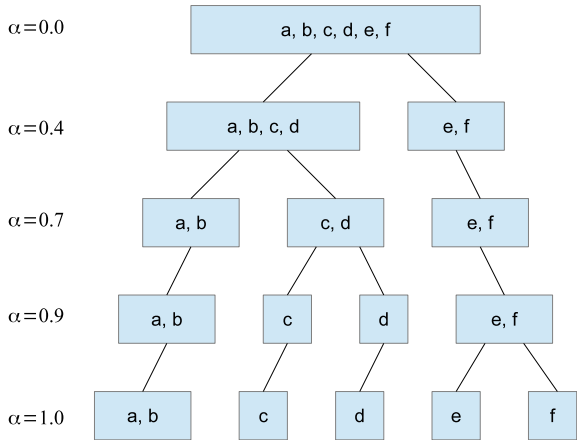
**Primjer:** 17

Kako bi se kupce u trgovačkom centru motiviralo da više kupuju, uprava trgovine napravila je analizu sedam proizvoda s ciljem ustanovljavanja koji se proizvodi često kupuju zajedno. Ideja je, dakako, takve proizvode postaviti jedne uz druge tako da, kada kupac potraži jedan od njih, po navici uzme i drugi, čak i ako nije došao po njega. U kojoj mjeri se svaki od proizvoda kupuje zajedno s drugim proizvodima prikazano je u matrici $R$ koja je definirana nad univerzalnim skupom od sedan proizvoda: $U = \{a, b, c, d, e, f\}$. Ispalo je da je prikazana relacija simetrična, refleksivna i tranzitivna. Kakve grupacije proizvoda se mogu dobiti iz te relacije?

$$R = \begin{bmatrix} 1.0 & 1.0 & 0.4 & 0.4 & 0.0 & 0.0 \\ 1.0 & 1.0 & 0.4 & 0.4 & 0.0 & 0.0 \\ 0.4 & 0.4 & 1.0 & 0.7 & 0.0 & 0.0 \\ 0.4 & 0.4 & 0.7 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.9 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.9 & 1.0 \end{bmatrix}.$$

**Rješenje:**

Potrebno je napraviti $\alpha$-presjeke relacije $R$ za vrijednosti $\alpha = 1.0$, $\alpha = 0.9$, $\alpha = 0.7$, $\alpha = 0.4$ i $\alpha = 0.0$. Ovo ostavljamo čitatelju za vježbu. Rezultat ćemo međutim prikazati grafički.




Za $\alpha = 1$ samo su proizvodi $a$ i $b$ grupirani zajedno, dok su svi ostali proizvodi grupa za sebe. Za $\alpha = 0.9$ pojavljuju se dvije dvočlane grupe: $\{a, b\}$ i $\{e, f\}$. Za $\alpha = 0.7$ proizvodi $c$ i $d$ se združuju pa imamo tri dvočlane grupe: $\{a, b\}$, $\{c, d\}$ i $\{e, f\}$. Za $\alpha = 0.4$ prve dvije grupe se spajaju tako da imamo jednu četveročlanu grupu $\{a, b, c, d\}$ i jednu dvočlanu grupu $\{e, f\}$. Konačno, uz $\alpha = 0.0$ svi proizvodi se združuju u jednu grupu.

### 4.4.2 Relacija kompatibilnosti i neizrazita relacija kompatibilnosti

Klasične binarne relacije koje su refleksivne i simetrične nazivamo relacijama kompatibilnosti ili relacijama tolerancije. Neizrazite binarne relacije koje su refleksivne i simetrične nazivamo neizrazitim relacijama kompatibilnosti ili neizrazitim relacijama tolerancije ili ponekad i relacijama blizine (engl. *proximity relations*).

Slično kao kod klasične relacije ekvivalencije, kod klasične relacije kompatibilnosti definira razred kompatibilnosti. Neka je binarna relacija $R$ definirana nad $U \times U$. Razred kompatibilnosti $P$ je podskup od $U$ takav da $\forall x \in P, \forall y \in P : \ R(x,y) = 1$; drugim riječima, svaka dva elementa iz razreda kompatibilnosti međusobno su kompatibilna. Razreda kompatibilnosti može biti mnogo. Posebno zanimljivi su maksimalni razredi kompatibilnosti: to su razredi kompatibilnosti koji nisu pravi podskup niti jednog drugog razreda kompatibilnosti – stoga naziv maksimalni. Ovi razredi ne čine particiju skupa $U$ jer ne moraju biti disjunktni (različiti razredi kompatibilnosti mogu sadržavati neke iste elemente). Skup svih maksimalnih razreda kompatibilnosti naziva se *potpuno prekrivanje* od $U$ inducirano s $R$.

Kod neizrazite relacije kompatibilnosti definira se razred $\alpha$-kompatibilnosti $P_\alpha$ (gdje je $\alpha$ minimalni stupanj pripadnosti) kao podskup od $U$ takav da $\forall x \in P_\alpha, \forall y \in P_\alpha : \ R(x,y) \geq \alpha$; drugim riječima, svaka dva elementa iz razreda $\alpha$-kompatibilnosti međusobno su kompatibilna barem s mjerom $\alpha$. $\alpha$-presjek neizrazite relacije kompatibilnosti je klasična relacija kompatibilnosti. Stoga se razred $\alpha$-kompatibilnosti neizrazite relacije kompatibilnosti može definirati i kao razred kompatibilnosti $\alpha$-presjeka neizrazite relacije kompatibilnosti. Pogledajmo primjer.

---

**Primjer:** 18

Nad univerzalnim skupom $U = \{a, b, c, d, e, f\}$ definirana je relacija $R$ kako je prikazano u nastavku. Izračunajte sva potpuna $\alpha$-prekrivanja.

$$R = \begin{bmatrix} 1.0 & 1.0 & 0.4 & 0.4 & 0.0 & 0.0 \\ 1.0 & 1.0 & 0.4 & 0.4 & 0.0 & 0.0 \\ 0.4 & 0.4 & 1.0 & 0.7 & 0.0 & 0.0 \\ 0.4 & 0.4 & 0.7 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.9 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.9 & 1.0 \end{bmatrix}.$$

**Rješenje:**

U relaciji $R$ imamo četiri različita stupnja pripadnosti; stoga ćemo raditi četiri $\alpha$-presjeka (za 1.0, 0.9, 0.5 i 0.0).

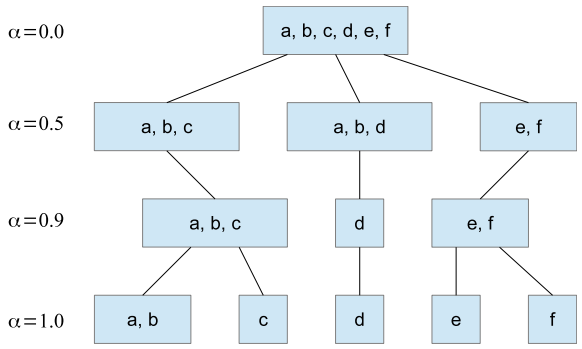
$$R_{\alpha=1.0} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

$$R_{\alpha=0.9} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

$$R_{\alpha=0.5} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

$$R_{\alpha=0.0} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Relacija kompatibilnosti $R_{\alpha=1.0}$ generira potpuno prekrivanje $\{a,b\} \cup \{c\} \cup \{d\} \cup \{e\} \cup \{f\}$. Relacija kompatibilnosti $R_{\alpha=0.9}$ generira potpuno prekrivanje $\{a,b,c\} \cup \{d\} \cup \{e,f\}$. Relacija kompatibilnosti $R_{\alpha=0.5}$ generira potpuno prekrivanje $\{a,b,c\} \cup \{a,b,d\} \cup \{e,f\}$. Konačno, relacija kompatibilnosti $R_{\alpha=0.0}$ generira potpuno prekrivanje $\{a,b,c,d,e,f\}$. Ovo je grafički prikazano na slici u nastavku.




### 4.4.3 Relacije uređaja

Svaka neizrazita relacija koja je refleksivna i tranzitivna naziva se neizrazita pred-uređajna relacija (engl. *Fuzzy pre-order relation*); ovdje se najčešće misli na *max-min* tranzitivnost (ali može se raditi i s drugim tranzitivnostima).

Svaka neizrazita relacija koja je refleksivna, antisimetrična i tranzitivna, naziva se neizrazita uređajna relacija (engl. *fuzzy order relation*). Klasična se relacija $R^*$ koja odgovara neizrazitoj uređajnoj relaciji $R$ može dobiti tako da se vrijednost karakteristične funkcije postavi na sljedeći način:

- ako je $\mu_R(x,y) \geq \mu_R(y,x)$, postavi $\mu_{R^*}(x,y) = 1$ i $\mu_{R^*}(y,x) = 0$;
- ako je $\mu_R(x,y) = \mu_R(y,x)$, postavi $\mu_{R^*}(x,y) = \mu_{R^*}(y,x) = 0$.

Primjer relacija $R$ definirane nad $U = \{a,b,c,d\}$ koja je refleksivna, antisimetrična i max-min tranzitivna prikazan je u nastavku.

$$R = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.1 & 0.0 \\ 0.4 & 0.9 & 1.0 & 0.3 \\ 0.5 & 0.0 & 0.0 & 1.0 \end{bmatrix}.$$

Odgovarajuća klasična relacija je:

$$R_* = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

Uočimo da se u prethodno definiranoj neizrazitoj relaciji $R$ (kao i njezinom klasičnom pandanu $R^*$) svi elementi međusobno ne daju usporediti (odnosno nisu u relaciji). Tako vidimo da elemente $a$ i $d$ možemo usporediti; $R(a,d) = 0$ ali $R(d,a) = 0.5$. S druge pak strane, elemente $a$ i $b$ ova relacija ne može usporediti: $R(a,b) = R(b,a) = 0$; niti uređeni par $(a,b)$ pripada relaciji, niti uređeni par $(b,a)$. Ovakve relacije stoga zovemo *relacija parcijalnog (ili djelomičnog) uređaja* (engl. *partial order relation*).

*Relacija potpunog uređaja* (ili linearnog uređaja) je svaka relacija uređaja (dakle, refleksivna, antisimetrična i tranzitivna) kod koje su svaka dva elementa usporediva: nije istina da postoji $x \in U, y \in U$ takav da je $R(x,y) = 0 \wedge R(y,x) = 0$.

## 4.5 Zatvaranje binarne relacije

Uz binarne relacije vezan je još i pojam "zatvaranja" s obzirom na neko svojstvo (engleski termin je *closure*). Tako se pojavljuju pojmovi poput refleksivnog zatvaranja relacije, simetričnog zatvaranja relacije, i možda najčešće: tranzitivnog zatvaranja relacije. Idemo stoga definirati te pojmove.

**Definicija 40** (Zatvaranje relacija)**.** *Neka je zadana binarna relacije $R$.* Zatvaranje *te relacije s obzirom na svojstvo $\xi$ je najmanje proširenje te relacije uz koje se postiže zadano svojstvo $\xi$.*

**Definicija 41** (Refleksivno zatvaranje relacije)**.** *Neka je zadana relacija $R$. Refleksivno zatvaranje relacije $R$ je relacija $R'$ ako i samo ako:*

1. *$R'$ je refleksivna relacija,*
2. *$R \subseteq R'$ te*
3. *za svaku relaciju $R''$, ako je $R \subseteq R''$ i ako je $R''$ također refleksivna, tada je nužno $R' \subseteq R''$; drugim riječima: $R'$ je najmanja relacija koja zadovoljava zahtjeve (1) i (2).*

Na identičan se način definiraju i simetrično zatvaranje relacije te tranzitivno zatvaranje relacije. Refleksivno zatvaranje relacije $R$ još ćemo označavati s $r(R)$, simetrično zatvaranje sa $s(R)$ a tranzitivno zatvaranje sa $t(R)$. Uz ove oznake sada možemo definirati i svojstva ovih zatvaranja.

1. Refleksivno zatvaranje: $r(R) = R \cup E$, gdje je $E$ relacija jednakosti (jedinice po dijagonali, svi ostali elementi su nula).

2. Simetrično zatvaranje: $s(R) = R \cup R^{-1}$ gdje je $R^{-1}$ inverzna relacija.

3. Tranzitivno zatvaranje: $t(R) = \cup_{i=1}^{\infty} R^i$. Ako je univerzalni skup $U$ nad kojim je definirana relacija konačan (pa ima kardinalitet $|U|$), vrijedi: $t(R) = \cup_{i=1}^{|U|} R^i$.

4. Relacija $R$ je refleksivna ako vrijedi $r(R) = R$.

5. Relacija $R$ je simetrična ako vrijedi $s(R) = R$.

6. Relacija $R$ je tranzitivna ako vrijedi $t(R) = R$.

Ako imamo neizrazitu relaciju kompatibilnosti (refleksivna i simetrična) definiranu nad konačnim univerzalnim skupom kardinaliteta $|U|$, kompozicija relacije $R$ sa samom sobom u najviše $|U| - 1$ koraka će dati relaciju koja ima i svojstvo tranzitivnosti, čime će takva relacija postati relacija sličnosti. Ovo se u praksi može zgodno iskoristiti. Pogledajmo to na primjeru.

---

**Primjer:** 19

Turistička zajednica želi izraditi web-sustav koji će turistima za svaku lokaciju koju odaberu predlagati i slične lokacije. Kako bi to ostvarili, u pet gradova su anketirali turiste vrlo jednostavnom anketom. Anketa se sastojala od jednog pitanja s četiri ponuđene opcije; pitanje je bilo formulirano na sljedeći način: "koja od ovih četiri opcije najbolje opisuje ovu lokaciju?". Prilikom obrade ankete za svaku je lokaciju i za svaku opciju izračunat postotak turista koji su odabrali tu opciju. Rezultati su prikazani u sljedećoj tablici.

| Lokacija | Opcija 1 | Opcija 2 | Opcija 3 | Opcija 4 | $\sum$ |
|---|---|---|---|---|---|
| $L_1$ | 0.1 | 0.1 | 0.4 | 0.4 | 1 |
| $L_2$ | 0.5 | 0.1 | 0.3 | 0.1 | 1 |
| $L_3$ | 0.4 | 0.1 | 0.4 | 0.1 | 1 |
| $L_4$ | 0.4 | 0.1 | 0.1 | 0.4 | 1 |
| $L_5$ | 0.1 | 0.3 | 0.3 | 0.3 | 1 |

Pojasnite kako biste temeljem tih podataka omogućili grupiranje lokacija?

**Rješenje:**

Na raspolaganju su nam podatci za 5 gradova. Temeljem tih podataka najprije možemo izgraditi relaciju kompatibilnosti koja se temelji na nekoj od mjera sličnosti. Naime, uočimo da podatke za svaku lokaciju možemo shvatiti kao 4-dimenzijski vektor realnih brojeva: $L_1 = [0.1 \quad 0.1 \quad 0.4 \quad 0.4]$, $L_2 = [0.5 \quad 0.1 \quad 0.3 \quad 0.1]$ i tako dalje. Sličnost između dva vektora možemo izgraditi izračunom kosinusa kuta između njih: vektori koji su jako različiti (tj. okomiti) imat će mjeru sličnosti 0, vektori koji su identični imat će mjeru sličnosti 1. Ako su $L_i$ i $L_j$ dva vektora, kosinus kuta između njih definiran je izrazom:

$$\cos(L_i, L_j) = \frac{L_i \cdot L_j}{\|L_i\| \cdot \|L_j\|}$$

odnosno kao omjer njihovog skalarnog produkta i umnoška njihovih normi.

Prvi korak je stoga izgraditi relaciju koja govori koliko su svake dvije lokacije međusobno slične odnosno kompatibilne. Uočite da će to biti relacija definirana nad univerzalnim skupom $L \times L$ gdje je $L = \{L_1, L_2, L_3, L_4, L_5\}$, i za koju će vrijediti:

$$\mu_R(l_i, l_j) = \cos(L_i, L_j).$$

Relacija koja se time dobiva jest:

$$R = \begin{bmatrix} 1.000 & 0.629 & 0.735 & 0.735 & 0.907 \\ 0.629 & 1.000 & 0.972 & 0.800 & 0.630 \\ 0.735 & 0.972 & 1.000 & 0.735 & 0.713 \\ 0.735 & 0.800 & 0.735 & 1.000 & 0.713 \\ 0.907 & 0.630 & 0.713 & 0.713 & 1.000 \end{bmatrix}$$

Uočite da je ova relacija, po definiciji, refleksivna i simetrična, jer je to svojstvo kosinusa između dva vektora. Stoga je ta relacija doista relacija kompatibilnosti.

Za podesivo grupiranje trebali smo relaciju ekvivalencije. Relacija ekvivalencije uz svojstvo refleksivnosti i simetričnosti zahtjeva i tranzitivnost; razmatrat ćemo max-min tranzitivnost. Relacija $R$, nažalost, nije max-min tranzitivna: to se vidi već iz para $(L_1, L_2)$ za koji tranzitivnost ne vrijedi. Idemo stoga izračunati relaciju $R^2 = R \circ R$:

$$R^2 = R \circ R = \begin{bmatrix} 1.000 & 0.735 & 0.735 & 0.735 & 0.907 \\ 0.735 & 1.000 & 0.972 & 0.800 & 0.713 \\ 0.735 & 0.972 & 1.000 & 0.800 & 0.735 \\ 0.735 & 0.800 & 0.800 & 1.000 & 0.735 \\ 0.907 & 0.713 & 0.735 & 0.735 & 1.000 \end{bmatrix}.$$

Relacija $R^2$ i dalje nije tranzitivna; primjerice, problem je par $(L_2, L_5)$. Napravimo još jednu kompoziciju, tj. izračunajmo $R^3$:

$$R^3 = R^2 \circ R = \begin{bmatrix} 1.000 & 0.735 & 0.735 & 0.735 & 0.907 \\ 0.735 & 1.000 & 0.972 & 0.800 & 0.735 \\ 0.735 & 0.972 & 1.000 & 0.800 & 0.735 \\ 0.735 & 0.800 & 0.800 & 1.000 & 0.735 \\ 0.907 & 0.735 & 0.735 & 0.735 & 1.000 \end{bmatrix}.$$

Relacija koju smo dobili je tranzitivna relacija. Stoga je $R^3$ koju ćemo dalje u primjeru označavati s $\tilde{R}$ neizrazita relacija ekvivalencije. I dalje je jednostavno: prisjetimo se – $\alpha$-presjek neizrazite relacije ekvivalencije je klasična relacija ekvivalencija koja generira particiju univerzalnog skupa, a time i grupiranje elemenata u razrede ekvivalencije.

Kako u relaciji imamo pet različitih vrijednosti: 0.735, 0.800, 0.907, 0.972 te 1.000, možemo napraviti pet $\alpha$-presjeka (čime dobivamo pet klasičnih relacija ekvivalencije) i pet particija skupa gradova. $\alpha$-presjeci i pripadne particije navedeni su u nastavku.

$$\tilde{R}_{\alpha=0.735} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Particija $L/\tilde{R}_{\alpha=0.735}$ je $\{\{L_1, L_2, L_3, L_4, L_5\}\}$.

$$\tilde{R}_{\alpha=0.800} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Particija $L/\tilde{R}_{\alpha=0.800}$ je $\{\{L_1, L_5\}, \{L_2, L_3, L_4\}\}$.

$$\tilde{R}_{\alpha=0.907} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Particija $L/\tilde{R}_{\alpha=0.907}$ je $\{\{L_1, L_5\}, \{L_2, L_3\}, \{L_4\}\}$.

$$\tilde{R}_{\alpha=0.972} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Particija $L/\tilde{R}_{\alpha=0.972}$ je $\{\{L_1\}, \{L_2, L_3\}, \{L_4\}, \{L_5\}\}$.

$$\tilde{R}_{\alpha=1.000} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Particija $L/\tilde{R}_{\alpha=1.000}$ je $\{\{L_1\}, \{L_2\}, \{L_3\}, \{L_4\}, \{L_5\}\}$.

## 4.6 Neinteraktivnost binarne relacije

U jednom od prethodnih primjera pitali smo se je li moguće izvornu relaciju rekonstruirati iz njezinih projekcija. Odgovor na to pitanje daje pojam neinteraktivnosti relacije.

**Definicija 42** (Neinteraktivnost binarne relacije)**.** *Neka je $R$ binarna relacija definirana nad $U_1 \times U_2$, pri čemu su $U_1$ i $U_2$ univerzalni skupovi. Neka je $X$ projekcija relacije $R$ na $U_1$ a $Y$ projekcija relacije $R$ na $U_2$. Relacija $R$ je neinteraktivna ukoliko vrijedi: $X \times Y = R$. Drugi način da ovo kažemo jest: relacija je neinteraktivna ako je* spoj *svojih projekcija: ako se $R$ dobije upravo kao presjek cilindričnih proširenja projekcija. Relacija je interaktivna ukoliko nije neinteraktivna.*

**Primjer:** 20

Utvrdite je li neizrazita relacija $R$ neinteraktivna.

$$R = \begin{bmatrix} 1.0 & 0.7 & 0.2 \\ 0.7 & 1.0 & 0.3 \\ 0.3 & 0.6 & 0.8 \\ 0.0 & 0.2 & 1.0 \end{bmatrix}.$$

**Rješenje:**

Najprije je potrebno utvrditi projekcije ove relacije.

$$R_{projU_1} = \begin{bmatrix} 1.0 \\ 1.0 \\ 0.8 \\ 1.0 \end{bmatrix} \quad R_{projU_2} = \begin{bmatrix} 1.0 & 1.0 & 1.0 \end{bmatrix}$$

Potom računamo presjek cilindričnih proširenja, što je ekvivalentno izračunu kartezijevog produkta:

$$R_{projU_1} \times R_{projU_2} = \begin{bmatrix} 1.0 \\ 1.0 \\ 0.8 \\ 1.0 \end{bmatrix} \times \begin{bmatrix} 1.0 & 1.0 & 1.0 \end{bmatrix} = \begin{bmatrix} 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \\ 0.8 & 0.8 & 0.8 \\ 1.0 & 1.0 & 1.0 \end{bmatrix}$$

Budući da je spoj projekcija relacije različit od izvorne relacije, relacija nije neinteraktivna, odnosno relacije je interaktivna.

Napomenimo i da je računanje kartezijevog produkta jednako računanju spoja cilindičnih proširenja projekcija (pri čemu *spoj* podrazumijeva operaciju presjeka). Pokažimo to i uvjerimo se da je rezultat jednak.

$$\begin{aligned} R_{cil\_projU_1 \to U_1 \times U_2} \cap R_{cil\_projU_2 \to U_1 \times U_2} &= \begin{bmatrix} 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \\ 0.8 & 0.8 & 0.8 \\ 1.0 & 1.0 & 1.0 \end{bmatrix} \cap \begin{bmatrix} 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \end{bmatrix} \\ &= \begin{bmatrix} 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 \\ 0.8 & 0.8 & 0.8 \\ 1.0 & 1.0 & 1.0 \end{bmatrix} \end{aligned}$$

---

Pogledajmo još jedan primjer.

---

**Primjer:** 21

Utvrdite je li neizrazita relacija $R$ neinteraktivna.

$$R = \begin{bmatrix} 0.7 & 0.5 & 0.7 & 0.2 \\ 0.9 & 0.5 & 1.0 & 0.2 \\ 0.5 & 0.5 & 0.5 & 0.2 \\ 0.4 & 0.4 & 0.4 & 0.2 \end{bmatrix}.$$

**Rješenje:**

Najprije je potrebno utvrditi projekcije ove relacije.

$$R_{projU_1} = \begin{bmatrix} 0.7 \\ 1.0 \\ 0.5 \\ 0.4 \end{bmatrix} \quad R_{projU_2} = \begin{bmatrix} 0.9 & 0.5 & 1.0 & 0.2 \end{bmatrix}$$

Potom računamo presjek cilindričnih proširenja, što je ekvivalentno izračunu kartezijevog produkta:

$$R_{projU_1} \times R_{projU_2} = \begin{bmatrix} 0.7 \\ 1.0 \\ 0.5 \\ 0.4 \end{bmatrix} \times \begin{bmatrix} 0.9 & 0.5 & 1.0 & 0.2 \end{bmatrix} = \begin{bmatrix} 0.7 & 0.5 & 0.7 & 0.2 \\ 0.9 & 0.5 & 1.0 & 0.2 \\ 0.5 & 0.5 & 0.5 & 0.2 \\ 0.4 & 0.4 & 0.4 & 0.2 \end{bmatrix} = R$$

Budući da je spoj projekcija relacije jednak izvornoj relaciji, relacija je neinteraktivna.

---

# Poglavlje 5

# Neizrazito zaključivanje i upravljanje

Jedna od temeljnih primjena neizrazite logike, neizrazitih skupova i neizrazitih relacija je pomoć u modeliranju složenih sustava te izrada sustava neizrazitog upravljanja. Stoga ćemo u ovom poglavlju zaokružiti sve što smo do sada obradili i pogledati konkretnu primjenu.

## 5.1 Modeliranje nelinearnih sustava

Relacije se mogu iskoristiti za modeliranje funkcijskog preslikavanja; štoviše, već smo i rekli da je relacija poopćenje pojma funkcija, jer omogućava da se jednom elementu domene pridruži više od jednog elementa kodomene. Neizrazite relacije tu su posebno pogodne jer omogućavaju da se svakoj uspostavljenoj vezi pridruži mjera s kojom ta veza vrijedi.

Pretpostavimo da smo mjerili ulazno-izlaznu nekog sustava koji je, što se nas tiče, crna kutija o kojoj ne znamo nikakve detalje. Mjerenja smo radili za 5 ulaznih vrijednosti napona i dobili smo rezultate koje prikazuje sljedeća tablica.

| Ulazni napon | 1V | 2V | 3V | 4V | 5V |
|---|---|---|---|---|---|
| Izlazni napon | 1V | 3V | 4V | 2V | 1V |

Ova tablica predstavlja prijenosnu karakteristiku sustava i sastoji se od pet parova (*ulazni napon*, *izlazni napon*): $(1,1)$, $(2,3)$, $(3,4)$, $(4,2)$ te $(5,1)$. Kako su vrijednosti ulaznog napona cijeli brojevi od 1 do 5, definirajmo s $U_u = \{1,2,3,4,5\}$ univerzalni skup koji definira moguće ulazne napone. Vrijednosti izlaznog napona se kreću slično, pa ćemo s $U_i = \{1,2,3,4,5\}$ označiti univerzalni skup koji definira moguće izlazne napone.

Sada možemo definirati relaciju $R$ definiranu nad $U_u \times U_i$ koja predstavlja prijenosnu karakteristiku i radi upravo preslikavanje koje je definirano tablicom odnosno prethodno navedenim parovima *ulaz-izlaz*; svaki navedeni par

relaciji će pripadati sa stupnjem pripadnosti 1. Time ćemo dobiti relaciju:

$$R = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

Uporabom ove relacije za bilo koji ulazni napon možemo izračunati izlazni napon uporabom kompozicije. Primjerice, neka smo na ulaz doveli napon od $3V$. Ovaj napon treba predstaviti neizrazitim skupom definiranim nad univerzalnim skupom $U_u$. Kako znamo da je ulazni napon točno $3V$, predstavit ćemo ga neizrazitim skupom $X$:

$$X = \frac{0.0}{1V} + \frac{0.0}{2V} + \frac{1.0}{3V} + \frac{0.0}{4V} + \frac{0.0}{5V}$$

Izračunajmo sada očekivani izlazni napon kompozicijom $Y = X \circ R$:

$$\begin{aligned} Y = X \circ R &= \begin{bmatrix} 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \end{bmatrix} \circ \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix} \\ &= \begin{bmatrix} 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \end{bmatrix} \\ &= \frac{0.0}{1V} + \frac{0.0}{2V} + \frac{0.0}{3V} + \frac{1.0}{4V} + \frac{0.0}{5V} \end{aligned}$$

Kako neizrazitom skupu $Y$ koji predstavlja vrijednost izlaznog napona pripada samo jedan element s mjerom pripadnosti većom od 0, taj ćemo element proglasiti izlaznim naponom, i to je, baš u skladu s očekivanjima, napon od $4V$. Na isti način možete provjeriti i za sve ostale ulazne napone – kompozicijom s relacijom $R$ dobit ćemo upravo očekivane izlazne napone (napravite to za vježbu).

Prikazani primjer ilustrira jednostavan scenarij gdje nam neizraziti skupovi praktički nisu niti trebali – sve vrijednosti funkcija pripadnosti bile su ili 0 ili 1 pa su u tom smislu to bile klasične relacije i klasični skupovi. Neizrazitost možemo iskoristiti ako postoji mjerna nesigurnost. Pretpostavimo da ulazni napon za promatrani sustav podešavamo preciznim naponskim generatorom, no izlazni napon mjerimo analognim voltmetrom koji je dosta neprecizan. Tada bismo izmjerene podatke mogli prikazati kako je to napravljeno u sljedećoj tablici.

| Ulazni napon | 1V | 2V | 3V | 4V | 5V |
|---|---|---|---|---|---|
| Izlazni napon | ≈1V | ≈3V | ≈4V | ≈2V | ≈1V |

Kako napraviti relaciju koja modelira ovakvu karakteristiku? Osnovno pitanje je sljedeće: što znači da je izlazni napon približno $3V$? Taj pojam možemo modelirati neizrazitim skupom. Iskoristimo primjerice trokutastu funkciju pripadnosti s poluširinom baze trokuta iznosa 3 – ako je trokut centriran na vrijednost $c$, za vrijednost $c$ funkcija pripadnosti će biti 1.00, za vrijednost $c \pm 1$ funkcija pripadnosti će biti 0.67, za vrijednost $c \pm 2$ funkcija pripadnosti će biti 0.33 i za vrijednost $c \pm 3$ funkcija pripadnosti past će na 0.00. U tom slučaju možemo reći da je ulaznom naponu $2V$ pridružen izlazni napon modeliran neizrazitim skupom:

$$\textit{"približno } 3V\textit{"} = \frac{0.33}{1V} + \frac{0.67}{2V} + \frac{1.00}{3V} + \frac{0.67}{4V} + \frac{0.33}{5V}.$$

Stoga će u relaciji koja modelira ovakvo preslikavanje u retku koji odgovara ulaznom naponu od $2V$ u stupcima biti upisane upravo vrijednosti funkcije pripadnosti izlaznog napona "približno 3V". Relacija koja se dobije na ovaj način je:

$$R = \begin{bmatrix} 1.00 & 0.67 & 0.33 & 0.00 & 0.00 \\ 0.33 & 0.67 & 1.00 & 0.67 & 0.33 \\ 0.00 & 0.33 & 0.67 & 1.00 & 0.67 \\ 0.67 & 1.00 & 0.67 & 0.33 & 0.00 \\ 1.00 & 0.67 & 0.33 & 0.00 & 0.00 \end{bmatrix}.$$

Utvrđivanje odnosa ulaz-izlaz i dalje se provodi na isti način. Ako znamo ulazni napon, izlazni ćemo dobiti kompozicijom ulaznog napona i relacije koja predstavlja prijenosnu karakteristiku. Provjerimo to na primjeru ulaznog napona $X \equiv 4V$:

$$\begin{aligned} Y = X \circ R &= \begin{bmatrix} 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \end{bmatrix} \circ \begin{bmatrix} 1.00 & 0.67 & 0.33 & 0.00 & 0.00 \\ 0.33 & 0.67 & 1.00 & 0.67 & 0.33 \\ 0.00 & 0.33 & 0.67 & 1.00 & 0.67 \\ 0.67 & 1.00 & 0.67 & 0.33 & 0.00 \\ 1.00 & 0.67 & 0.33 & 0.00 & 0.00 \end{bmatrix} \\ &= \begin{bmatrix} 0.67 & 1.00 & 0.67 & 0.33 & 0.00 \end{bmatrix} \\ &= \frac{0.67}{1V} + \frac{1.00}{2V} + \frac{0.67}{3V} + \frac{0.33}{4V} + \frac{0.0}{5V} \quad \equiv \textit{" približno } 2V\textit{"}. \end{aligned}$$

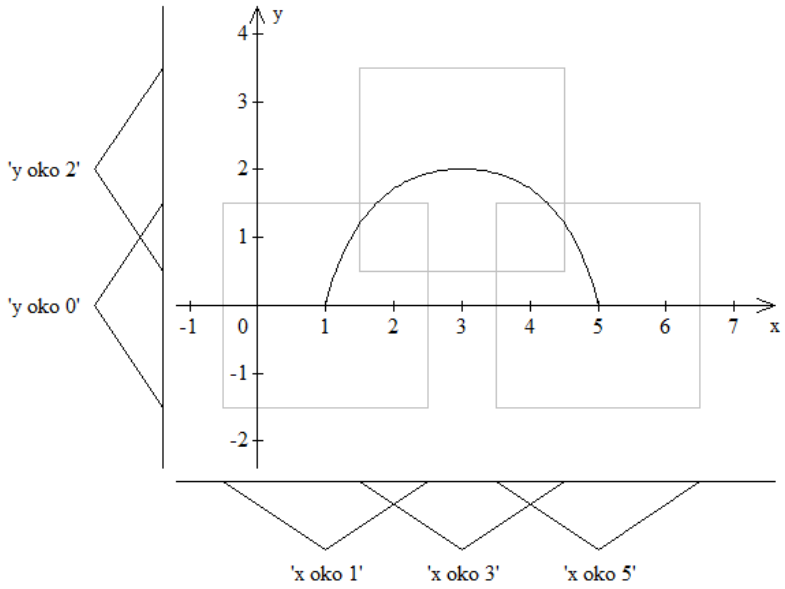
Konačno, sustav koji modeliramo može biti takav da za njega niti imamo precizne ulazne podatke, niti imamo precizne izlazne podatke. A moguć je i drugačiji scenarij: promatrani sustav ne želimo modelirati do najsitnijeg detalja već želimo dobiti grubu aproksimaciju prijenosne funkcije sustava. Uzmimo za primjer sustav koji ima prijenosnu karakteristiku $y = -0.5x^2 + 3x - 2.5$ (ali pretpostavimo da mi to ne znamo). Da bismo došli do prijenosne karakteristike, napravit ćemo nekoliko mjerenja; dovodit ćemo na ulaz različite vrijednosti napona ($x$) i mjeriti ćemo odziv (napon $y$). Kako vrijednosti koje daju instrumenti nisu baš precizne, dolazimo do sljedećeg niza *AKO-ONDA* pravila:

- **AKO je** "$x$ oko 1" **ONDA je** "$y$ oko 0",
- **AKO je** "$x$ oko 3" **ONDA je** "$y$ oko 2",
- **AKO je** "$x$ oko 5" **ONDA je** "$y$ oko 0".

Definirajmo korištene neizrazite skupove:

- "$x$ oko 1" = `FuzzyTrokut`(-0.5,1,2.5),
- "$x$ oko 3" = `FuzzyTrokut`( 1.5,3,4.5),
- "$x$ oko 5" = `FuzzyTrokut`( 3.5,5,6.5),
- "$y$ oko 0" = `FuzzyTrokut`(-1.5,0,1.5),
- "$y$ oko 2" = `FuzzyTrokut`( 0.5,2,3.5).

**AKO** $A$ **ONDA** $B$ pravila možemo shvatiti kao binarne relacije koje se dobiju kartezijevim produktom $A \times B$. Sada još samo treba napraviti uniju kartezijevih produkata koji su nastali iz pravila koja imamo, i dobili smo relaciju koja odgovara prijenosnoj karakteristici (slika 5.1). Ovo se može koristiti uvijek kada je prijenosna karakteristika sustava nepoznata, ili pak previše kompliciran da bi se mogla opisati direktno konvencionalnim matematičkim modelima.




Slika 5.1: Prijenosna karakteristika sustava.

Evo i pojašnjenja postupka. Svako **AKO-ONDA** pravilo definira jednu relaciju. Ako imamo $k$ pravila, imat ćemo $k$ relacija: $R_1$, $R_2$, ..., $R_k$. Pri tome između svakog pravila stoji logički vezik *ili*. Stoga izlaz možemo definirati kao:

$$y = (x \circ R_1) \cup (x \circ R_2) \cup \ldots \cup (x \circ R_k)$$

a to je pak jednako:

$$y = x \circ (R_1 \cup R_2 \cup \ldots \cup R_k).$$

Ako uniju relacija proglasimo konačnom relacijom $R$:

$$R = R_1 \cup R_2 \cup \ldots \cup R_k,$$

slijedi da je:

$$y = x \circ R.$$

Relacija $R$ definirana kao unija relacija koje definira svako od pravila predstavlja relaciju koja modelira prijenosnu karakteristiku sustava.

## 5.2 Dekodiranje neizrazitosti

Kroz sva poglavlja do sada govorili smo i radili s neizrazitim skupovima. Međutim, ponekad se iz neizrazitosti treba vratiti i natrag. Npr. neka imamo neizraziti sustav koji na temelju nekoliko ulaznih senzora mjeri temperaturu i vlažnost zraka. Od takvog sustava ipak očekujemo da na kraju da odgovor oblika: "trenutna temperatura je $x$ stupnjeva celzijusa", a nikako odgovor tipa "stupanj pripadnosti temperature od 15° trenutnoj temperaturi je 0.7, stupanj pripadnosti temperature od 16° trenutnoj temperaturi je 0.73, stupanj pripadnosti...". Proces kojim se vrši ova transformacija naziva se *dekodiranje neizrazitosti* (engl. *defuzzyfication*). I odmah dobra (ili loša) vijest. Ne postoji jednoznačan, najbolji način da se ovo obavi. Postoje samo načini na koje se to obično radi, iako se svakim danom javljaju i nove metode.

Najčešće metode dekodiranja neizrazitosti su sljedeće: težište (engl. *Center of Area*) koja određuje točku na $x$-osi koja u fizikalnom smislu odgovara težištu lika koji tvori funkcija pripadnosti neizrazitoga skupa, polovište (engl. *Bisector of Area*) koja određuje onu točku na $x$-osi koja predstavlja točku kroz koju prolazi okomiti pravac koji lik što ga tvori funkcija pripadnosti neizrazitoga skupa dijeli na dvije površinom jednake polovice, srednji maksimum (engl. *Mean of Max*) koja određuje srednju vrijednosti onih točaka za koje je funkcija pripadnosti neizrazitog skupa maksimalna, najmanji maksimum (engl. *Smallest of Max*) koja daje najmanji element za koji je funkcija pripadnosti neizrazitog skupa maksimalna, najveći maksimum (engl. *Largest of Max*) koja daje najveći element za koji je funkcija pripadnosti neizrazitog skupa maksimalna te druge.

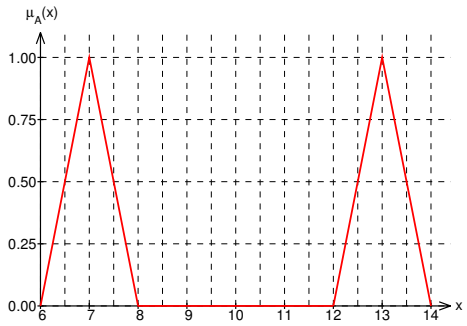
*CenterOfArea* definiran je za konačni neizraziti skup $A$ kao:

$$X_{CoA}(A) = \frac{\sum_i \mu_A(x_i) \cdot x_i}{\sum_i \mu_A(x_i)} \tag{5.1}$$

dok za kontinuirane skupove vrijedi:

$$X_{CoA}(A) = \frac{\int \mu_A(x) \cdot x \, dx}{\int \mu_A(x) \, dx}. \tag{5.2}$$

Metoda *BisectorOfArea* u općem slučaju ne daje jednoznačno rješenje, odnosno može se dogoditi da postoji beskonačno točaka koje ga zadovoljavaju. Ovo je jasno ilustrirano na slici 5.2 na kojoj je prikazan neizraziti skup definiran nad univerzalnim skupom relanih brojeva – kao rezultat dekodiranja neizrazitosti metodom *BisectorOfArea* može se uzeti bilo koji $x$ iz skupa $[8, 12]$, a takvih je beskonačno. Za koji god $x$ iz tog intervala, površina s lijeve strane bit će jednaka površini s desne strane.




Slika 5.2: Nejednoznačnost pri dekodiranju neizrazitosti metodom *BisectorOfArea*.

Spomenimo još jednu metodu dekodiranja neizrazitosti koja se povremeno koristi jer je dosta jednostavna. Pretpostavimo da je neizraziti skup $A$ koji dekodiramo nastao kao unija $r$ neizrazitih skupova $A_i$, gdje je $i \in \{1, \ldots, r\}$. Neka je $x_i^c$ centar neizrazitog skupa $A_i$ a $w_i$ njegova visina: $w_i = hgt(A_i)$.

Tada se kao rezultat dekodiranja neizrazitosti može uzeti *uprosječeni centar* pri čemu računamo težinski prosjek s visinama kao težinama:

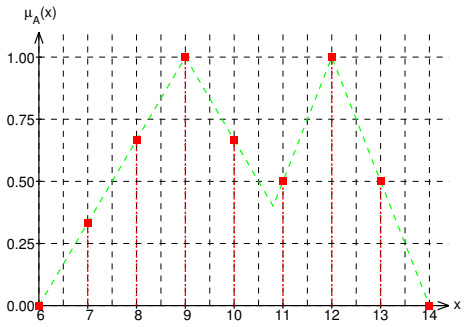
$$X_{CA}(A) = \frac{\sum_{i=1}^{r} x_i^c \cdot w_i}{\sum_{i=1}^{r} w_i}. \tag{5.3}$$

Engleski naziv ove metode je *center average defuzzifier*.

---

**Primjer:** 22

Neizraziti skup $A$ definiran je nad univerzalnim skupom $U = \{6, 7, 8, 9, 10, 11, 12, 13, 14\}$ i nastao je kao rezultat unije dvaju neizrazitih skupova čije su funkcije pripadnosti bile trokutaste. Funkcija pripadnosti skupa $A$ prikazana je na slici u nastavku.




Provedite postupak dekodiranja neizrazitosti metodom *CenterOfArea* te metodom *MeanOfMax*.

## Rješenje:

Sa slike možemo rekonstruirati parametre trokutastih funkcija pripadnosti izvornih skupova i temeljem toga možemo očitati (tj. izračunati) vrijednosti funkcije pripadnosti za sve elemente univerzalnog skupa:

$$A = \frac{0}{6} + \frac{1/3}{7} + \frac{2/3}{8} + \frac{1}{9} + \frac{2/3}{10} + \frac{0.5}{11} + \frac{1}{12} + \frac{0.5}{13} + \frac{0}{14}.$$

Dekodiranje metodom *CenterOfArea* daje:

$$\begin{aligned} X_{CoA}(A) &= \frac{\sum_i \mu_A(x_i) \cdot x_i}{\sum_i \mu_A(x_i)} \\ &= \frac{0 \cdot 6 + 1/3 \cdot 7 + 2/3 \cdot 8 + 1 \cdot 9 + 2/3 \cdot 10 + 0.5 \cdot 11 + 1 \cdot 12 + 0.5 \cdot 13 + 0 \cdot 14}{0 + 1/3 + 2/3 + 1 + 2/3 + 0.5 + 1 + 0.5 + 0} \\ &= \frac{142/3}{14/3} \\ &= \frac{71}{7} \approx 10.143 \end{aligned}$$

Kada bismo temeljem neizrazitog skupa $A$ morali generirati jednu jasnu vrijednost koja ga predstavlja, metodom *CenterOfArea* dobivamo vrijednost koja je približno 10.143 – najbliži predstavnik u univerzalnom skupu je element 10.

Za dekodiranje metodom *MeanOfMax* trebamo utvrditi za koje sve vrijednosti univerzalnog skupa funkcija pripadnosti zadanog neizrazitog skupa ima maksimalne vrijednosti. Uvidom u sliku (ili u skup koji smo očitali) vidimo da funkcija pripadnosti kao maksimalnu vrijednost poprima vrijednost 1, i ona se postiže za dva elementa univerzalnog skupa – za 9 i 12. Stoga je rezultat:

$$\begin{aligned} X_{MoM}(A) &= \frac{9 + 12}{2} \\ &= \frac{21}{2} \\ &= 10.5 \end{aligned}$$

Rezultat dekodiranja neizrazitosti ovom metodom je 10.5. U univerzalnom skupu imamo dva elementa koja su najbliža toj vrijednosti: 10 i 11 i bilo koja od njih može biti odgovor.

---

**Primjer:** 23

Pretpostavite da je neizraziti skup iz prethodnog zadatka definiran nad univerzalnim skupom realnih brojeva. U tom slučaju njegova je funkcija pripadnosti skicirana zelenom crtkanom linijom. Provedite postupak dekodiranja neizrazitosti metodama *CenterOfArea* te *BisectorOfArea*.

## Rješenje:

Kako je u ovom slučaju funkcija pripadnosti definirana nad skupom realnih brojeva, metoda *CenterOfArea* koristit će izraz koji je definiran preko integrala. Pročitajmo stoga najprije kako je točno definirana funkcija pripadnosti skupa $A$. Kao pomoć izračunajmo gdje se sijeku dva izvorna trokuta čiju uniju analiziramo: dobit ćemo vrijednosti $x = 10.8$ i $\mu_A(x) = 0.4$. Čitava funkcija pripadnosti tada je:

$$\mu_A(x) = \begin{cases} 0, & x \leq 6, \\ \frac{x-6}{3}, & x \leq 9, \\ \frac{12-x}{3}, & x \leq 10.8, \\ \frac{x-10}{2}, & x \leq 12, \\ \frac{14-x}{2}, & x \leq 14, \\ 0, & x > 14. \end{cases}$$

Da bismo proveli dekodiranje neizrazitosti, koristimo izraz:

$$X_{CoA}(A) = \frac{\int_{-\infty}^{\infty} \mu_A(x) \cdot x \, dx}{\int_{-\infty}^{\infty} \mu_A(x) \, dx}.$$

S obzirom da je funkcija pripadnosti zadana po dijelovima, gornji integral ćemo računati po dijelovima, što ćemo zapisati simbolički kao:

$$\begin{aligned} \int_{-\infty}^{\infty} \mu_A(x) \cdot x \, dx &= \int_{-\infty}^{6} + \int_{6}^{9} + \int_{9}^{10.8} + \int_{10.8}^{12} + \int_{12}^{14} + \int_{14}^{\infty} \\ &= \int_{6}^{9} + \int_{9}^{10.8} + \int_{10.8}^{12} + \int_{12}^{14} \end{aligned}$$

U svakom od tih dijelova za $\mu_A(x)$ ćemo uvrstiti izraz koji ga definira. Račun je prikazan u nastavku.

$$\begin{aligned} \int_{6}^{9} \left(\frac{1}{3}x - 2\right) x \, dx &= \int_{6}^{9} \left(\frac{1}{3}x^2 - 2x\right) \, dx = \left(\frac{1}{9}x^3 - x^2\right)\Big|_{6}^{9} = 12. \\ \int_{9}^{10.8} \left(4 - \frac{1}{3}x\right) x \, dx &= \int_{6}^{9} \left(4x - \frac{1}{3}x^2\right) \, dx = \left(2x^2 - \frac{1}{9}x^3\right)\Big|_{9}^{10.8} = 12.312. \\ \int_{10.8}^{12} \left(\frac{1}{2}x - 5\right) x \, dx &= \int_{6}^{9} \left(\frac{1}{2}x^2 - 5x\right) \, dx = \left(\frac{1}{6}x^3 - \frac{5}{2}x^2\right)\Big|_{10.8}^{12} = 9.648. \\ \int_{12}^{14} \left(7 - \frac{1}{2}x\right) x \, dx &= \int_{6}^{9} \left(7x - \frac{1}{2}x^2\right) \, dx = \left(\frac{7}{2}x^2 - \frac{1}{6}x^3\right)\Big|_{12}^{14} = 13 - \frac{1}{3}. \end{aligned}$$

Ukupna vrijednost integrala u nazivniku jednaka je sumi upravo izračunatih dijelova:

$$\int_{-\infty}^{\infty} \mu_A(x) \cdot x \, dx = 12 + 12.312 + 9.648 + 13 - \frac{1}{3} = 46.96 - \frac{1}{3} \approx 46.6266667.$$

Integral u nazivniku predstavlja površinu ispod funkcije pripadnosti; čitateljima se ostavlja za vježbu da postupak provedu direktno integriranjem, a mi ćemo se poslužiti činjenicom da su

funkcije početnih skupova bile trokutaste, pa se površina može direktno izračunati kako slijedi:

$$\begin{aligned}\int_{-\infty}^{\infty} \mu_A(x)\,dx &= \int_{-\infty}^{6} + \int_{6}^{9} + \int_{9}^{10.8} + \int_{10.8}^{12} + \int_{12}^{14} + \int_{14}^{\infty} \\ &= \int_{6}^{9} + \int_{9}^{10.8} + \int_{10.8}^{12} + \int_{12}^{14} \\ &= \text{površina 1 } + \text{površina 2 } + \text{površina 3 } + \text{površina 4} \\ &= (1.5) + (1.5 - \frac{1.2 \cdot 0.4}{2}) + (1 - \frac{0.8 \cdot 0.4}{2}) + (1) \\ &= 4.6.\end{aligned}$$

Time možemo izračunati i traženi omjer:

$$X_{CoA}(A) = \frac{\int_{-\infty}^{\infty} \mu_A(x) \cdot x\,dx}{\int_{-\infty}^{\infty} \mu_A(x)\,dx} = \frac{46.6266667}{4.6} = 10.136.$$

Provedimo dekodiranje i metodom *BisectorOfArea*. Već smo utvrdili da se čitava površina ispod funkcije pripadnosti raspada na četiri podpovršine ukupne sume 4.6. Pola od toga je 2.3 što se postiže zbrajanjem prve površine i dijela druge površine; traženi $x$ stoga je negdje u rasponu $x$-eva koji pripadaju drugoj površini. Možemo postaviti sljedeću jednadžbu:

$$P1 + P2a = P2b + P3 + P4.$$

Raspisivanjem imamo:

$$(1.5) + \left(\frac{3 \cdot 1}{2} - \frac{(12 - x)(4 - {}^1\!/_3\, x)}{2}\right) = \left(\frac{(12 - x)(4 - {}^1\!/_3\, x)}{2} - \frac{1.2 \cdot 0.4}{2}\right) + (0.84) + (1).$$

pri čemu su zagradama grupirani dijelovi koji predstavljaju svaku od površina. Ova jednadžba može se svesti na klasičnu kvadratnu jednadžbu:

$$5x^2 - 120x + 699 = 0$$

čije je prihvatljivo rješenje:

$$x = 9.951,$$

i to je rješenje jedinstveno.

---

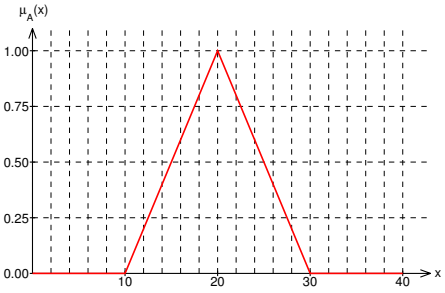
### 5.2.1 Numerički postupci pri dekodiranju neizrazitosti

Ako su neizraziti skupovi definirani nad diskretnim univerzalnim skupovima, tada se postupci dekodiranja neizrazitosti mogu vrlo jednostavno implementirati na računalu – naime, imamo jasne definicije koje se tipično svode na izračune suma određenih izraza nad elementima univerzalnog skupa. Problem međutim nastaje ako sustav implementiramo na računalu i kao domenu koristimo podskup realnih brojeva. U tom slučaju na raspolaganju su nam formule koje se svode na izračune integrala. Računanje točnog iznosa integrala pri tome može biti neprihvatljivo složen (ili čak nemoguć) zadatak, pa takvo računanje treba nekako aproksimirati. U ovom podpoglavlju osvrnut ćemo se stoga na jednostavan postupak numeričkog aproksimiranja vrijednosti integrala.

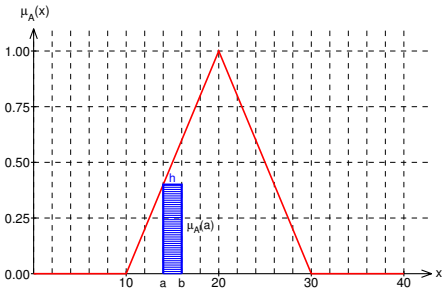
Pretpostavimo da je zadan neizraziti skup prikazan na slici 5.3a. Potrebno je izračunati površinu ispod funkcije pripadnosti tog neizrazitog skupa, odnosno integral:

$$I = \int_{x_{min}}^{x_{max}} \mu_A(x)\, dx$$

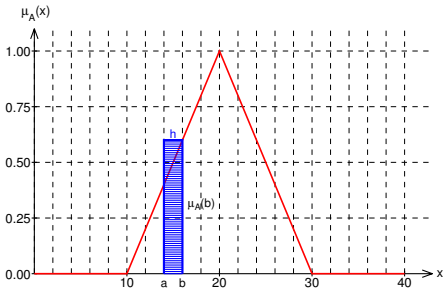
pri čemu je pretpostavka da je univerzalni skup nad koji je definiran ovaj neizraziti skup jednak $[x_{min}, x_{max}] \in \mathbb{R}$. Ako promatramo samo onaj dio koji je prikazan slikom, možemo pretpostaviti da je to $[0, 40]$.




(a) Funkcija čija se računa površina.




(b) Aproksimacija dijela površine s lijeve strane.




(c) Aproksimacija dijela površine s desne strane.

Slika 5.3: Aproksimacija površine tj. vrijednosti integrala trapeznom formulom.

Postoji više načina kako se može pristupiti ovom problemu, i mi ćemo se ovdje fokusirati na najjednostavniji koji će u praksi često biti dovoljno dobar. Čitav interval $[x_{min}, x_{max}]$ podijelit ćemo u $n$ uniformnih podintervala, svaki širine $h$:

$$h = \frac{x_{max} - x_{min}}{n}.$$

Vrijednost integrala tada možemo zapisati kao sumu integrala nad svakim

podintervalom, tj.:

$$I = \sum_{i=0}^{n-1} I_{[x_{min}+i\cdot h, x_{min}+(i+1)\cdot h]}.$$

Time smo problem određivanja jednog integrala sveli na problem određivanja $n$ integrala; međutim, varijabilnost funkcije unutar svakog od tih $n$ integrala nadamo se da je značajno manja u odnosu na varijabilnost funkcije nad čitavim univerzalnim skupom, i pretpostavka je da će uz dovoljno velik $n$ (odnosno uz dovoljno mali $h$) na svakom podintervalu funkcija poprimiti linearni karakter. Pogledajmo stoga jedan istaknuti podinterval $[a, b]$ širine $h = b - a$. I bez pretpostavke da je unutar tog funkcija strogo linearna, površinu možemo aproksimirati na dva načina, kako to prikazuju slike 5.3b i 5.3c. Prva mogućnost je da izračunamo vrijednost funkcije u točki $a$ i to uzmemo kao visinu stupca, te površinu nad intervalom aproksimiramo s $h \cdot \mu_A(a)$. Alternativa je da izračunamo vrijednost funkcije u točki $b$ i to uzmemo kao visinu stupca, te površinu nad intervalom aproksimiramo s $h \cdot \mu_A(b)$. Kako je $\mu_A(a) < \mu_A(b)$, vidimo da smo u prvom slučaju površinu podcijenili, a u drugom precijenili. Stoga se kao konačna aproksimacija može uzeti aritmetička sredina tih dviju procjena površina:

$$I_{[a,b]} = \frac{h \cdot \mu_A(a) + h \cdot \mu_A(b)}{2} = h \cdot \frac{\mu_A(a) + \mu_A(b)}{2} = (b-a) \cdot \frac{\mu_A(a) + \mu_A(b)}{2}.$$

Izraz koji smo upravo izveli poznat je pod nazivom *trapezna aproksimacija vrijednosti integrala funkcije* $\mu_A(x)$. Uočite da, za slučaj da se funkcija na svakom od intervala doista ponaša linearno, ova aproksimacija daje pravu vrijednost površine. Ako se funkcija ne ponaša linearno, dobit ćemo aproksimaciju; to bolju što je $h$ manji, ali kako time $n$ nužno raste, postupak računanja će biti sve dulji i dulji. Dakle, kao općeniti izraz za aproksimaciju integrala:

$$I = \int_{x_{min}}^{x_{max}} f(x)\, dx \tag{5.4}$$

može se koristiti uniformna diskretizacija u $n$ podintervala:

$$h = \frac{x_{max} - x_{min}}{n} \tag{5.5}$$

te izraz:

$$I = \sum_{i=0}^{n-1} I_{[x_{min}+i\cdot h, x_{min}+(i+1)\cdot h]} \tag{5.6}$$

$$= \sum_{i=0}^{n-1} h \cdot \frac{f(x_{min} + i \cdot h) + f(x_{min} + (i+1) \cdot h)}{2}. \tag{5.7}$$

Primjerice, u slučaju dekodiranja neizrazitosti metodom *CenterOfArea* potrebno je izračunati dva integrala: $\int \mu_A(x) \cdot x \, dx$ i $\int \mu_A(x) \, dx$; za izračun prvog integrala uzet ćemo $f(x) = \mu_A(x) \cdot x$ i provesti postupak dok ćemo za izračun drugog uzeti $f(x) = \mu_A(x)$ i ponoviti postupak.

Osim opisanog postupka, moguće je posegnuti i za drugim načinom izračuna integrala. Prethodno opisani postupak aproksimirao je vrijednost integrala funkcije koju nismo mogli točno integrirati. Umjesto toga, problem bismo mogli postaviti i ovako: pokušajmo funkciju koju integriramo aproksimirati nekom funkcijom koju znamo točno integrirati, i potom obavimo integraciju. Dakle, umjesto da aproksimiramo površinu ispod funkcije, mi ćemo aproksimirati funkciju nekom drugom (ali takvom da nju znamo integrirati) i potom ćemo provesti integraciju nad aproksimacijskom funkcijom. I tu se otvara čitav niz mogućnosti kako to napraviti, čime se bavi jedan velik dio numeričke matematike.

Jedna od vrsta funkcija koje znamo integrirati su polinomi. Naime, ako imamo zadan polinom:

$$P(x) = \sum_{i=0}^{n} a_i \cdot x^i$$

neodređeni integral tog polinoma dan je sljedećim izrazom:

$$\int P(x) \, dx = \sum_{i=0}^{n} \frac{a_i}{i+1} \cdot x^{i+1} = Q(x)$$

što je opet polinom. Tada vrijedi:

$$\int_{x_{min}}^{x_{max}} P(x) \, dx = Q(x)|_{x_{min}}^{x_{max}} = Q(x_{max}) - Q(x_{min}) \tag{5.8}$$

i što je jednostavno za izračunati (i isprogramirati).

Problem koji time ostaje za riješiti jest kako pronaći polinom koji dobro aproksimira zadanu funkciju $f(x)$ koju trebamo integrirati, i na to pitanje nema jednoznačnog odgovora. Jedna mogućnost jest odlučiti se zadanu funkciju $f(x)$ uniformno uzorkovati u $n+1$ točki, i zatim pronaći polinom minimalnog stupnja koji točno prolazi kroz svih $n+1$ točku. Takav polinom je jednoznačno određen, i to je polinom $n$-tog stupnja. Do koeficijenata polinoma možemo doći na više načina, a ovdje ćemo opisati način koji koristi Lagrangeov zapis polinoma.

Neka je funkcija $f(x)$ uniformno uzorkovana u $n+1$ točki, čime smo dobili $n+1$ par $(x_i, y_i)$ gdje je $y_i = f(x_i)$. Za slučaj interpolacije u $n+1$ točki definira se $n+1$ bazni Lagrangeov polinom, pri čemu je $j$-ti bazni Lagrangeov polinom $l_j(x)$ definiran kao:

$$l_j(x) = \prod_{\substack{0 \le m \le n \\ m \ne j}} \frac{(x - x_m)}{x_j - x_m} = \frac{(x - x_0) \cdots (x - x_{j-1})(x - x_{j+1}) \cdots (x - x_n)}{(x_j - x_0) \cdots (x_j - x_{j-1})(x_j - x_{j+1}) \cdots (x_j - x_n)}.$$

Uočite da je brojnik razlomka upravo polinom $n$-tog stupnja čije su nultočke svih $n$ vrijednosti $x_i$ osim $x_j$, i čiji je nazivnik konstanta koja ovisi o svim vrijednostima $x_i$ u kojima je napravljeno uzorkovanje. Stoga slijedi da za $i \neq j$ vrijedi $l_j(x_i) = 0$. Ako pak u $l_j$ uvrstimo vrijednost $x_j$, brojnik i nazivnik postaju identični i pokrate se; tada vrijedi: $l_j(x_j) = 1$. $j$-ti bazni Lagrangeov polinom, dakle, u točki $x_j$ poprima vrijednost 1 a u svim ostalim točkama u kojima je rađeno uzorkovanje poprima vrijednost 0. Ako s $L(x)$ označimo aproksimaciju funkcije $f(x)$ (točnije njezinu interpolaciju u $n+1$ uniformno distribuiranoj točki), tada možemo pisati:

$$L(x) = \sum_{j=0}^{n} y_j \cdot l_j(x). \tag{5.9}$$

Funkcija $L(x)$ doista jest interpolacija funkcije $f(x)$ – onda prolazi kroz svih $n+1$ zadanih točaka zbog svojstva funkcije $l_j$ koja je u $x = x_j$ jednaka 1 pa je tada doprinos $y_j \cdot l_j(x_j) = y_j$ a u $x = x_i$ za $i \neq j$ jednaka 0 pa je doprinos $y_i \cdot l_j(x_i) = 0$ čime je čitava suma jednaka upravo $y_j$. Međutim, isto tako, funkcija $L(x)$ je samo aproksimacija funkcije $f(x)$ jer ona sigurno poprima iste vrijednosti kao funkcija $f(x)$ samo u $n+1$ zadanoj točki; u svim ostalim međutočkama razlika između $L(x)$ i $f(x)$ može biti zamjetna. Ovo je posebice istina zbog Rungeovog fenomena, pojave koju je spoznao Carl David Tolmé Runge, njemački matematičar i fizičar, i koja se ispoljava kao pojava sve većih oscilacija interpolacijskog polinoma $n$-tog reda uz krajeve intervala, kako $n$ raste. To pak znači da će se, za određene funkcije, s povećanjem broja točaka u kojima se radi uzorkovanje aproksimacija pogoršavati, a ne postajati sve bolja. Načini zaobilaženja ovog problema izlaze iz opsega ovog teksta, i zainteresirane čitatelje se upućuje da pogledaju interpolacijske pristupe koji se temelje na neuniformnom uzorkovanju (ideja je da se rjeđe uzorkuje oko sredine intervala a sve gušće kako se približavamo rubovima intervala, jer tamo inače dolazi do oscilacija – vidi Chebyshevljevo uzorkovanje). Osim Lagrangeovog oblika polinoma koristi se i Newtonov oblik polinoma te drugi.

Jednom kad smo utvrdili svih $n+1$ baznih Lagrangeovih polinoma (koji su svaki stupnja $n$), konačni aproksimacijski polinom $L(x)$ koji je težinska suma baznih polinoma (gdje su vrijednosti funkcije $f$ u uzorkovanim točkama težine) također je polinom $n$-tog stupnja. Koeficijente uz sve potencije od $x$ je moguće programski izračunati, a to znači da možemo primjeniti izraz 5.8 za izračun točne vrijednosti integrala ispod aproksimacijske funkcije.

## 5.3 Približno zaključivanje

Približno zaključivanje služi nam za predstavljanje znanja i zaključivanje kada je znanje izraženo prirodnim jezikom, ili kada je neprecizno ili nejasno definirano. Pri tome se koriste jezične varijable i neizraziti skupovi. U klasičnoj logici osnova svega su propozicije. To je atomarna jedinica kojoj se pri-

djeljuje vrijednost istinitosti; primjerice, propozicija "Pero je doktor" može biti istinita ili lažna, ili može poprimiti neki nivo istinitosti ako govorimo o viševrijednosnoj logici. U neizrazitoj logici radimo s neizrazitim propozicijama koje se u općem obliku zapisuju kao "$X$ je $A$", pri čemu je $x$ varijabla a $A$ neizraziti skup. Npr. u neizrazitoj propoziciji "temperatura je vrlo visoka", "temperatura" je varijabla, a "vrlo visoka" je neizraziti skup koji opisuje koncept vrlo visoke temperature. Uočite odmah da je značenje ove propozicije potpuno subjektivno jer je definicija "vrlo visoke" temperature subjektivna. Nije isto ako govorimo da čovjek (pacijent) ima visoku temperaturu ($\approx 40°$C) ili da je talište nekog metala $M_2$ na vrlo visokoj temperaturi (par tisuća °C) jer smo kao referentnu temperaturu uzeli talište nekog metala $M_1$ koja iznosi par stotina °C. Vrijednost istinitosti propozicije "$x$ je $A$" definirana je s funkcijom pripadnosti $\mu_A(x)$.

Složene neizrazite propozicije grade se uporabom veznika *i*, *ili*, *ne*, *ako-onda* i slično. Tako primjerice od jednostavnih propozicija "$x$ je $A$" i "$x$ je $B$" možemo izgraditi složene propozicije "$x$ je $A$ ILI $x$ je B", "Ako $x$ je $A$ ONDA $x$ je $B$" i tako dalje. Pri tome se vrijednosti istinitosti složenih propozicija određuju na način kako to prikazuje tablica 5.1.

Tablica 5.1: Složene neizrazite propozicije

| Složena propozicija | Rezultat | Način računanja istinitosti |
|---|---|---|
| $x$ je $A$ I $x$ je $B$ | $x$ je $C$, $C = A$ I $B$ | $\mu_C(x) = \mu_A(x) \cap \mu_B(x)$, $\cap$ označava bilo koju $t$-normu iako se najčešće koristi Zadehova operacija minimuma |
| $x$ je $A$ ILI $x$ je $B$ | $x$ je $C$, $C = A$ ILI $B$ | $\mu_C(x) = \mu_A(x) \cup \mu_B(x)$, $\cup$ označava bilo koju $s$-normu iako se najčešće koristi Zadehova operacija maksimuma |
| $x$ nije $A$ | $x$ je $C$, $C =$ ne $A$ | $\mu_C(x) = \neg\mu_A(x)$, $\neg$ označava bilo koju operaciju generaliziranog komplementa, iako se najčešće koristi Zadehova operacija komplementa definirana kao $1 - \mu_A(x)$ |

Osim ovih operacija u uporabi su česta i neizrazita produkcijska pravila koja su oblika **AKO** $x$ je $A$ **ONDA** $y$ je $B$, pri čemu je $A$ neizraziti skup definiran nad univerzalnim skupom $U_1$, $B$ neizraziti skup definiran nad univerzalnim skupom $U_2$. Značenje ove implikacije predstavljeno je relacijom nad $U_1 \times U_2$, sa svojstvom: $\mu_R(x, y) = \mu_A(x) * \mu_B(y)$, pri čemu je $*$ bilo koji operator implikacije (a jedan smo već upoznali iako ga tada nismo zvali implikacijom $\rightarrow$ kartezijev produkt).

U klasičnoj logici jedno od poznatih pravila zaključivanja je modus ponens. Prisjetimo se što nam govori to pravilo:

| | |
|---|---|
| Premisa 1: | $x$ je $A$ |
| Premisa 2: | **AKO** $x$ je $A$ **ONDA** $y$ je $B$ |
| Zaključak: | $y$ je $B$. |

Pri tome su "$x$ je $A$" i "$y$ je $B$" klasične propozicije. Neizrazita logika omogućava definiranje *generaliziranog modus ponensa* koji radi s neizrazitim propozicijama:

| | |
|---|---|
| Premisa 1: | $x$ je $A$ |
| Premisa 2: | **AKO** $x$ je $B$ **ONDA** $y$ je $C$ |
| Zaključak: | $y$ je $D$. |

Uporaba generaliziranog modus ponensa omogućava provođenje zaključivanja u situacijama u kojima ulazna vrijednost varijable nije baš identična antecedentu u pravilu. Zaključivanje generaliziranim modus ponensom svodi se na izračun kompozicije neizrazitog skupa koji predstavlja ulaznu premisu i relacije koju generira **AKO-ONDA** pravilo:

| | |
|---|---|
| Premisa 1: | $x$ je $A$ |
| Premisa 2: | $(x, y) \in R$ |
| Zaključak: | $y$ je $D$. |

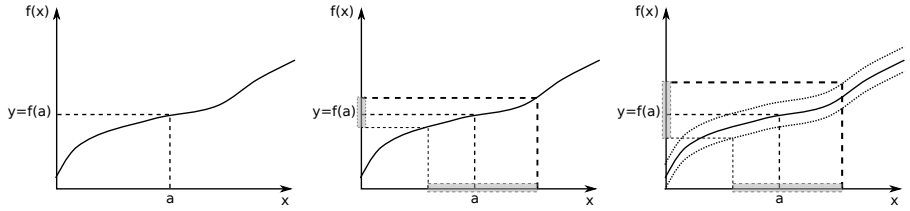
Ovo pravilo pretpostavlja da relacija $R$ već postoji, i može biti proizvoljna. Ako je $A$ definiran nad univerzalnim skupom $U_1$, tada relacija $R$ mora biti definirana nad $U_1 \times U_2$, čime će rezultat $Y$, tj. kompozicija $A \circ R$, biti definiran nad $U_2$. **AKO-ONDA** pravila jedan su način dobivanja relacije $R$ – već smo rekli da je značenje **AKO-ONDA** pravila relacija koja se dobije nekim od operatora implikacije. Najčešći operatori implikacije navedeni su u nastavku.

U klasičnoj binarnoj logici vrijedi: $(p \rightarrow q) \equiv (\neg p \vee q) \equiv ((p \wedge q) \vee \neg p)$. Koristeći $s$-norme i $t$-norme te generalizirane komplemente izvodi se većina operatora implikacije u neizrazitoj logici.

Pogledajmo malo detaljnije: zašto se zaključivanje *modus ponensom* svodi na kompoziciju. Neka je dano sljedeće pravilo.

**AKO** $x = a$ **I** $y = f(x)$ **TADA** $y = f(a)$

Pravilo se čini sasvim zdravorazumsko. No pogledajmo njegove dijelove. Antecedent se sastoji od dva dijela: prvi dio utvrđuje vrijednost varijable $x$ a drugi dio definira da se varijabla $y$ dobiva kao primjena funkcije $f(.)$ na varijablu $x$. Ako antecedent vrijedi, pravilo kaže da je tada vrijednost varijable $y$ jednaka rezultatu koji se dobije uvrštavanjem varijable $x$ u funkciju




(a) Izrazite vrijednosti (b) Neizrazit $x$ (c) Neizraziti $f$ i $x$

Slika 5.4: Inspiracija za postupak izračuna generaliziranog modus ponensa

$f$. No funkcija nije ništa drugo doli specijalan oblik relacije. Stoga se dano pravilo može direktno proširiti i to na sljedeći način (vidi sliku 5.4).

Ako imamo klasičnu funkciju i izrazitu vrijednost varijable (slučaj na slici 5.4a), tada se svaki $x$ preslikava točno u jedan $y$. Ako sada dopustimo da $x$ nije izrazit broj već neki interval (slučaj na slici 5.4b), rezultat preslikavanja bit će također interval. Dapače, ako dopustimo da $x$ bude neizraziti skup a ne običan interval, tada ćemo za svaki $y$ u koji $f$ preslika $x$ definirati i stupanj pripadnosti koji će biti jednak stupnju pripadnosti kojim $x$ pripada ulaznom neizrazitom skupu.

Konačno, sada možemo proširiti i $f$ tako da nije funkcija u strogom matematičkom smislu već da je neizrazita funkcija: funkcija koja isti fiksirani $x$ preslikava u različite vrijednosti $y$ sa zadanim stupnjevima pripadnosti (slika 5.4c). Ovakvo zadana funkcija zapravo je neizrazita relacija. Da bismo odredili neizrazit skup $Y$ u koji se preslikava ulazni neizraziti skup $X$, potrebno je za svaki $y \in V$ (gdje je $V$ univerzalni skup nad kojim je definiran $Y$ a $U$ univerzalni skup nad kojim je definiran $X$) odrediti mjeru pripadnosti kojom taj element pripada neizrazitom skupu $Y$. Pogledamo li sliku i fiksiramo li neki $y = y_i$, uočit ćemo da se više različitih elemenata $x_j \in U$ preslikava u taj fiksirani $y_i$, ali s različitim stupnjevima pripadnosti. Za svaki takav $x_j$ treba odrediti u kojoj mjeri taj $x_j$ pripada neizrazitom skupu $X$ te u kojoj mjeri relacija (odnosno zadana neizrazita funkcija) taj $x_j$ preslikava u zadani $y_i$. Ta dva stupnja pripadnosti uobičajeno se kombiniraju nekom $t$-normom (primjerice, minimumom). Npr. ako odabrani $x_j$ neizrazitom skupu $X$ pripada s mjerom 0.4 ($\mu_X(x_j) = 0.4$) te ako neizrazita funkcija taj $x_j$ preslikava u $y_i$ sa stupnjem pripadnosti 1 ($\mu_R(x_j, y_i) = 1$), zaključit ćemo da smo tim putem utvrdili da $y_i$ neizrazitom skupu pripada sa stupnjem pripadnosti od $\min(0.4, 1) = 0.4$. Naravno, kako se u isti $y_i$ potencijalno preslikava više različitih $x_j$, želimo utvrditi mjeru kojom $y_i$ pripada u $Y$ preko svakog mogućeg preslikavanja i onda zaključiti da u $Y$ pripada mjerom koja je najveća utvrđena preko svih promatranih $x_j$. Neformalno,postupak bismo mogli opisati ovako: mjera kojom $y_i$ pripada neizrazitom skupu $Y$ jednaka je (mjeri kojom $x_1$ pripada neizrazitom skupu $X$ **I** mjeri kojom relacija $f$ taj $x_1$ preslikava u $y_i$) **ILI** (mjeri kojom $x_2$ pripada neizrazitom skupu $X$ **I** mjeri kojom relacija $f$ taj $x_2$ preslikava u $y_i$) **ILI** ... U prethodnoj rečenici namjerno su korištene

zagrade u svrhu grupiranja kako bismo prepoznali mjesto na kojem se koriste $t$-norme za kombiniranje dviju mjera te mjesta na kojima se koriste $s$-norme za kombiniranje dviju mjera.

Isti postupak možemo prikazati na još jedan način (pokušajte se uvjeriti da je to isto). Ulazni neizraziti skup $X$ proširimo na neizrazitu relaciju $R'$ definiranu nad $U \times V$ koja opisuje neku funkciju $f'$ koja svaki $x_j \in U$ preslikava u sve $y_i \in V$ onom mjerom kojom $x_j$ pripada neizrazitom skupu $X$. Ovakva relacija direktno odgovara cilindričnom proširenju. Naša zadana funkcija $f$ definirana je neizrazitom relacijom $R$. Sada izgradimo konačnu relaciju $R''$ kao presjek ovih dviju relacija: $R'' = R' \cap R$, koristeći bilo koju $t$-normu. Time smo dobili relaciju koja za svaki $(x_j, y_i)$ sadrži supanj pripadnosti koji reflektira i mjeru kojom $x_j$ pripada ulaznom neizrazitom skupu $X$ i mjeru kojom relacija $R$ (tj. funkcija $f$) taj $x_j$ preslikava u $y_i$. Da bismo utvrdili kojom mjerom $y_i$ pripada izlaznom neizrazitom skupu $Y$, radimo projekciju relacije $R''$ na $V$ pri čemu za kombiniranje stupnjeva pripadnosti možemo koristiti bilo koju $s$-normu.

Dajmo sada i formalnu definiciju postupka. Pretpostavimo da znamo da vrijedi premisa $x$ je $A$. Pretpostavimo također da je temeljem **AKO-ONDA** pravila izgrađena relacija $R$ nad $U \times V$, odnosno da vrijedi i druga premisa. Provodimo sljedeći postupak.

1. korak: neizraziti skup $A$ želimo proširiti na istu domenu nad kojom je definirana u relacija kako bismo dobili relaciju koju možemo usporediti sa zadanom relacijom $R$. Računamo stoga skup $A_{cil}$ kao cilindrično proširenje $A$ na $V$: $\mu_{A_{cil}}(x, y) = \mu_A(x)$. Time su $A_{cil}$ i $R$ definirani nad istom domenom.

2. korak: računamo presjek cilindričnog proširenja i relacije:

$$\mu_{A_{cil} \cap R}(x, y) = T(\mu_{A_{cil}}(x, y), \mu_R(x, y)).$$

   gdje je $T$ odabrana $t$-norma.

3. korak: računamo projekciju tog presjeka na $V$; slijedi:

$$\mu_B(y) = \sup_x \mu_{A_{cil} \cap R}(x, y).$$

Cjelokupni postupak može se kraće zapisati kao:

$$\mu_B(y) = \sup_x T(\mu_A(x), \mu_R(x, y))$$

Odaberemo li kao $t$-normu operator minimum ili produkt, direktno dobivamo *sup-min* odnosno *sup-produkt* kompozicije.

### 5.3.1 Najčešći operatori implikacije u neizrazitoj logici

U nastavku je dan kratak pregled najčešćih operatora implikacije koji se koriste u neizrazitoj logici.

**Implikacija *Kleene-Diens***

Ova implikacija definirana je izrazom:

$$(p \to q) \equiv (\neg p \vee q)$$

pri čemu se kao operator $\vee$ koristi Zadehov operator maksimuma a kao negacija Zadehov operator negacije. Stoga je pripadna relacija definirana funkcijom pripadnosti:

$$\mu_R(x, y) = \max(1 - \mu_A(x), \mu_B(y)).$$

**Implikacija *Lukasiewicz***

Ova implikacija definirana je izrazom:

$$(p \to q) \equiv (\neg p \vee q)$$

pri čemu se kao operator $\vee$ koristi ograničena suma a kao negacija Zadehov operator negacije. Stoga je pripadna relacija definirana funkcijom pripadnosti:

$$\mu_R(x, y) = \min(1, 1 - \mu_A(x) + \mu_B(y)).$$

**Implikacija *Zadeh***

Ova implikacija definirana je izrazom:

$$(p \to q) \equiv ((p \wedge q) \vee \neg p)$$

pri čemu se kao operator $\vee$ koristi Zadehov operator maksimuma, kao operator $\wedge$ koristi Zadehov operator minimuma a kao negacija Zadehov operator negacije. Stoga je pripadna relacija definirana funkcijom pripadnosti:

$$\mu_R(x, y) = \max(\min(\mu_A(x), \mu_B(y)), 1 - \mu_A(x)).$$

**Implikacija *Gödel***

Ova implikacija definirana je izrazom:

$$(p \to q) \equiv \begin{cases} 1, & p \leq q, \\ q, & p > q \end{cases}$$

što je definicija implikacije kakva se inače koristi u viševrijednosnoj logici. Stoga je pripadna relacija definirana funkcijom pripadnosti:

$$\mu_R(x, y) = \begin{cases} 1, & \mu_A(x) \leq \mu_B(y), \\ \mu_B(y), & \mu_A(x) > \mu_B(y) \end{cases}$$

Prije no što damo još i definiciju implikacije prema Mamdaniju, osvrnimo se malo na uporabu implikacije u neizrazitoj logici. **AKO-ONDA** pravila koja su sastavni dio sustava neizrazitog upravljanja uobičajeno se shvaćaju kao implikacije. Primjerice, neka je dano sljedeće pravilo.

**AKO** struja je *mala* **ONDA** napon je *visok*.

Postoje dva tumačenja značenja danog pravila. Jedno moguće tumačenje bi bilo: pravilo nam kaže da je napon visok samo ako je struja mala, ali ako struja nije mala, tada napon nije visok. Donekle blisko tumačenje bilo bi i reći da ako struja nije mala, napon može biti i visok i nizak. Ovakva tumačenja kod kojih je interpretacija pravila u određenom smislu globalna odgovara implikaciji u klasičnoj logici. Sve prethodne definicije implikacija izvedene su upravo iz klasične implikacije.

Pogledajmo to na primjeru. Neka su dani neizraziti skupovi *Mala struja* $= \left\{\frac{1}{1A} + \frac{0.6}{2A} + \frac{0}{3A}\right\}$ i *Visoki napon* $= \left\{\frac{0.1}{1V} + \frac{0.5}{2V} + \frac{1}{3V}\right\}$. Relacija koja se dobije uporabom implikacije prema Zadehu je:

$$R = \begin{bmatrix} 0.1 & 0.5 & 1.0 \\ 0.4 & 0.5 & 0.6 \\ 1.0 & 1.0 & 1.0 \end{bmatrix}.$$

Očekivano, iz relacije je vidljivo da ako je struja mala (prvi redak koji odgovara struji 1A), napon je visok jer za tu struju samo napon od 3V pripada relaciji sa stupnjem pripadnosti 1: $\mu_R(1A, 3V) = 1$ dok ostali naponi pripadaju s manjim stupnjem. Međutim, pogledajmo što se događa za srednje ili velike struje: razmotrimo primjerice treći redak relacije koji opisuje vezu između različitih napona i struje 3A. Svi stupnjevi pripadnosti u tom retku su 1, čime definirana relacija uspostavlja vezu između struje od 3A i svih napona. Mogli bismo reći da je ova relacija zapravo relacija koja definira što nije moguće: pogledajte distribuciju niskih stupnjeva pripadnosti u toj relaciji. Relaciju bismo mogli protumačiti ovako: nije istina da je struja mala i napon mali: to je proturječno pravilu; sve ostalo vrijedi. Ako je struja mala, pravilo definira da je napon visok; za sve ostale struje ne znamo pa ne isključujemo vezu između takvih struja i svih mogućih napona.

Često, ovakva interpretacija **AKO-ONDA** pravila nam ne odgovara. Umjesto globalnog karaktera, pravilu želimo dali lokalni karakter: pravilo uspostavlja vezu između zadovoljenog antecedenta i danog konsekventa. Ako antecedent ne vrijedi, pravilo ne daje nikakvu dodatnu informaciju, odnosno ne uspostavlja relaciju između elemenata koji ne zadovoljavaju antecedent. Posljedično, u takvoj situaciji htjeli bismo implikaciju koja će u relaciju uzeti samo one elemente koji zadovoljavaju i antecedent i konsekvent. U konkretnom slučaju, visok stupanj pripadnosti očekujemo od elemenata koji predstavljaju male struje **i** visoke napone. Što je struja dalja od "male struje" i

napon dalji od "visokog napona", takav par će imati manji stupanj pripadnosti relaciji, jer se pravilo ne odnosi na takvu situaciju. Stoga se kao mogući model implikacije za **AKO-ONDA** pravilo kojemu dajemo lokalni karakter nameću upravo $t$-norme (odnosno logički I). Računamo li implikaciju kao minimum (što smo radili i kada smo gradili kartezijev produkt dvaju neizrazitih skupova), za prethodno **AKO-ONDA** pravilo dobit ćemo sljedeću relaciju (provjerite):

$$R = \begin{bmatrix} 0.1 & 0.5 & 1.0 \\ 0.1 & 0.5 & 0.6 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}.$$

Uočite sada kako su raspoređene vrijednosti mjere pripadnosti u relaciji. Dajmo sada i formalnu definiciju implikacije koja odgovara lokalnoj interpretaciji **AKO-ONDA** pravila.

**Implikacija *Mamdani***

Ova implikacija definirana je izrazom:

$$(p \to q) \equiv (p \wedge q)$$

pri čemu se kao operator $\wedge$ koristi ili Zadehov operator minimuma ili Algebarski produkt. Stoga je, u slučaju da se koristi Zadehov operator minimuma, pripadna relacija definirana funkcijom pripadnosti:

$$\mu_R(x, y) = \min(\mu_A(x), \mu_B(y)) \text{ tj. } R = A \times B$$

a u slučaju da se koristi Algebarski produkt, pripadna relacija definirana je funkcijom pripadnosti:

$$\mu_R(x, y) = \mu_A(x) \cdot \mu_B(y).$$

Uočite da je ovakva definicija implikacije i računski najjednostavnija. Stoga se upravo ova definicija implikacije najčešće koristi u neizrazitom upravljanju.

## 5.4 Neizrazito upravljanje

Jedno od područja u kojima se koristi neizrazita logika jest upravljanje, odnosno neizrazito upravljanje. Pogledajmo koje se metode koriste i na koji način rade sustavi neizrazitog upravljanja. Najčešće se za opisivanje ponašanja sustava koriste **AKO-ONDA** pravila. Npr. **AKO** $x$ je $A$ **ONDA** $y$ je $B$. Pri tome je $A$ neizraziti skup definiran nad univerzalnim skupom $U_1$, a $B$ je neizraziti skup definiran nad univerzalnim skupom $U_2$. Ulazi u sustav mogu ali i ne moraju biti neizraziti (često nisu što olakšava izvođenje zaključaka). Postavlja se pitanje kako sada na temelju ulazne vrijednosti

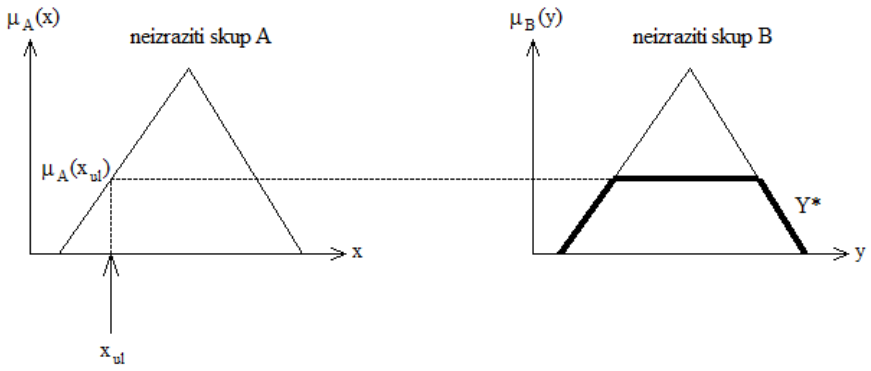
dobiti odgovarajuću izlaznu vrijednost. I opet, metoda je nekoliko, i naravno, nema one koja bi bila najbolja u svim situacijama.

Prvo ćemo opisati metodu koju smo već koristili. Ako imamo običan ulaz (npr. realni broj $b$), transformirat ćemo taj broj u neizraziti skup $x$ (npr. "približno broj $b$", ili "točno broj $b$"). Neizrazito pravilo **AKO** $x$ je $A$ **ONDA** $y$ je $B$ predstavlja relaciju $R$ koja se dobije kao implikacija (metode su opisane u podpoglavlju 5.3.1, primjerice npr. $R = A \times B$). Koristeći pravilo kompozicije izvodimo vrijednost izlaza $y = x \circ R$, pri čemu je $y$ opet neizraziti skup nad $U_2$. Ako kao izlaz ne želimo neizraziti skup, još je potrebno provesti postupak dekodiranja neizrazitosti (poglavlje 5.2).

Drugu metodu koja se često koristi opisuje slika 5.5. Postupak se provodi tako da se izračuna s kojim stupnjem pripadnosti ulazna vrijednosti $x_{ul}$ pripada neizrazitom skupu $A$. Zatim se kao rezultat $y^*$ uzima modifikacija neizrazitog skupa $B$, na sljedeći način:

$$\mu_{y^*}(y) = \begin{cases} \mu_B(y), & \mu_B(y) \leq \mu_A(x_{ul}) \\ \mu_A(x_{ul}), & \mu_B(y) > \mu_A(x_{ul}) \end{cases} .$$

Ovaj postupak još se zove odsijecanje (engl. *clipping*). Kao rezultat se opet dobije neizraziti skup $y^*$, pa ako je potrebno, može se još izvršiti i dekodiranje neizrazitosti.



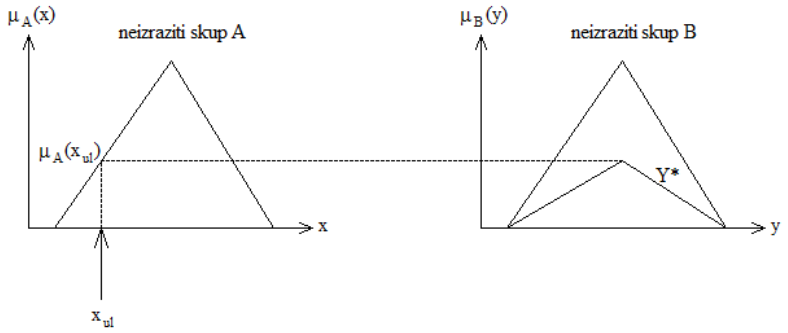

Slika 5.5: Izvođenje zaključka odsijecanjem.

Treća metoda samo je mala modifikacija upravo pokazanog postupka, slika 5.6.

Umjesto da skup $B$ odsiječemo na visini $\mu_A(x_{ul})$, rezultantni skup $y^*$ dobije se skaliranjem skupa $B$ na visinu $\mu_A(x_{ul})$:

$$\mu_{y^*}(y) = \mu_A(x_{ul}) \cdot \mu_B(y).$$

Sada smo se upoznali s razrješavanjem jednog **AKO-ONDA** pravila. No što ako takvih pravila ima više?




Slika 5.6: Izvođenje zaključka skaliranjem.

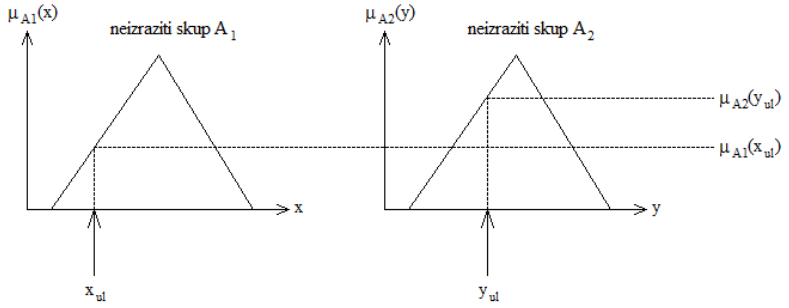
- **AKO** $x$ je $A_1$ **ONDA** $y$ je $B_1$
- **AKO** $x$ je $A_2$ **ONDA** $y$ je $B_2$
- ...
- **AKO** $x$ je $A_n$ **ONDA** $y$ je $B_n$

U tom slučaju možemo postupiti na dva načina. Prvi način svodi se na izračunavanje unije svih relacija koje se dobiju iz ovih **AKO-ONDA** pravila. Naime, $i$-to **AKO-ONDA** pravilo prestavlja relaciju $R_i$. Skup od $n$ **AKO-ONDA** pravila tada predstavlja relaciju $R = \cup_{i=1}^{n} R_i$. Sada možemo pravilom kompozicije izvoditi zaključivanje $y = x \circ R$.

Možemo postupiti i na drugi način. Možemo izvoditi pravilo po pravilo te dobiti $n$ izlaznih neizrazitih skupova. Izvođenjem $i$-tog pravila dobiti ćemo izlazni neizraziti skup $y_i^*$. Kada ovo izračunamo za sva pravila, konačni izlazni skup $y^*$ možemo dobiti kao uniju svih pojedinačnih izlaznih skupova: $y^* = \cup_{i=1}^{n} y_i^*$.

I na kraju, **AKO-ONDA** pravila mogu biti puno složenija od oblika "**AKO** $x$ je $A$ **ONDA** $y$ je $B$". Pogledajmo još kako se razrješavaju takve situacije. Ukoliko imamo oblik: "**AKO** $x$ je $A_1$ I $x$ je $A_2$ ILI $x$ je $A_3$ **ONDA** $y$ je $B$", pravilo se može svesti na oblik "**AKO** $x$ je $A^*$ **ONDA** $y$ je $B$" budući da sve propozicije imaju istu varijablu $x$ pa se "I", "ILI" i slično može izračunati prema tablici 5.1. No što uraditi s pravilima oblika: "**AKO** $x$ je $A_1$ I $y$ je $A_2$ **ONDA** $z$ je $B$"? Varijabla $x$ i neizraziti skup $A_1$ definirani su nad univerzalnim skupom $U_1$ dok su varijabla $y$ i neizraziti skup $A_2$ definirani su nad univerzalnim skupom $U_2$. Očito ne možemo izraze "I", "ILI" i sl. računati kao presjek ili uniju, jer nisu po domenama kompatibilni. U tom slučaju postupa se na način kako to pokazuje slika 5.7.

Ideja je da se razriješe "visine" za svaku varijablu koja se pojavljuje: kod nas su to varijable $x$ i $y$ pa imamo visine $\mu_{A_1}(x_{ul})$ i $\mu_{A_2}(y_{ul})$. Zatim



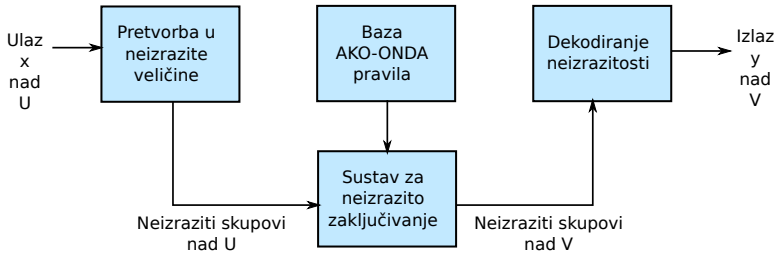

Slika 5.7: Izvođenje pri složenim izrazima.

se ovisno o vezniku između tih varijabli rezultantna visina računa pomoću neke od $t$-normi (ako je veznik I), pomoću neke od $s$-normi (ako je veznik ILI), odnosno pomoću nekog od generaliziranih komplemenata (ako imamo negaciju). Kada se izračuna rezultantna visina, može se izvesti odsijecanje ili pak skaliranje, ovisno o tome na koji način dalje želimo računati.

Prethodno opisani metode djeluju dosta "proizvoljno" definirane. Međutim, u pozadini postoji i formalni model koji ćemo opisati u nastavku.

### 5.4.1 Zaključivanje u sustavima neizrazitog upravljanja

Sustavi neizrazitog upravljanja koje ovdje razmatramo su *produkcijski sustavi*. Njihova baza znanja sastoji se od niza **AKO-ONDA** pravila. Građa jednog takvog sustava prikazana je na slici 5.8.




Slika 5.8: Građa sustava neizrazitog upravljanja.

Pretpostavimo da sustav kao ulaz dobiva $r$ podataka – označit ćemo ih s $x_1$ do $x_r$ koji kao vrijednosti poprimaju elemente univerzalnih skupova $U_1$ do $U_n$. Također neka se baza pravila sastoji od $n_R$ pravila kako je prikazano u nastavku, pri čemu je $A_ij$ neizraziti skup definiran nad univerzalim skupom $U_j$. Svi neizraziti skupovi $B_i$ definirani su nad istim univerzalnim skupom $V$.

- **AKO** $x_1$ je $A_{11}$ **I** $\cdots$ **I** $x_r$ je $A_{1r}$ **ONDA** $y$ je $B_1$
- ...
- **AKO** $x_1$ je $A_{i1}$ **I** $\cdots$ **I** $x_r$ je $A_{ir}$ **ONDA** $y$ je $B_i$
- ...
- **AKO** $x_1$ je $A_{n_R1}$ **I** $\cdots$ **I** $x_r$ je $A_{n_Rr}$ **ONDA** $y$ je $B_{n_R}$

$x_1$ do $x_r$ su, općenito govoreći, neizraziti skupovi koji predstavljaju trenutne ulaze sustava.

U opisanom scenariju razvijena su dva načina izvođenja zaključaka:

- zaključivanje temeljeno na kompoziciji te
- zaključivanje temeljeno na pojedinačnim pravilima.

U nastavku ćemo razmotriti oba načina.

### Zaključivanje temeljeno na kompoziciji

Razmotrimo $i$-to pravilo. Antecedent se sastoji od konjunkcije $r$ zahtjeva. Čitav antecedent možemo prikazati kao neizrazitu relaciju koja je definirana nad $U_1 \times \cdots \times U_r$ kao kartezijev produkt neizrazitih skupova $A_{i1}$ do $A_{ir}$. Drugim riječima, definiramo relaciju čija je funkcija pripadnosti dana sljedećim izrazom:

$$\mu_{A_{i1} \times \cdots \times A_{ir}}(x_1, \ldots, x_r) = T(\mu_{A_{i1}}(x_1), \ldots, \mu_{A_{ir}}(x_r)).$$

U ovom izrazu $T(...)$ predstavlja bilo koju $t$-normu a uobičajeno je koristiti minimum. Činjenica da smo dobili relaciju nas ne treba smetati – relacija jest neizraziti skup. No sada kada smo to utvrdili, navedeno **AKO-ONDA** pravilo možemo promatrati kao pravilo čiji je antecedent jedan neizraziti skup a konsekvent drugi neizraziti skup. Stoga ćemo čitavo pravilo predstaviti kao jednu relaciju definiranu nad univerzalnim skupom $U_1 \times \cdots \times U_r \times V$ uporabom bilo kojeg od operatora implikacije koje smo prethodno opisali.

Temeljem $i$-tog pravila dobili smo relaciju $R_i$. Kako imamo $n_R$ pravila, dobit ćemo ukupno $n_R$ relacija: $R_1, \cdots, R_{n_R}$.

Sljedeći korak je izgradnja konačne relacije $R$ koja će predstavljati cjelokupni sustav. Prisjetimo se sada prethodne diskusije o lokalnosti semantike pravila. Ako **AKO-ONDA** pravila imaju lokalnu semantiku (odnosno koristimo implikaciju takvih karakteristika), tada svako pravilo generira relaciju koja ima veće vrijednosti funkcije pripadnosti samo u onom podprostoru na koji se to pravilo odnosi dok je na svim ostalim mjestima vrijednost funkcije pripadnosti nula (ili mala). Stoga se u takvom slučaju ukupna relacija

dobiva kao *unija* pojedinačnih relacija (računamo je uporabom bilo koje od $s$-normi):

$$R = R_1 \cup \ldots \cup R_{n_R}.$$

U konačnoj relaciji trebaju biti oni parovi elemenata koji su uključeni u bilo koju od relacija.

Ako pak za **AKO-ONDA** pravila koristimo implikaciju koja daje globalnu semantiku, prisjetimo se, u dobivenoj relaciji nisu vrijednost funkcije pripadnosti imat će oni parovi elemenata koji su kontradiktorni pravilu a svi ostali parovi će imati visoke vrijednosti funkcije pripadnosti. Stoga u konačnoj relaciji visok stupanj funkcije pripadnosti trebaju imati samo oni parovi elemenata koje niti jedno pravilo nije isključilo, što znači da ćemo u tom slučaju koristiti *presjek* pojedinačnih relacija (računamo ga uporabom bilo koje od $t$-normi):

$$R = R_1 \cap \ldots \cap R_{n_R}.$$

Ovime smo osigurali da je par elemenata koji je bilo koja relacija isključila isključen i u konačnoj relaciji.

Prethodne korake bilo je potrebno provesti kako bismo došli do relacije $R$ koja predstavlja cjelokupni neizraziti sustav (odnosno koja u sebi ima ugrađena sva pravila). Jednom kada imamo relaciju $R$, zaključivanje se svodi na uporabu *modus ponensa*. Pretpostavimo da je korisnik kao ulaz neizrazitog sustava definirao $x_1$ je $C_1$, ..., $x_r$ je $C_r$, gdje su $C_1$ do $C_r$ neizraziti skupovi. Gradimo neizraziti skup $C$ kao kartezijev produkt neizrazitih skupova $C_1 \times \cdots \times C_r$ i potom izvodimo kao kompoziciju neizrazitog skupa $C$ i relacije $R$:

Kompozicijsko zaključivanje

$$\mu_B(y) = \sup_{x_1,\ldots,x_r} T(\mu_C(x_1,\ldots,x_r), \mu_R(x_1,\ldots,x_r,y)). \tag{5.10}$$

Izračun zaključka, kako se vidi iz prethodnog pravila, računski će biti vrlo skup: supremum se traži po kartezijevom produktu univerzalnih skupova nad kojima su definirane ulazne varijable. Stoga se u praksi često uvodi jedno pojednostavljenje koje će osigurati efikasniji izračun zaključka: uvodi se restrukcija da neizraziti skupovi $C_i$ moraju biti jednoelementni neizraziti skup (singletoni). Označimo u svakom od neizrazitih skupova $C_i$ oznakom $x_i^*$ taj element domene za koji je $\mu_{C_i}$ jednaka 1:

$$\mu_{C_i}(x_i) = \begin{cases} 1, & x_i = x_i^* \\ 0, & \text{inače} \end{cases}$$

i pogledajmo malo bolje što se pretražuje u izrazu 5.10. U slučaju da su $C_i$ jednoelementni neizraziti skupovi, funkcija pripadnosti $\mu_C(x_1,\ldots,x_r)$ poprimit će vrijednost 1 samo kada je $x_1 = x_1^* \wedge \ldots \wedge x_r = x_r^*$ dok će u svim drugim slučajevima biti 0. No iz rubnih uvjeta $t$-normi znamo da će rezultat $t$-norme biti 0 ako je jedan od operanada jednak 0. Slijedi da se traži supremum (maksimum) od jako puno nula te one jedne specifične situacije u kojoj

je $x_1 = x_1^* \wedge \ldots \wedge x_r = x_r^*$ za koju je $\mu_C(x_1^*, \ldots, x_r^*) = 1$ pa se tada računa $t$-norma između 1 i $\mu_R(x_1^*, \ldots, x_r^*, y)$; opet iz rubnih uvjeta $t$-normi znamo da je tada rezultat jednak upravo ovom posljednjem operandu. Slijedi da se cjelokupni zaključak tada svodi na:

Pojednostavljenje

$$\mu_B(y) = \mu_R(x_1^*, \ldots, x_r^*, y). \tag{5.11}$$

što se, uz pretpostavku da je $R$ unaprijed izračunat, može vrlo efikasno odrediti te nije potrebno provoditi vremenski zahtjevnu kompoziciju.

### Zaključivanje temeljeno na pojedinačnim pravilima

Kod zaključivanje temeljenog na kompoziciji prvi korak je bio izgradnja relacije koja u sebe ugrađuje znanje dano kroz sva pravila sustava. Jednom kada je relacija izgrađena, pravila se više ne koriste. Umjesto pomoću njih, za svaki zadani ulaz radi se njegova kompozicija s dobivenom relacijom i tako generira izlaz.

Za razliku od zaključivanja temeljenog na kompoziciji, zaključivanje temeljeno na pravilima čitavo vrijeme radi direktno s pravilima. Pretpostavimo da je korisnik kao ulaz neizrazitog sustava definirao $x_1$ je $C_1$, ..., $x_r$ je $C_r$, gdje su $C_1$ do $C_r$ neizraziti skupovi.

1. Za svako pravilo potrebno je izgraditi relaciju koja predstavlja to pravilo. Konkretno, za $i$-to pravilo gradimo relaciju $R_i(x_1, \ldots, x_r, y)$ na jednak način kako smo to napravili kod zaključivanja temeljenog na kompoziciji.

2. Označimo s $C(x_1, \ldots, x_r)$ relaciju koja je dobivena kao kartezijev produkt neizrazitih skupova $C_i$ (ulaza u sustav), koristeći odabranu $t$-normu.

3. Za svako pravilo računamo neizrazit skup koji predstavlja lokalni zaključak $B_i'$ kao generalizirani modus ponens pri čemu je prva premisa "$x$ je $C$" a druga promatrano pravilo. Tako dobiveni zaključak je neizraziti skup sa sljedećom funkcijom pripadnosti:

   Izvođenje pojedinačnih zaključaka

   $$\mu_{B_i'}(y) = \sup_{x_1,\ldots,x_r} T(\mu_C(x_1, \ldots, x_r), \mu_{R_i}(x_1, \ldots, x_r, y)). \tag{5.12}$$

   Provođenjem ovog koraka završavamo s $n_R$ različitih zaključaka (svako pravilo generiralo je jedan; to su neizraziti skupovi $B_1'$ do $B_{n_R}'$).

4. Zaključke svakog pravila sada je potrebno kombinirati u jedan jedinstveni zaključak i to kao uniju zaključaka (ako je korištena implikacija koja pravilu daje lokalnu semantiku):

   Kombiniranje zaključaka

   $$\mu_{B'}(y) = \bigcup_{i=1,\ldots,n_R} \mu_{B_i'}(y) = S(\mu_{B_1'}(y), \ldots, \mu_{B_{n_R}'}(y)) \tag{5.13}$$

ili kao presjek zaključaka (ako je korištena implikacija koja svakom pravilu daje globalnu semantiku):

$$\mu_{B'}(y) = \bigcap_{i=1,\ldots,n_R} \mu_{B'_i}(y) = T(\mu_{B'_1}(y), \ldots, \mu_{B'_{n_R}}(y)). \tag{5.14}$$

**Stroj za zaključivanje koji se temelji na minimumu**

Stroj za zaključivanje koji se temelji na minimumu (engl. *Minimum inference engine*) je implementacija zaključivanja koje:

- se temelji na zaključivanju prema pojedinačnim pravilima,
- kao operator implikacije koristi Mamdanijevu minimum-implikaciju,
- za sve $s$-norme koristi operator maksimuma,
- za sve $t$-norme koristi operator minimuma te
- pojedinačne zaključke svakog pravila kombinira unijom u konačan zaključak.

$$\mu_{B'}(y) = \max_{i=1,\ldots,n_R} \left( \sup_{x_1,\ldots,x_r} \min(\mu_C(x_1, \ldots, x_r), \mu_{R_i}(x_1, \ldots, x_r, y)) \right) \tag{5.15}$$

$$= \max_{i=1,\ldots,n_R} \left\{ \sup_{x_1,\ldots,x_r} \min\left(\mu_C(x_1, \ldots, x_r), \min(\mu_{A_{i1}}(x_1), \ldots, \mu_{A_{ir}}(x_r), \mu_{B_i}(y))\right) \right\} \tag{5.16}$$

$$= \max_{i=1,\ldots,n_R} \left( \sup_{x_1,\ldots,x_r} \min(\mu_{C_1}(x_1), \ldots, \mu_{C_r}(x_r), \mu_{A_{i1}}(x_1), \ldots, \mu_{A_{ir}}(x_r), \mu_{B_i}(y)) \right) \tag{5.17}$$

U slučaju da se kao ulaz koriste singleton neizraziti skupovi i ako označimo s $x_1^*$ do $x_r^*$ konkretne vrijednosti koje jedine pripadaju ulaznim neizrazitim skupovima $C_1$ do $C_r$, tada se prethodni izraz pojednostavljuje u:

$$\mu_{B'}(y) = \max_{i=1,\ldots,n_R} \left\{ \min(\mu_{A_{i1}}(x_1^*), \ldots, \mu_{A_{ir}}(x_r^*), \mu_{B_i}(y)) \right\}. \tag{5.18}$$

**Stroj za zaključivanje koji se temelji na produktu**

Stroj za zaključivanje koji se temelji na produktu (engl. *Product inference engine*) je implementacija zaključivanja koje:

- se temelji na zaključivanju prema pojedinačnim pravilima,

- kao operator implikacije koristi Mamdanijevu produkt-implikaciju,
- za sve $s$-norme koristi operator maksimuma,
- za sve $t$-norme koristi operator produkt te
- pojedinačne zaključke svakog pravila kombinira unijom u konačan zaključak.

$$\mu_{B'}(y) = \max_{i=1,\ldots,n_R} \left( \sup_{x_1,\ldots,x_r} \mu_C(x_1,\ldots,x_r) \cdot \mu_{R_i}(x_1,\ldots,x_r,y) \right) \tag{5.19}$$

$$= \max_{i=1,\ldots,n_R} \left\{ \sup_{x_1,\ldots,x_r} \mu_C(x_1,\ldots,x_r) \cdot \left( \prod_{j=1,\ldots,r} \mu_{A_{ij}}(x_j) \right) \cdot \mu_{B_i}(y) \right\} \tag{5.20}$$

$$= \max_{i=1,\ldots,n_R} \left\{ \sup_{x_1,\ldots,x_r} \left( \prod_{j=1,\ldots,r} \mu_{C_j}(x_j) \right) \cdot \left( \prod_{j=1,\ldots,r} \mu_{A_{ij}}(x_j) \right) \cdot \mu_{B_i}(y) \right\} \tag{5.21}$$

U slučaju da se kao ulaz koriste singleton neizraziti skupovi i ako označimo s $x_1^*$ do $x_r^*$ konkretne vrijednosti koje jedine pripadaju ulaznim neizrazitim skupovima $C_1$ do $C_r$, tada se prethodni izraz pojednostavljuje u:

$$\mu_{B'}(y) = \max_{i=1,\ldots,n_R} \left\{ \left( \prod_{j=1,\ldots,r} \mu_{A_{ij}}(x_j^*) \right) \cdot \mu_{B_i}(y) \right\}. \tag{5.22}$$

Kod prethodna dva opisana stroja može se javiti problem male visine generiranog zaključka. Naime, ako je visina barem jednog od ulaznih neizrazitih skupova mala, ta visina ograničava i visiku generiranog rezultata (što je posebice istaknuto kod stroj za zaključivanje koji se temelji na produktu). Zbog toga ćemo još navesti i strojeve koji koriste implikacije koje pravilima daju globalnu semantiku i koji u određenoj mjeri prevladavaju ovaj problem.

### Dienes-Rescherov stroj za zaključivanje

Dienes-Rescherov stroj za zaključivanje (engl. *Dienes-Rescher inference engine*) je implementacija zaključivanja koje:

- se temelji na zaključivanju prema pojedinačnim pravilima,
- kao operator implikacije koristi Dienes-Rescherovu implikaciju,
- za sve $s$-norme koristi operator maksimuma,

- za sve $t$-norme koristi operator minimum te
- pojedinačne zaključke svakog pravila kombinira presjekom u konačan zaključak.

$$\mu_{B'}(y) = \min_{i=1,\ldots,n_R} \sup_{x_1,\ldots,x_r} \min( \mu_C(x_1,\ldots,x_r), \max(1 - \min_{j=1,\ldots,r}(\mu_{A_{ij}}(x_j)), \mu_{B_i}(y)) )$$

U slučaju da se kao ulaz koriste singleton neizraziti skupovi i ako označimo s $x_1^*$ do $x_r^*$ konkretne vrijednosti koje jedine pripadaju ulaznim neizrazitim skupovima $C_1$ do $C_r$, tada se prethodni izraz pojednostavljuje u:

$$\mu_{B'}(y) = \min_{i=1,\ldots,n_R} \max(1 - \min_{j=1,\ldots,r}(\mu_{A_{ij}}(x_j^*)), \mu_{B_i}(y))$$

**Zadehov stroj za zaključivanje**

Zadehov stroj za zaključivanje (engl. *Zadeh inference engine*) je implementacija zaključivanja koje:

- se temelji na zaključivanju prema pojedinačnim pravilima,
- kao operator implikacije koristi Zadehovu implikaciju,
- za sve $s$-norme koristi operator maksimuma,
- za sve $t$-norme koristi operator minimum te
- pojedinačne zaključke svakog pravila kombinira presjekom u konačan zaključak.

$$\mu_{B'}(y) = \min_{i=1,\ldots,n_R} \sup_{x_1,\ldots,x_r} \min( \mu_C(x_1,\ldots,x_r), \max(\min(\min_{j=1,\ldots,r}(\mu_{A_{ij}}(x_j)), \mu_{B_i}(y)), 1 - \min_{j=1,\ldots,r}(\mu_{A_{ij}}(x_j))) )$$

U slučaju da se kao ulaz koriste singleton neizraziti skupovi i ako označimo s $x_1^*$ do $x_r^*$ konkretne vrijednosti koje jedine pripadaju ulaznim neizrazitim skupovima $C_1$ do $C_r$, tada se prethodni izraz pojednostavljuje u:

$$\mu_{B'}(y) = \min_{i=1,\ldots,n_R} \max(\min(\min_{j=1,\ldots,r}(\mu_{A_{ij}}(x_j^*)), \mu_{B_i}(y)), 1 - \min_{j=1,\ldots,r}(\mu_{A_{ij}}(x_j^*)))$$

**Lukasiewiczev stroj za zaključivanje**

Lukasiewiczev stroj za zaključivanje (engl. *Lukasiewicz inference engine*) je implementacija zaključivanja koje:

- se temelji na zaključivanju prema pojedinačnim pravilima,
- kao operator implikacije koristi Lukasiewiczevu implikaciju,
- za sve $s$-norme koristi operator maksimuma,
- za sve $t$-norme koristi operator minimum te
- pojedinačne zaključke svakog pravila kombinira presjekom u konačan zaključak.

$$\begin{aligned}\mu_{B'}(y) = \min_{i=1,\ldots,n_R} \sup_{x_1,\ldots,x_r} \min( \\ \mu_C(x_1,\ldots,x_r), \\ \min(1, 1 - (\min_{j=1,\ldots,r}(\mu_{A_{ij}}(x_j))) + \mu_{B_i}(y)) \\ )\end{aligned}$$

što se dalje može pojednostavniti u:

$$\begin{aligned}\mu_{B'}(y) = \min_{i=1,\ldots,n_R} \sup_{x_1,\ldots,x_r} \min( \\ \mu_C(x_1,\ldots,x_r), \\ 1 - (\min_{j=1,\ldots,r}(\mu_{A_{ij}}(x_j))) + \mu_{B_i}(y) \\ )\end{aligned}$$

U slučaju da se kao ulaz koriste singleton neizraziti skupovi i ako označimo s $x_1^*$ do $x_r^*$ konkretne vrijednosti koje jedine pripadaju ulaznim neizrazitim skupovima $C_1$ do $C_r$, tada se prethodni izraz pojednostavljuje u:

$$\mu_{B'}(y) = \min_{i=1,\ldots,n_R}(1, 1 - \min_{j=1,\ldots,r}(\mu_{A_{ij}}(x_j^*)) + \mu_{B_i}(y))$$

## 5.5 Primjer sustava neizrazitog upravljanja: okrenuto njihalo

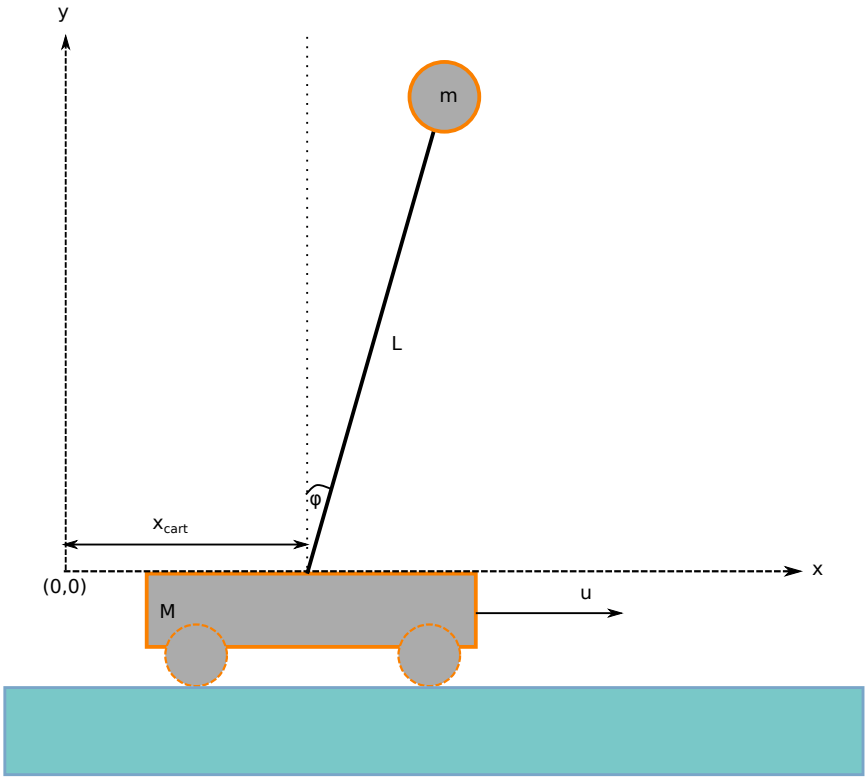
Okrenuto njihalo jedan je od sustava koji se koriste za ispitivanje i demonstraciju sustava upravljanja. Taj je sustav nelinearan i moguće ga je opisati

preko dvije diferencijalne jednadžbe drugog reda, što je izvedeno u podpoglavlju 5.5.2. Te su jednadžbe sljedeće.

$$(M+m)\ddot{x} - mL\sin(\phi)\dot{\phi}^2 + mL\cos(\phi)\ddot{\phi} = u \tag{5.23}$$

$$m\ddot{x}\cos(\phi) + mL\ddot{\phi} = mg\sin(\phi). \tag{5.24}$$

Sustav je prikazan na slici 5.9.

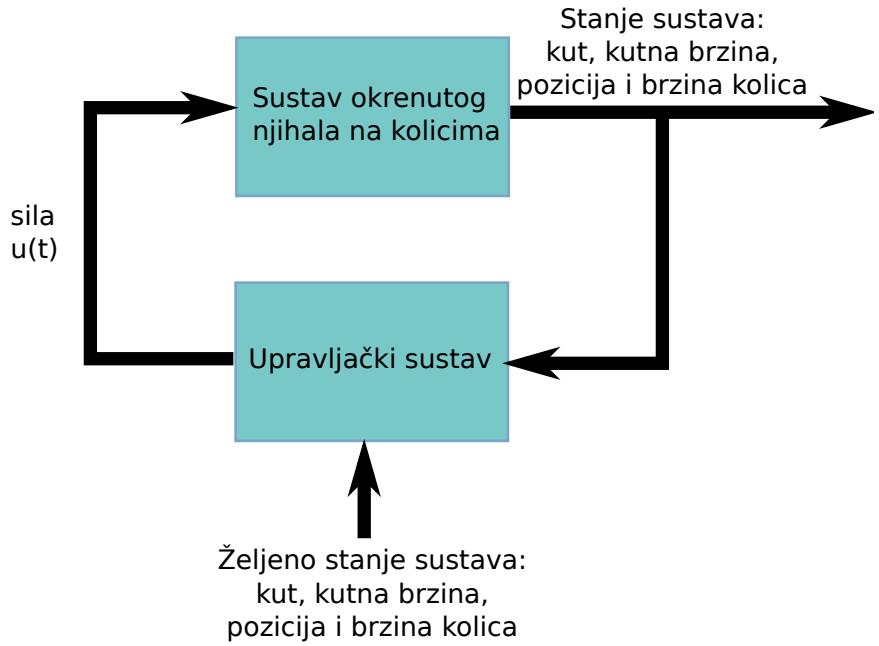



Slika 5.9: Obrnuto njihalo.

Na kolica mase $M$ koja se mogu gibati samo vodoravno montirana je šipka duljine $L$ na zglob koji nema trenja niti drugih gubitaka i koji omogućava rotaciju šipke samo u $xy$ ravnini (drugim riječima, gledano sliku, ili prema naprijed, ili prema natrag). Na vrh šipke montirana je kugla mase $m$. Za potrebe izvoda pretpostavit ćemo da je šipka zanemarive mase a kolica i kuglu aproksimirat ćemo točkastim masama. U svakom trenutku na kolica djeluje vanjska sila $u(t)$ čije je djelovanje strogo horizontalno.

Pogledajmo na tren sliku 5.9 i pretpostavimo da nema vanjske sile: uslijed sile gravitacije koja djeluje na kuglu očekujemo da će štap početi rotirati prema desno. Potrebno je razviti upravljački sustav koji će određivati iznos i smjer sile koju treba primijeniti sa svrhom da se štap postavi u okomiti

položaj (kut $\phi = 0$) te da se kolica pozicioniraju u ishodište koordinatnog sustava ($x_{cart} = 0$). Neovisno o načinu izvedbe takvog upravljačkog sustava, njegovo uklapanje u okolinu prikazano je na slici 5.10.

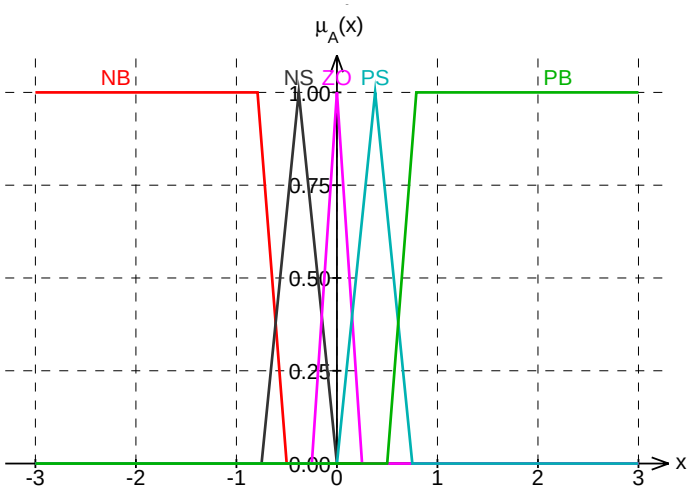



Slika 5.10: Sustav i upravljački sklop.

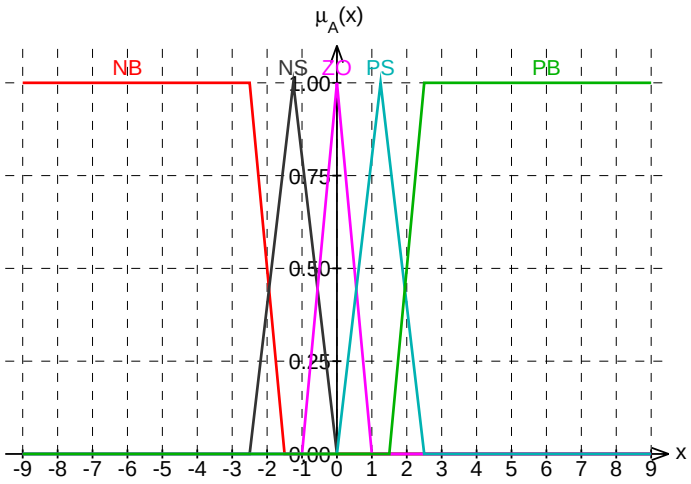
Upravljački sklop od sustava može dobiti četiri podatka koja ujedno čine i stanje sustava:

1. vodoravni položaj kolica $x_{cart}$ (u $m$),
2. brzinu kojom se kolica gibaju $v_{cart}$ (u $m/s$),
3. kut koji njihalo zatvara s okomicom (u $rad$; za okomiti položaj $\phi = 0$, za otklon u desno kut je pozitivan, za otklon u lijevo kut je negativan) te
4. kutnu brzinu (u $rad/s$; pozitivna vrijednost znači da se kut povećava, negativna da se smanjuje).

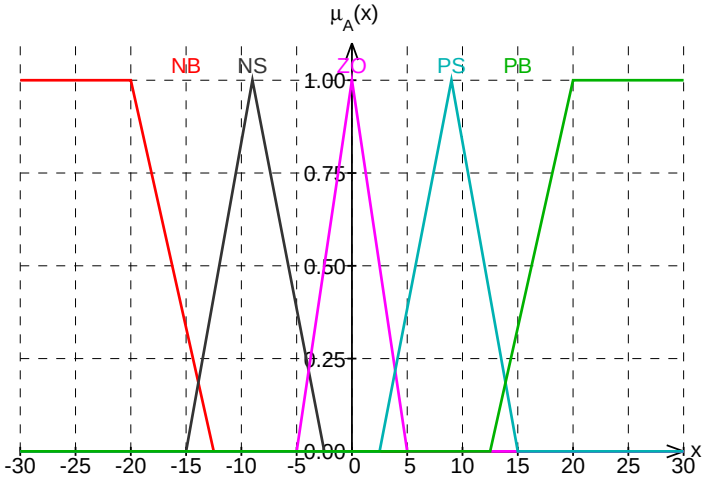
Temeljem tih podataka zadaća upravljačkog sustava je prilagoditi silu kja će djelovati na kolica a u svrhu stabilizacije njihava u okomitom položaju. Opisana konfiguracija predstavlja zatvorenu petlju: u nekom trenutku sustav se nalazi u nekom stanju, temeljem tog stanja upravljački sklop generira novi ulaz za sustav; vrijednost ulaza pak u sljedećem trenutku dovodi do potecijalne promjene stanja i postupak se zatvara u krug.




(a) Jezična varijabla KUT.




(b) Jezična varijabla KUTNA BRZINA.




(c) Jezična varijabla SILA.

Slika 5.11: Jezične varijable sustava za upravljanje okrenutim njihalom.

Upravljački sustav koji ćemo opisati razmatrat će samo dvije od varijabli stanja: trenutni kut te trenutnu kutnu brzinu. Temeljem tih podataka sustav će generirati iznos sile koji je potrebno primijeniti na kolica. Definicije triju odgovarajućih jezičnih varijabli dane su u nastavku a grafički prikaz daje slika 5.11.

Neizraziti skupovi koji pripadaju jezičnoj varijabli *KUT* definirani su na sljedeći način.

$$\begin{array}{ll} \text{NB} & \mu_{NB}(x) = L(x; -0.79, -0.50) \\ \text{NS} & \mu_{NS}(x) = \Lambda(x; -0.75, -0.38, 0.00) \\ \text{ZO} & \mu_{ZO}(x) = \Lambda(x; -0.25, 0.00, 0.25) \\ \text{PS} & \mu_{PS}(x) = \Lambda(x; 0.00, 0.38, 0.75) \\ \text{PB} & \mu_{PB}(x) = \Gamma(x; 0.50, 0.79) \end{array}$$

Neizraziti skupovi koji pripadaju jezičnoj varijabli *KUTNA BRZINA* definirani su na sljedeći način.

$$\begin{array}{ll} \text{NB} & \mu_{NB}(x) = L(x; -2.5, -1.5) \\ \text{NS} & \mu_{NS}(x) = \Lambda(x; -2.5, -1.25, 0.0) \\ \text{ZO} & \mu_{ZO}(x) = \Lambda(x; -1.0, 0.0, 1.0) \\ \text{PS} & \mu_{PS}(x) = \Lambda(x; 0.0, 1.25, 2.5) \\ \text{PB} & \mu_{PB}(x) = \Gamma(x; 1.5, 2.5) \end{array}$$

Konačno, neizraziti skupovi koji pripadaju jezičnoj varijabli *SILA* definirani su na sljedeći način.

$$\begin{array}{ll} \text{NB} & \mu_{NB}(x) = L(x; -20.0, -12.5) \\ \text{NS} & \mu_{NS}(x) = \Lambda(x; -15.0, -9, -2.5) \\ \text{ZO} & \mu_{ZO}(x) = \Lambda(x; -5.0, 0.0, 5.0) \\ \text{PS} & \mu_{PS}(x) = \Lambda(x; 2.5, 9, 15.0) \\ \text{PB} & \mu_{PB}(x) = \Gamma(x; 12.5, 20.0) \end{array}$$

Sustav neizrazitog upravljanja koristit će pravila oblika:
**AKO** *kutna brzina* je $A_{i1}$ **I** *kut* je $A_{i2}$ **TADA** *sila* je $B_i$
pri čemu će neizraziti skupovi $A_{i1}$ biti definirani nad univerzalnim skupom $[-9, +9]$, neizraziti skupovi $A_{i2}$ bit će definirani nad univerzalnim skupom $[-3, +3]$ dok će neizraziti skupovi $B_i$ biti definirani nad univerzalnim skupom $[-35, +35]$. Ulazne vrijednosti pretvarat će se u jednoelementne neizrazite skupove (singletone). Stroj za zaključivanje koji ćemo koristiti je stroj koji se temelji na produktu.

Kako jezične varijable *KUTNA BRZINA* i *KUT* svaka može poprimiti po 5 različitih temeljnih vrijednosti, slijedi da ćemo imati $5 \times 5 = 25$ pravila. Pravila ćemo sažeto prikazati tablicom u nastavku: za svaki redak (kutnu brzinu) i svaki stupac (kut) sadržaj ćelije sadrži silu koju daje pravilo koje predstavlja odabranu kutnu brzinu i kut.

| Kutna brzina \ Kut | NB | NS | ZO | PS | PB |
|---|---|---|---|---|---|
| NB | NB | NB | NB | NS | ZO |
| NS | NB | NB | NS | ZO | PS |
| ZO | NB | NS | ZO | PS | PB |
| PS | NS | ZO | PS | PB | PB |
| PB | ZO | PS | PB | PB | PB |

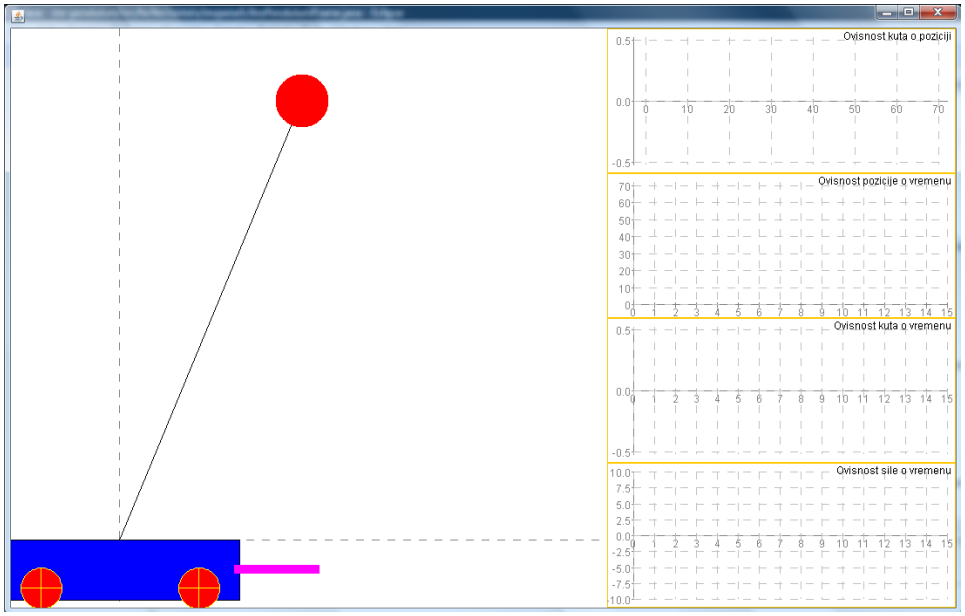
Tako primjerice, ako je kutna brzina negativna i velika (prvi redak; kugla se kreće u lijevo) i ako je kut negativan i velik (prvi stupac; kugla je već daleko lijevo od ravnotežnog položaja), potrebno je na kolica djelovati silom velikog iiznosa koja je negativna (djeluje u lijevo) kako bi se stvorio zakretni moment koji će zaustaviti pad kugle i kutnu brzinu pokušati okrenuti u pozitivnu; takva sila modelirana je neizrazitim skupom *NB* što je upravo sadržaj prve ćelije. Sličnom analizom možemo proanalizati svih 25 situacija i odrediti kakvom silom trebamo djelovati.

Primijetite da je tablica u određenom smislu simetrična: situacija u gornjem lijevom uglu tablice opisuje kuglu s lijeve strane koja pada lijevo dok donji desni ugao opisuje situaciju u kojoj je kugla desno i pada desno: zaključak što treba poduzeti je isti (u smislu iznosa sile), samo se mijenja smjer sile, pa kako je u gornjem lijevom uglu zaključak *NB*, u donjem desnom je *PB*.

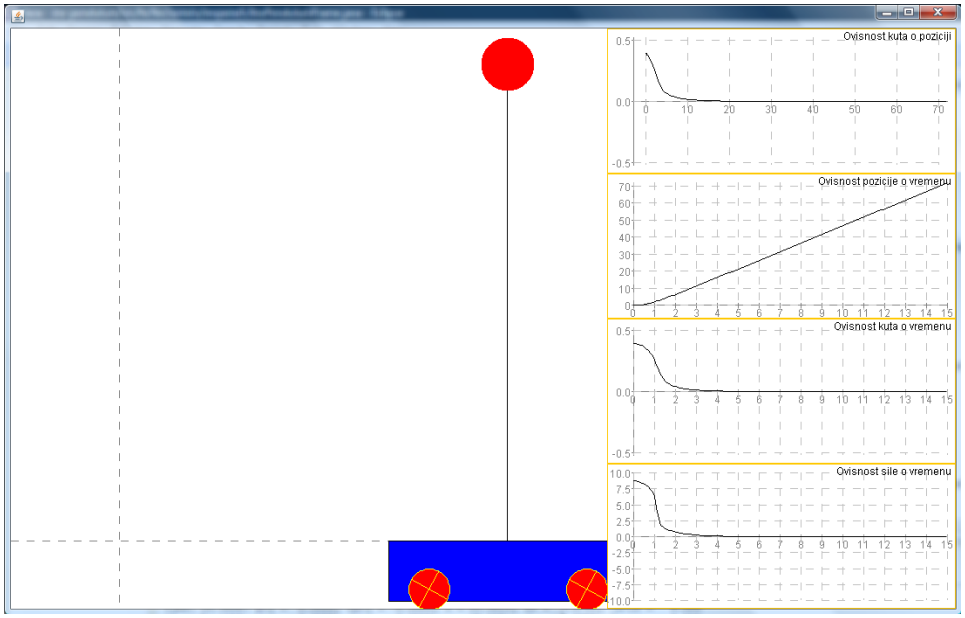
Rad opisanog neizrazitog upravljačkog sustava prikazan je na slici 5.12. Parametri sustava kojim je upravljano su: $M = 2\,kg$, $m = 0.1\,kg$, $L = 0.5\,m$, $g = 9.81\,m/s^2$. Kut pod kojim je početno postavljeno njihalo je $\frac{\pi}{8}$. Ponašanje njihala simulirano je metodom Runge-Kutta četvrtog reda uz korak integracije $2\,ms$. Isti vremenski razmak korišten je i od strane razvijenog sustava neizrazitog upravljanja: svakih $2\,ms$ očitavan je ulaz i generirana je odgovarajuća sila.

Za potrebe dekodiranja neizrazitosti domena nad kojom je definiran neizraziti skup *SILA* diskretizirana je na elemente $\{-35, -34.9, -34.8, \ldots, 34.8, 34.9, 35\}$. Potom je dekodiranje neizrazitosti provedeno metodom pronalaska težišta (engl. *Center of Area*).

Je li opisani sustav uspješan u stabilizaciji njihala? Slika 5.12a prikazuje početno stanje a slika 5.12b stanje nakon 15 sekundi. Pogledamo li graf ovisnosti kuta o vremenu (treći po redu s desne strane na slici), vidimo da je odgovor pozitivan: upravljački sustav je kut uspio svesti na kut od 0 radijana. Međutim, položaj kolica nije niti blizu početnog položaja; dapače, pogledamo li graf ovisnosti položaja kolica o vremenu (drugi po redu s desne strane na slici), uočit ćemo da se nakon početnih promjena položaj kolica u odnosu na početni položaj linearno mijenja – kolica se udaljavaju konstantnom brzinom i na kraju simulacije su preko $70\,m$ udaljena od početnog položaja. Ovo svakako nije željeni rezultat. Međutim, ne možemo kriviti upravljački sklop: sklop koji smo razvili kao jedini ulaz dobiva kut i kutnu brzinu: upravljački




(a) Početno stanje




(b) Nakon 15 sekundi

Slika 5.12: Prikaz rada sustava i upravljanje razvijenim upravljačkim sklopom.

sklop obje vrijednosti uspješno je sveo na nulu.

Da bismo dobili i stabilizaciju položaja, odnosno osigurali da se kolica vrate na položaj $x_{cart} = 0$ i da tamo ostanu stabilna morat ćemo modificirati razvijeni upravljački sklop. Prva modifikacija jest koristiti sve četiri raspoložive varijable stanja: kut i kutnu brzinu njihala te položaj i brzinu kolica. Potom imamo dvije mogućnosti.

1. pristup. Definirat ćemo još dvije jezične varijable: jednu za položaj kolica a drugu za brzinu kolica. Potom ćemo modificirati pravila tako da uključuju i te jezične varijable. Za slučaj da za svaku od njih definiramo po 5 temeljnih vrijednosti, prostor koji možemo prekriti pravilima penje se s $5 \times 5 = 25$ na $5 \times 5 \times 5 \times 5 = 625$. Za takav upravljački sustav svakako nam ne bi bilo jednostavno (a niti praktično) definirati pravila.

2. pristup. Integrirat ćemo informaciju o položaju i brzini kolica u kut i kutnu brzinu i time ostaviti upravljački sustav nepromijenjen.

S obzirom na jednostavnost, odabrat ćemo drugi pristup: upravljački sustav koristit će kao prividni kut stvarni kut modificiran informacijom o položaju:

$$\phi^{'} = \phi + k_1 \cdot x_{cart}$$

te kao prividnu kutnu brzinu stvarnu kutnu brzinu modificiranu informacijom o brzini kolica:

$$\dot{\phi}^{'} = \dot{\phi} + k_2 \cdot \dot{x}_{cart}.$$

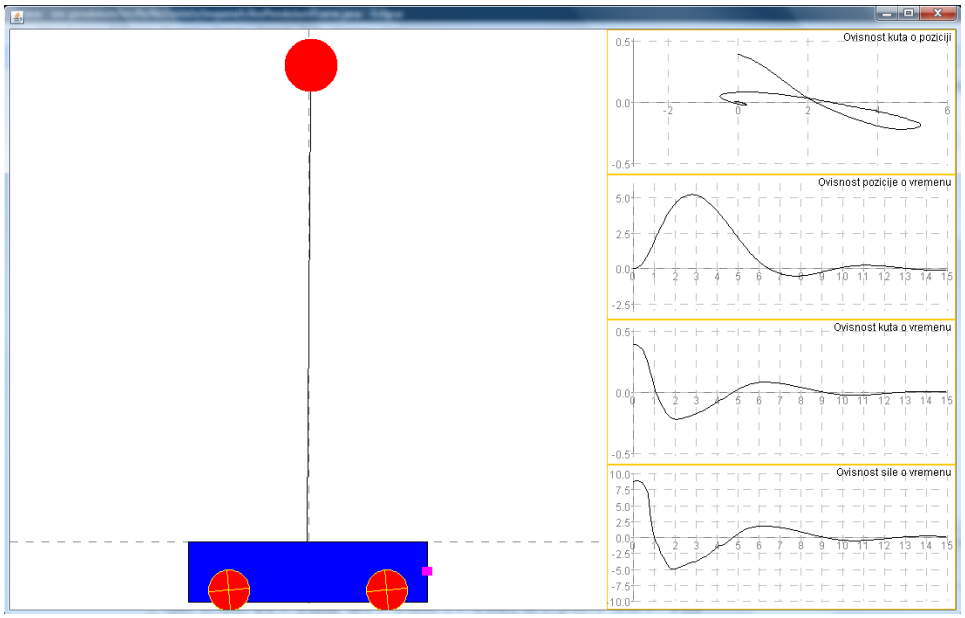
Pitanje je samo kako odabrati konstante $k_1$ i $k_2$. Uz nepovoljan odabir ovih konstanti rad sustava se može destabilizati. Razmislimo: upravljački sustav nužno mora osigurati da njihalo ne padne – podešavanje položaja je manje važan cilj. Osim toga, skale kuteva i pomaka nisu iste: pomak za $1\,m$ na stazi ima puno manje posljedice od rotacije za 1 radijan (što je nešto manje od $60°$). Stoga je očito da konstanta $k_1$ mora biti manja od 1 i to dosta. Stoga ćemo za konstante odabrati sljedeće vrijednosti: $k_1 = 10^{-2}$ i $k_2 = 10^{-1}$ čime dobivamo konačne izraze:

$$\phi^{'} = \phi + \frac{x_{cart}}{100},$$

$$\dot{\phi}^{'} = \dot{\phi} + \frac{\dot{x}_{cart}}{10}.$$

Rad upravljačkog sustava koji koristi ovako definirani kut i kutnu brzinu prikazan je na slici 5.13.

Sa slike je jasno vidljivo da smo postigli traženo: na početku, položaj kolica raste u desno jer je potrebno stabilizirati njihalo pa upravljački sklop na kolica primjenjuje pozitivnu silu. Međutim, potom se kolica počinju vraćati u lijevo i polako se stabiliziraju oko ishodišta. Ovo se lijepo vidi na prvom grafikonu prikazanom u desnom dijelu slike 5.13 koji prikazuje ovisnost kuta o položaju kolica.




Slika 5.13: Simulacija rada sustava i upravljanje razvijenim izmijenjenim upravljačkim sklopom.

### Izračun početne sile

U trenutku $t = 0$ stanje sustava je sljedeće: $\phi = \frac{\pi}{8}\,rad$, $\dot{\phi} = 0\,rad/s$, $x_{cart} = 0\,m$ te $\dot{x}_{cart} = 0\,m/s$. Izračunajmo koju će silu vratiti upravljački sklop ako koristimo modificiranu verziju upravljačkog sklopa.

Računamo:

$$\phi' = \frac{\pi}{8} + \frac{0}{100} = \frac{\pi}{8} \approx 0.393\,rad,$$

$$\dot{\phi}' = 0 + \frac{0}{10} = 0\,rad/s.$$

Ulazne vrijednosti pretvaraju se u jednoelementne neizrazite skupove: $C_1 = \{\frac{1}{0.393\,rad}\}$, $C_2 = \{\frac{1}{0\,rad/s}\}$. Stoga se zaključivanje provodi prema izrazu (5.22). Trebamo stoga najprije za svaku jezičnu varijablu utvrditi stupanj pripadnosti ulazne vrijednosti temeljnim neizrazitim skupovima jezične varijable koji su korišteni u pravilima.

| Jezična varijabla | $\mu_{NB}(\cdot)$ | $\mu_{NS}(\cdot)$ | $\mu_{ZO}(\cdot)$ | $\mu_{PS}(\cdot)$ | $\mu_{PB}(\cdot)$ |
|---|---|---|---|---|---|
| KUTNA BRZINA = 0 | 0 | 0 | 1 | 0 | 0 |
| KUT = 0.393 | 0 | 0 | 0 | 0.965 | 0 |

Svih 25 pravila koja imamo razmatraju jednu vrijednost za jezičnu varijablu *KUTNA BRZINA* i jednu vrijednost za jezičnu varijablu *KUT*. Kod stroja za zaključivanje koji koristimo jakost antecedenta jednaka je umnošku

stupnjeva pripadnosti obaju jezičnih varijabli neizrazitim skupovima navedenim u pravilu. Stoga će u našem slučaju samo jedno pravilo generirati neprazan zaključak: pravilo koje ispituje je li kutna brzina *ZO* i kut *PS*. U tom pravilu jakost antecedenta bit će $1 \cdot 0.965 = 0.965$ i to će pravilo generirati zaključak da je sila *PB'* gdje je $\mu_{sila=PS'} = 0.965 \cdot \mu_{sila=PS}$, odnosno zaključak je opisan neizrazitim denormaliziranim trokutastim skupom (odnosno lambda funkcijom s parametrima 2.5, 9 i 15 te visinom 0.965). Zaključci svih ostalih pravila su neizraziti skupovi visine 0 pa je i konačna unija jednaka ovom jedinom zaključku. Dekodiranje neizrazitosti metodom težišta daje $sila = 8.333$, i to je upravo sila koja je prikazana u simulaciji u trenutku $t = 0$. Programski, dekodiranje je provedeno diskretizacijom domene. Pravimo li se da je domena kontinuirana, konačni zaključak je neizraziti skup dan funkcijom pripadnosti $\mu(x) = 0.965 \cdot \Lambda(2.5, 9, 15)$. Dekodiranje neizrazitosti tada možemo provesti i analitički.

$$X_{CoA} = \frac{\int_{-35}^{35} x \cdot \mu(x) \cdot dx}{\int_{-35}^{35} \mu(x) \cdot dx} = \frac{\int_{2.5}^{15} x \cdot \mu(x) \cdot dx}{\int_{2.5}^{15} \mu(x) \cdot dx}$$

Ovo se dalje može raspisati kako slijedi:

$$\begin{aligned} X_{CoA} &= \frac{\int_{2.5}^{9} x \cdot 0.965 \cdot \frac{x-2.5}{9-2.5} \cdot dx + \int_{9}^{15} x \cdot 0.965 \cdot \frac{15-x}{15-9} \cdot dx}{\int_{2.5}^{9} 0.965 \cdot \frac{x-2.5}{9-2.5} \cdot dx + \int_{9}^{15} 0.965 \cdot \frac{15-x}{15-9} \cdot dx} \\ &= \frac{\frac{1}{6.5} \cdot \left(\frac{x^3}{3} - \frac{2.5x^2}{2}\right)\Big|_{2.5}^{9} + \frac{1}{6} \cdot \left(\frac{15x^2}{2} - \frac{x^3}{3}\right)\Big|_{9}^{15}}{\frac{1}{6.5} \cdot \left(\frac{x^2}{2} - 2.5x\right)\Big|_{2.5}^{9} + \frac{1}{6} \cdot \left(15x - \frac{x^2}{2}\right)\Big|_{9}^{15}} \\ &= \frac{55.2083333332}{6.25} \\ &= 8.83328. \end{aligned}$$

### Složeniji slučaj

Pretpostavimo da je nekom trenutku stanje sustava je sljedeće: $\phi = 0.55\,rad$, $\dot{\phi} = 0\,rad/s$, $x_{cart} = 0\,m$ te $\dot{x}_{cart} = 0\,m/s$. Izračunajmo koju će silu vratiti upravljački sklop ako koristimo modificiranu verziju upravljačkog sklopa.

Računamo:

$$\phi' = 0.55 + \frac{0}{100} = 0.55\,rad,$$

$$\dot{\phi}' = 0 + \frac{0}{10} = 0\,rad/s.$$

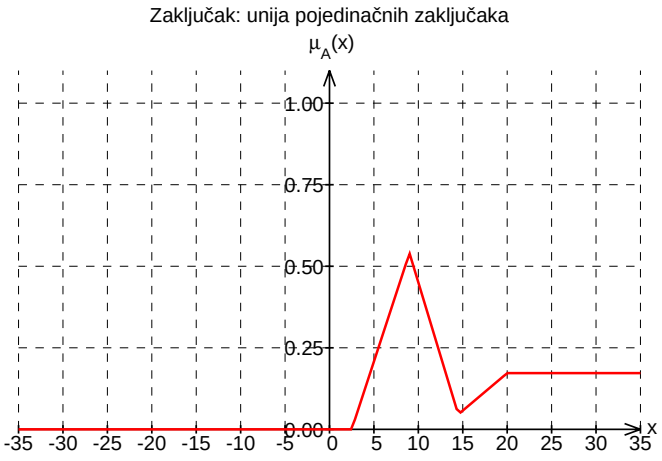
Ulazne vrijednosti pretvaraju se u jednoelementne neizrazite skupove: $C_1 = \{\frac{1}{0.55\,rad}\}$, $C_2 = \{\frac{1}{0\,rad/s}\}$. Utvrdimo za svaku jezičnu varijablu stupanj pripadnosti ulazne vrijednosti temeljnim neizrazitim skupovima te jezične varijable.

| Jezična varijabla | $\mu_{NB}(\cdot)$ | $\mu_{NS}(\cdot)$ | $\mu_{ZO}(\cdot)$ | $\mu_{PS}(\cdot)$ | $\mu_{PB}(\cdot)$ |
|---|---|---|---|---|---|
| KUTNA BRZINA = 0 | 0 | 0 | 1 | 0 | 0 |
| KUT = 0.55 | 0 | 0 | 0 | 0.5405 | 0.1724 |

Sada imamo dva pravila koja generiraju neprazan zaključak. Prvo je pravilo:
**AKO** *kutna brzina* je *ZO* **I** *kut* je *PS* **TADA** *sila* je *PS*
dok je drugo pravilo:
**AKO** *kutna brzina* je *ZO* **I** *kut* je *PB* **TADA** *sila* je *PB*.
Prvo pravilo izvodi zaključak uz iznos jakosti antecedenta od $1 \cdot 0.5405 = 0.5405$ dok drugo pravilo izvodi zaključak uz iznos jakosti antecedenta od $1 \cdot 0.1724 = 0.1724$. Konačni zaključak je unija ovih zaključaka pa je to neizraziti skup definiran funkcijom pripadnosti:

$$\mu(x) = \max(0.5405 \cdot \Lambda(x; 2.5, 9, 15), 0.1724 \cdot \Gamma(x; 12.5, 20.0).$$

Ova funkcija pripadnosti prikazana je na slici 5.14.




Slika 5.14: Izvedeni zaključak.

Dekodiranjem neizrazitosti metodom težišta uz domenu $[-35, 35]$ za silu dobivamo vrijednost 17.033. I opet, u ovom slučaju postupak bismo mogli provesti i analitički da se uvjerimo u korektnost rezultata. Očito je sada da će integraciju trebati podijeliti u četiri segmenta. Ovo ostavljamo čitatelju za vježbu.

### 5.5.1 Ubrzavanje rada upravljačkog sustava

Ako se na rad upravljačkog sustava postavljaju vremenski strogi zahtjevi, tada se, posebice u slučaju da postoji puno pravila može razmotriti pretvorba sustava u oblik *pregledne tablice* (engl. *look-up table*). Da bi to bilo moguće, sve ulazne domene treba diskretizirati.

Na primjeru upravljanja okrenutim njihalom to bi značilo sljedeće. Definirali bismo diskretnu domenu, primjerice, $\{-3, -2.9, \ldots, 2.9, 3\}$ nad kojom bismo izgradili neizrazite skupove koji opisuju kut. Označimo kardinalni broj ovog skupa oznakom $K_1$. Također, definirali bismo diskretnu domenu, primjerice, $\{-9, -8.9, \ldots, 8.9, 9\}$ nad kojom bismo izgradili neizrazite skupove koji opisuju kutnu brzinu. Označimo kardinalni broj ovog skupa oznakom $K_2$. U slučaju da se diskretizacija radi linearno od minimalne vrijednosti $L$ do maksimalne vrijednosti $U$ uz korak $\Delta$, kardinalni broj tako dobivenog skupa bio bi:

$$\frac{U - L}{\Delta} + 1$$

što bi u danom primjeru dalo $K_1 = 61$ te $K_2 = 181$.

Sljedeći korak je definiranje funkcija $\Xi_i$ koja bi elemente tih diskretnih skupova preslikalo u redni broj elementa (ako elemente skupa razmatramo poredano). Tako bi definirali funkciju $\Xi_1$ na način da vrijedi $\Xi_1(-3) = 0$, $\Xi_1(-2.9) = 1$, $\Xi_1(-2.8) = 2$, $\ldots$, $\Xi_1(3) = 61$ te funkciju $\Xi_2$ na način da vrijedi $\Xi_1(-9) = 0$, $\Xi_1(-8.9) = 1$, $\Xi_1(-8.8) = 2$, $\ldots$, $\Xi_1(9) = 181$.

Konačno, sada bismo za svaku diskretnu vrijednost kuta i kutne brzine (a takvih je konačno mnogo – točnije $K_1 \cdot K_2$) izračunali vrijednost sile koristeći razvijeni upravljački sustav. Ovo je mnoštvo izračuna, ali kako se radi jednom, ne predstavlja problem. Izračunatu vrijednost zapamtili bismo u matrici na poziciji $(\Xi_1(\phi), \Xi_2(\dot{\phi}))$; primjerice, silu izračunatu za kut $-3\,rad$ i kutnu brzinu od $-9\,rad/s$ pohranili bismo u matricu na poziciju $(0, 0)$. Dimenzije ovako dobivene matrice su $K_1 \times K_2$.

Nakon što je matrica izgrađena, možemo napraviti novi sustav neizrazitog upravljanja koji prima iste ulaze kao i postojeći, definira nove funkcije $\Psi_1(x)$ i $\Psi_2(x)$ koje za ulaznu vrijednost $x$ vraćaju najbliži element diskretne domene zadane vrijednosti te koji u memoriji čuva pohranjenu prethodno izračunatu matricu. Ovakav sustav upravljanja odgovor računa izuzetno efikasno:

$$sila = matrica[\Xi_1(\Psi_1(\phi)), \Xi_2(\Psi_2(\dot{\phi}))]$$

što se praktički svodi na dohvat pohranjene vrijednosti u matrici.

Prilikom posezanja za ovakvim rješenjem treba imati na umu da ono nije uvijek prikladno. Primjerice, ako su kardinalni brojevi diskretnih skupova veliki, matrica bi mogla biti neprihvatljivo velika. Isto se može dogoditi ako su kardinalni brojevi diskretnih skupova relativno mali ali ako pravila razmatraju više varijabli: broj elemenata matrice jednak je umnošku kardinalnih brojeva svih diskretnih skupova.

Drugo pitanje koje treba razmotriti jest kako napraviti diskretizaciju? Je li linearna (odnosno uniformna) diskretizacija prihvatljiva? U našem konkretnom slučaju gdje je zadatak balansirati njihalo oko kuta $0°$, možda bi bolje bilo imati finije uzorkovan prostor koji je u blizini te vrijednosti a ostatak prostora grublje uzorkovan. Tu se onda postavlja pitanje kako to napraviti? Da li ručno prostor podijeliti u podprostore i svaki zasebno diskretizirati ili se pak za gustoću diskretizacije osloniti na neku prikladnu matematičku funkciju? Koliko to komplicira definiranje funkcija $\Psi_i(x)$?

Unatoč tim svim pitanjima, neosporna je međutim prednost ovog pristupa: ako ga je moguće provesti, sustav neizrazitog upravljanja može biti izuzetno brz.

### 5.5.2 Izvod jednadžbi gibanja okrenutog njihala

Za potrebe izvoda jednadžbi gibanja na slici 5.15 ponovno je prikazan sustav kolica i okrenutog njihala. Na sliku su dodana djelovanja svih relevantnih sila.

Izvod radimo uz pretpostavku da je šipka zanemarive mase te da se kolica i kugla mogu aproksimirati točkastim masama. U tom slučaju imamo sustav koji se sastoji od dvije mase: $M$ i $m$ koje su u prostoru uvijek razmaknute za udaljenost $L$ što osigurava šipka. Kako se položaj kolica po okomici ne može mijenjati, sila $N$ predstavlja reakciju podloge koja će osigurati da se takvo gibanje spriječi. Uočimo da je gibanje kugle ograničeno na kružno gibanje oko fiksirane osi: sila $T$ prikazana kod kugle je napetost koja osigurava takvo gibanje. Međutim, s obzirom da je šipka kruta, protusila jednakog iznosa ili suprotnog smjera javlja se na zglobu kolica oko kojega kugla rotira.

Gibanje kod razmatranog sustava događa se u dvije dimenzije. Stoga ćemo razmatrati vektore položaja kolica $\vec{p}_{cart}$ i kugle $\vec{p}_{pend}$. Kako smo na slici naznačili sile koje djeluju na svaku od masa, gibanje mase možemo razmatrati pojedinačno.

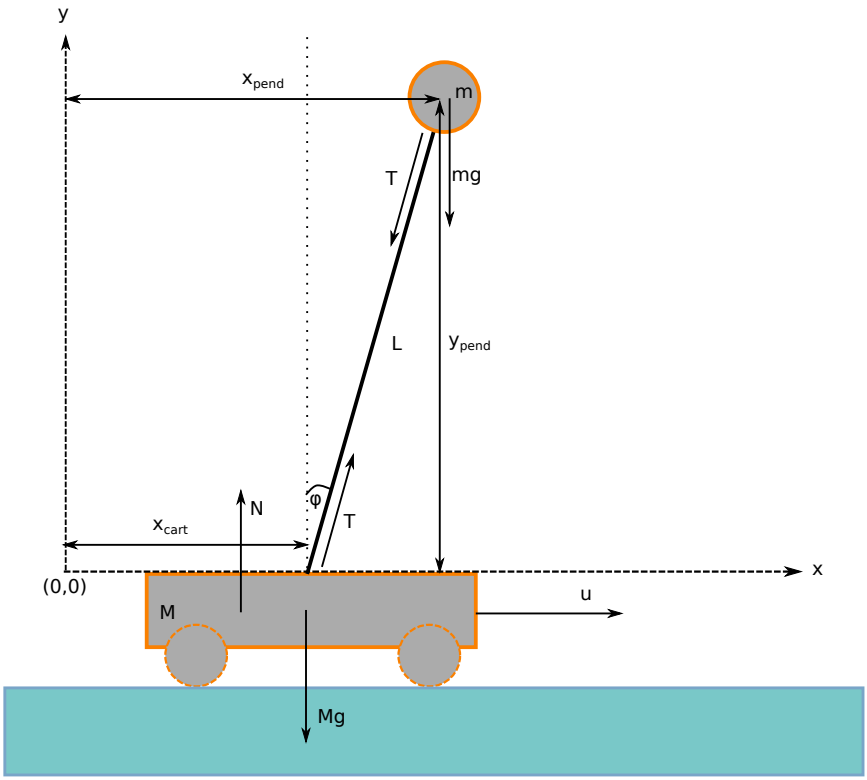
Pozicija kolica određena je s:

$$\vec{p}_{cart} = \begin{pmatrix} x_{cart} & y_{cart} \end{pmatrix} \tag{5.25}$$

$$= \begin{pmatrix} x_{cart} & 0 \end{pmatrix}, \tag{5.26}$$

brzina kolica je:

$$\vec{v}_{cart} = \dot{\vec{p}}_{cart} = \begin{pmatrix} \dot{x}_{cart} & 0 \end{pmatrix}, \tag{5.27}$$




Slika 5.15: Obrnuto njihalo.

a akceleracija:

$$\vec{a}_{cart} = \dot{\vec{v}}_{cart} = \begin{pmatrix} \ddot{x}_{cart} & 0 \end{pmatrix}. \tag{5.28}$$

Pozicija kugle definirana je s:

$$\vec{p}_{pend} = \begin{pmatrix} x_{pend} & y_{pend} \end{pmatrix} = \begin{pmatrix} x_{cart} + L \cdot \sin(\phi) & L \cdot \cos(\phi) \end{pmatrix}, \tag{5.29}$$

brzina kugle je:

$$\vec{v}_{pend} = \dot{\vec{p}}_{pend} = \begin{pmatrix} \dot{x}_{cart} + L \cdot \cos(\phi) \cdot \dot{\phi} & -L \cdot \sin(\phi) \cdot \dot{\phi} \end{pmatrix}, \tag{5.30}$$

a akceleracija:

$$\vec{a}_{pend} = \dot{\vec{v}}_{pend} \tag{5.31}$$

$$= \begin{pmatrix} \ddot{x}_{cart} - L \cdot \sin(\phi) \cdot \dot{\phi}^2 + L \cdot \cos(\phi) \cdot \ddot{\phi} & -L \cdot \cos(\phi) \cdot \dot{\phi}^2 - L \cdot \sin(\phi) \cdot \ddot{\phi} \end{pmatrix}. \tag{5.32}$$

Ukupna sila koja djeluje na kolica je:

$$\vec{F}_{cart} = N\vec{j} - Mg\vec{j} + u\vec{i} + T\sin(\phi)\vec{i} + T\cos(\phi)\vec{j} \tag{5.33}$$

što je prema trećem Newtonovom zakonu jednako:

$$\vec{F}_{cart} = M \cdot \vec{a}_{cart}. \tag{5.34}$$

Ukupna sila koja djeluje na kuglu je:

$$\vec{F}_{pend} = -mg\vec{j} - T\sin(\phi)\vec{i} - T\cos(\phi)\vec{j} \tag{5.35}$$

što je prema trećem Newtonovom zakonu jednako:

$$\vec{F}_{pend} = m \cdot \vec{a}_{pend}. \tag{5.36}$$

Izjednačavanjem sile $\vec{F}_{cart}$ iz izraza 5.33 i 5.34 dobiva se vektorska jednadžba odnosno dvije jednadžbe po komponentama. Uz supstituciju komponenata $\vec{a}_{cart}$ iz izraza 5.28 dobiva se jedna jednadžba po komponenti $\vec{i}$:

$$M \cdot \ddot{x}_{cart} = u + T\sin(\phi) \tag{5.37}$$

te druga po komponenti $\vec{j}$:

$$0 = N - Mg + T\cos(\phi). \tag{5.38}$$

Izjednačavanjem sile $\vec{F}_{pend}$ iz izraza 5.35 i 5.36 dobiva se također vektorska jednadžba odnosno dvije jednadžbe po komponentama. Uz supstituciju komponenata $\vec{a}_{pend}$ iz izraza 5.32 dobiva se jedna jednadžba po komponenti $\vec{i}$:

$$m \cdot \left(\ddot{x}_{cart} - L \cdot \sin(\phi) \cdot \dot{\phi}^2 + L \cdot \cos(\phi) \cdot \ddot{\phi}\right) = -T\sin(\phi) \tag{5.39}$$

te druga po komponenti $\vec{j}$:

$$m \cdot \left(-L \cdot \cos(\phi) \cdot \dot{\phi}^2 - L \cdot \sin(\phi) \cdot \ddot{\phi}\right) = -mg - T\cos(\phi). \tag{5.40}$$

Uvrštavanjem $T\sin(\phi)$ iz izraza 5.39 u izraz 5.37 slijedi:

$$M \cdot \ddot{x}_{cart} = u - m \cdot \left(\ddot{x}_{cart} - L \cdot \sin(\phi) \cdot \dot{\phi}^2 + L \cdot \cos(\phi) \cdot \ddot{\phi}\right)$$

što nakon grupiranja daje:

$$(M + m) \cdot \ddot{x}_{cart} - m \cdot L \cdot \sin(\phi) \cdot \dot{\phi}^2 + m \cdot L \cdot \cos(\phi) \cdot \ddot{\phi} = u. \tag{5.41}$$

Izraz 5.40 pomnožit ćemo sa $\sin(\phi)$ i presložiti:

$$-T\cos(\phi)\sin(\phi) = m \cdot \left(-L \cdot \cos(\phi) \cdot \dot{\phi}^2 - L \cdot \sin(\phi) \cdot \ddot{\phi}\right) \cdot \sin(\phi) + mg \cdot \sin(\phi).$$

Izraz 5.39 pomnožit ćemo s $\cos(\phi)$ i presložiti:

$$-T\sin(\phi)\cos(\phi) = m \cdot \left(\ddot{x}_{cart} - L \cdot \sin(\phi) \cdot \dot{\phi}^2 + L \cdot \cos(\phi) \cdot \ddot{\phi}\right) \cdot \cos(\phi).$$

Izjednačavanjem desnih strana prethodna dva izraza, kraćenjem izraza koji se poništavaju te sažimanjem $mL\sin^2(\phi)\ddot{\phi} + mL\cos^2(\phi)\ddot{\phi}$ u $mL\ddot{\phi}$ dobiva se:

$$m\ddot{x}\cos(\phi) + mL\ddot{\phi} = mg\sin(\phi). \tag{5.42}$$

Izrazi 5.41 i 5.42 predstavljaju tražene jednadžbe gibanja. Ovaj sustav jednadžbi može se uz malo algebarskih manipulacija dalje raspisati tako da se dobiju dvije jednadžbe pri čemu svaka ima drugu derivaciju samo jedne varijable (ili $\ddot{x}_{cart}$ ili $\ddot{\phi}$). Jednadžbe su sljedeće.

$$\ddot{x}_{cart} = \frac{u + mL\sin(\phi)\dot{\phi}^2 - mg\cos(\phi)\sin(\phi)}{M + m - m\cos^2(\phi)} \tag{5.43}$$

$$\ddot{\phi} = \frac{u\cos(\phi) - (M+m)g\sin(\phi) + mL\sin(\phi)\cos(\phi)\dot{\phi}^2}{mL\cos^2(\phi) - (M+m)L} \tag{5.44}$$

Kako bismo sustav mogli simulirati na računalu, prikladno ga je pretvoriti u oblik u kojem je opisan preko jednadžbi stanja. Uvedemo li vektor stanja $\vec{z} = \begin{array}{cccc} z_1 & z_2 & z_3 & z_4 \end{array}$ čije su četiri varijable stanja: $z_1 = \phi$ (kut otklona), $z_2 = \dot{\phi}$ (kutna brzina rotacije), $z_3 = x_{cart}$ (pozicija kolica) te $z_4 = \dot{x}_{cart}$ (brzina kolica) te uočimo li da vrijedi: $\dot{z}_1 = z_2$ i $\dot{z}_3 = z_4$ dobivamo traženi sustav prikazan u nastavku.

$$\frac{d}{dt}\vec{z} = \frac{d}{dt}\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \frac{d}{dt}\begin{bmatrix} z_2 \\ \frac{u\cos(z_1) - (M+m)g\sin(z_1) + mL\sin(z_1)\cos(z_1)z_2^2}{mL\cos^2(z_1) - (M+m)L} \\ z_4 \\ \frac{u + mL\sin(z_1)z_2^2 - mg\sin(z_1)\cos(z_1)}{M + m - m\cos^2(z_1)} \end{bmatrix} \tag{5.45}$$

Uz poznato početno stanje (vektor $\vec{z}_0$) sada je moguće primijeniti bilo koji od algoritama za numeričku integraciju kako bi se izračunalo ponašanje sustava u traženom vremenskom intervalu. Zainteresiranog čitatelja se upućuje na metode Runge-Kutta četvrtog reda ili čak na metodu Dormand-Prince koja omogućava adaptivno podešavanje koraka integracije čime može biti računski dosta efikasnija.

# Poglavlje 6

# Princip proširenja i neizrazita aritmetika

Do sada smo se već upoznali s načinima na koje teorija neizrazitih skupova pomaže u savladavanju ograničenja koja postavlja klasična teorija skupova: pripadnost elementa skupu više ne mora biti binarna odluka (pripada ili ne) već nam je omogućeno modeliranje mjere u kojoj pojedini elementi pripadaju neizrazitim skupovima. Pojam klasične relacije također je trivijalno proširiv na neizrazite relacije koje su, u svojoj osnovi, opet neizraziti skupovi.

U ovom poglavlju pokazat ćemo kako se uporabom principa proširenja neizrazitost može uvesti i u općenita funkcijska preslikavanja, preslikavanja temeljena na relacijama pa čak i preslikavanja temeljena na neizrazitim relacijama. Potom ćemo pokazati na koji se način neizrazitost može uvesti u *aritmetiku*, tj. kako računati ako već u startu imamo nesigurnost u ulaznim podatcima.

## 6.1 Princip proširenja

U klasičnoj matematici, *funkcija* je preslikavanje elemenata domene u elemente kodomene uz ograničenje da se svaki element domene može preslikati u točno jedan element kodomene. Istovremeno, da bismo preslikavanje smatrali funkcijom, ne postavlja se nikakvo ograničenje na broj elemenata domene koji se preslikavaju u isti element kodomene. Uvođenjem dodatnih restrikcija tada se dolazi do funkcija koje imaju neka posebna svojstva, primjerice, injektivnost, surjektivnost ili pak bijektivnost.

Polazeći međutim od ideje uvođenja neizrazitosti, ograničenje s kojim smo ovdje suočeni upravo leži u definiciji funkcije: to je preslikavanje koje svakom elementu domene pridružuje neki element kodomene. Primjerice, neka je definirana funkcija $f : \mathbb{Z} \to \mathbb{Z}$ na sljedeći način $f(x) = 2 \cdot x$.

Elementi nad kojima djeluje funkcija $f$ tada su cijeli brojevi. Primjerice, funkcija $f$ preslikat će element domene $-2$ u element kodomene $-4$, a element

domene 3 u element kodomene 6. Pri tome za element kodomene $-4$ kažemo da je *slika* elementa domene $-2$ a za element kodomene 6 kažemo da je *slika* elementa domene 3. Drugim riječima, za preslikavanje $y = f(x)$ još kažemo i da je *$y$ slika od $x$ pod djelovanjem funkcije $f$*. Uočimo da, u općem slučaju, inverzno preslikavanje više ne mora biti funkcija u strogom smislu definicije funkcije. Naime, ako s $f^{-1}$ označimo inverzno preslikavanje, tj. $x = f^{-1}(y)$, $x$ više ne mora biti jednoznačno definiran već može biti skup. Ovo se jasno vidi na primjeru funkcije $g : \mathbb{Z} \to \mathbb{N}_0$ definirane na sljedeći način $g(x) = x^2$. Ova funkcija elementu domene $-2$ kao i elementu domene $+2$ pridružuje istu sliku: element kodomene 4. Stoga inverzno preslikavanje odnosno funkcija $g^{-1}$ elementu kodomene 4 pridružuje skup elemenata domene $\{-2, +2\}$ te kažemo da je skup $\{-2, +2\}$ originalna slika od 4.

Pitanje na koje valja odgovoriti jest kako izračunati koliko iznosi $g(x)$ ako već početno nemamo jasno definiranu vrijednost na koju primjenjujemo funkciju već ako znamo da je, primjerice, *$x$ broj oko -1* definiran nad univerzalnim skupom $\mathbb{Z}$ na sljedeći način:

$$A = \left\{ \frac{0.2}{-3} + \frac{0.7}{-2} + \frac{1.0}{-1} + \frac{0.7}{0} + \frac{0.2}{1} \right\}.$$

Odgovor na ovo pitanje daje nam Zadehov *princip proširenja* koji kaže sljedeće.

**Definicija 43** (Princip proširenja)**.** *Neka je $f$ preslikavanje s $X_1 \times X_2 \times \cdots \times X_n$ na $Y$, odnosno $f : X_1 \times X_2 \times \cdots \times X_n \to Y$. Neka su nad univerzalnim skupovima $X_i$ definirani neizraziti skupovi $A_i$. Princip proširenja kaže da se djelovanjem funkcije $f$ nad neizrazitim skupovima $A_1 \times A_2 \times \cdots \times A_n$ dobiva neizraziti skup $B$ definiran nad univerzalnim skupom $Y$ čija je vrijednost funkcije pripadnosti definirana sljedećim izrazom:*

$$\mu_B(y) = \max_{y=f(x_1,x_2,\ldots,x_n)} \{\min(\mu_{A_1}(x_1), \mu_{A_2}(x_2), \ldots, \mu_{A_n}(x_n))\}.$$

Pogledajmo to na primjeru.

---

**Primjer:** 24

Neka je $f : \mathbb{Z} \to \mathbb{N}_0$ definirana izrazom $f(x) = x^2$. Potrebno je utvrditi u što se preslikava $x$ ako ga definiramo kao *broj oko -1* (vidi prethodnu definiciju).

**Rješenje:**

Potpora zadanog neizrazitog skupa je klasični skup $\{-3, -2, -1, 0, 1\}$. Stoga ćemo pogledati u koji se podskup kodomene preslikavaju ti elementi. Vrijedi $(-3)^2 = 9$, $(-2)^2 = 4$, $(-1)^2 = 1^2 = 1$ te $0^2 = 0$. Elementi skupa $\{-3, -2, -1, 0, 1\}$ preslikavaju se dakle na skup $\{0, 1, 4, 9\}$. Utvrdimo za svaki od elemenata skupa $\{0, 1, 4, 9\}$ mjeru pripadnosti kojom oni pripadaju rezultatu djelovanja funkcije $f$ na neizraziti skup *broj oko -1*.

Element kodomene 0 dobiva se preslikavanjem elementa domene 0. Kako broj 0 neizrazitom skupu $A$ pripada sa stupnjem pripadnosti $\mu_A(0) = 0.7$, tada rezultat preslikavanja 0 pripada neizrazitom skupu $B$ sa stupnjem pripadnosti $\mu_B(0) = \max(\mu_A(0)) = \max(0.7) = 0.7$.

Element kodomene 1 dobiva se preslikavanjem elemenata domene $-1$ i $+1$. Kako broj $-1$ neizrazitom skupu $A$ pripada sa stupnjem pripadnosti $\mu_A(-1) = 1.0$ a broj $+1$ neizrazitom skupu $A$ pripada sa stupnjem pripadnosti $\mu_A(1) = 0.2$, tada rezultat preslikavanja 1 pripada neizrazitom skupu $B$ sa stupnjem pripadnosti $\mu_B(1) = \max(\mu_A(-1), \mu_A(+1)) = \max(1.0, 0.2) = 1.0$.

Element kodomene 4 dobiva se preslikavanjem elementa domene $-2$. Kako broj $-2$ neizrazitom skupu $A$ pripada sa stupnjem pripadnosti $\mu_A(-2) = 0.7$, tada rezultat preslikavanja 4 pripada neizrazitom skupu $B$ sa stupnjem pripadnosti $\mu_B(4) = \max(\mu_A(-2)) = \max(0.7) = 0.7$.

Element kodomene 9 dobiva se preslikavanjem elementa domene $-3$. Kako broj $-3$ neizrazitom skupu $A$ pripada sa stupnjem pripadnosti $\mu_A(-3) = 0.2$, tada rezultat preslikavanja 9 pripada neizrazitom skupu $B$ sa stupnjem pripadnosti $\mu_B(9) = \max(\mu_A(-3)) = \max(0.2) = 0.2$.

Slijedi da se neizraziti skup:

$$A = \left\{ \frac{0.2}{-3} + \frac{0.7}{-2} + \frac{1.0}{-1} + \frac{0.7}{0} + \frac{0.2}{1} \right\}$$

pod djelovanjem funkcije $f$ preslikava u neizraziti skup:

$$B = \left\{ \frac{0.7}{0} + \frac{1.0}{1} + \frac{0.7}{4} + \frac{0.2}{9} \right\}.$$

---

**Primjer:** 25

Neka je $f : \mathbb{R} \to [-1, 1]$ definirana izrazom $f(x) = sin(x)$. Potrebno je utvrditi neizraziti skup $B$ u koji se preslikava neizraziti skup $A =$*"kut oko 45°"* definiran izrazom `FuzzyTrokut(40,45,50)`.

**Rješenje:**

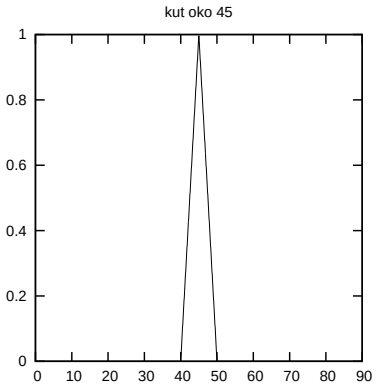
Neizraziti skup $A$ koji odgovara konceptu *kut oko 45°* možemo prikazati na sljedeći način.

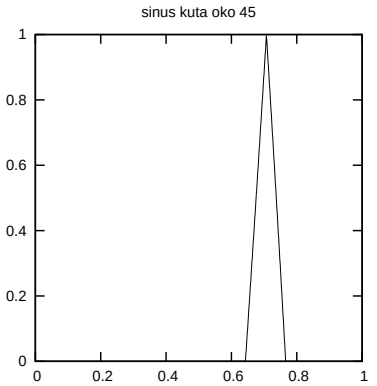
$$\mu_A(x) = \begin{cases} x < 40 & 0 \\ x >= 40 \wedge x < 45 & \frac{x-45}{5} \\ x >= 45 \wedge x < 50 & \frac{50-x}{5} \\ x >= 50 & 0 \end{cases}$$

Prema principu proširenja, za svaki element kodomene potrebno je pronaći elemente domene koje funkcija u njih preslikava i razmotriti njihov stupanj pripadnosti. Kako je funkcija $sin(x)$ na intervalu $[0 - 90]$ injektivna, rezultat je lagano odrediti.

$$\mu_B(y) = \begin{cases} arcsin(y) < 40 & 0 \\ arcsin(y) >= 40 \wedge arcsin(y) < 45 & \frac{arcsin(y)-45}{5} \\ arcsin(y) >= 45 \wedge arcsin(y) < 50 & \frac{50-arcsin(y)}{5} \\ arcsin(y) >= 50 & 0 \end{cases}$$

Grafički prikaz oba neizrazita skupa dan je u nastavku.



**Primjer:** 26

Neka je $f : \mathbb{Z} \times \mathbb{Z} \to \mathbb{N}_0$ definirana izrazom $f(x, y) = x^2 + y^2$. Potrebno je utvrditi neizraziti skup koji nastaje pod djelovanjem funkcije $f$ ako je *x broj oko -1* a *y broj oko 1*. Neka pri tome neizraziti skup $A$ označava koncept *broj oko -1* a neizraziti skup $B$ označava koncept *broj oko 1* te neka su ti neizraziti skupovi definirani kako slijedi. Neizraziti skup $C$ tada je rezultat od $f(A, B)$.

$$A = \left\{ \frac{0.2}{-3} + \frac{0.7}{-2} + \frac{1.0}{-1} + \frac{0.7}{0} + \frac{0.2}{1} \right\}$$

$$B = \left\{ \frac{0.2}{-1} + \frac{0.7}{0} + \frac{1.0}{1} + \frac{0.7}{2} + \frac{0.2}{3} \right\}$$

**Rješenje:**

Potpora neizrazitog skupa $A$ je klasični skup $\{-3, -2, -1, 0, 1\}$; potpora neizrazitog skupa $B$ je klasični skup $\{-1, 0, 1, 2, 3\}$. Stoga ćemo pogledati u koji se podskup kodomene preslikavaju elementi kartezijevog produkta tih skupova. Drugim riječima, trebamo pogledati $f(x, y)$ za sve $x \in \{-3, -2, -1, 0, 1\}$ i $y \in \{-1, 0, 1, 2, 3\}$, čime imamo ukupno $5 \cdot 5 = 25$ parova. Uz zadanu funkciju $f$ ti se parovi preslikavaju u $\{0, 1, 2, 4, 5, 8, 9, 10, 13, 18\}$. Za svaki od tih brojeva potrebno je utvrditi elemente domene koji funkcijsko preslikavanje preslikava u te brojeve.

Element kodomene 0 dobiva se funkcijskim preslikavanjem elementa domene $(0, 0)$. Kako broj 0 neizrazitom skupu $A$ pripada sa stupnjem pripadnosti $\mu_A(0) = 0.7$ i kako broj 0 neizrazitom skupu $B$ pripada sa stupnjem pripadnosti $\mu_B(0) = 0.7$, tada rezultat preslikavanja 0 pripada neizrazitom skupu $C$ sa stupnjem pripadnosti $\mu_C(0) = \max(\min(\mu_A(0), \mu_B(0))) = \max(\min(0.7, 0.7)) = 0.7$.

Element kodomene 1 dobiva se preslikavanjem elemenata domene $(-1, 0)$, $(0, -1)$, $(0, 1)$ i $(1, 0)$. Rezultat preslikavanja 1 pripada neizrazitom skupu $C$ sa stupnjem pripadnosti $\mu_C(1) = \max(\min(\mu_A(-1), \mu_B(0)), \min(\mu_A(0), \mu_B(-1)), \min(\mu_A(0), \mu_B(1)), \min(\mu_A(1), \mu_B(0))) = \max(\min(1, 0.7), \min(0.7, 0.2), \min(0.7, 1), \min(0.2, 0.7)) = \max(0.7, 0.2, 0.7, 0.2) = 0.7$.

Na sličan način dolazi se i do vrijednosti funkcije pripadnosti za preostale elemente. Konačna vrijednost neizrazitog skupa $C$ tada je:

$$C = \left\{ \frac{0.7}{0} + \frac{0.7}{1} + \frac{1}{2} + \frac{0.7}{4} + \frac{0.7}{5} + \frac{0.7}{8} + \frac{0.2}{9} + \frac{0.2}{10} + \frac{0.2}{13} + \frac{0.2}{18} \right\}.$$

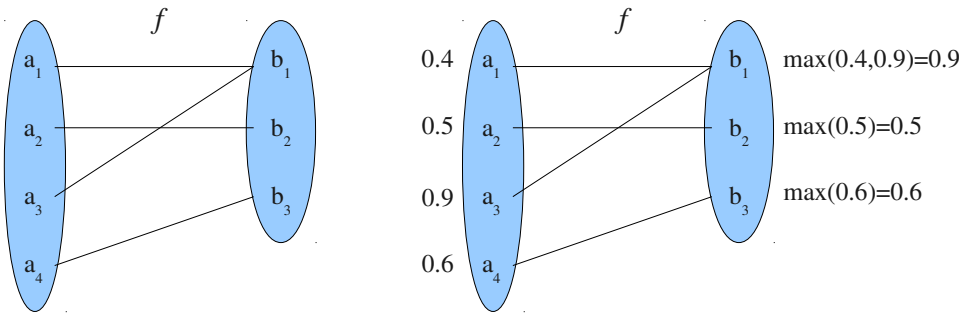
Nakon što smo na ovim primjerima ilustrirali uporabu principa proširenja na primjeru funkcije, slijedi još nekoliko primjera.

### 6.1.1 Primjena principa proširenja na funkcije

Neka je $f$ funkcija definirana nad $X \to Y$. Neka je $A$ neizraziti skup definiran nad univerzalnim skupom $X$. Funkcija $f$ tada inducira neizraziti skup $B$ definiran nad univerzalnim skupom $Y$ kako slijedi.

$$\mu_B(y) = \begin{cases} \max_{x \in f^{-1}(y)}(\mu_A(x)) & \text{ako je } f^{-1}(y) \neq \emptyset \\ 0 & \text{ako je } f^{-1}(y) = \emptyset \end{cases}$$

Primjer je prikazan na slici u nastavku. Skup $X = \{a_1, a_2, a_3, a_4\}$ a skup $Y = \{b_1, b_2, b_3\}$. Funkcija $f$ preslikava elemente skupa $X$ na skup $Y$ (lijevi dio slike).




Neka je sada definiran neizraziti skup $A$:

$$A = \left\{ \frac{0.4}{a_1} + \frac{0.5}{a_2} + \frac{0.9}{a_3} + \frac{0.6}{a_4} \right\}.$$

Funkcija $f$ tada kao rezultat djelovanja nad neizrazitim skupom $A$ inducira neizraziti skup $B$ (desni dio slike):

$$B = \left\{ \frac{0.9}{b_1} + \frac{0.5}{b_2} + \frac{0.6}{b_3} \right\}.$$

### 6.1.2 Primjena principa proširenja na klasične relacije

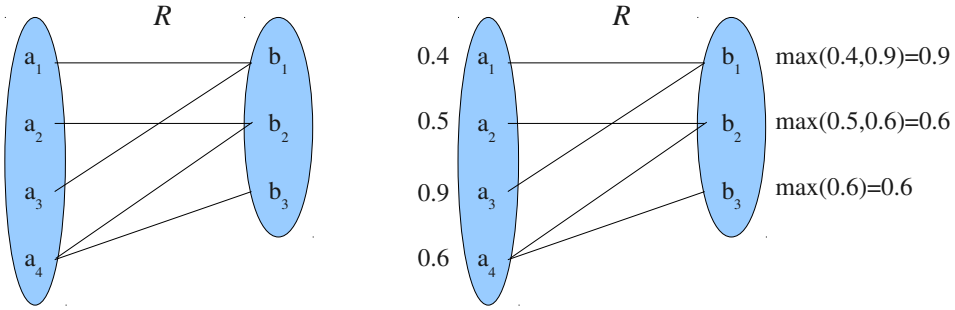
Relacija između dva skupa poopćenje je pojma funkcijskog preslikavanja: dok funkcijsko preslikavanje svakom elementu domene pridružuje točno jedan element kodomene, relacije dozvoljavaju da je svaki element domene istovremeno povezan s više elemenata kodomene.

Neka je $R$ relacija definirana nad kartezijevim produktom $X \times Y$. Neka je $A$ neizraziti skup definiran nad univerzalnim skupom $X$. Relacija $R$ tada inducira neizraziti skup $B$ definiran nad univerzalnim skupom $Y$ kako slijedi.

$$\mu_B(y) = \begin{cases} \max_{x \in R^{-1}(y)}(\mu_A(x)) & \text{ako je } R^{-1}(y) \neq \emptyset \\ 0 & \text{ako je } R^{-1}(y) = \emptyset \end{cases}$$

Pri tome je $R^{-1}$ inverzna relacija relacije $R$.

Primjer je prikazan na slici u nastavku. Skup $X = \{a_1, a_2, a_3, a_4\}$ a skup $Y = \{b_1, b_2, b_3\}$. Relacija $R$ preslikava elemente skupa $X$ na skup $Y$ (lijevi dio slike).




Neka je sada definiran neizraziti skup $A$:

$$A = \left\{ \frac{0.4}{a_1} + \frac{0.5}{a_2} + \frac{0.9}{a_3} + \frac{0.6}{a_4} \right\}.$$

Relacija $R$ tada kao rezultat djelovanja nad neizrazitim skupom $A$ inducira neizraziti skup $B$ (desni dio slike):

$$B = \left\{ \frac{0.9}{b_1} + \frac{0.6}{b_2} + \frac{0.6}{b_3} \right\}.$$

### 6.1.3 Primjena principa proširenja na neizrazite relacije

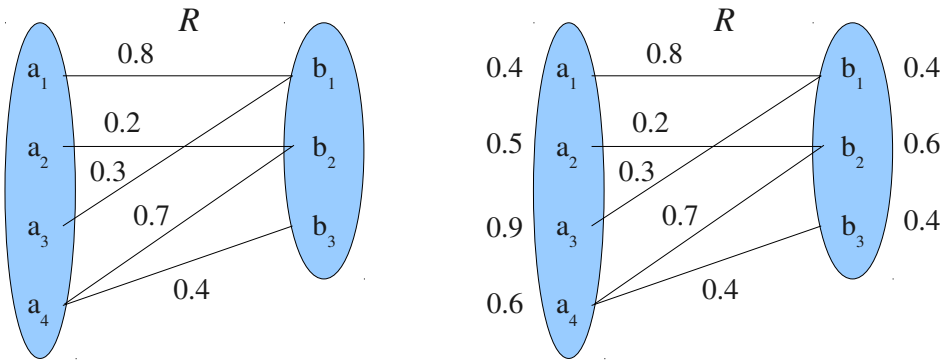
Neizrazita relacija poopćenje je klasične relacije; to je preslikavanje između dva skupa pri čemu je dodatno definirana i vrijednost funkcije pripadnost za svako preslikavanje.

Neka je $R$ neizrazita relacija definirana nad kartezijevim produktom $X \times Y$. Neka je $A$ neizraziti skup definiran nad univerzalnim skupom $X$. Neizrazita relacija $R$ tada inducira neizraziti skup $B$ definiran nad univerzalnim skupom $Y$ kako slijedi.

$$\mu_B(y) = \begin{cases} \max_{x \in R^{-1}(y)}(\min(\mu_A(x), \mu_R(x,y))) & \text{ako je } R^{-1}(y) \neq \emptyset \\ 0 & \text{ako je } R^{-1}(y) = \emptyset \end{cases}$$

Pri tome je $R^{-1}$ inverzna relacija relacije $R$.

Primjer je prikazan na slici u nastavku. Skup $X = \{a_1, a_2, a_3, a_4\}$ a skup $Y = \{b_1, b_2, b_3\}$. Neizrazita relacija $R$ izgrađena je nad kartezijevim produktom $X \times Y$ (lijevi dio slike).




Neka je sada definiran neizraziti skup $A$:

$$A = \left\{ \frac{0.4}{a_1} + \frac{0.5}{a_2} + \frac{0.9}{a_3} + \frac{0.6}{a_4} \right\}.$$

Neizrazita relacija $R$ tada kao rezultat djelovanja nad neizrazitim skupom $A$ inducira neizraziti skup $B$ (desni dio slike):

$$B = \left\{ \frac{0.4}{b_1} + \frac{0.6}{b_2} + \frac{0.4}{b_3} \right\}.$$

Naime, do elementa $b_1$ može se doći na dva načina. Neizrazita relacija $R$ uspostavlja vezu između $a_1$ i $b_1$ uz mjeru pripadnosti 0.8; kako $a_1$ pripada neizrazitom skupu $A$ mjerom pripadnosti 0.4, tim putem zaključujemo da $b_1$ pripada induciranom neizrazitom skupu $B$ mjerom pripadnosti $\min(0.4, 0.8) = 0.4$. Međutim, do $b_1$ se može doći i drugim putem: neizrazita relacija $R$ uspostavlja vezu između $a_3$ i $b_1$ uz mjeru pripadnosti 0.3; kako $a_3$ pripada neizrazitom skupu $A$ mjerom pripadnosti 0.9, tim putem zaključujemo da $b_1$ pripada induciranom neizrazitom skupu $B$ mjerom pripadnosti $\min(0.9, 0.3) = 0.3$. Konačna mjera pripadnosti elementa $b_1$ neizrazitom skupu $B$ određena je maksimumom svih utvrđenih mjera: $max(0.4, 0.3) = 0.4$. Sličnim postupkom dolazi se do vrijednosti mjera pripadnosti za sve ostale elemente skupa $Y$.

### 6.1.4 Neizrazita udaljenost

Da bismo definirali neizrazitu udaljenost, prisjetimo se najprije pojma metrike.

**Definicija 44** (Metrika)**.** *Neka je $X$ neprazni klasični skup. Neka je za svaka dva elementa $x_1, x_2 \in X$ definirana funkcija $d(x_1, x_2) : X \times X \to \mathbb{R}^+$ koja zadovoljava sljedeća svojstva:*

1. *$d(x_1, x_2) \geq 0$ uz $d(x_1, x_2) = 0$ ako i samo ako $x_1 = x_2$,*

2. $d(x_1, x_2) = d(x_2, x_1)$ *te*

3. $d(x_1, x_2) \leq d(x_1, x_3) + d(x_3, x_2) \quad \forall x_3 \in X$.

*Funkcija* $d$ *tada se zove* metrika *a uređen par* $(X, d)$ metrički prostor.

**Definicija 45** (Neizrazita točka)**.** *Neka je* $A = (X, \mu_A(x))$ *neizrazit skup definiran nad univerzalnim skupom* $X$ *i neka je* $X$ *linearni prostor. Takav neizraziti skup* $A$ *naziva se* neizrazitom točkom *ako je* $A$ *konveksan i normalan. Neizrazita točka* $A = (\mathbb{R}, \mu_A(x))$ *naziva se* neizrazitim brojem. *Neizrazit broj je* ne-negativan *ako i samo ako je* $\mu_A(x) = 0 \ \ \forall x < 0$.

Pojam neizrazite točke možemo ilustrirati neizrazitim skupovima koje smo već koristili; primjerice: *broj oko -1* te *broj oko 1*. Promatramo li jednodimenzijski prostor, u njemu dvije točke $a = -1$ i $b = 1$, Manhattan udaljenost možemo izračunati na uobičajen način: $|(-1) - (1)| = 2$. Ideja neizrazite udaljenosti koju ćemo definirati u nastavku jest opisati udaljenost između dvije neizrazite točke – primjerice, kolika je udaljenost između *broja oko -1* i *broja oko 1*. Kako se može naslutiti, neizrazita će udaljenost biti neizraziti skup.

**Definicija 46** (Neizrazita udaljenost)**.** *Neka je* $(X, d)$ *linearni metrički prostor. Neka su* $A$ *i* $B$ *dvije neizrazite točke definirane nad* $X$. Neizrazita udaljenost neizrazitih točaka $A$ *i* $B$ *je neizraziti skup* $d_f(A, B)$ *definiran nad* $\mathbb{R}$ *čija je funkcija pripadnosti definirana na sljedeći način:*

$$\mu_{d_f(A,B)}(y) = \max_{\substack{(x_1, x_2) \in X^2 \\ d(x_1, x_2) = y}} \min\{\mu_A(x_1), \mu_B(x_2)\}.$$

---

**Primjer:** 27

Zadani su neizraziti skupovi $A =$*broj oko -1* te $B =$*broj oko 1*. Ti su neizraziti skupovi definirani kako slijedi.

$$A = \left\{ \frac{0.2}{-3} + \frac{0.7}{-2} + \frac{1.0}{-1} + \frac{0.7}{0} + \frac{0.2}{1} \right\}$$

$$B = \left\{ \frac{0.2}{-1} + \frac{0.7}{0} + \frac{1.0}{1} + \frac{0.7}{2} + \frac{0.2}{3} \right\}$$

Potrebno je izračunati neizrazitu udaljenost ovih neizrazitih skupova.

**Rješenje:**

Potpora neizrazitog skupa $A$ je klasični skup $\{-3, -2, -1, 0, 1\}$; potpora neizrazitog skupa $B$ je klasični skup $\{-1, 0, 1, 2, 3\}$. Za sve parove gdje je prvi element uzet iz potpore skupa $A$ a drugi element iz potpore skupa $B$ potrebno je izračunati udaljenost tih brojeva kao i mjeru pripadnosti kojom ta udaljenost pripada traženoj neizrazitoj udaljenosti. Alternativno, kako je skup $X$ diskretan, postupak možemo provesti i na način da za zadanu udaljenost tražimo sve parove koji je generiraju, utvrdimo mjeru pripadnosti za taj par te uzmemo maksimum od mjera pripadnosti parova koji daju fiksiranu udaljenost. Taj ćemo postupak i provesti.

Udaljenost 0 dobit ćemo za sljedeće parove ($a \in A, b \in B$): $(-1, -1)$, $(0, 0)$ i $(1, 1)$; parovima odgovaraju sljedeće vrijednosti funkcije pripadnosti: $\min(1.0, 0.2) = 0.2$ za prvi par,

$\min(0.7, 0.7) = 0.7$ za drugi par te $\min(0.2, 1.0) = 0.2$ za treći par. Stoga element 0 neizrazitoj udaljenosti između $A$ i $B$ pripada s mjerom $\max(0.2, 0.7, 0.2) = 0.7$.

Udaljenost 1 dobit ćemo za sljedeće parove ($a \in A, b \in B$): $(-2, -1)$, $(-1, 0)$, $(0, 1)$, $(0, -1)$, $(1, 2)$ i $(1, 0)$; pripadni iznosi funkcije pripadnosti po parovima su $\min(0.7, 0.2) = 0.2$, $\min(1.0, 0.7) = 0.7$, $\min(0.7, 1.0) = 0.7$, $\min(0.7, 0.2) = 0.2$, $\min(0.2, 0.7) = 0.2$ i $\min(0.2, 0.7) = 0.2$. Stoga element 1 neizrazitoj udaljenosti između $A$ i $B$ pripada s mjerom $\max(0.2, 0.7, 0.7, 0.2, 0.2, 0.2) = 0.7$.

Udaljenost 2 dobit ćemo za sljedeće parove ($a \in A, b \in B$): $(-3, -1)$, $(-2, 0)$, $(-1, 1)$, $(0, 2)$, $(1, 3)$ i $(1, -1)$; pripadni iznosi funkcije pripadnosti po parovima su $\min(0.2, 0.2) = 0.2$, $\min(0.7, 0.7) = 0.7$, $\min(1.0, 1.0) = 1.0$, $\min(0.7, 0.7) = 0.7$, $\min(0.2, 0.2) = 0.2$ i $\min(0.2, 0.2) = 0.2$. Stoga element 2 neizrazitoj udaljenosti između $A$ i $B$ pripada s mjerom $\max(0.2, 0.7, 1.0, 0.7, 0.2, 0.2) = 1.0$.

Udaljenost 3 dobit ćemo za sljedeće parove ($a \in A, b \in B$): $(-3, 0)$, $(-2, 1)$, $(-1, 2)$ i $(0, 3)$; pripadni iznosi funkcije pripadnosti po parovima su $\min(0.2, 0.7) = 0.2$, $\min(0.7, 1.0) = 0.7$, $\min(1.0, 0.7) = 0.7$ i $\min(0.7, 0.2) = 0.2$. Stoga element 3 neizrazitoj udaljenosti između $A$ i $B$ pripada s mjerom $\max(0.2, 0.7, 0.7, 0.2) = 0.7$.

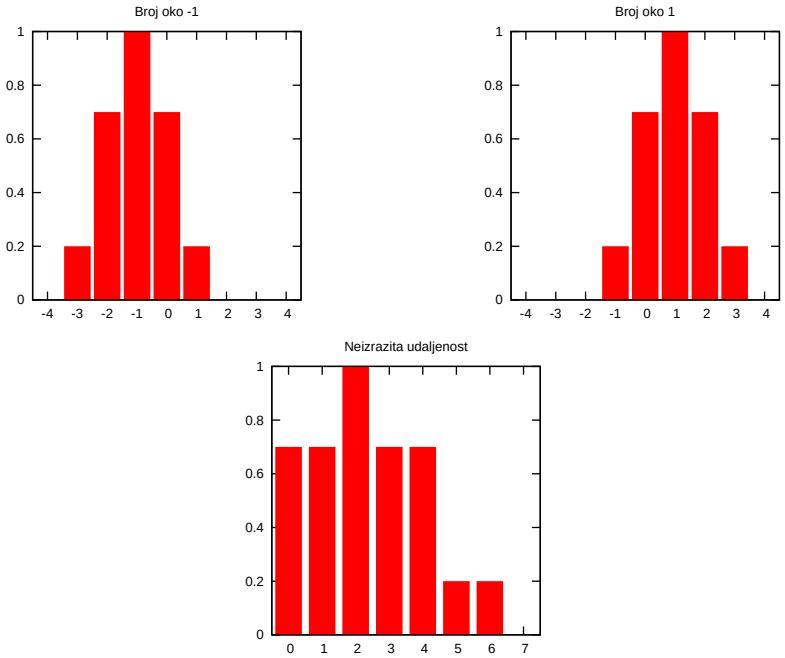
Udaljenost 4 dobit ćemo za sljedeće parove ($a \in A, b \in B$): $(-3, 1)$, $(-2, 2)$ i $(-1, 3)$; pripadni iznosi funkcije pripadnosti po parovima su $\min(0.2, 1.0) = 0.2$, $\min(0.7, 0.7) = 0.7$ i $\min(1.0, 0.2) = 0.2$. Stoga element 4 neizrazitoj udaljenosti između $A$ i $B$ pripada s mjerom $\max(0.2, 0.7, 0.2) = 0.7$.

Udaljenost 5 dobit ćemo za sljedeće parove ($a \in A, b \in B$): $(-3, 2)$ i $(-2, 3)$; pripadni iznosi funkcije pripadnosti po parovima su $\min(0.2, 0.7) = 0.2$ i $\min(0.7, 0.2) = 0.2$. Stoga element 5 neizrazitoj udaljenosti između $A$ i $B$ pripada s mjerom $\max(0.2, 0.2) = 0.2$.

Udaljenost 6 dobit ćemo samo za jedan par ($a \in A, b \in B$): $(-3, 3)$; pripadni iznosi funkcije pripadnosti za taj par je $\min(0.2, 0.2) = 0.2$. Stoga element 6 neizrazitoj udaljenosti između $A$ i $B$ pripada s mjerom $\max(0.2) = 0.2$.

Time smo za udaljenost dobili sljedeći neizraziti skup:

$$d_f(A, B) = \left\{ \frac{0.7}{0} + \frac{0.7}{1} + \frac{1.0}{2} + \frac{0.7}{3} + \frac{0.7}{4} + \frac{0.2}{5} + \frac{0.2}{6} \right\}.$$




U slučaju da su neizrazite točke definirane nad kontinuiranim univerzalnim skupovima (primjerice nad $\mathbb{R}$), neizrazitu udaljenost trebalo bi računati

za beskonačno mnogo vrijednosti. Tada bismo koristili izraz:

$$\forall \delta \in \mathbb{R}^+ \quad \mu_{d_f(A,B)}(\delta) = \sup_{\substack{(x_1,x_2) \in X^2 \\ d(x_1,x_2)=\delta}} \min\{\mu_A(x_1), \mu_B(x_2)\}.$$

## 6.2 Neizrazita aritmetika

Riječima Lotfija Zadeha, neizrazita aritmetika omogućava nam *računanje s riječima*. U osnovi neizrazite aritmetike nalazi se pojam neizrazitog broja. Taj pojam omogućava nam modeliranje koncepata poput *broj vrlo blizak broju 3*, *broj oko 5*, *broj manje-više jednak broju 12* i slično. Kako nam je pojam neizrazitog broja vrlo važan, ponovimo još jednom njegovu definiciju.

**Definicija 47** (Neizraziti broj)**.** *Neka je $A = (\mathbb{R}, \mu_A(x))$ neizrazit skup definiran nad univerzalnim skupom $\mathbb{R}$. Takav neizraziti skup $A$ naziva se* neizraziti broj *ako je $A$ konveksan i normalan. Ponekad se od neizrazitog broja zahtjeva još i da ima po dijelovima neprekinutu funkciju pripadnosti, da za samo jedan element poprima visinu 1 te da je funkcija pripadnosti simetrična oko tog elementa. Iako poželjna, ta svojstva nisu nužna.*

Posljedica ovakve definije neizrazitog broja jest konveksnost i zatvorenost svih njegovih $\alpha$-presjeka. Naime, $\alpha$-presjek svakog neizrazitog broja bit će $A_\alpha = [a_1^\alpha, a_2^\alpha]$. Također, ako razmotrimo dva $\alpha$-presjeka, konkretno $A_{\alpha_1}$ i $A_{\alpha_2}$ te ako vrijedi $\alpha_1 < \alpha_2$, tada će vrijediti $a_1^{\alpha_1} \leq a_1^{\alpha_2}$ i $a_2^{\alpha_1} \geq a_2^{\alpha_2}$; posljedica toga je i sljedeće svojstvo: $A_{\alpha_2} \subseteq A_{\alpha_1}$.

### 6.2.1 Intervalna aritmetika

Za potrebe proširenja aritmetike na neizrazitu aritmetiku, pogledajmo najprije kako bismo proširili aritmetičke operacije ako našu nesigurnost o vrijednosti broja modeliramo intervalom mogućih vrijednosti koje taj broj može poprimiti.

**Definicija 48** (Intervalna aritmetika)**.** *Neka je nesigurnost u točnu vrijednost brojeva $\hat{a}$ i $\hat{b}$ modelirana zatvorenim intervalima iz kojih ti brojevi mogu poprimiti vrijednost. Konkretno, neka je vrijednost broja $\hat{a}$ modelirana intervalom $[a_1, a_2]$ a vrijednost broja $\hat{b}$ modelirana intervalom $[b_1, b_2]$, pri čemu su $a_1$, $a_2$, $b_1$ i $b_2 \in \mathbb{R}$. Operacije zbrajanja, oduzimanja, množenja i dijeljenja tada možemo definirati na sljedeći način.*

- *Nesigurnost u vrijednost zbroja brojeva $\hat{a}$ i $\hat{b}$ tada modeliramo intervalom:*

$$\hat{a} + \hat{b} = [a_1 + b_1, a_2 + b_2].$$

- *Nesigurnost u vrijednost razlike brojeva $\hat{a}$ i $\hat{b}$ tada modeliramo intervalom:*

$$\hat{a} - \hat{b} = [a_1 - b_2, a_2 - b_1].$$

- *Nesigurnost u vrijednost umnoška brojeva $\hat{a}$ i $\hat{b}$ tada modeliramo intervalom:*

$$\hat{a} \cdot \hat{b} = [\min(a_1 \cdot b_1, a_1 \cdot b_2, a_2 \cdot b_1, a_2 \cdot b_2), \max(a_1 \cdot b_1, a_1 \cdot b_2, a_2 \cdot b_1, a_2 \cdot b_2)].$$

- *Nesigurnost u vrijednost kvocijenta brojeva $\hat{a}$ i $\hat{b}$ tada modeliramo intervalom:*

$$\begin{aligned} \frac{\hat{a}}{\hat{b}} &= [a_1, a_2] \cdot \left[\frac{1}{b_2}, \frac{1}{b_1}\right] \\ &= \left[\min\left(\frac{a_1}{b_1}, \frac{a_1}{b_2}, \frac{a_2}{b_1}, \frac{a_2}{b_2}\right), \max\left(\frac{a_1}{b_1}, \frac{a_1}{b_2}, \frac{a_2}{b_1}, \frac{a_2}{b_2}\right)\right] \end{aligned}$$

  *uz ograničenje da 0 ne smije biti u intervalu kojim modeliramo $\hat{b}$, tj.* $0 \notin [b_1, b_2]$.

Lagano se je uvjeriti da ove definicije doista vrijede. Primjerice, ako za $\hat{a}$ znamo da je broj iz intervala $[1, 3]$ i za $\hat{b}$ znamo da je broj iz intervala $[2, 4]$, suma ta dva broja bit će negdje u intervalu $[3, 7]$.

Sada možemo definirati četiri osnovne operacije nad neizrazitim brojevima oslanjajući se upravo na intervalnu aritmetiku i teorem predstavljanja.

### 6.2.2 Neizrazita aritmetika definirana preko intervalne aritmetike

Podsjetimo se, teorem predstavljanja nam govori da se svaki neizraziti skup može prikazati kao zbroj njegovih $\alpha$-presjeka pomnoženih s vrijednošću svakog $\alpha$, odnosno:

$$A = \sum_{\alpha} \alpha \cdot A_\alpha.$$

Uočimo pri tome da je kod neizrazitog broja svaki $\alpha$-presjek po definiciji konveksan zatvoreni interval. Stoga se rezultat proizvoljne aritmetičke operacije $\diamond$ nad neizrazitim brojevima $A$ i $B$, dakle, $A \diamond B$ definira direktno nad intervalima koji odgovaraju $\alpha$-presjecima neizrazitih brojeva $A$ i $B$. Time se

definira:

$$\begin{aligned} A \diamond B &= \sum_{\alpha} \alpha \cdot A_\alpha \diamond \sum_{\alpha} \alpha \cdot B_\alpha \\ &= \sum_{\alpha} \alpha \cdot [a_1^\alpha, a_2^\alpha] \diamond \sum_{\alpha} \alpha \cdot [b_1^\alpha, b_2^\alpha] \\ &= \sum_{\alpha} \alpha \cdot ([a_1^\alpha, a_2^\alpha] \diamond [b_1^\alpha, b_2^\alpha]) \\ &= \sum_{\alpha} \alpha \cdot (A_\alpha \diamond B_\alpha) \\ &= \sum_{\alpha} \alpha \cdot (A \diamond B)_\alpha. \end{aligned}$$

Pogledajmo to na slučaju dva neizrazita broja koja su definirana trokutastim funkcijama pripadnosti. Neka je neizraziti broj $A$ definiran funkcijom `FuzzyTrokut(`$a_1$`,`$a_2$`,`$a_3$`)`, odnosno:

$$\mu_A(x) = \begin{cases} 0 & x < a_1 \\ \frac{x-a_1}{a_2-a_1} & a_1 \leq x \wedge x < a_2 \\ \frac{a_3-x}{a_3-a_2} & a_2 \leq x \wedge x < a_3 \\ 0 & x \geq a_3 \end{cases}$$

Kako je $\alpha$-presjek tog skupa zatvoreni interval, uvedimo za njega oznaku $[a_L^\alpha, a_D^\alpha]$ pri čemu $a_L^\alpha$ označava granicu s lijeve strane a $a_R^\alpha$ granicu s desne strane. S obzirom na definiciju trokutaste funkcije pripadnosti, za neku konkretnu vrijednost $\alpha$ granice možemo dobiti iz sljedećih izraza:

$$\frac{a_L^\alpha - a_1}{a_2 - a_1} = \alpha$$

$$\frac{a_3 - a_R^\alpha}{a_3 - a_2} = \alpha$$

iz čega dalje slijedi:

$$a_L^\alpha = \alpha(a_2 - a_1) + a_1 \qquad a_R^\alpha = \alpha(a_2 - a_3) + a_3.$$

$\alpha$-presjek neizrazitog broja s trokutastom funkcijom pripadnosti tada je klasični skup, odnosno interval, $[\alpha(a_2 - a_1) + a_1, \alpha(a_2 - a_3) + a_3]$. Ako je neizrazit broj $B$ zadan funkcijom `FuzzyTrokut(`$b_1$`,`$b_2$`,`$b_3$`)`, njegov $\alpha$-presjek bit će, analogno, $[\alpha(b_2 - b_1) + b_1, \alpha(b_2 - b_3) + b_3]$.

Sada možemo koristeći teorem o predstavljanju definirati zbroj dvaju trokutastih neizrazitih brojeva. Prema prethodno napisanom izrazu za neki općeniti operator $\diamond$, mora vrijediti:

$$A \diamond B = \sum_{\alpha} \alpha \cdot ([a_1^\alpha, a_2^\alpha] \diamond [b_1^\alpha, b_2^\alpha])$$

što uz pretpostavku da radimo operaciju zbrajanja te da su u prethodnom izrazu $a_1^\alpha = a_L^\alpha$, $a_2^\alpha = a_R^\alpha$, $b_1^\alpha = b_L^\alpha$ i $b_2^\alpha = b_R^\alpha$ daje:

$$\begin{aligned}
A + B &= \sum_\alpha \alpha \cdot ([a_L^\alpha, a_R^\alpha] + [b_L^\alpha, b_R^\alpha]) \\
&= \sum_\alpha \alpha \cdot ([a_L^\alpha + b_L^\alpha, a_R^\alpha + b_R^\alpha]) \\
&= \sum_\alpha \alpha \cdot ([\alpha(a_2 - a_1) + a_1 + \alpha(b_2 - b_1) + b_1, \alpha(a_2 - a_3) + a_3 + \alpha(b_2 - b_3) + b_3]) \\
&= \sum_\alpha \alpha \cdot ([\alpha(a_2 - a_1 + b_2 - b_1) + a_1 + b_1, \alpha(a_2 - a_3 + b_2 - b_3) + a_3 + b_3]) \\
&= \sum_\alpha \alpha \cdot ([\alpha((a_2 + b_2) - (a_1 + b_1)) + (a_1 + b_1), \alpha((a_2 + b_2) - (a_3 + b_3)) + (a_3 + b_3)]) \\
&= \sum_\alpha \alpha \cdot ([\alpha(c_2 - c_1) + c_1, \alpha(c_2 - c_3) + c_3]) \\
&= \sum_\alpha \alpha \cdot ([c_L^\alpha, c_R^\alpha]) \\
&= \texttt{FuzzyTrokut}(c_1, c_2, c_3) \\
&= \texttt{FuzzyTrokut}(a_1 + b_1, a_2 + b_2, a_3 + b_3)
\end{aligned}$$

čime smo pokazali da je zbroj dvaju neizrazitih trokutastih brojeva opet neizraziti trokutasti broj.

Na sličan se način može provesti analiza i za ostale operacije što ostavljamo čitatelju za vježbu. Konačni rezultat za operaciju oduzimanja dan je u nastavku uz pretpostavku da su $A$ i $B$ zadani kao prethodno definirani trokutasti neizraziti brojevi.

$$A - B = \texttt{FuzzyTrokut}(a_1 - b_3, a_2 - b_2, a_3 - b_1)$$

Množenje i dijeljenje trokutastih brojeva ne daje trokutaste brojeve.

### 6.2.3 Neizrazita aritmetika definirana preko principa proširenja

Princip proširenja primijenjen na definiranje neizrazite aritmetike kaže da za proizvoljan operator $\diamond$ primijenjen nad neizrazitim brojevima definiranim nad $\mathbb{R}$ mora vrijediti:

$$\mu_{A \diamond B}(z) = \max_{z = x \diamond y} \min(\mu_A(x), \mu_B(y)).$$

To znači da redom vrijedi $\forall x, y, z \in \mathbb{R}$:

$$\mu_{A+B}(z) = \max_{z = x+y} \min(\mu_A(x), \mu_B(y)).$$

$$\mu_{A-B}(z) = \max_{z=x-y} \min(\mu_A(x), \mu_B(y)).$$

$$\mu_{A\cdot B}(z) = \max_{z=x\cdot y} \min(\mu_A(x), \mu_B(y)).$$

$$\mu_{\frac{A}{B}}(z) = \max_{z=\frac{x}{y}} \min(\mu_A(x), \mu_B(y)).$$

Primjenom ovog principa dolazi se do identičnih definicija kao i preko intervalne aritmetike i teorema o predstavljanju. Čitatelju se ostavlja za vježbu da to pokaže na primjeru operacije zbrajanja.

---

**Primjer:** 28

Zadana su dva neizrazita broja:

$$A(x) = \begin{cases} 0, & x < -1 \text{ ili } x > 3, \\ \frac{x+1}{2}, & -1 \le x \le 1, \\ \frac{3-x}{2}, & 1 \le x \le 3, \end{cases} \qquad B(x) = \begin{cases} 0, & x < 1 \text{ ili } x > 5, \\ \frac{x-1}{2}, & 1 \le x \le 3, \\ \frac{5-x}{2}, & 3 \le x \le 5. \end{cases}$$

Izračunajte njihov zbroj, razliku, umnožak i kvocijent.

**Rješenje:**

Izračunajmo najprije $\alpha$-presjeke brojeva $A$ i $B$. Vrijedi: $A_\alpha = [a_L^\alpha, a_R^\alpha]$ te $B_\alpha = [b_L^\alpha, b_R^\alpha]$. Vrijedi:

$$\alpha = \frac{a_L^\alpha + 1}{2} \quad \Rightarrow \quad a_L^\alpha = 2\alpha - 1$$

$$\alpha = \frac{3 - a_R^\alpha}{2} \quad \Rightarrow \quad a_R^\alpha = 3 - 2\alpha$$

$$\alpha = \frac{b_L^\alpha - 1}{2} \quad \Rightarrow \quad b_L^\alpha = 2\alpha + 1$$

$$\alpha = \frac{5 - b_R^\alpha}{2} \quad \Rightarrow \quad b_R^\alpha = 5 - 2\alpha$$

Traženi $\alpha$-presjeci tada su:

$$A_\alpha = [2\alpha - 1, 3 - 2\alpha],$$

$$B_\alpha = [2\alpha + 1, 5 - 2\alpha].$$

Neizraziti broj koji predstavlja sumu, prema teoremu predstavljanja i s obzirom na intervalnu aritmetiku, definiran je kao:

$$\begin{aligned} A + B &= \sum \alpha \cdot (A + B)_\alpha \\ &= \sum \alpha \left( [2\alpha - 1, 3 - 2\alpha] + [2\alpha + 1, 5 - 2\alpha] \right) \\ &= \sum \alpha [4\alpha, 8 - 4\alpha] \\ &= \begin{cases} 0, & x < 0 \text{ ili } x > 8, \\ \frac{x}{4}, & 0 \le x \le 4, \\ \frac{8-x}{4}, & 4 \le x \le 8. \end{cases} \\ &= \text{FuzzyTrokut}(0, 4, 8). \end{aligned}$$

Neizraziti broj koji predstavlja razliku, prema teoremu predstavljanja i s obzirom na intervalnu aritmetiku, definiran je kao:

$$\begin{aligned} A - B &= \sum \alpha \cdot (A-B)_\alpha \\ &= \sum \alpha \left([2\alpha - 1, 3 - 2\alpha] - [2\alpha + 1, 5 - 2\alpha]\right) \\ &= \sum \alpha [4\alpha - 6, 2 - 4\alpha] \\ &= \begin{cases} 0, & x < -6 \text{ ili } x > 2, \\ \frac{x+6}{4}, & -6 \leq x \leq -2, \\ \frac{2-x}{4}, & -2 \leq x \leq 2. \end{cases} \\ &= \text{FuzzyTrokut}(0, 4, 8). \end{aligned}$$

Neizraziti broj koji predstavlja umnožak, prema teoremu predstavljanja i s obzirom na intervalnu aritmetiku, definiran je kao:

$$\begin{aligned} A \cdot B &= \sum \alpha \cdot (A \cdot B)_\alpha \\ &= \sum \alpha \left([2\alpha - 1, 3 - 2\alpha] \cdot [2\alpha + 1, 5 - 2\alpha]\right) \end{aligned}$$

Kako bismo razriješili interval koji odgovara umnošku dobivenih intervala, označimo s $S$ skup vrijednosti:

$$S = \{(2\alpha - 1)(2\alpha + 1), (2\alpha - 1)(5 - 2\alpha), (3 - 2\alpha)(2\alpha + 1), (3 - 2\alpha)(5 - 2\alpha)\}.$$

Vrijedi:

$$A \cdot B = \sum \alpha [\min(S), \max(S)]$$

Potrebno je odrediti vrijednost minimuma i maksimuma. Pri traženju minimalne vrijednosti može se vidjeti da postoje dva područja. Za $\alpha \in [0, 0.5]$ vrijedi $\min(S) = (2\alpha - 1)(5 - 2\alpha) = -4\alpha^2 + 12\alpha - 5$, dok je za $\alpha \in [0.5, 1]$ vrijedi $\min(S) = (2\alpha - 1)(2\alpha + 1) = -4\alpha^2 - 1$. Maksimalna vrijednost definirana je samo jednim članom za čitavo legalno područje od $\alpha$, i to je $\max(S) = (3 - 2\alpha)(5 - 2\alpha) = 4\alpha^2 - 16\alpha + 15$. Stoga je $\alpha$-presjek umnoška definiran po dijelovima:

$$(A \cdot B)_{alpha} = \begin{cases} [-4\alpha^2 + 12\alpha - 5, 4\alpha^2 - 16\alpha + 15] & 0 \leq \alpha \leq 0.5, \\ [-4\alpha^2 - 1, 4\alpha^2 - 16\alpha + 15] & 0.5 < \alpha \leq 1. \end{cases}$$

Za $\alpha = 1$ dobit ćemo jezgru neizrazitog skupa koji predstavlja umnožak. Uvrštavanjem dobivamo da je jezgra $[3, 3]$. Za $\alpha = 0$ dobit ćemo potporu neizrazitog skupa koji predstavlja umnožak. Uvrštavanjem dobivamo da je potpora $(-5, 15)$. Za $\alpha = 0.5$ donja granica intervala (koja je definirana po dijelovima) postaje $-4 \cdot 0.5^2 + 12 \cdot 0.5 - 5 = 0$. To znači da za donju granicu intervala u rasponu $x \in [-5, 0]$ vrijedi izraz $x = -4\alpha^2 + 12\alpha - 5$ a u rasponu od $x \in [0, 3]$ vrijedi izraz $x = -4\alpha^2 - 1$; gornja granica intervala kreće se u intervalu $[3, 15]$ i vrijedi $x = 4\alpha^2 - 16\alpha + 15$. Iz $x = -4\alpha^2 + 12\alpha - 5$ slijedi:

$$\alpha_{1,2} = \frac{-12 \pm \sqrt{144 - 4 \cdot 4 \cdot (5 + x)}}{-8} = \frac{-12 \pm 4\sqrt{9 - (5 + x)}}{-8} = \frac{3 \mp \sqrt{4 - x}}{2}$$

Kako na intervalu $x \in [-5, 0]$ vrijednost za $\alpha$ mora biti u intervalu $[0, 1]$, pravo rješenje je $\alpha_1$ tj.:

$$\alpha = \frac{3 - \sqrt{4 - x}}{2}.$$

Iz $x = -4\alpha^2 - 1$ slijedi:

$$\alpha_{1,2} = \pm \frac{1}{2}\sqrt{1 + x}.$$

Kako na intervalu $x \in [0, 3]$ vrijednost za $\alpha$ mora biti u intervalu $[0, 1]$, pravo rješenje je $\alpha_1$ tj.:

$$\alpha = \frac{1}{2}\sqrt{1 + x}.$$

Iz $x = 4\alpha^2 - 16\alpha + 15$ slijedi:

$$\alpha_{1,2} = \frac{16 \pm \sqrt{256 - 16 \cdot (15 - x)}}{8} = \frac{16 \pm 4\sqrt{16 - (15 - x)}}{8} = \frac{4 \pm \sqrt{1 + x}}{2}$$

Kako na intervalu $x \in [3, 15]$ vrijednost za $\alpha$ mora biti u intervalu $[0, 1]$, pravo rješenje je $\alpha_2$ tj.:

$$\alpha = \frac{4 - \sqrt{1 + x}}{2}.$$

Sada konačni rezultat možemo raspisati po dijelovima:

$$(A \cdot B)(x) = \begin{cases} 0, & x < -5 \text{ ili } x > 15, \\ \frac{3 - \sqrt{4 - x}}{2}, & -5 \leq x \leq 0, \\ \frac{1}{2}\sqrt{1 + x}, & 0 \leq x \leq 3, \\ \frac{4 - \sqrt{1 + x}}{2}, & 3 \leq x \leq 15. \end{cases}$$

Uočimo da ova funkcija pripadnosti nije trokutasta funkcija – umnožak dvaju trokutastih brojeva više nije trokutasti broj.
Neizraziti broj koji predstavlja kvocijent, prema teoremu predstavljanja i s obzirom na intervalnu aritmetiku, definiran je kao:

$$\begin{aligned} A \cdot B &= \sum \alpha \cdot (A/B)_\alpha \\ &= \sum \alpha \left( [2\alpha - 1, 3 - 2\alpha] / [2\alpha + 1, 5 - 2\alpha] \right) \\ &= \sum \alpha \left( [2\alpha - 1, 3 - 2\alpha] \cdot \left[ \frac{1}{5 - 2\alpha}, \frac{1}{2\alpha + 1} \right] \right) \end{aligned}$$

Kako bismo razriješili interval koji odgovara umnošku dobivenih intervala, označimo s $T$ skup vrijednosti:

$$T = \left\{ \frac{2\alpha - 1}{5 - 2\alpha}, \frac{2\alpha - 1}{2\alpha + 1}, \frac{3 - 2\alpha}{5 - 2\alpha}, \frac{3 - 2\alpha}{2\alpha + 1} \right\}.$$

Vrijedi:

$$A/B = \sum \alpha [\min(T), \max(T)]$$

Nakon malo računanja dobiva se:

$$(A \cdot B)_{alpha} = \begin{cases} \left[ \frac{2\alpha - 1}{2\alpha + 1}, \frac{3 - 2\alpha}{2\alpha + 1} \right] & 0 \leq \alpha \leq 0.5, \\ \left[ \frac{2\alpha - 1}{5 - 2\alpha}, \frac{3 - 2\alpha}{2\alpha + 1} \right] & 0.5 \leq \alpha \leq 1. \end{cases}$$

te nakon još ponešto računa koji ostavljamo čitatelju za vježbu slijedi:

$$(A/B)(x) = \begin{cases} 0, & x < -1 \text{ ili } x > 3, \\ \frac{x + 1}{2 - 2x}, & -1 \leq x \leq 0, \\ \frac{5x + 1}{2x + 2}\sqrt{1 + x}, & 0 \leq x \leq \frac{1}{3}, \\ \frac{3 - x}{2x + 2}, & \frac{1}{3} \leq x \leq 3. \end{cases}$$

Uočimo da ova funkcija pripadnosti nije trokutasta funkcija – kvocijent dvaju trokutastih brojeva više nije trokutasti broj.

# Dio II

# Umjetne neuronske mreže

# Poglavlje 7

# Općenito o neuronskim mrežama

Razvoj neuroračunarstva započeo je sredinom 20. stoljeća spojem niza bioloških spoznaja i njihovim modeliranjem prikladnim za računalnu obradu. Još davne 1943. Warren McCulloch i Walter Pitts u radu *A Logical Calculus of Ideas Immanent in Nervous Activity* definirali su model umjetnog neurona (tzv. TLU perceptron) i pokazali da se njime mogu računati logičke funkcije I, ILI i NE. Uzevši u obzir da se uporabom tih triju funkcija može izgraditi proizvoljna Booleova funkcija, slijedi da se izgradnjom (tj. prikladnom konstrukcijom) umjetnom neuronskom mrežom sastavljenom od TLU neurona može dobiti proizvoljna Booleova funkcija. Međutim, problem koji ovdje valja još jednom istaknuti jest da se radi o rješavanju problema konstrukcijom, a ne učenjem.

Nekoliko godina kasnije, britanski biolog Donald Hebb u svojoj knjizi *The Organization of Behavior* iz 1949. godine objavljuje svoje spoznaje o radu i interakciji bioloških neurona. Konkretno, njegovo je zapažanje da, ako dva neurona često zajedno pale, tada dolazi do metaboličkih promjena kojima efikasnost kojom jedan neuron pobuđuje drugoga s vremenom raste.

Tom spoznajom stvoreni su svi preduvjeti za definiranje algoritma učenja TLU perceptrona, i to je upravo kroz izvještaj *The Perceptron: A Perceiving and Recognizing Automaton* iz 1957. i knjigu *Principles of Neurodynamics* iz 1962. napravio Frank Rosenblatt definirajući algoritam učenja TLU perceptrona, iskoristivši Hebbovu ideju da *učiti znači mijenjati jakost veza između neurona*.

## 7.1 Motivacija

Razvoj umjetnih neuronskih mreža započeo je kao pokušaj da se objasni inteligentno ponašanje ljudi i životinja. Jedan od pokušaja objašnjavanja i postizanja inteligentnog ponašanja kod strojeva je poznati simbolički pristup.

Međutim, spoznaje o funkcioniranju bioloških organizama do danas nisu uspjele pokazati i dokazati da se u mozgovima odsnosno u živčanom sustavu odvija simbolička obrada podataka i simboličko zaključivanje. Umjesto toga, čini se da inteligentno ponašanje izranja iz velikog broja jednostavnih procesnih elemenata (neurona) koji podatke obrađuju masovno paralelno i uz veliku redundanciju – što je temelj pristupa danas poznatog pod nazivom *konektivizam*. Istraživanja su pokazala da se u ljudskom mozgu nalazi poprilici $10^{11}$ neurona koji su međusobno povezani s oko $10^{15}$ veza. To znači da je svaki neuron povezan s u prosjeku $10^4$ veza iz čega je jasno da je obrada podataka u ljudskom mozgu masovno paralelna. Dodatni argument tezi o masovno parelnoj obradi podataka slijedi i iz vrlo sporog rada neurona; naime, vrijeme paljenja neurona je poprilici 1 ms. Uzmemo li u obzir da čovjek može izuzetno kompleksnu obradu podataka riješiti u vrlo kratkom vremenu (primjerice, vlastitu majku možemo prepoznati sa slike u vremenu od $\approx 0.1$ s), očito je da se obrada ne može odvijati slijedno jer je broj koraka koje je u tako kratkom vremenu moguće napraviti vrlo mali.

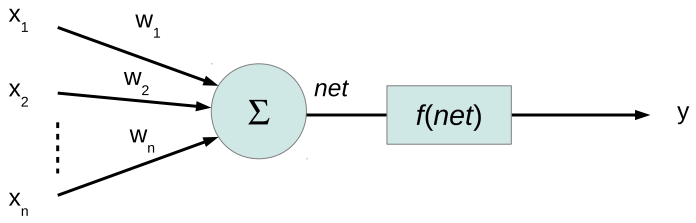
S obzirom da se područje umjetnih neuronskih mreža razvija već posljednjih sedamdesetak godina, do danas je napravljen niz uspješnih primjena. Neuronske mreže su se pokazale kao uspješan model za aproksimaciju funkcija, klasifikaciju podataka, predviđanje trendova i slično. Zgodno je spomenuti i eksperimentalnu neuronsku mrežu *Alvin* razvijenu na sveučilištu Carnegie Mellon. Radi se o umjetnoj neuronskoj mreži koja uči kako voziti automobil promatrajući kako to radi čovjek. Kroz napravljene eksperimente ta je mreža uspješno upravljala vozilom na dionici duljine 100 km vozeći pri tome vršnom brzinom od 110 km/h.

S obzirom na niz dobrih svojstava neuronskih mreža, poput njihove robusnosti na pogreške u podatcima, otpornosti na šum, masovno paralelnu obradu podataka te mogućnost učenja, ova je paradigma i danas u fokusu istraživanja i primjene.

## 7.2 Osnovni model neurona

Upoznavanje s umjetnim neuronskim mrežama započet ćemo promatranjem jednog neurona. Pojednostavljeno, biološki neuron sastoji se od dendrita, tijela neurona i aksona. U prosjeku, svaki je neuron povezan s $10^4$ drugih neurona preko dendrita, i preko njih prikuplja električke impulse koje akumulira u tijelu. Nakon što se akumulira određena količina naboja (ugrubo određena pragom), neuron *pali* – akumulirani naboj šalje kroz svoj akson prema drugim neuronima i time se prazni. U tom smislu, dendriti predstavljaju ulaze preko kojih neuron prikuplja informacije, tijelo stanice ih obrađuje i na kraju generira rezultat koji se prenosi kroz akson – izlaz neurona.

Temeljem ovako pojednostavljenog opisa definiran je osnovni model neurona koji je prikazan na slici 7.1. Model se sastoji od ulaza $x_1$ do $x_n$




Slika 7.1: Osnovni model neurona

("dendriti"), težina $w_1$ do $w_n$ koje određuju u kojoj mjeri svaki od ulaza pobuđuje neuron, tijela neurona koje računa ukupnu pobudu (koju ćemo često označavati s $net$) te prijenosne funkcije $f(net)$ ("akson" neurona) koja pobudu obrađuje i proslijeđuje na izlaz neurona.

Ukupna pobuda $net$ računa se prema izrazu:

$$net = w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n - \theta$$

pri čemu $\theta$ predstavlja prag paljenja neurona. Kako se iznos praga ne bi morao posebno tretirati u izrazima za $net$, najčešće se definira da neuron ima još jedan "fiktivni" ulaz $x_0$ na koji je uvijek dovedena vrijednost 1, a prag $\theta$ tada se zamjenjuje težinom $w_0$ čime se dobije konačni izraz:

$$\begin{aligned} net &= w_0 \cdot x_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n \\ &= \sum_{i=0}^{n} w_i \cdot x_i \\ &= \vec{w} \cdot \vec{x} \end{aligned}$$

pri čemu je $\vec{x} = (1, x_1, x_2, \ldots, x_n)$ $(n+1)$-dimenzijski vektor koji predstavlja ulaz u neuron a $\vec{w} = (w_0, \ldots, w_n)$ $(n+1)$-dimenzijski vektor težina.

Prijenosna funkcija modelira ponašanje aksona i u praksi se koristi nekoliko tipičnih prijenosnih funkcija koje su prikazane na slici 7.2.

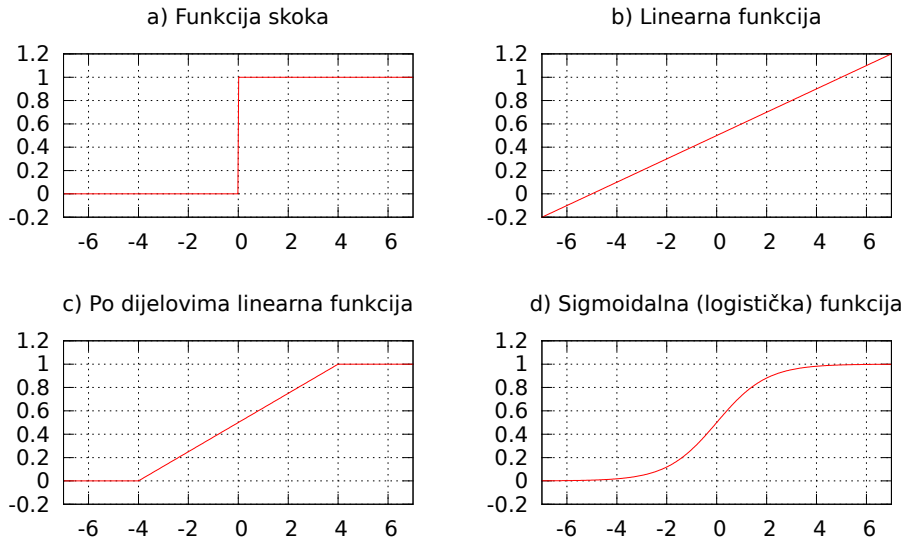
Prijenosna funkcija prikazana na slici 7.2 (a) je funkcija skoka (engl. *step-function*). Ta je funkcija definirana na sljedeći način:

$$f(net) = \begin{cases} 0, & net \leq 0 \\ 1, & \text{inače} \end{cases}$$

Ovu prijenosnu funkciju koristili su Warren McCulloch i Walter Pitts 1943. godine prilikom definiranja modela TLU perceptrona (engl. *Threshold Logic Unit*). Temeljeći se na takvom modelu neurona pokazali su kako je njime mogu računati osnovne Booleove funkcije I, ILI i NE.

Prijenosna funkcija prikazana na slici 7.2 (b) je linearna funkcija definirana izrazom:

$$f(net) = k \cdot net + l$$




Slika 7.2: Tipične prijenosne funkcije

gdje su $k$ i $l$ konstante. Neuroni s ovakvim prijenosnim funkcijama često se koriste u izlaznim slojevima neuronskih mreža. Valja međutim napomenuti da se samo ovakvim neuronima ne mogu graditi složene neuronske mreže jer se od samo linearnih elemenata ne može dobiti ništa složenijega (odnosno, čitava se mreža opet pretvara u jedan linearni element).

Kako bi se riješio uočeni problem, umjesto linearne prijenosne funkcije može se koristiti funkcija prikazana na slici 7.2 (c): po dijelovima linearna funkcija koja je definirana izrazom:

$$f(net) = \begin{cases} 0, & net \leq l \\ \frac{net-l}{u-l}, & l < net < u \\ 1, & net \geq u \end{cases}$$

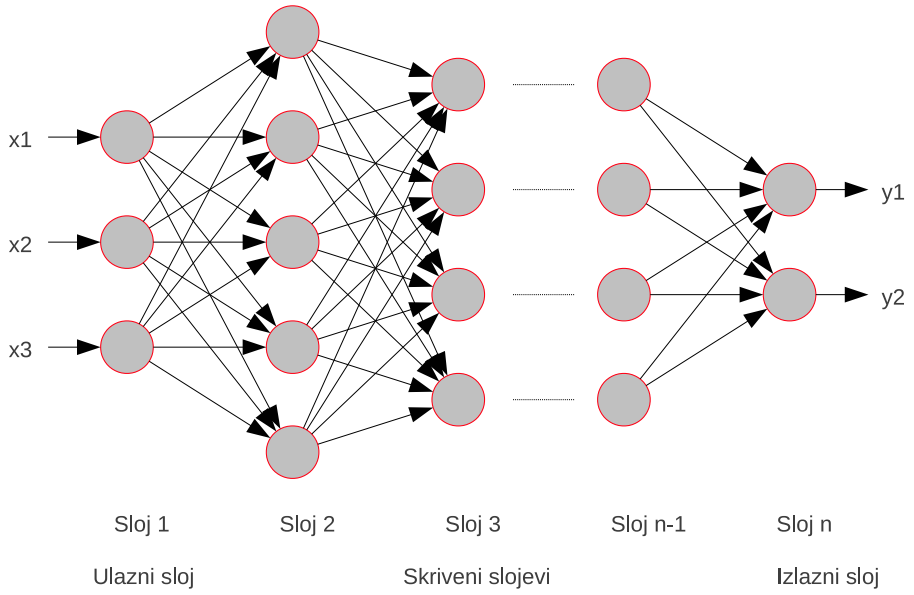
Međutim, uz sve navedene prijenosne funkcije nije se znalo kako definirati algoritam učenja složenijih neuronskih mreža koje su sastavljene od takvih elemenata. Rješenje u obliku algoritma *Backpropagation* pojavilo se tek uvođenjem derivabilnih prijenosnih funkcija čiji je najpoznatiji predstavnik sigmoidalna (ili logistička) prijenosna funkcija koja je prikazana na slici 7.2 (d) i definirana izrazom:

$$f(net) = \frac{1}{1 + e^{-net}}$$

Uz ovu funkciju zgodno je još i primijetiti da je derivacija te funkcije:

$$\frac{df(net)}{dnet} = f(net) \cdot (1 - f(net))$$

što ćemo kasnije često koristiti. Pokušajte stoga to izvesti za vježbu.




Slika 7.3: Tipična građa neuronske mreže

## 7.3 Osnovni model neuronske mreže

Izolirani neuron, kako smo vidjeli u prethodnom poglavlju, može obavljati samo jednostavnu obradu podataka. Stoga se više neurona povezuje u različite strukture poznate pod općenitim nazivom *umjetne neuronske mreže*. Jedan primjer takve mreže prikazan je na slici 7.3.

S lijeve strane mreže nalaze se ulazni neuroni. To su neuroni na koje se dovodi informacija koju je potrebno obraditi. Ulazni neuroni pri tome ne obavljaju nikakvu obradu već ulaz dovode do ostalih neurona u mreži. Unutrašnjost mreže sastoji se od neurona koji se izvana ne vide i koji nemaju nikakvog doticaja niti s ulazom, niti s izlazom mreže. Konačno, krajnje desno nalazi se izlazni sloj neurona koji rezultat obrade dostavljaju okolini.

Mreža prikazana u ovom primjeru je slojevita mreža. Međutim, općenito, mreža ne mora biti takva. U općem slučaju moguće je da bilo koji neuron bude povezan s bilo kojim drugim neuronom, čime se otvara mogućnost definiranja vrlo velikog broja različitih arhitektura umjetnih neuronskih mreža. Pri tome valja voditi računa o tome da ponašanje neuronske mreže kao i učenje, odnosno lakoća učenja, ovisi velikim dijelom o samoj arhitekturi neuronske mreže. Primjerice, slojevite neuronske mreže uz fiksan ulaz generirat će stabilan izlaz i u tom smislu odgovarat će funkcijskom preslikavanju. Dozvoli li se pak da u neuronskoj mreži postoje ciklički povezani neuroni, izlaz takve mreže čak će i uz fiksne ulaze biti vremenski promjenjiv jer će unurašnjost mreže tada moći pohranjivati *stanje*. Algoritmi za učenje takvih mreža bit će bitno kompleksniji (i neefikasniji).

## 7.4 Vrsta i podjela neuronskih mreža

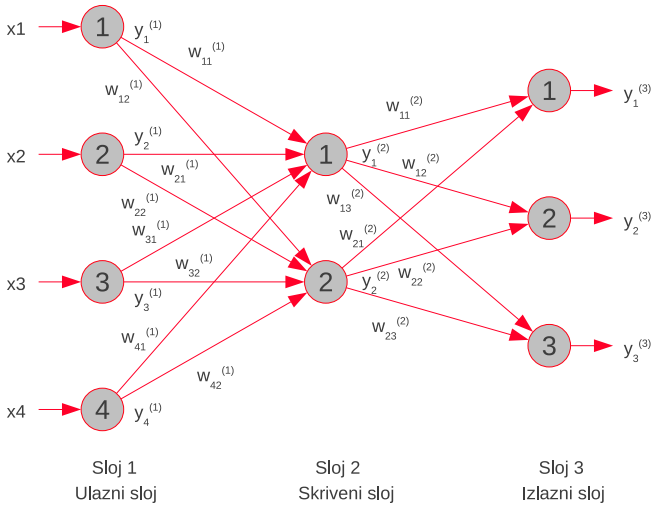
Ovisno o građi i namjeni umjetnih neuronskih mreža, do danas je razvijen čitav niz različitih vrsta umjetnih neuronskih mreža, od kojih ćemo u nastavku nabrojiti samo neke.

- Meže za učenje s učiteljem
  - Unaprijedne
    - Linearne
      - Perceptron: Rosenblatt (1958), Minsky and Papert (1969/1988)
      - Adaline: Widrow and Hoff (1960)
    - Višeslojni perceptron
      - *Backpropagation*: Rumelhart, Hinton, and Williams (1986)
    - RBF mreže (mreže s radijalnim baznim funkcijama)
  - Mreže s povratnim vezama
    - Boltzmanov stroj
    - Mreže za vremenske serije (Backpropagation kroz vrijeme, Mreže s vremenskim pomakom – Time Delay NN)
  - Mreže s natjecanjem
    - Counterpropagation mreže
- Mreže za učenje bez učitelja
  - Mreže s natjecanjem
    - Mreže za kvantizaciju vektora
    - Samoorganizirajuće mreže (npr. Kohonenov SOM)

# Poglavlje 8

# Unaprijedne neuronske mreže

Unaprijedna slojevita neuronska mreža (često još poznata i pod nazivom *višeslojni perceptron*, odnosno engl. *multilayer perceptron*) je vrsta neuronske mreže čija je struktura prikazana na slici u nastavku.




Prikazana neuronska mreža sastoji se od tri sloja.

- 1. sloj odnosno *ulazni sloj* sastoji se od neurona koji ne obavljaju nikakvu funkciju već naprosto preslikavaju dovedene ulazne podatke i čine ih dostupnima ostatku mreže. U tom smislu to su neuroni čija je prijenosna funkcija upravo funkcija identiteta. Mreža prikazana na slici ima 4 neurona u ulaznom sloju pa radi s 4-dimenzijskim podatcima. Ulaz stoga možemo zapisati kao $\vec{x} = (x_1, x_2, x_3, x_4)$. Za neurone ovog sloja vrijedi $y_i^{(1)} = x_i$.
- 2. sloj odnosno *skriveni sloj* sastoji se od neurona koji na svoje ulaze dobivaju isključivo vrijednosti izlaza neurona 1. sloja. Pri tome izlaz neurona $i$ iz 1. sloja na neuron $j$ u 2. sloju utječe s težinom $w_{ij}^{(1)}$.

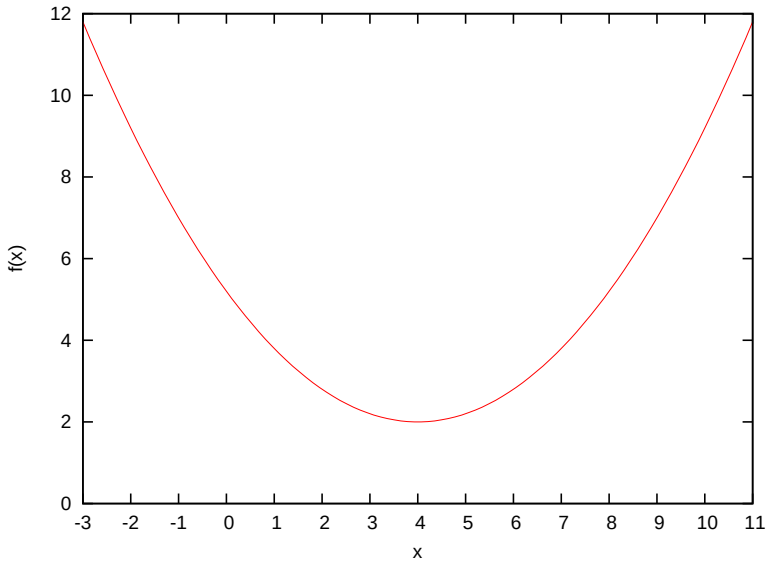
- 3. sloj odnosno *izlazni sloj* sastoji se od neurona koji na svoje ulaze dobivaju isključivo vrijednosti izlaza neurona 2. sloja. Na prikazanoj slici neurona u izlaznom sloju ima 3 što znači da prikazana neuronska mreža generira 3-dimenzijske izlaze, odnosno $\vec{y}^{(3)} = (y_1^{(3)}, y_2^{(3)}, y_3^{(3)})$. Prikazana neuronska mreža, dakle, ostvaruje preslikavanje podataka iz 4-dimenzijskog prostora $\vec{x}$ u 3-dimenzijski prostor $\vec{y}^{(3)}$.

Za mrežu kažemo da je slojevita ako u svakom sloju $k$ vrijedi da se izlazi neurona tog sloja računaju samo i isključivo temeljem izlaza neurona sloja $k-1$. Kod slojevitih mreža lateralne veze (veze između neurona u istom sloju) nisu dozvoljene.

Da bi mreža bila unaprijedna, nije nužno da bude slojevita: slojevite neuronske mreže samo su jedan od posebnih oblika unaprijednih neuronskih mreža. Da bi mreža bila unaprijedna, nužan i dovoljan uvjet je da nema ciklusa. Algoritam *Backpropagation* za učenje neuronskih mreža koji je opisan u nastavku primjenjiv je na općenitu kategoriju unaprijednih neuronskih mreža, iako ćemo ga izvesti na primjeru slojevite mreže.

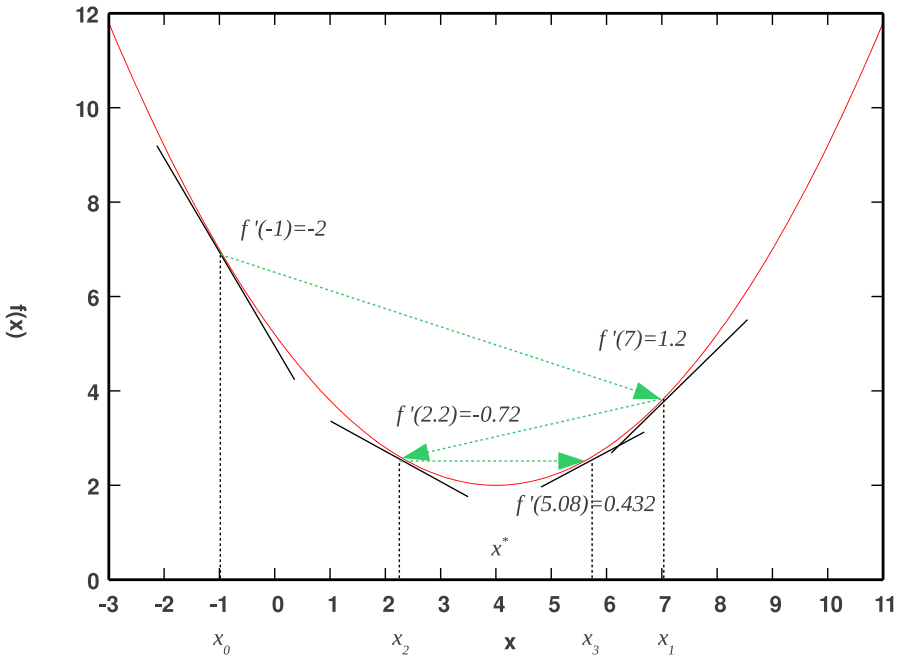
## 8.1 Učenje gradijentnim spustom

Prije no što obradimo najpoznatiji algoritam za učenje unaprijednih neuronskih mreža, podsjetimo se osnovne ideje pronalaska minimuma funkcije koja se temelji na uporabi derivacije funkcije odnosno općenitije na uporabi gradijenta. Neka je dana funkcija $f(x) = \frac{1}{5}(x-4)^2 + 2$ koja je prikazana na slici u nastavku.




Zadatak je pronaći onu vrijednost varijable $x^*$ za koju funkcija poprima minimalnu vrijednost. Sa slike je jasno da je $x^* = 4$. Ideja gradijentnog

spusta je krenuti od neke početne slučajno odabrane vrijednosti za $x$ i potom tu vrijednost umanjivati ili uvećavati ovisno o iznosu derivacije funkcije u toj točki. Grafičko je to prikazano na slici u nastavku.




Za zadanu funkciju derivaciju je trivijalno odrediti:

$$f'(x) = \frac{df(x)}{dx} = \frac{2}{5}(x-4).$$

Pretpostavimo da smo za početnu vrijednost varijable $x$ odabrali $x_0 = -1$. Vrijednost derivacije funkcije u toj točki je $\frac{df(x)}{dx}|_{x=x_0} = -2$. Kako je predznak derivacije negativan a sa slike je jasno da je $x$ potrebno uvećati, pomak koji je potrebno napraviti bit će proporcionalan negativnoj vrijednosti derivacije. Koristit ćemo sljedeći izraz:

$$x_{i+1} = x_i - \psi \cdot \frac{df(x)}{dx}|_{x=x_i}$$

pri čemu uobičajeno mala pozitivna konstanta $\psi$ kontrolira veličinu korekcije. Prvih nekoliko iteracije uzvrijednost $\psi = 4$ daju:

$x_0 = -1$ (slučajno odabrano)

$$x_1 = x_0 - \psi \cdot \frac{df(x)}{dx}|_{x=x_0} = -1 - 4 \cdot \frac{df(x)}{dx}|_{x=-1} = -1 - 4 \cdot (-2) = 7$$

$$x_2 = x_1 - \psi \cdot \frac{df(x)}{dx}|_{x=x_1} = 7 - 4 \cdot \frac{df(x)}{dx}|_{x=7} = 7 - 4 \cdot 1.2 = 2.2$$

$$x_3 = x_2 - \psi \cdot \frac{df(x)}{dx}|_{x=x_2} = 2.2 - 4 \cdot \frac{df(x)}{dx}|_{x=2.2} = 2.2 - 4 \cdot (-0.72) = 5.08$$

$$x_4 = x_3 - \psi \cdot \frac{df(x)}{dx}|_{x=x_3} = \ldots$$

Za prikazani primjer, postupak će u konačnici konvergirati prema $x^*$. Općenito, postoje tri situacije koje se mogu pojaviti prilikom provođenja gradijentnog spusta.

1. *Monotona konvergencija.* Ako je faktor $\psi$ dovoljno mali, vrijednosti za $x$ bi se monotono približavale prema $x^*$. U primjeru koji smo proveli to nije bio slučaj. No da je, primjerice, $\psi$ bio postavljen na vrijednost 1, dobili bismo slijed $x$-eva: -1, 1, 2.2, 2.92, 3.352, 3.6112, 3.76672, 3.86003, 3.91602, 3.94961, 3.96977, 3.98186, 3.98912, ... koji polako teži prema vrijednosti $x^* = 4$. Ovaj hod $x$-eva u našem primjeru odgovarao bi spustu po lijevom dijelu parabole prema minimumu.

2. *Alternirajuća konvergencija.* Ako je faktor $\psi$ umjereno velik, vrijednosti za $x$ više se ne bi monotono približavale prema $x^*$ već bi alternirale oko njega (svaki puta sve bliže i bliže). To je upravo slučaj u primjeru koji smo proveli gdje je $\psi = 4$. Uz malo više napravljenih koraka, slijed $x$-eva koji se dobiva je: -1, 7, 2.2, 5.08, 3.352, 4.3888, 3.76672, 4.13997, 3.91602, 4.05039, 3.96977, 4.01814, 3.98912, .... Taj niz opet polako teži prema vrijednosti $x^* = 4$ ali uočite kako su svaka dva susjedna elementa niza sa suprotnih strana optimalne vrijednosti za $x$.

3. *Divergencija.* Ako je faktor $\psi$ prevelik, vrijednosti za $x$ više se uopće ne bi približavale optimalnoj vrijednosti $x$-a već bi alternirale i svaki puta na sve većim udaljenostima. Primjerice, uz $\psi = 8$, slijed $x$-eva koji se dobiva je: -1, 11, -5.8, 17.72, -15.208, 30.8912, -33.64768, 56.70675, -69.78945, 107.30523, -140.62733, 206.47826, -279.46956, .... U zadanom primjeru granični slučaj uz koji još uvijek nema divergencije je $\psi = 5$ uz koji postupak niti konvergira niti divergira već oscilira; uz $x_0 = -1$ dobiva se slijed -1, 9, -1, 9, -1, 9, .... Treba napomenuti da je ova vrijednost karakteristična upravo za ovaj konkretan primjer: uz druge funkcije granična će vrijednost biti drugačija.

Ako znamo algebarski oblik funkcije koju optimirano i ako je on dovoljno jednostavan, prethodna tri slučaja možemo izvesti i formalno. Evo samo ideje ukratko.

1. *Ograda divergencije.* Pretpostavimo da je $\psi$ takav da iz koraka u korak rješenja alterniraju. Sustav još uvijek neće divergirati ako vrijedi:

$$x_1 = x_0 - \psi \cdot \frac{df(x)}{dx}|_{x=x_0}$$

$$x_2 = x_1 - \psi \cdot \frac{df(x)}{dx}|_{x=x_1} = x_0$$

odnosno ako se po povratku na početnu stranu vratimo u točku iz koje smo krenuli (a ne dalje od nje). Uvrštavanjem izraza za $x_1$ u izraz za $x_2$ i izjednačavanjem tog izraza s $x_0$ te uz zadanu funkciju $f$ s kojom smo radili i čija je derivacija $\frac{2}{5}(x-4)$ direktno slijedi da je granična vrijednost $\psi_{gr} = 5$ (probajte to za vježbu).

2. *Ograda monotonosti.* Promotrimo što se događa kada $\psi$ postupno smanjujemo od $\psi_{gr}$ na niže. Postupak će alternirati sa sve manjom i manjom amplitudom sve do trenutka kada korekcija neće biti dovoljno velika da $x_i$ koji se nalazi s jedne strane $x^*$ prebaci na njegovu drugu stranu – slučaj od interesa je dakle situacija koja $x_i$ promijeni točno toliko da postane jednak $x^*$. No sada je lagano. Možemo pisati:

$$x_1 = x_0 - \psi \cdot \frac{df(x)}{dx}|_{x=x_0} = x^*$$

što u slučaju naše zadane funkcije (i uz $x^* = 4$) direktno daje iznos $\psi_{alt} = 2.5$ (napravite taj izvod za vježbu).

Uz ovako određene vrijednosti sada možemo opisati ponašanje gradijentnog spusta za zadanu funkciju:

1. Uz $\psi \leq \psi_{alt}$ konvergencija je monotona.
2. Uz $\psi_{alt} < \psi < \psi_{gr}$ konvergencija je alternirajuća.
3. Uz $\psi = \psi_{gr}$ postupak oscilira (niti konvergira niti divergira).
4. Uz $\psi > \psi_{gr}$ postupak divergira.

## 8.2 Algoritam *Backpropagation*

Najpoznatiji algoritam za učenje unaprijednih neuronskih mreža izvest ćemo krenuvši od općenite slojevite unaprijedne neuronske mreže koja ima $d$ ulaza, $m$ izlaza i za čije nam je učenje na raspolaganju $N$ parova $(\vec{x}_i, \vec{t}_i)$ gdje je $\vec{x}_i$ $d$-dimenzijski vektor koji predstavlja $i$-ti ulazni uzorak a $\vec{t}_i$ $m$-dimenzijski

vektor koji predstavlja željeni odziv mreže za promatrani $i$-ti ulazni uzorak. Također, pretpostavka će biti da svi neuroni imaju sigmoidalnu prijenosnu funkciju, tj. da vrijedi:

$$y = \frac{1}{1 + e^{-net}},$$

gdje je *net* težinska suma za pronatrani neuron. Promatrat ćemo neuronsku mrežu koja ima $k+1$ sloj: 1. sloj je ulazni sloj, 2. sloj do $k$-ti sloj su skriveni slojevi i $k+1$-vi sloj je izlazni sloj. Neurone u svakom sloju ćemo numerirati pa ćemo tako oznakom $y_i^{(k)}$ označavati izlaz $i$-tog neurona u $k$-tom sloju.

Za potrebe definiranja postupka učenja moramo definirati kriterijsku funkciju koja mjeri kvalitetu neuronske mreže. U ovom slučaju definirat ćemo da je kriterijska funkcija jednaka polovici srednjeg kvadratnog odstupanja između svakog željenog izlaza mreže i stvarne vrijednosti koju mreža generira na izlazu i to kumulativno za sve raspoložive uzorke. Kako je izlazni sloj promatrane mreže $k+1$-vi sloj, možemo pisati:

$$E = \frac{1}{2N} \sum_{s=1}^{N} \sum_{o=1}^{m} \left( t_{s,o} - y_{s,o}^{(k+1)} \right)^2$$

pri čemu indeks $s$ ide po svim uzorcima za učenje a indeks $o$ ide po svim izlaznim neuronima. Oznaka $t_{s,o}$ tada predstavlja $o$-tu dimenziju željenog izlaza koji je pridružen $s$-tom uzorku za učenje a oznaka $y_{s,o}^{(k+1)}$ predstavlja izlaz $o$-tog neurona izlaznog sloja (dakle $k+1$-vog sloja) koji je mreža generirala kada joj je na ulaz doveden upravo $s$-ti uzorak za učenje.

Definirana funkcija $E$ funkcija je od zadanog skupa uzoraka (koji je za postupak učenja nepromjenjiv) te vrijednosti težinskih faktora, uz pretpostavku da je struktura mreže također nepromjenjiva. U tom slučaju na iznos funkcije $E$ možemo utjecati samo promjenom vrijednosti težinskih faktora. Stoga će zadaća postupka učenja biti pronaći takve vrijednosti težinskih faktora uz koje će iznos funkcije $E$ biti minimalan.

U skladu s prethodno pojašnjenom idejom gradijentnog spusta, to znači da je potrebno izračunati gradijent od $E$ kojeg čine redom parcijalne derivacije od $E$ po svakom od težinskih faktora, tj. želimo izračunati:

$$\frac{\partial E}{\partial w_{ij}^{(k)}}.$$

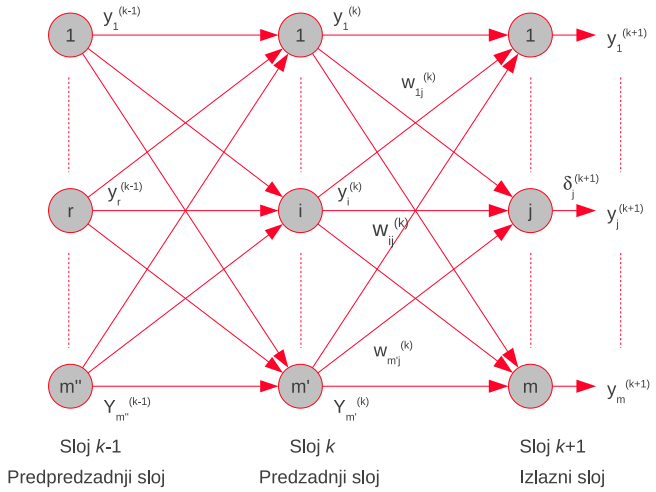
Tada ćemo, u skladu s pravilom gradijentnog spusta, moći ažurirati težinu $w_{ij}^{(k)}$ na sljedeći način:

$$w_{ij}^{(k)} \leftarrow w_{ij}^{(k)} - \psi \cdot \frac{\partial E}{\partial w_{ij}^{(k)}} \tag{8.1}$$

gdje će $\psi$ biti mala pozitivna konstanta.

### 8.2.1 Slučaj 1: izlazni sloj

Najprije ćemo razmotriti slučaj u kojem težinski faktor pripada neuronu izlaznog sloja. Takva situacija prikazana je na sljedećoj slici.




Potrebno je izračunati $\frac{\partial E}{\partial w_{ij}^{(k)}}$. Možemo pisati:

$$\frac{\partial E}{\partial w_{ij}^{(k)}} = \frac{\partial}{\partial w_{ij}^{(k)}} \left[ \frac{1}{2N} \sum_{s=1}^{N} \sum_{o=1}^{m} \left( t_{s,o} - y_{s,o}^{(k+1)} \right)^2 \right] \tag{8.2}$$

$$= \frac{1}{2N} \sum_{s=1}^{N} \sum_{o=1}^{m} 2 \cdot \left( t_{s,o} - y_{s,o}^{(k+1)} \right) \cdot (-1) \cdot \frac{\partial y_{s,o}^{(k+1)}}{\partial w_{ij}^{(k)}} \tag{8.3}$$

$$= -\frac{1}{N} \sum_{s=1}^{N} \sum_{o=1}^{m} \left( t_{s,o} - y_{s,o}^{(k+1)} \right) \cdot \frac{\partial y_{s,o}^{(k+1)}}{\partial w_{ij}^{(k)}}. \tag{8.4}$$

Kako je težina $w_{ij}^{(k)}$ težina koja spaja $i$-ti neuron u $k$-tom sloju i $j$-ti neuron u $k+1$-vom sloju, on utječe samo na izlaz neurona $j$ u $k+1$-vom sloju. Stoga parcijalne derivacije $\frac{\partial y_{s,o}^{(k+1)}}{\partial w_{ij}^{(k)}}$ za slučaj da je $o \neq j$ iščezavaju te vrijedi:

$$\sum_{o=1}^{m} \frac{\partial y_{s,o}^{(k+1)}}{\partial w_{ij}^{(k)}} = \frac{\partial y_{s,j}^{(k+1)}}{\partial w_{ij}^{(k)}}$$

tako da možemo pisati:

$$\frac{\partial E}{\partial w_{ij}^{(k)}} = -\frac{1}{N} \sum_{s=1}^{N} \left( t_{s,j} - y_{s,j}^{(k+1)} \right) \cdot \frac{\partial y_{s,j}^{(k+1)}}{\partial w_{ij}^{(k)}}. \tag{8.5}$$

Za izračun preostale parcijalne derivacije poslužit ćemo se pravilom ulančavanja derivacija pa možemo pisati:

$$\frac{\partial y_{s,j}^{(k+1)}}{\partial w_{ij}^{(k)}} = \frac{\partial y_{s,j}^{(k+1)}}{\partial net_{s,j}^{(k+1)}} \cdot \frac{\partial net_{s,j}^{(k+1)}}{\partial w_{ij}^{(k)}} \tag{8.6}$$

Uočimo da za sigmoidalnu prijenosnu funkciju vrijedi:

$$\frac{\partial y_{s,j}^{(k+1)}}{\partial net_{s,j}^{(k+1)}} = y_{s,j}^{(k+1)} \cdot (1 - y_{s,j}^{(k+1)}).$$

Također, uočimo da je $net_{s,j}^{(k+1)}$ težinska suma za neuron $k+1$-vog sloja, odnosno da se računa kao:

$$net_{s,j}^{(k+1)} = w_{1,j}^{(k)} \cdot y_{s,1}^{(k)} + w_{2,j}^{(k)} \cdot y_{s,2}^{(k)} + \cdots + w_{i,j}^{(k)} \cdot y_{s,i}^{(k)} + \cdots$$

Uočimo sada da, zbog svojstva slojevitosti mreže (nema cikličkih i lateralnih veza) niti jedan od izlaza $y_{s,1}^{(k)}$, $y_{s,2}^{(k)}$, ... temeljem kojih se računa $net_{s,j}^{(k+1)}$ ne ovise o težini $w_{i,j}^{(k)}$ već su s obzirom na nju konstante. Tada je tražena parcijalna derivacija jednaka:

$$\frac{\partial net_{s,j}^{(k+1)}}{\partial w_{ij}^{(k)}} = y_{s,i}^{(k)}.$$

Slijedi da je:

$$\frac{\partial y_{s,j}^{(k+1)}}{\partial w_{ij}^{(k)}} = y_{s,j}^{(k+1)} \cdot (1 - y_{s,j}^{(k+1)}) \cdot y_{s,i}^{(k)}$$

što uvrštavanjem u izraz (8.5) daje:

$$\frac{\partial E}{\partial w_{ij}^{(k)}} = -\frac{1}{N} \sum_{s=1}^{N} y_{s,j}^{(k+1)} \cdot \left(1 - y_{s,j}^{(k+1)}\right) \cdot \left(t_{s,j} - y_{s,j}^{(k+1)}\right) \cdot y_{s,i}^{(k)} \tag{8.7}$$

$$= -\frac{1}{N} \sum_{s=1}^{N} \delta_{s,j}^{(k+1)} \cdot y_{s,i}^{(k)} \tag{8.8}$$

pri čemu je:

$$\delta_{s,j}^{(k+1)} = y_{s,j}^{(k+1)} \cdot \left(1 - y_{s,j}^{(k+1)}\right) \cdot \left(t_{s,j} - y_{s,j}^{(k+1)}\right).$$

Veličina $\delta_{s,j}^{(k+1)}$ predstavlja pogrešku $j$-tog izlaznog neurona za $s$-ti uzorak za učenje. Pravilo za ažuriranje težine $w_{ij}^{(k)}$ tada glasi:

$$w_{ij}^{(k)} \leftarrow w_{ij}^{(k)} - \psi \cdot \frac{\partial E}{\partial w_{ij}^{(k)}} \tag{8.9}$$

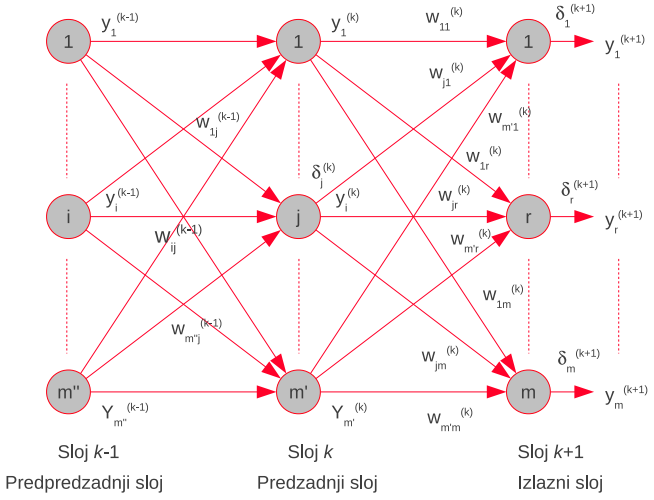
$$\leftarrow w_{ij}^{(k)} - \psi \cdot \left( -\frac{1}{N} \sum_{s=1}^{N} \delta_{s,j}^{(k+1)} \cdot y_{s,i}^{(k)} \right) \tag{8.10}$$

$$\leftarrow w_{ij}^{(k)} + \eta \cdot \left( \sum_{s=1}^{N} \delta_{s,j}^{(k+1)} \cdot y_{s,i}^{(k)} \right) \tag{8.11}$$

gdje smo umjesto umnoška konstanti $\psi \cdot \frac{1}{N}$ uveli novu konstantu $\eta$.

### 8.2.2 Slučaj 2: težina pripada skrivenom sloju

Ovaj slučaj razmotrit ćemo na primjeru težine koja se nalazi u zadnjem skrivenom sloju i potom napraviti poopćenje na proizvoljni sloj. Ovaj slučaj prikazan je na sljedećoj slici.




Potrebno je izračunati parcijalnu derivaciju kriterijske funkcije po promatranoj težini $w_{ij}^{(k-1)}$.

$$\frac{\partial E}{\partial w_{ij}^{(k-1)}} = \frac{\partial}{\partial w_{ij}^{(k-1)}} \left[ \frac{1}{2N} \sum_{s=1}^{N} \sum_{o=1}^{m} \left( t_{s,o} - y_{s,o}^{(k+1)} \right)^2 \right] \tag{8.12}$$

$$= \frac{1}{2N} \sum_{s=1}^{N} \sum_{o=1}^{m} 2 \cdot \left( t_{s,o} - y_{s,o}^{(k+1)} \right) \cdot (-1) \cdot \frac{\partial y_{s,o}^{(k+1)}}{\partial w_{ij}^{(k-1)}} \tag{8.13}$$

$$= -\frac{1}{N} \sum_{s=1}^{N} \sum_{o=1}^{m} \left( t_{s,o} - y_{s,o}^{(k+1)} \right) \cdot \frac{\partial y_{s,o}^{(k+1)}}{\partial w_{ij}^{(k-1)}} \tag{8.14}$$

$$= -\frac{1}{N} \sum_{s=1}^{N} \sum_{o=1}^{m} \left( t_{s,o} - y_{s,o}^{(k+1)} \right) \cdot \frac{\partial y_{s,o}^{(k+1)}}{\partial net_{s,o}^{(k+1)}} \cdot \frac{\partial net_{s,o}^{(k+1)}}{\partial w_{ij}^{(k-1)}} \tag{8.15}$$

$$\tag{8.16}$$

Raspišimo svaku od prikaznih parcijalnih derivacija. Prva je derivacija sigmoidalne prijenosne funkcije što je trivijalno:

$$\frac{\partial y_{s,o}^{(k+1)}}{\partial net_{s,o}^{(k+1)}} = y_{s,o}^{(k+1)} \cdot (1 - y_{s,o}^{(k+1)}).$$

Da bismo razriješili preostalu parcijalnu derivaciju, fiksirajmo na trenutak $o$ i pogledajmo kako se računa $net_{s,o}^{(k+1)}$. Vrijedi:

$$net_{s,o}^{(k+1)} = w_{1o} \cdot y_{s,1}^{(k)} + w_{2o} \cdot y_{s,2}^{(k)} + \cdots w_{jo} \cdot y_{s,j}^{(k)} + \cdots$$

Zbog toga što je mreža slojevita, težina $w_{ij}^{(k-1)}$ koja se nalazi u težinskoj sumi $j$-tog neurona u $k$-tom sloju utječe samo na njegov izlaz, odnosno na $y_{s,j}^{(k)}$. Stoga $\forall o \neq j$ vrijedi $\frac{\partial y_{s,o}^{(k)}}{\partial w_{ij}^{(k-1)}} = 0$ jer promjena te težine nema nikakvog utjecaja na te izlaze. Slijedi da je samo $\frac{\partial y_{s,j}^{(k)}}{\partial w_{ij}^{(k-1)}} \neq 0$, pa je:

$$\frac{\partial net_{s,o}^{(k+1)}}{\partial w_{ij}^{(k-1)}} = \frac{\partial}{\partial w_{ij}^{(k-1)}} \left( w_{1o} \cdot y_{s,1}^{(k)} + w_{2o} \cdot y_{s,2}^{(k)} + \cdots w_{jo} \cdot y_{s,j}^{(k)} + \cdots \right)$$

$$= w_{1o} \cdot \frac{\partial y_{s,1}^{(k)}}{\partial w_{ij}^{(k-1)}} + w_{2o} \cdot \frac{\partial y_{s,2}^{(k)}}{\partial w_{ij}^{(k-1)}} + \cdots w_{jo} \cdot \frac{\partial y_{s,j}^{(k)}}{\partial w_{ij}^{(k-1)}} + \cdots$$

$$= w_{jo} \cdot \frac{\partial y_{s,j}^{(k)}}{\partial w_{ij}^{(k-1)}}.$$

Kombiniranjem ovih rezultata slijedi:

$$\begin{aligned}\frac{\partial E}{\partial w_{ij}^{(k-1)}} &= -\frac{1}{N}\sum_{s=1}^{N}\sum_{o=1}^{m}\left[\left(t_{s,o}-y_{s,o}^{(k+1)}\right)\cdot y_{s,o}^{(k+1)}\cdot(1-y_{s,o}^{(k+1)})\cdot w_{jo}\cdot\frac{\partial y_{s,j}^{(k)}}{\partial w_{ij}^{(k-1)}}\right]\\ &= -\frac{1}{N}\sum_{s=1}^{N}\left[\frac{\partial y_{s,j}^{(k)}}{\partial w_{ij}^{(k-1)}}\sum_{o=1}^{m}\left(t_{s,o}-y_{s,o}^{(k+1)}\right)\cdot y_{s,o}^{(k+1)}\cdot(1-y_{s,o}^{(k+1)})\cdot w_{jo}\right]\\ &= -\frac{1}{N}\sum_{s=1}^{N}\left[\frac{\partial y_{s,j}^{(k)}}{\partial w_{ij}^{(k-1)}}\sum_{o=1}^{m}\delta_{s,o}^{(k+1)}\cdot w_{jo}\right]\end{aligned}$$

Konačno, preostalu parcijalnu derivaciju $\frac{\partial y_{s,j}^{(k)}}{\partial w_{ij}^{(k-1)}}$ sada je trivijalno razriješiti jer je težina $w_{ij}^{(k-1)}$ upravo težina neurona čiji je izlaz $y_{s,j}^{(k)}$. Koristeći pravilo ulančavanja možemo pisati:

$$\begin{aligned}\frac{\partial y_{s,j}^{(k)}}{\partial w_{ij}^{(k-1)}} &= \frac{\partial y_{s,j}^{(k)}}{\partial net_{s,j}^{(k)}}\cdot\frac{\partial net_{s,j}^{(k)}}{\partial w_{ij}^{(k-1)}}\\ &= y_{s,j}^{(k)}\cdot(1-y_{s,j}^{(k)})\cdot y_{s,i}^{(k-1)}\end{aligned}$$

Uvrštavanjem slijedi:

$$\begin{aligned}\frac{\partial E}{\partial w_{ij}^{(k-1)}} &= -\frac{1}{N}\sum_{s=1}^{N}\left[y_{s,j}^{(k)}\cdot(1-y_{s,j}^{(k)})\cdot y_{s,i}^{(k-1)}\cdot\sum_{o=1}^{m}\delta_{s,o}^{(k+1)}\cdot w_{jo}\right]\\ &= -\frac{1}{N}\sum_{s=1}^{N}\left[y_{s,i}^{(k-1)}\cdot\left\{y_{s,j}^{(k)}\cdot(1-y_{s,j}^{(k)})\cdot\sum_{o=1}^{m}\delta_{s,o}^{(k+1)}\cdot w_{jo}\right\}\right]\\ &= -\frac{1}{N}\sum_{s=1}^{N}\left[y_{s,i}^{(k-1)}\cdot\delta_{s,j}^{(k)}\right]\end{aligned}$$

pri čemu je pogreška skrivenog neurona $j$ u sloju $k$ za uzorak $s$ definirana kao derivacija njegove prijenosne funkcije pomnožena težinskom sumom pogrešaka neurona sljedećeg sloja, odnosno:

$$\delta_{s,j}^{(k)} = y_{s,j}^{(k)}\cdot(1-y_{s,j}^{(k)})\cdot\sum_{o=1}^{m}\delta_{s,o}^{(k+1)}\cdot w_{jo}.$$

Ažuriranje težine $w_{ij}^{(k-1)}$ tada se obavlja na sljedeći način:

$$w_{ij}^{(k-1)} \leftarrow w_{ij}^{(k-1)} - \psi \cdot \frac{\partial E}{\partial w_{ij}^{(k-1)}} \tag{8.17}$$

$$\leftarrow w_{ij}^{(k-1)} - \psi \cdot \left( -\frac{1}{N} \sum_{s=1}^{N} \left[ y_{s,i}^{(k-1)} \cdot \delta_{s,j}^{(k)} \right] \right) \tag{8.18}$$

$$\leftarrow w_{ij}^{(k-1)} + \eta \cdot \left( \sum_{s=1}^{N} \left[ \delta_{s,j}^{(k)} \cdot y_{s,i}^{(k-1)} \right] \right) \tag{8.19}$$

gdje smo umjesto umnoška konstanti $\psi \cdot \frac{1}{N}$ uveli novu konstantu $\eta$.

Razmotrimo li umjesto neurona iz zadnjeg skrivenog sloja neki od neurona iz prethodnih slojeva, dolazi se do strukturno istih formula: za svaki neuron iz skrivenog sloja $l$ pogreška se računa kao derivacija njegove prijenosne funkcije pomnožena težinskom sumom pogrešaka neurona sljedećeg sloja na koje je promatrani neuron direktno spojen.

### 8.2.3 Opći slučaj

Analizom neurona izlaznog sloja dobili smo izraz za ažuriranje težina:

$$w_{ij}^{(k)} \leftarrow w_{ij}^{(k)} + \eta \cdot \left( \sum_{s=1}^{N} \delta_{s,j}^{(k+1)} \cdot y_{s,i}^{(k)} \right)$$

a analizom neurona skrivenog sloja izraz:

$$w_{ij}^{(k-1)} \leftarrow w_{ij}^{(k-1)} + \eta \cdot \left( \sum_{s=1}^{N} \delta_{s,j}^{(k)} \cdot y_{s,i}^{(k-1)} \right)$$

Pravilo ažuriranja je dakle isto – razlika je samo u načinu kako se računa pogreška. Za neurone izlaznog sloja pogrešku je moguće izračunati direktno: ona je jednaka derivaciji prijenosne funkcije pomnoženoj s razlikom između očekivane vrijednosti izlaza i dobivene vrijednosti izlaza. Pogreške neurona skrivenih slojeva potom se računaju na opisani način temeljem pogrešaka izlaznih neurona.

Težinski faktore koje do sada nismo razmatrali su slobodni težinski faktori (odnosno ono što u *net*-u odgovara težini $w_0$). Napravimo li i taj izvod (ostavljamo čitatelju za vježbu), dobit će se izrazi koji odgovaraju prethodno izvedenima uz postavljanje $y_{s,i}^{(k)} = 1$ odnosno $y_{s,i}^{(k-1)} = 1$; drugim riječima, možemo se praviti da su te težine spojene između fiksnih neurona koji stalno na izlazu generiraju vrijednost 1.

## 8.3 Inačice algoritma *Backpropagation*

Direktnom primjenom metode gradijentnog spusta na minimizaciju srednje kvadratne pogreške nad cjelokupnim skupom uzoraka za učenje dobili smo sljedeći izraz za ažuriranje težinskih faktora:

$$w_{ij}^{(k)} \leftarrow w_{ij}^{(k)} + \eta \cdot \left( \sum_{s=1}^{N} \delta_{s,j}^{(k+1)} \cdot y_{s,i}^{(k)} \right).$$

U tom izrazu suma:

$$\sum_{s=1}^{N} \delta_{s,j}^{(k+1)} \cdot y_{s,i}^{(k)}$$

dolazi iz izraza za točnu vrijednost gradijenta kriterijske funkcije u točki koja je određena trenutnim vrijednostima težinskih faktora. Algoritam učenja ostvaren na ovaj način spada u porodicu algoritama *grupnog* učenja (engl. *batch learning algorithm*) kod kojih se učenje događa tek po predočenju svih uzoraka za učenje. U tom smislu, jedna je epoha jednaka jednoj iteraciji ovako izvedenog algoritma.

S obzirom da ovako izvedeni algoritam u svakoj točki računa točan iznos gradijenta, algoritam garantira da se nakon svakog koraka ukupna pogreška (uz dovoljno malu konstantu $\eta$) neće povećavati: ili će stagnirati, ili će se smanjivati. Problem ove izvedbe, međutim, leži u tome što je funkcija koja se minimizira visoko multimodalana te gradijent u svakoj točki uvijek pokazuje u smjeru najbližeg lokalnog optimuma a ne globalnog. Stoga ovakva izvedba algoritma učenja pokazuje vrlo malu otpornost na zaglavljivanje u lokalnim optimumima.

Kako bi se riješio ovaj problem, moguće je napraviti modifikaciju algoritma koja umjesto prave vrijednosti gradijenta u svakoj točki radi procjenu gradijenta samo na temelju jednog uzorka i zatim odmah radi korekciju težina temeljem te procjene – takva izvedba poznata je pod nazivom *stohastički Backpropagation*. To se postiže tako da se umjesto sume:

$$\sum_{s=1}^{N} \delta_{s,j}^{(k+1)} \cdot y_{s,i}^{(k)}$$

u $s$-toj iteraciji koristi aproksimacija gradijenta:

$$\delta_{s,j}^{(k+1)} \cdot y_{s,i}^{(k)}$$

i korekcija radi samo temeljem te aproksimacije:

$$\text{Za svaki } s \text{ iz [1, ..., N]} \qquad w_{ij}^{(k)} \leftarrow w_{ij}^{(k)} + \eta \cdot \delta_{s,j}^{(k+1)} \cdot y_{s,i}^{(k)}.$$

Time se postiže da algoritam uči nakon svakog predočenog uzorka. Takva varijanta algoritma pripada porodici algoritama *pojedinačnog* učenja (engl.

*online learning algorithm*). Kod ove izvedbe algoritma jednu *iteraciju* čini predočavanje točno jednog uzorka i korigiranje težina dok se slijed od $N$ iteracija u kojima se redom mreži predoče svi uzorci naziva jednom *epohom*. Faktor $\eta$ koji se ovdje koristi može biti $N$ puta veći od faktora koji se koristi kod klasičnog algoritma *Backpropagation* (razmislite zašto?).

Za razliku od klasičnog algoritma *Backpropagation*, stohastički Backpropagation pokazuje povećanu otpornost na zapinjanje u lokalnim optimumima i time je algoritam koji se najčešće bira.

### 8.3.1 Dodavanje inercije

Još jedan od načina kako pospješiti otpornost na lokalne optimume jest u izraz za ažuriranje težine dodati novi član koji će trenutnoj korekciji dodati određen iznos korekcije iz prethodnog koraka. Time se simulira utjecaj inercije koji može pospiješiti prolazak kroz lokalni optimum. Izraz koji se tada koristi opisan je u nastavku, na primjeru stohastičkog algoritma *Backpropagation*.

Prikažimo izvedeno pravilo ažuriranja težina na malo drugačiji način:

$$w_{ij}^{(k)} \leftarrow w_{ij}^{(k)} + \Delta w_{ij}^{(k)}$$

pri čemu smo uveli posebnu oznaku za iznos promjene:

$$\Delta w_{ij}^{(k)} = \eta \cdot \delta_{s,j}^{(k+1)} \cdot y_{s,i}^{(k)}.$$

Novo pravilo ažuriranja koje ima inercijski član tada glasi:

$$\Delta w_{ij}^{(k)} = \eta \cdot \delta_{s,j}^{(k+1)} \cdot y_{s,i}^{(k)} + \alpha \cdot \Delta w_{ij}^{(k)'}$$

gdje je s $\Delta w_{ij}^{(k)'}$ označena vrijednost korekcije iz prethodnog koraka. Varijanta pravila koje dodaje moment također je:

$$\Delta w_{ij}^{(k)} = (1-\alpha) \cdot \eta \cdot \delta_{s,j}^{(k+1)} \cdot y_{s,i}^{(k)} + \alpha \cdot \Delta w_{ij}^{(k)'}.$$

Kod ove varijante odabirom različitih vrijednosti parametra $\alpha$ može se direktno podesiti u kojoj mjeri ažuriranje težinskih faktora uzima u obzir trenutnu vrijednost gradijenta pogreške a u kojoj mjeri ažuriranje iz prethodnog koraka.

## 8.4 Dodatne napomene

### 8.4.1 Kriteriji zaustavljanja postupka učenja

Prilikom postupka učenja treba odlučiti u kojem je trenutku najbolje prekinuti postupak učenja. Tih kriterija može biti nekoliko.

1. *Kriterij maksimalne pogreške.* Za svaki se uzorak $s$ i za svaki izlaz $o$ provjerava je li zadovoljeno:

$$\left| t_{s,o} - y_{s,o}^{(k+1)} \right| \leq \epsilon$$

gdje je $\epsilon$ unaprijed zadan mali pozitivni broj. Time se postupak prekida tek kada je i najveća pogreška na bilo kojem od izlaza manja ili jednaka zadanoj ogradi $\epsilon$.

2. *Kriterij ukupne kvadratne pogreške.* Zahtjevamo da je:

$$\sum_{s=1}^{N} \sum_{o=1}^{m} \left( t_{s,o} - y_{s,o}^{(k+1)} \right)^2 \leq \epsilon$$

odnosno postupak se zaustavlja čim ukupna kvadratna pogreška postane manja ili jednaka danoj ogradi $\epsilon$. U ovom slučaju moguće je da mreža neke uzorke nauči jako loše a druge jako dobro jer jedini kriterij koji se koristi je provjera združene pogreške.

3. *Uporaba unakrsne validacije.* Dostupan skup uzoraka veličine $M$ temeljem kojih bismo htjeli obaviti učenje podijeli se u dva skupa: skup za učenje, obično veličine $\frac{9}{10}M$ i skup za validaciju, obično veličine $\frac{1}{10}M$. Mreža se uči samo temeljem skupa za učenje, a ponašanje mreže na skupu za validaciju se periodički provjerava. Inicijalno, kako mreža uči, pogreška na oba skupa će padati. Međutim, u jednom trenutku mreža će se početi jako dobro prilagođavati uzorcima skupa za učenje i počet će gubiti sposobnost generalizacije. To ćemo primjetiti na skupu za validaciju jer će tada pogreška na tom skupu početi rasti. Postupak učenja stoga treba prekinuti u trenutku kada pogreška na skupu na validaciju dosegne svoj minimum – nakon toga mreža postaje *pretrenirana.*

### 8.4.2 Ubrzavanje postupka učenja

Početne vrijednosti koje se dodjeljuju težinskim faktorima biraju se sasvim slučajno. Međutim, to može imati značajne posljedice na brzinu kojom mreža počinje učiti. Pogledajmo to na primjeru jednostavnog sigmoidalnog neurona s 5 ulaza. Njegov je izlaz definiran izrazom:

$$y = f(net), \qquad net = \sum_{i=1}^{5} w_i x_i + w_0, \qquad f(net) = \frac{1}{1 + e^{-net}}.$$

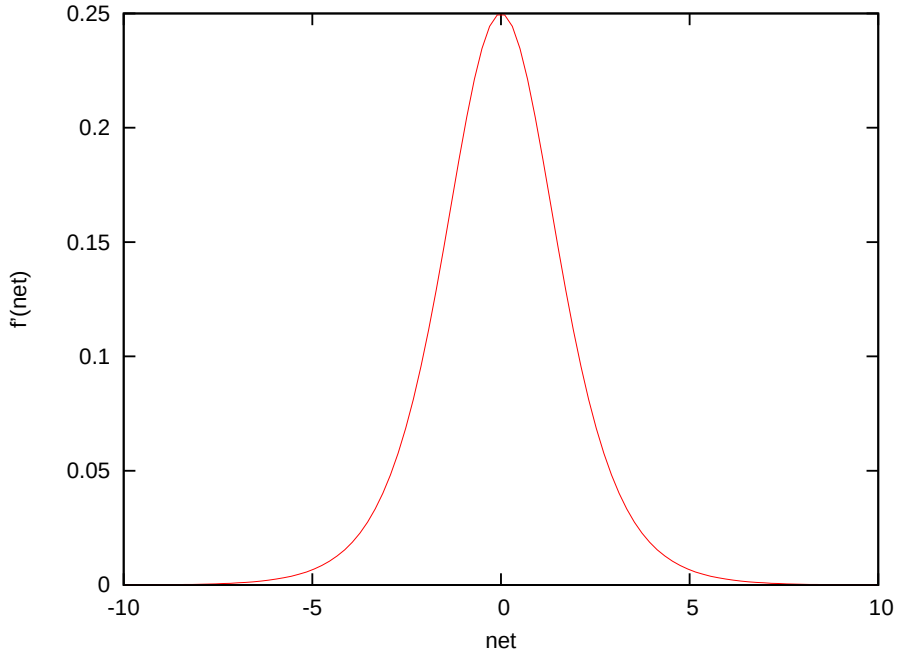
Prisjetimo se također kako je definirana korekcija težine:

$$\begin{aligned} \Delta w_{ij}^{(k)} &= \eta \cdot \delta_{s,j}^{(k+1)} \cdot y_{s,i}^{(k)} \\ &= \eta \cdot y_{s,j}^{(k+1)} \cdot \left(1 - y_{s,j}^{(k+1)}\right) \cdot \left(t_{s,j} - y_{s,j}^{(k+1)}\right) \cdot y_{s,i}^{(k)} \end{aligned}$$

Primijetimo da se korekcija množi s derivacijom sigmoidalne funkcije odnosno članom $y_{s,j}^{(k+1)} \cdot \left(1 - y_{s,j}^{(k+1)}\right)$. Raspišemo li to kao funkciju od $net$, imamo:

$$f'(net) = \frac{1}{1+e^{-net}} \cdot \left(1 - \frac{1}{1+e^{-net}}\right)$$

što je grafički prikazano na slici u nastavku.




Ta funkcija već i za relativno male vrijednosti $net$-a poprima vrlo mali iznos kojim prigušuje ukupnu korekciju i time značajno usporava postupak učenja. Stoga se preporuča početne vrijednosti za težinske faktore birati slučajno iz intervala koji će garantirati da inicijalna vrijednost $net$-a ne bude prevelika. Primjerice, za $net = 2.4$ vrijednost umnoška je nešto manja od 0.1. Postavimo li stoga kao uvjet $net <= 2.4$ i uzevši u obzir da u našem primjeru s 5-ulaznim neuronom ulaze generiraju opet drugi neuroni čiji je

izlaz ograničen s +1, imamo:

$$\begin{aligned} net &\leq 2.4 \\ \sum_{i=1}^{5} w_i x_i + w_0 &\leq 2.4 \\ \sum_{i=1}^{5} w_i + w_0 &\leq 2.4 \\ 6 \cdot w &\leq 2.4 \\ w &\leq \frac{2.4}{6} \\ w &\leq 0.4 \end{aligned}$$

iz čega slijedi da bi svaku težinu inicijalno trebali postaviti na slučajni broj iz intervala $[-0.4, +0.4]$.

Osim prikladnog odabira početnih vrijednosti za težinske faktore, treba obratiti pažnju na još nekoliko pojedinosti.

- Različiti uzorci za mrežu mogu biti različito zahtjevni za naučiti. Stoga umjesto da se uzorci mreži prikazuju slijedno i svaki u trajanju od jedne iteracije, algoritam se može modificirati na način da se za svaki uzorak uči tako dugo dok se ne nauči pa se tek potom krene na sljedeći uzorak. Takav se postupak potom ponavlja iz epohe u epohu.

- Parametri $\eta$ i $\alpha$ mogu se dinamički mijenjati. U literaturi je za tu svrhu opisano nekoliko mogućnosti koje mogu doprinijeti ubrzanju postupka učenja.

- Korekcija težinskog faktora može se skalirati faktorom $e^{\rho \cdot cos(\phi)}$ gdje je $\rho \approx 0.2$ a $\phi$ kut između trenutnog vektora gradijenta i vektora gradijenta iz prethodnog koraka. Ideja je sljedeća: ako je taj kut 0 ili vrlo mali, korekcije idu u istom smjeru pa se isplati pomak i dodatno povećati u tom smjeru jer je za očekivati da bi i sljedeća korekcija tada mogla biti u istom smjeru. S druge pak strane, što je taj kut veći (ili čak veći od 90°) iznos faktora pada čime se korekcija prigušuje jer smo očito u području gdje se gradijent jako mijenja pa treba raditi male korake.

### 8.4.3 Primjeri učenja unaprijedne slojevite neuronske mreže

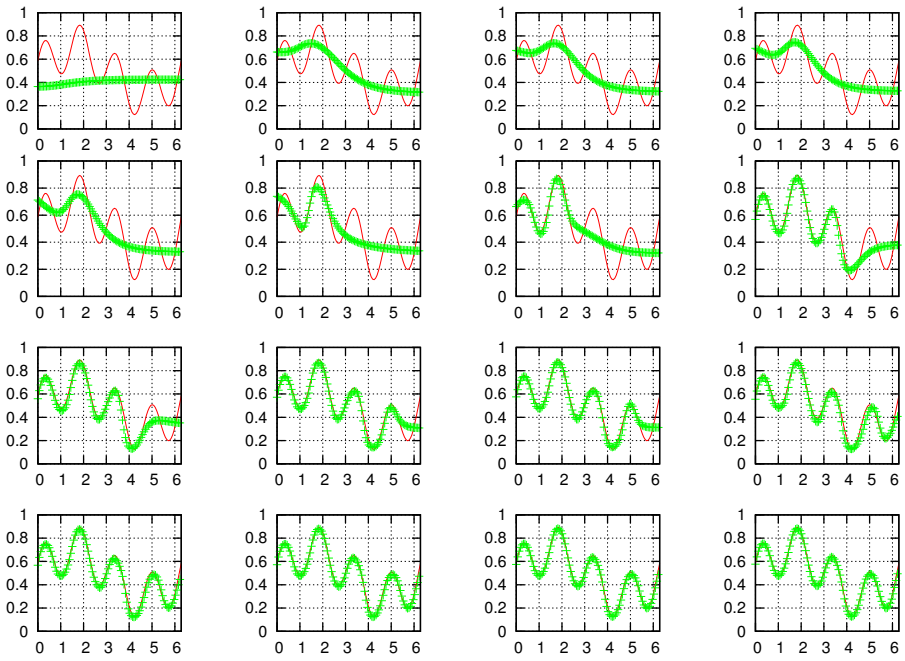
Primjer učenja pokazat ćemo na primjeru učenja funkcije:

$$f(x) = 0.2sin(x) + 0.2sin(4x + \frac{\pi}{7}) + 0.5.$$

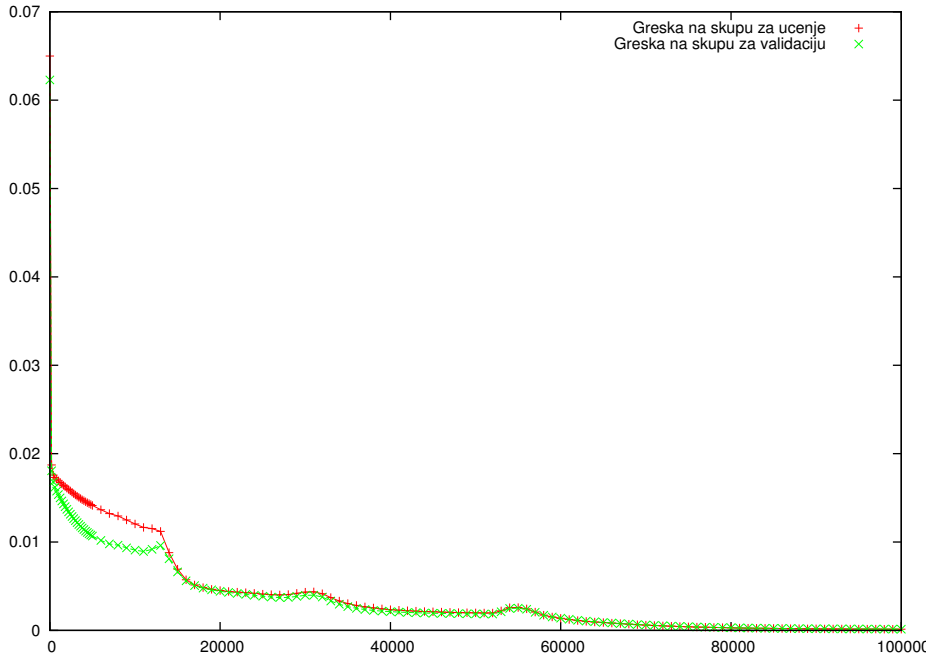
Za potrebe pripreme podataka funkcija $f$ uzorkovana je na intervalu $[0, 2\pi]$ u 200 točaka. Time je prikupljen skup od 200 uzoraka za učenje. Taj je skup potom podijeljen (slučajno) u 100 uzoraka koji čine skup za učenje te 100 uzoraka koji čine skup za validaciju. Prilikom postupka učenja, neuronska mreža težinske faktore mijenja isključivo temeljem skupa za učenje.

Tijek učenja provedenog postupkom *Backpropagation* prikazan je na slici u nastavku. Postupak učenja je proveden kroz 100 000 epoha. Kako se radi o klasičnom postupku *Backpropagation*, svaka se epoha sastoji od predočavanja svih 100 uzoraka iz skupa za učenje neuronskoj mreži i akumuliranja korekcija za svaki uzorak. Tek na kraju epohe kada je za svaku težinu poznata konačna kumulativna korekcija mijenjaju se težinski faktori. Prisjetimo se, ovo je potrebno kako bi se u trenutku korekcije znao pravi iznos gradijenta i kako bi pomak bio u suprotnom smjeru od smjera pravog gradijenta. Tijekom postupka učenja, težine su, dakle, promijenjene 100 000 puta.

Na slici u nastavku prikazan su stanja (smjer čitanja je s lijeva u desno pa odozgo prema dolje): stanje s nasumično odabranim težinama, stanje nakon 500 epoha, 1000 epoha, 1500 epoha, 2000 epoha, 5000 epoha, 10000 epoha, 20000 epoha, 30000 epoha, 40000 epoha, 50000 epoha, 60000 epoha, 70000 epoha, 80000 epoha, 90000 epoha i konačno stanje nakon 100000 epoha.

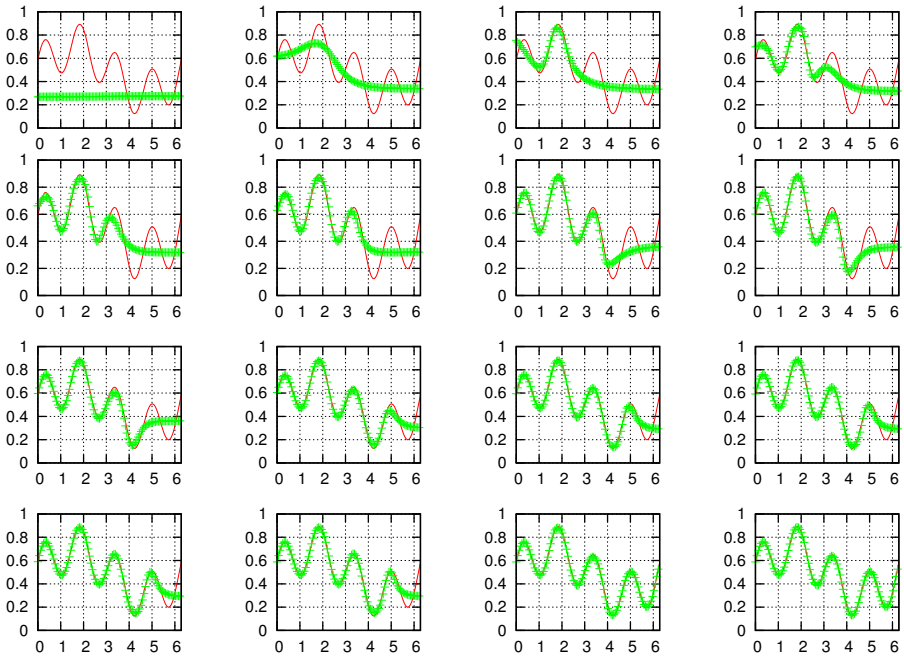


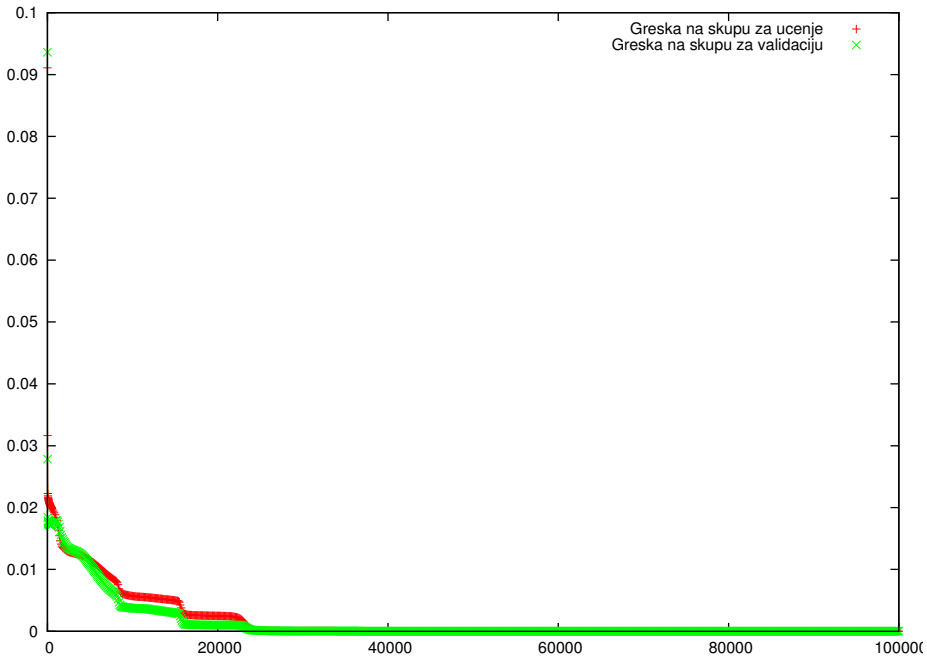
Pogreške na skupovima za učenje i validaciju prikazane su na slici u nastavku.



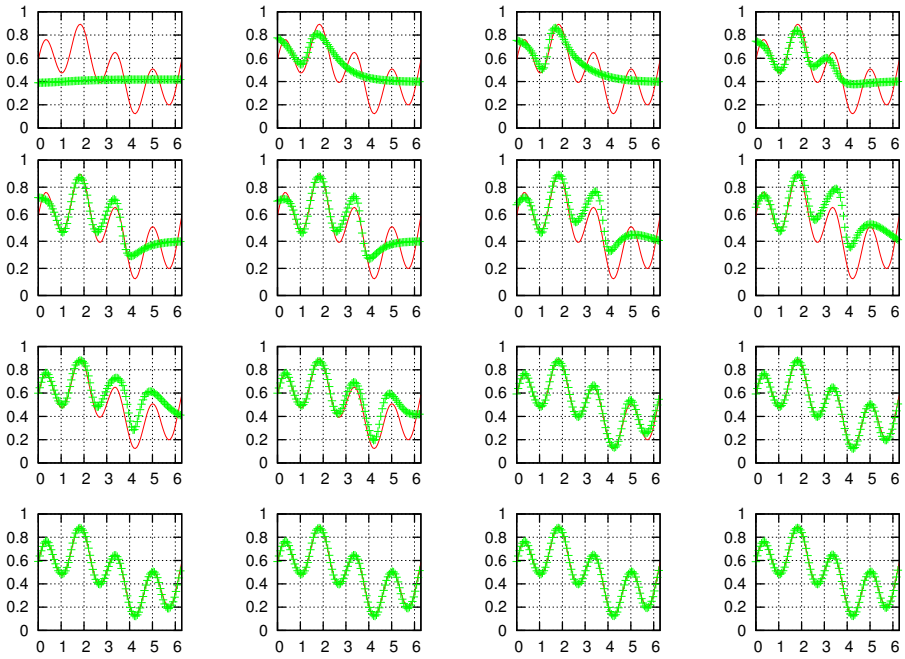

S obzirom da se radi o učenju klasičnim algoritmom *Backpropagation*, pogreška na skupu za učenje očekivano relativno monotono pada (razmislite zašto nemamo garanciju da će greška doista monotono padati?).

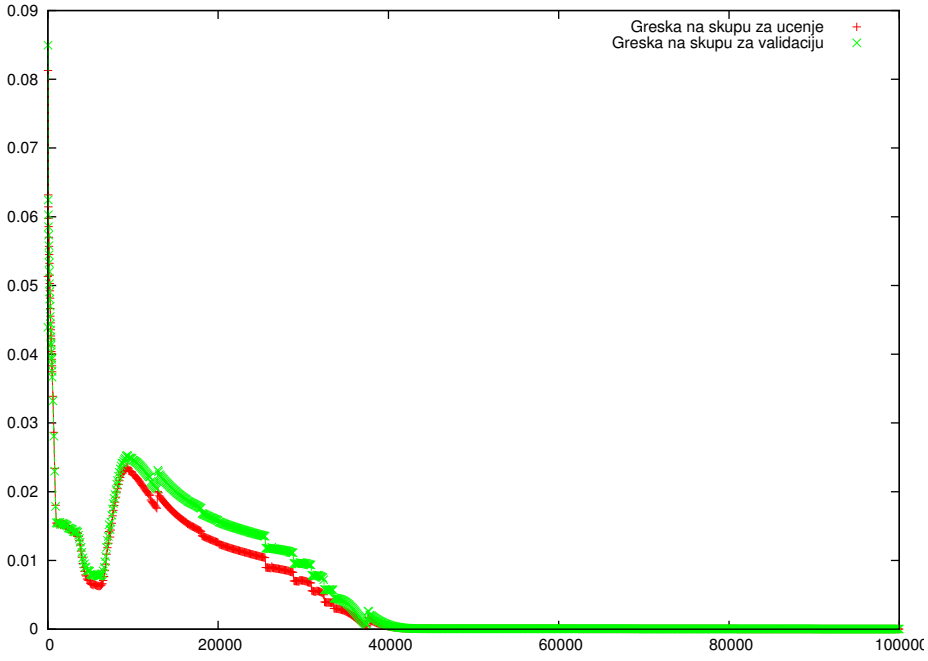
Postupak učenja stohastičkom izvedbom algoritma *Backpropagation* prikazan je na slikama u nastavku.






Konačno, postupak učenja stohastičkom izvedbom algoritma *Backpropagation* uz dodatno učenje svakog pojedinog uzorka dok greška na njemu nije dovoljno mala (ili dok se ne dođe do zadanog broja pokušaja) prikazan je na slikama u nastavku.





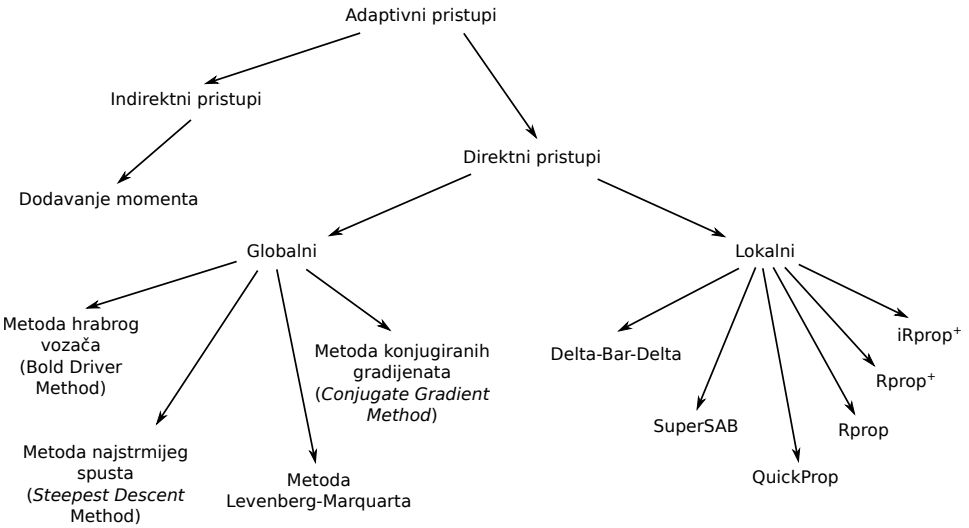

## 8.5 Pregled algoritama učenja

Najveći dio ovog poglavlja bio je fokusiran na, s povijesnog stajališta gledano, osnovni algoritam za učenje višeslojnih unaprijednih neuronskih mreža: al-

goritam *Backpropagation* [Rumelhart et al.(1986)Rumelhart, Hinton, and Williams]. U ovom podpoglavlju dat ćemo pregled sličnih metoda koje se koriste za ovaj zadatak i koje su nastale kako bi poboljšale efikasnost tog algoritma.

Algoritam *Backpropagation* razvijen je kao rješenje problema *Credit Assignment*: to je algoritam koji je omogućio učenje višeslojnih unaprijednih neuronskih mreža. Međutim, praksa je pokazala da je upravo to zadatak na kojem ovaj algoritam zakazuje: umjesto da smo sada u stanju učiti neuronske mreže s pet, šest ili više slojeva, na takvim arhitekturama mreža algoritam ne radi dobro. Osnovni razlog tome leži u činjenici da se gradijent koji se vraća iz kasnijeg sloja u raniji konzistentno i eksponencijalno smanjuje od izlaznog sloja gdje ima maksimalnu vrijednost. To nije pretjeran problem ako mreža ima samo jedan skriveni sloj, ali ako mreža ima primjerice 5 skrivenih slojeva, vrijednosti gradijenta koje dolaze do prvog sloja praktički su zanemarive. Stoga ovaj algoritam najčešće zakazuje upravo na ovakvim zadatcima.

Stoga je razvijen niz algoritama (od kojih ćemo ovdje spomenuti samo neke) koji na različite načine pokušavaju popraviti određena svojstva algoritma *Backpropagation*. Osnovna ideja svih tih algoritama jest na određen način utjecati na iznos stope učenja koja se koristi u svakom koraku. Jednostavna podjela ovakvih algoritama prikazana je na slici 8.1.




Slika 8.1: Pristupi podešavanja stope učenja.

*Indirektni pristupi* su pristupi koji stopu učenja ne mijenjaju direktno već nekim drugim mehanizmima postižu da se učenje ponaša kao da je stopa učenja modificirana. Primjer takvog pristupa je prethodno opisan postupak dodavanja inercije pravilu učenja. Stopa učenja $\eta$ se tijekom učenja ne

mijenja. Međutim, ako u dva uzastopna koraka gradijenti pogreške pokazuju u istom smjeru, prilikom druge korekcije težina ukupna korekcija bit će jednaka trenutno izračunatoj zbog stope učenja uvećanoj za isti takav iznos iz prethodnog koraka čime se algoritam ponaša kao da radi korekciju *efektivnom stopom* učenja koja je veća od $\eta$. S druge pak strane, ako smo u prethodnom koraku bili s jedne strane minimuma a u trenutnom smo s druge strane, gradijenti u te dvije točke bit će suprotni pa će ukupna korekcija biti napravljena za iznos koji je manji od onoga koji definira $\eta$.

*Direktne pristupe* možemo podijeliti na *globalne* i *lokalne*. *Globalni pristupi* su pristupi koji koriste jednu (globalnu) stopu učenja i nju direktno modificiraju. *Lokalni pristupi* su pristupi koji za svaku težinu uvode zasebnu stopu učenja i informacije o promjenama pojedine stope učenja tipično donose na temelju samo lokalno dostupnih informacija (primjerice, prethodne korekcije pripadne težine ili pak iznosa parcijalne derivacije funkcije pogreške s obzirom na pripadnu težinu).

### 8.5.1 Algoritam najstrmijeg spusta

Algoritam najstrmijeg spusta (engl. *Steepest Descent Algorithm*) je klasični algoritam matematičke optimizacije primijenjen na traženje optimalnih parametara neuronske mreže.

U svakom koraku algoritam ponavlja sljedeće korake.

1. Izračunaj vrijednost gradijenta funkcije pogreške $\nabla E(k)$.
2. Klasični *Backpropagation* težine bi ažurirao prema izrazu

$$\vec{w}(k+1) = \vec{w}(k) - \eta \cdot \nabla E(k)$$

radeći korekciju uz fiksni korak $\eta \cdot \nabla E(k)$. Algoritam najstrmijeg spusta umjesto toga zahtjeva da se pronađe $\eta^*$ koji minimizira funkciju $E(\vec{w}(k) - \eta^* \cdot \nabla E(k))$ i da se potom napravi ažuriranje:

$$\vec{w}(k+1) = \vec{w}(k) - \eta^* \cdot \nabla E(k).$$

Pojasnimo malo detaljnije drugi korak. U koraku $k$ težine su postavljene na vrijednost $\vec{w}(k)$. Uz te težine i sve uzorke za učenje možemo utvrditi iznos funkcije pogreške $E(k)$ kao i njezin gradijent $\nabla E(k)$ u točki $\vec{w}(k)$. Taj gradijent je vektor koji pokazuje u smjeru najvećeg porasta funkcije pogreške. Možemo stoga definirati vektor $\vec{d}(k) = -\nabla E(k)$ kao vektor koji pokazuje kako treba mijenjati težine gledano iz točke $\vec{w}(k)$ a da bi funkcija imala maksimalni pad vrijednost. Potom možemo definirati pomoćnu skalarnu funkciju:

$$\Psi(\eta) = E(\vec{w}(k) + \eta \cdot \vec{d}(k)).$$

S obzirom da su $\vec{w}(k)$ i $\vec{d}(k)$ fiksirani vektori, ono što zapravo radimo jest argument funkcije $E$ mijenjamo krenuvši od točke $\vec{w}(k)$ po pravcu koji je

određen vektorom $\vec{d}(k)$; koliko se pomićemo po tom fiksiranom pravcu određeno je parametrom $\eta$. Za pretpostaviti je da, kako $\eta$ raste, iznos funkcije $\Psi(\eta)$ će najprije padati (u okolici točke $\vec{w}(k)$ to nam garantira činjenica da kretanje radimo u smjeru negativnog gradijenta) a potom rasti. Želimo stoga pronaći onaj $\eta$ (označit ćemo ga s $\eta^*$) za koji je funkcije $\Psi(\eta^*)$ minimalna.

Jednostavan algoritam za pronalazak $\eta^*$ bismo mogli implementirati na sljedeći način. Krenemo s nekim malim $\eta$ izračunamo vrijednost funkcije $E(\vec{w}(k) + \eta \cdot \vec{d}(k))$ u tako dobivenoj točki. Ako je vrijednost funkcije manja, povećamo $\eta$ (primjerice, pomnožimo ga s 1.5) i izračunamo vrijednost funkcije $E$ u tako dobivenoj točki; postupak ponavljamo sve dok vrijednosti funkcije $E$ padaju. Prvi puta kada vrijednost funkcije $E$ poraste, kao $\eta^*$ uzmemo $\eta$ iz posljednjeg koraka prije porasta.

Treba uočiti da algoritam koji pronalazi optimalan $\eta$ zahtjeva izračun funkcije $E$ u različitim točkama u prostoru težina: to međutim znači da za svaki izračun funkcije $E$ moramo na ulaz mreže dovesti sve uzorke za učenje i izračunati ukupnu pogrešku što može biti skupo. Stoga je prikladnije posegnuti za nekim boljim algoritmom za pronalazak optimalnog $\eta$ koji traži što je moguće manje različitih točaka u kojima treba računati funkciju $E$ (čitatelj se upućuje na porodicu algoritama poznatu pod nazivom *line-search algorithms*).

Jednom kada je pronađen optimalan $\eta^*$, radi se korekcija težina. S obzirom da je $\eta^*$ optimalan, u smjeru vektora $\vec{d}(k)$ pogrešku više sigurno nije moguće smanjiti. Lako je stoga pokazati da će sljedeća korekcija težina koja će biti provedena u sljedećem koraku biti napravljena u smjeru vektora $\vec{d}(k+1)$ koji je okomit na prethodni vektor $\vec{d}(k)$.

### 8.5.2 Algoritam hrabrog vozača

Algoritam hrabrog vozača (engl. *Bold Driver Method*) ažuriranje težina radi prema uobičajenom pravilu:

$$w_{ij}(k+1) = w_{ij}(k) - \eta \cdot \frac{\partial E}{\partial w_{ij}}(k)$$

gdje je $\frac{\partial E}{\partial w_{ij}}(k)$ pravi iznos parcijalne derivacije dobiven temeljem svih raspoloživih uzoraka za učenje. Početnu stopu učenja $\eta_0$ definira korisnik (primjerice, $\eta_0 = 0.1$).

Prilikom učenja, u svakom se koraku $k$ promatra ponašanje funkcije pogreške $E$. Ako je nakon promjene težina iznos funkcije pogreške u sljedećem koraku smanjen, stopa učenja se povećava množenjem s konstantom $\rho > 1$. Ovo će osigurati da se pomicanje u podprostoru funkcije pogreške u kojem je vrijednost gradijenta mala ubrza. Ako se iznos funkcije pogreške u sljedećem koraku poveća, pretpostavka je da je stopa učenja bila prevelika. Stoga se poduzimaju sljedeće akcije.

1. Posljednja napravljena korekcija težina (koja je dovela do povećanja pogreške) se otkazuje. Težine se postavljaju na one koje su bile prije te korekcije (treba dakle pamtiti težinske faktore iz prethodnog koraka).

2. Stopa učenja se smanjuje množenjem s faktorom $0 < \sigma < 1$.

3. Radi se korekcija težina prema novoj stopi. Ako se pogreška smanji, nova stopa se prihvaća; ako pogreška i dalje raste, stopa se ponovno smanjuje množenjem s faktorom $\sigma$ i provjerava se funkcija pogreške – postupak se ponavlja sve dok funkcije pogreške ne počne padati. Predložene vrijednosti za koeficijente su $\rho = 1.1$ i $\sigma = 0.5$ [Battiti(1989)].

### 8.5.3 Algoritam Delta-Bar-Delta

Algoritam Delta-Bar-Delta [Jacobs(1988)] pripada u porodicu lokalnih algoritama. Svaka težina $w_{ij}$ ima svoju stopu učenja $\eta_{ij}$. Za svaku težinu algoritam promatra ponašanje parcijalne derivacije funkcije pogreške s obzirom na tu težinu. Prisjetimo se, parcijalna derivacija funkcije s obzirom na veku varijablu govori kako će se funkcija promijeniti ako se promatrana varijabla poveća. Stoga je ideja algoritma sljedeća: ako se u dva uzastopna koraka predznak parcijalne derivacije funkcije pogreške s obzirom na promatranu težinu ne mijenja, tada se stopa učenja pridružena toj težini može povećati kako bismo ubrzali promjenu težine; algoritam propisuje povećanje za fiksni iznos $\kappa^+ > 0$. Ako se pak predznak parcijalne derivacije funkcije pogreške s obzirom na promatranu težinu promijeni iz prethodnog koraka u trenutni korak, to znači da je težina upravo preskočila optimalnu vrijednost (prethodni korak je bio prevelik); stoga se stopa učenja eksponencijalno smanjuje množenjem trenutne stope s faktorom $0 < \kappa^- < 1$. Postupak je opisan u nastavku.

$$\eta_{ij}(k) = \begin{cases} \kappa^+ + \eta_{ij}(k-1), & \text{ako je } \frac{\partial E}{\partial w_{ij}}(k-1) \cdot \frac{\partial E}{\partial w_{ij}}(k) > 0, \\ \kappa^- \cdot \eta_{ij}(k-1), & \text{ako je } \frac{\partial E}{\partial w_{ij}}(k-1) \cdot \frac{\partial E}{\partial w_{ij}}(k) < 0, \\ \eta_{ij}(k-1), & \text{inače.} \end{cases}$$

Pri tome mora vrijediti $0 < \kappa^- < 1$. $\eta_{ij}(k)$ označava vrijednost stope učenja težine $w_{ij}$ u trenutku $k$; $\frac{\partial E}{\partial w_{ij}}(k-1)$ odnosno $\frac{\partial E}{\partial w_{ij}}(k)$ su vrijednosti parcijalne derivacije funkcije pogreške s obzirom na težinu $w_{ij}$ u trenutcima $k-1$ i $k$. Ažuriranje težina obavlja se prema izrazima:

$$\Delta w_{ij}(k) = -\eta_{ij}(k) \cdot \frac{\partial E}{\partial w_{ij}}(k) + \alpha \cdot \Delta w_{ij}(k-1),$$

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}(k).$$

### 8.5.4 Algoritam SuperSAB

Algoritam SuperSAB [Tollenaere(1990)] također pripada u porodicu lokalnih algoritama gdje svaka težina ima svoju stopu učenja. Algoritam je sličan algoritmu Delta-Bar-Delta ali uz dvije modifikacije: povećanje stope učenja je također eksponencijalno a u slučaju preskakanja optimuma poništava se prethodni korak ažuriranja težine. Postupak je opisan u nastavku.

$$\eta_{ij}(k) = \begin{cases} \kappa^+ \cdot \eta_{ij}(k-1), & \text{ako je } \frac{\partial E}{\partial w_{ij}}(k-1) \cdot \frac{\partial E}{\partial w_{ij}}(k) > 0, \\ \kappa^- \cdot \eta_{ij}(k-1), & \text{ako je } \frac{\partial E}{\partial w_{ij}}(k-1) \cdot \frac{\partial E}{\partial w_{ij}}(k) < 0, \\ \eta_{ij}(k-1), & \text{inače.} \end{cases}$$

Pri tome mora vrijediti $0 < \kappa^- < 1 < \kappa^+$. Primjerice, parametri se mogu postaviti na sljedeći način $\kappa^- = 0.5$, $\kappa^+ = 1.1$. U slučaju da je nastupio slučaj $\frac{\partial E}{\partial w_{ij}}(k-1) \cdot \frac{\partial E}{\partial w_{ij}}(k) < 0$ (postoji razlika u predznaku derivacije u prethodnom i trenutnom koraku), vrijednost težine $w_{ij}$ se restaurira na onu iz prethodnog koraka. U ostalim slučajevima radi se ažuriranje prema izrazima:

$$\Delta w_{ij}(k) = -\eta_{ij}(k) \cdot \frac{\partial E}{\partial w_{ij}}(k) + \alpha \cdot \Delta w_{ij}(k-1),$$

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}(k).$$

### 8.5.5 Algoritam Quickprop

Algoritam QuickProp [Fahlman(1988a), Fahlman(1988b)] još je jedan od algoritama koji pripadaju u porodicu lokalnih algoritama gdje svaka težina ima svoju stopu učenja. Međutim, po načinu rada bitno se razlikuje od prethodno opisanih algoritama. Pretpostavka ovog algoritma je da je u okolici težine $w_{ij}$ funkcija pogreške kvadratnog karaktera s obzirom na tu težinu (i to je pretpostavka koja vrijedi za svaku težinu u mreži). Algoritam stoga funkciju pogreške kada razmatra težinu $w_{ij}$ aproksimira Taylorovim razvojem drugog reda:

$$E(w_{ij} + \Delta w_{ij}) = E(w_{ij}) + \frac{\partial E(w_{ij})}{\partial w_{ij}} \cdot \Delta w_{ij} + \frac{1}{2} \frac{\partial^2 E(w_{ij})}{\partial w_{ij}^2} \cdot (\Delta w_{ij})^2$$

Shvatimo li ovdje $\Delta w_{ij}$ kao slobodnu varijablu, funkcija $E(w_{ij} + \Delta w_{ij})$ za različite vrijednosti varijable $\Delta w_{ij}$ poprima različite iznose. Mi tražimo onu vrijednost $\Delta w_{ij}$ za koju funkcija postaje minimalna: računamo stoga derivaciju te funkcije s obzirom na $\Delta w_{ij}$ i tu derivaciju izjednačavamo s 0 (uočite da su u prethodnom izrazu parcijalne derivacije i $E(w_{ij})$ konstante s obzirom na $\Delta w_{ij}$ pa se tako i tretiraju):

$$\frac{dE(w_{ij} + \Delta w_{ij})}{d(\Delta w_{ij})} = \frac{\partial E(w_{ij})}{\partial w_{ij}} + \frac{\partial^2 E(w_{ij})}{\partial w_{ij}^2} \cdot \Delta w_{ij} = 0$$

Slijedi da je traženi korak jednak:

$$\Delta w_{ij} = -\frac{\frac{\partial E(w_{ij})}{\partial w_{ij}}}{\frac{\partial^2 E(w_{ij})}{\partial w_{ij}^2}}.$$

Drugu parcijalnu derivaciju aproksimirat ćemo omjerom variranja prve parcijalne derivacije u trenutnom i prethodnom koraku i variranja težine u trenutnom i prethodnom koraku:

$$\frac{\partial^2 E(w_{ij})}{\partial w_{ij}^2} = \frac{\frac{\partial E(w_{ij})}{\partial w_{ij}}(k) - \frac{\partial E(w_{ij})}{\partial w_{ij}}(k-1)}{w_{ij}(k) - w_{ij}(k-1)} = \frac{\frac{\partial E(w_{ij})}{\partial w_{ij}}(k) - \frac{\partial E(w_{ij})}{\partial w_{ij}}(k-1)}{\Delta w_{ij}(k-1)}$$

Uvrštavanjem ovog izraza u prethodno dobiveni izraz za $\Delta w_{ij}$ dobivamo pravilo:

$$\Delta w_{ij}(k) = -\frac{\frac{\partial E(w_{ij})}{\partial w_{ij}}(k)}{\frac{\partial E(w_{ij})}{\partial w_{ij}}(k) - \frac{\partial E(w_{ij})}{\partial w_{ij}}(k-1)} \cdot \Delta w_{ij}(k-1).$$

Kako izraz u nazivniku može postati vrlo malen, promjena težine bi mogla ispasti vrlo velika pa se uvodi ograda $\gamma$ koja određuje koliko puta korekcija $\Delta w_{ij}(k)$ smije biti veća od korekcije $\Delta w_{ij}(k-1)$.

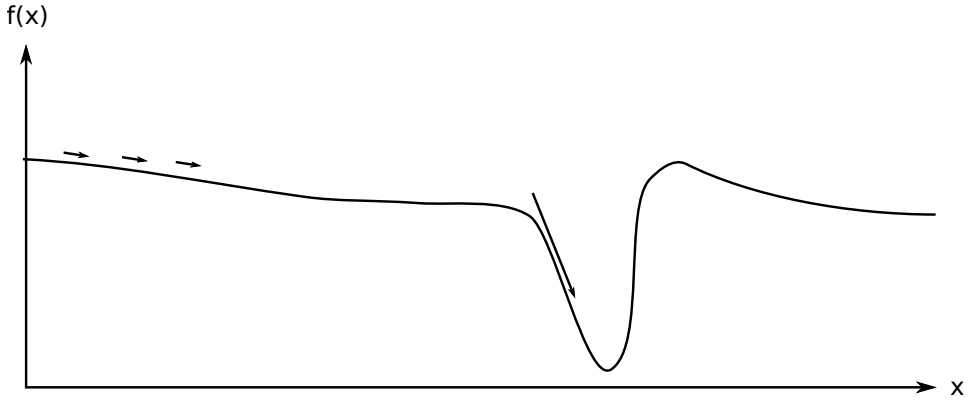
Konačni izraz koji se koristi korekciju još skalira stopom učenja $\eta$ pa imamo konačan izraz:

$$\Delta w_{ij}(k) = -\eta \cdot \frac{\frac{\partial E(w_{ij})}{\partial w_{ij}}(k)}{\frac{\partial E(w_{ij})}{\partial w_{ij}}(k) - \frac{\partial E(w_{ij})}{\partial w_{ij}}(k-1)} \cdot \Delta w_{ij}(k-1).$$

Kod prethodno opisanih algoritama koji dinamički ažuriraju stopu učenja (bilo globalno, bilo lokalno), može se pojaviti problem ilustriran na slici 8.2.

Pretpostavimo da je početna vrijednost težine na lijevoj strani grafa. Kako je u tom dijelu funkcija pogreške vrlo malog nagiba, algoritmi koji dinamički modificiraju stopu učenja na tom dijelu stopu će dosta povećati; međutim, koraci će ostati relativno mali jer je gradijent u tom području vrlo malen (prisjetite se: promjena težine proporcionalna je umnošku negativnog gradijenta i stope učenja). Pretpostavite sada da je pretraga pomaknula težinu na ulaz minimuma prikazanog u desnom dijelu slike. Sada imamo situaciju da je stopa učenja vrlo velika i gradijent je istovremeno postao vrlo velik (jer je oko prikazanog optimuma funkcija dosta strma): njihov umnožak bit će stoga vrlo velik i algoritam koji napravi promjenu težine s trenutne na sljedeću uz takvu korekciju može izletiti daleko izvan područja ovog optimuma.

Rješenje ovog problema nude algoritmi koji samostalno upravljaju kompletnim iznosom korekcije: korekcija se više neće računati kao umnožak nečega što algoritam drži pod kontrolom i nečega nad čim algoritam nema kontrolu. Takav algoritam je Rprop koji je razvijen u nekoliko inačica. Osnovna inačica opisana je u nastavku.




Slika 8.2: Jedan od problema adaptivnih pristupa.

### 8.5.6 Algoritam Rprop

Algoritam Rprop (engl. *Resiliant Backpropagation*) [Riedmiller and Braun(1993)] pripada u porodicu lokalnih algoritama gdje svaka težina ima svoju stopu učenja. Algoritam u potpunosti upravlja stopom učenja i parcijalne derivacije (odnosno gradijent) koristi samo za indikaciju ide li u dobrom smjeru ili ne – gleda se samo predznak dok je iznos nebitan.

Amplitudu korekcije težine $w_{ij}$ označit ćemo s $\Delta_{ij}$; to je broj koji je nenegativan. Amplituda korekcije iz koraka u korak se korigira na sljedeći način:

$$\Delta_{ij}(k) = \begin{cases} \kappa^+ \cdot \Delta_{ij}(k-1), & \text{ako je } \frac{\partial E}{\partial w_{ij}}(k-1) \cdot \frac{\partial E}{\partial w_{ij}}(k) > 0, \\ \kappa^- \cdot \Delta_{ij}(k-1), & \text{ako je } \frac{\partial E}{\partial w_{ij}}(k-1) \cdot \frac{\partial E}{\partial w_{ij}}(k) < 0, \\ \Delta_{ij}(k-1), & \text{inače.} \end{cases}$$

uz $0 < \kappa^- < 1 < \kappa^+$. Razmišljanje iza ovog pravila bi trebalo biti jasno: ako se preznak derivacije ne mijenja, amplitudu korekcije možemo povećati; ako se mijenja, amplitudu moramo smanjiti. Inicijalno, svi se $\Delta_{ij}$ postavljaju na vrijednost $\Delta_0$ (npr. 0.1). Uvodi se gornja ograda $\Delta_{max}$ koja ograničava maksimalni iznos koji $\Delta_{ij}(k)$ može poprimiti (npr. $\Delta_{max} = 50$) a moguće je uvesti i donju ogradu $\Delta_{min}$. Kao vrijednosti parametara mogu se koristiti primjerice $\kappa^- = 0.5$ i $\kappa^+ = 1.2$.

Jednom kada je utvrđena amplituda korekcije, određuje se i iznos korekcije prema pravilu:

$$\Delta w_{ij}(k) = \begin{cases} -\Delta_{ij}(k), & \text{ako je } \frac{\partial E}{\partial w_{ij}}(k) > 0, \\ \Delta_{ij}(k), & \text{ako je } \frac{\partial E}{\partial w_{ij}}(k) < 0, \\ 0, & \text{inače.} \end{cases}$$

Ovo pravilo je direktna implementacija algoritma koji sluša gradijent: ako

je u točki gradijent pozitivan, varijablu treba umanjiti; ako je negativan, varijablu treba povećati.

### 8.5.7 Algoritam Rprop$^+$

Algoritam Rprop$^+$ [Igel and Hüsken(2000)] dodatno je poboljšanje algoritma Rprop. Razlika u odnosu na algoritam je Rprop je ponašanje u slučaju kada se detektira promjena predznaka derivacije. U tom slučaju algoritam Rprop$^+$ poništava prethodnu korekciju dotične težine na način da postavlja $\Delta w_{ij}(k) = -\Delta w_{ij}(k-1)$ te zabranjuje promjenu $\Delta_{ij}$ u sljedećem koraku algoritma. Ovo se programski može postići tako da se za težinu $w_{ij}$ postavi $\Delta w_{ij}(k) = -\Delta w_{ij}(k-1)$ i da se $\frac{\partial E}{\partial w_{ij}}(k)$ postavi na nulu. Time će u sljedećem koraku pravilo za ažuriranje $\Delta_{ij}(k)$ završiti u *inače* grani koja ne mijenja vrijednost. Isto se, naravno, može postići uvođenjem po jedne zastavice za svaku težinu i upravljanja njome.

### 8.5.8 Algoritam iRprop$^+$

Konačno, posljednja modifikacija algoritma Rprop$^+$ je algoritam iRprop$^+$ [Igel and Hüsken(2000)] u nekim slučajevima dodatno popravlja svojstva algoritma, a radi se o kombiniranju lokalne informacije s globalnom informacijom. Konkretno, algoritam Rprop$^+$ uvodi poništavanje prethodne korekcije težine ako njezina parcijalna derivacija promijeni predznak, ne uzimajući u obzir što se događa s globalnim iznosom funkcije pogreške. Algoritam iRprop$^+$ zahtjeva da se uz iznos funkcije pogreške $E(k)$ pamti i iznos funkcije pogreške u prethodnom koraku $E(k-1)$ (što je samo jedan broj više koji treba zapamtiti). Potom se poništavanje prethodne korekcije (odnosno postavljanje $\Delta w_{ij}(k) = -\Delta w_{ij}(k-1)$ i $\frac{\partial E}{\partial w_{ij}}(k) = 0$) radi samo u slučaju da je došlo do promjene predznaka derivacije i da je istovremeno došlo i do povećanja funkcije pogreške, tj. da je $E(k) > E(k-1)$; u suprotnom se korekcija računa na uobičajeni način.

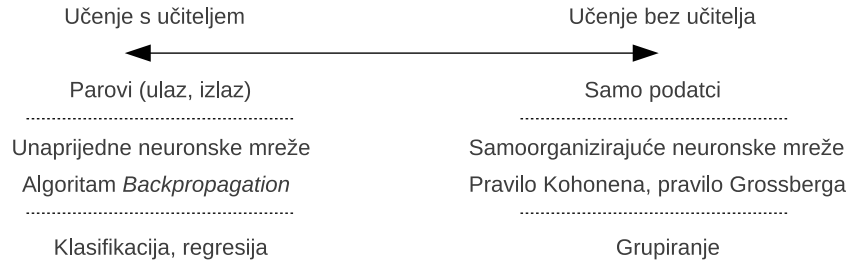
# Poglavlje 9

# Samoorganizirajuće neuronske mreže

Neuronske mreže koje smo do sada obradili bile su prikladne za *učenje s učiteljem*: temeljem poznatog ulaza i željenog izlaza korigirani su težinski faktori kako bi stvarni izlaz mreže bio bliži željenom izlazu. Samoorganizirajuće neuronske mreže predstavnik su mreža koje *uče bez učitelja*: takvim neuronskim mrežama daje se samo niz podataka; neuronska mreža potom korigira težinske faktore kako bi postigla neko željeno ponašanje. Ovo je ilustrirano na slici 9.1. Za razliku od mreža koje uče s učiteljem i koje obavljaju klasifikaciju podataka ili regresiju funkcije, mreže koje uče bez učitelja uobičajeno obavljaju zadatak grupiranja podataka te su u tom smislu jednake po funkciji kao različiti algoritmi za grupiranje poput $k$-srednjih vrijednosti i sličnih.

## 9.1 Natjecanje

Kod samoorganizirajućih neuronskih mreža centralni je pojam *natjecanje*. Kao što je poznato i iz biologije, priroda često poseže za ovim mehanizmom u svrhu optimizacije: prisjetimo se samo Darwinovih spoznaja o evoluciji




Slika 9.1: Vrste učenja, primjeri mreža i zadatci koje rješavaju

vrsta – evoluciji koja je direktno potaknuta borbom za razmnožavanje i preživljavanje u zadanom okolišu. Mehanizam natjecanja vodi do optimizacije direktno na lokalnoj razini, bez ikakve potrebe za globalnom kontrolom. Stoga je primjena ovog mehanizma vrlo interesantna, posebice u raspodijeljenim sustavima.

Kod samoorganizirajućih neuronskih mreža natjecanje će se odvijati direktno između procesnih elemenata (odnosno, neurona). Mehanizam natjecanja se pri tome može ostvariti na dva načina:

1. lateralnim vezama ili

2. formulacijom principa učenja.

Kod neuronskih mreža koje mehanizam natjecanja implementiraju lateralnim vezama, ideja je sljedeća. Svaki procesni element direktno dobiva ulazni podatak. Izlaz neurona računa se kao težinska suma dovedenog ulaznog podatka i izlaza svih preostalih neurona pri čemu su težine na vezama prema preostalim neuronima (tzv. lateralne veze) negativne. Time će svaki od neurona to više gušiti izlaz drugih neurona što je njegov vlastiti izlaz veći. Primjer ovakve arhitekture je mreža *Winner-takes-all* koju ćemo obraditi u nastavku.

Natjecanje formulacijom principa učenja uobičajeno se pak izvodi na algoritamskoj razini i najčešće podrazumijeva da u sustavu postoji globalni nadzornik koji promatra odziv mreže za svaku ulaznu pobudu i definira koji se procesni elementi smiju prilagoditi i u kolikoj mjeri kako bi njihov odziv na trenutnu pobudu postao još veći. Primjeri ovakvih mreža bit će mreže *INSTAR* te Kohonenov *SOM* koje ćemo također obraditi u nastavku.

### 9.1.1 Zahtjevi na samoorganizirajuće neuronske mreže

Da bismo neuronsku mrežu zvali samoorganizirajućom, uobičajeno se očekuje da takva mreža zadovoljava sljedeća četiri zahtjeva.

1. Težinski faktori u procesnim elementima trebaju biti predstavnici grupa ulaznih podataka. Time se kod ovakvih sustava uobičajeno odstupa od semantike težine kao broja s kojim se ulaz množi. Umjesto toga, procesni elementi će preko težina pamtiti primjerke ulaznih podataka ili predstavnike grupa ulaznih podataka. Time automatski svaki neuron postaje, u određenom smislu, predstavnik jednog od razreda ulaznih podataka.

2. Nema skrivenih neurona. Ulazni podatak dovodi se do svih neurona i svaki je neuron ujedno i izlazni. Vrijednost izlaza pri tome je, u određenom smislu, mjera sličnosti između uzorka dovedenog na ulaz mreže i predstavnika razreda pohranjenog u težinama neurona.

3. Koristi se strategija učenja koja se temelji na mehanizmu natjecanja (engl. *competitive learning strategy*) koja odabire neuron s najvećim izlazom.

4. Da bi čitav algoritam funkcionirao, postoje metode poticanja najvećeg izlaza, odnosno postupak prilagodbe težina provodi se upravo na način da neuron tim promjenama pokušava doći do stanja u kojem mu vrijednost izlaza raste, kako bi, potencijalno, što češće postajao pobjednik u natjecanju.

Prilikom implementacije mehanizma natjecanja koriste se dva tipa natjecanja: čvrsto natjecanje i meko natjecanje.

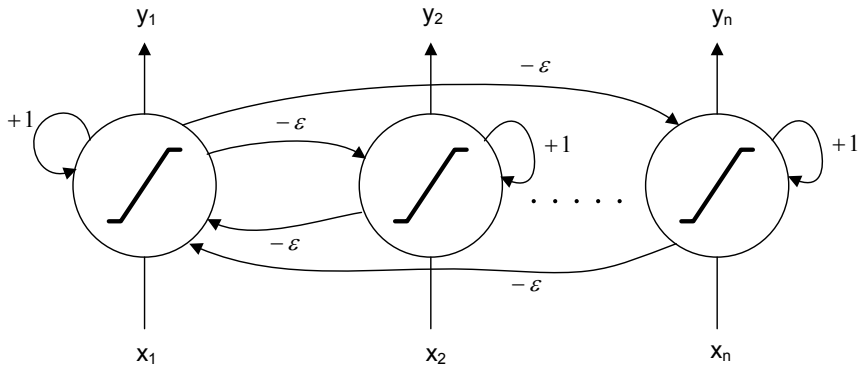
- *Čvrsto natjecanje* (engl. *hard competition*) je vrsta natjecanja kod koje uvijek postoji samo jedan pobjednik i samo je njemu dopušteno da uči, odnosno da prilagodi svoje težinske faktore kako bi dodatno povećao vrijednost svojeg izlaza.

- *Meko natjecanje* (engl. *soft competition*) je vrsta natjecanja kod koje uvijek postoji samo jedan pobjednik no učenje se dopušta njemu i u manjoj mjeri njegovoj bliskoj okolini. Na taj način pobjednik i njegova okolina u različim mjerama smiju prilagoditi svoje težinske faktore kako bi im se dodatno povećala vrijednost njihovog izlaza. Kako se točno definira okolina i na koji se način definira mjera u kojoj se smije raditi prilagodba ovisit će o svakoj pojedinoj vrsti samoorganizirajuće neuronske mreže koja koristi ovakav oblik natjecanja.

## 9.2 Mreža *Winner-Takes-All*

Prva od samoorganizirajućih neuronskih mreža koju ćemo obraditi je mreža *Winner-Takes-All* koja je prikazana na slici 9.2. To je mreža koja dobiva $n$-dimenzijski ulazni vektor podataka i ima upravo $n$ neurona. Svaki neuron pri tome ima prema sebi samome povratnu vezu s pozitivnom težinom (iznosa nešto većeg od +1, primjerice +1.05) te prema svim drugim neuronima ima veze ali s negativnim težinama. Iznosi tih težina uobičajeno su mali negativni brojevi (u prikazanom primjeru oznaka $-\epsilon$, gdje je $0 < \epsilon < \frac{1}{N}$).

Zadaća koju obavlja ovakva mreža jest pronalazak neurona na koji je doveden ulazni podatak s najvećim iznosom. Primjerice, zamislimo da imamo mrežu koja radi s 4-dimenzijskim vektorima. Takva će mreža za ulaz (0.1, 0.5, 0.3, 0.8) generirati izlaz (0.0, 0.0, 0.0, 1.0) dok će ulaz (0.1, 0.5, 0.3, 0.4) generirati izlaz (0.0, 1.0, 0.0, 0.0); dakle, samo će onaj neuron na koji je dovoden najveći podatak na svojem izlazu generirati vrijednost 1.0 dok će svi ostali neuroni na svojim izlazima generirati vrijednost 0.0. Ova mreža to će postići bez postojanja centralnog nadzornika. Evo kako.




Slika 9.2: Mreža *Winner-Takes-All*

### 9.2.1 Rad mreže

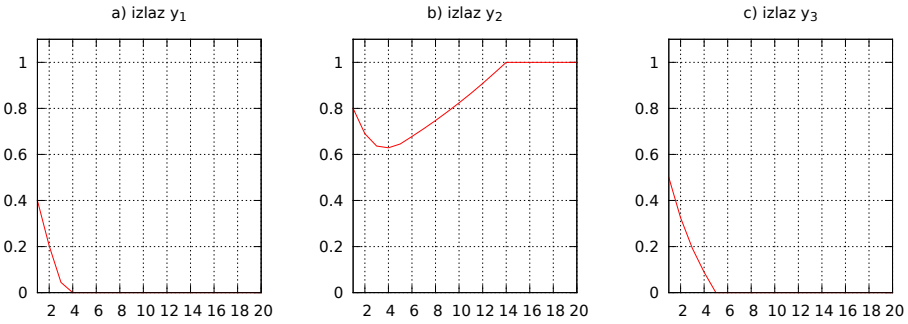
Izlaz $i$-tog neurona računa se prema izrazu:

$$y_i \leftarrow 1.05 \cdot y_i - \epsilon \cdot \sum_{j=1, j \neq i}^{N} y_j \quad \text{uz ogradu } 0 \leq y_i \leq 1.$$

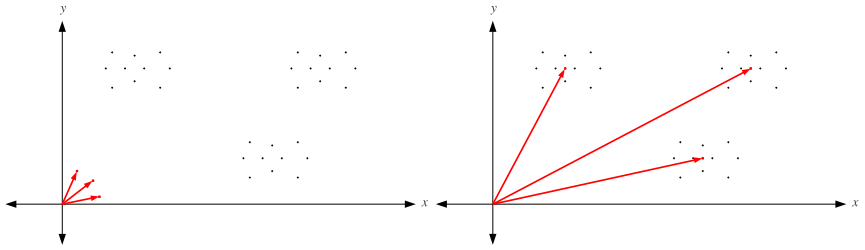
Međutim, mreža *Winner-Takes-All* s obzirom na postojanje povratnih veza ispoljava vremenski-ovisan odziv, pa se odziv (odnosno izlazi) računaju po koracima sve do postizanja stabilnog stanja. U trenutku $t = 0$ izlazi svih neurona postavljaju se na vrijednost 0 i na ulaz mreže se dovodi ulazni vektor. Svaki neuron po prvi puta računa svoj izlaz koji će biti upravo jednak dovedenom podatku s obzirom da su dotadašnji izlazi svih neurona jednaki 0.0 pa nema gušenja. Tako izračunate vrijednosti upisuju se na izlaze neurona i s ulaza se uklanja ulazni vektor podataka (tj. na sve se ulaze dalje postavlja vrijednost 0.0).

U sljedećem koraku, svaki od neurona opet računa svoj izlaz. Pozitivni doprinos izračunatom iznosu pri tome dobiva samo od svojeg vlastitog izlaza uvećanog za mali postotak dok izlazi svih ostalih neurona umanjuju izračunatu vrijednost jer se množe negativnom težinom $-\epsilon$. Nakon što su svi neuroni izračunali novu vrijednost, ta se vrijednost postavlja na izlaze neurona i postupak se ponavlja.

Primjer odziva troulazne mreže *Winner-Takes-All* uz ulaz (0.4, 0.8, 0.5), težinu pozitivne povratne veze +1.05 i vrijednost $\epsilon = \frac{1}{6}$ prikazan je na slici 9.3. U prvih nekoliko koraka jasno je vidljivo da svaki neuron svojim izlazom guši sve preostale neurone, pa izlazi svih neurona postaju sve manji i manji. Međutim, jednom kada izlaz pojedinog neurona postane dovoljno malen, gušenje postaje zanemarivo i pobjednički neuron vrlo brzo postiže maksimalnu vrijednost izlaza.




Slika 9.3: Odziv mreže *Winner-Takes-All* uz zadanu pobudu



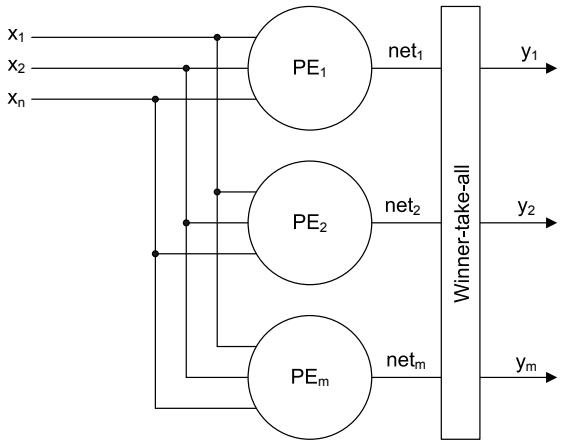

(a) Početna situacija (b) Željeno krajnje stanje

Slika 9.4: Zadatak kvantizacije vektora

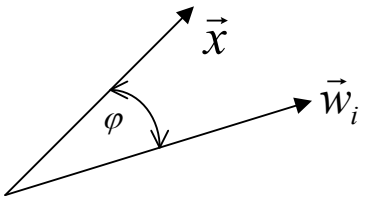
## 9.3 Mreža *INSTAR*

Mreža *INSTAR* primjer je mreže kod koje se neuroni specijaliziraju kako bi postali predstavnici razreda ulaznih uzoraka. Primjer je prikazan na slici 9.4. Pretpostavka je da će podatci koji se dovode kao ulaz neuronske mreže biti u prostoru grupirani u 3 grupe (razreda). U istom primjeru radi se s mrežom *INSTAR* koja ima tri neurona (svaki je prikazan crvenom bojom). U tom primjeru pretpostavka je da težine neurona predstavljaju pozicije reprezentanata svakog od razreda, i te su pozicije inicijalno odabrane sasvim slučajno (slika 9.4a). Kroz postupak učenja ove mreže konačno će stanje odgovarati onome prikazanome na slici 9.4b gdje će svaki od tri neurona postati reprezentant jednog od razreda.

Primjer općenite mreže *INSTAR* prikazan je na slici 9.5. Prikazana mreža radi s $n$-dimenzijskim uzorcima, odnosno ulaz je vektor $\vec{X} = (x_1, x_2, \ldots, x_n)$. Mreža na slici sastoji se od $m$ neurona. Svaki neuron $i$ ima $n$ težina kojima je povezan sa svakim od ulaza, i računa $net_i$ na uobičajen način: $\vec{W}_i^T \vec{X}$. Izlaz neurona $y_i$ trebao bi odgovarati mjeri sličnosti između pohranjenog težinskog vektora $\vec{W}_i$ i predočenog ulaznog vektora $\vec{X}$. Prisjetimo se da je skalarni produkt dvaju vektora zapravo jednak umošku njihovih normi i




Slika 9.5: Mreža *INSTAR*




Slika 9.6: Mjera bliskosti temeljena na kutu između vektora

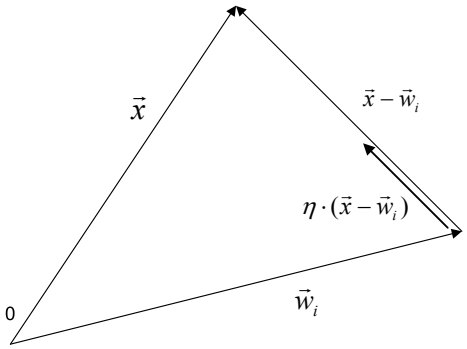
kosinusa kuta koji ti vektori zatvaraju:

$$\vec{W}_i \cdot \vec{X} = |\vec{W}_i| \cdot |\vec{X}| \cdot cos(\phi)$$

gdje je $\phi$ kut koji ti vektori zatvaraju. Ako pretpostavimo da su vektori $\vec{W}_i$ i $\vec{X}$ normirani, tada je njihov skalarni produkt upravo jednak kosinusu kuta. Kako je kosinus upravo funkcija koja za manje kuteve (sličnije vektore) daje veću vrijednost, skalarni produkt normiranih vektora može se koristiti kao tražena mjera sličnosti (slika 9.6).

Pretpostavimo sada da su ulazni podatci normirani. Postavljanjem podatka na ulaz mreže, taj se podatak dovodi do svakog neurona, i svaki neuron računa vrijednost skalarnog produkta. Ta će vrijednost, s obzirom na normiranost ulaza i težina biti broj iz intervala $[-1, 1]$. *net* svakog neurona potom se vodi na mrežu *Winner-Takes-All* koja generira konačni izlaz. Mreža *Winner-Takes-All* odabrat će kao pobjednika onaj neuron $j$ koji je spojen na neuron koji je generirao najveći iznos skalarnog produkta. Nakon što je utvrđen pobjednik, samo se njemu dopušta učenje: taj neuron obavlja ažuriranje težinskog vektora u skladu s Kohonenovim pravilom učenja:

$$\vec{W}_j \leftarrow \vec{W}_j + \eta(\vec{X} - \vec{W}_j)y_j$$




Slika 9.7: Mehanizam učenja kod mreže *INSTAR*

pri čemu je $y_j = 1$ jer se radi o pobjedniku. Raspisano po komponentama, imamo:

$$w_{ij}(k+1) \leftarrow w_{ij}(k) + \eta(x_i - w_{ij})y_j.$$

Ideja učenja na ovakav način prikazana je na slici 9.7: nakon učenja, dobiveni novi težinski vektor sličniji je ulaznom uzorku (zatvara s njime manji kut) i time će ovaj neuron sljedeći puta kada vidi taj uzorak na svojem izlazu dati još veću vrijednost izlaza. Nakon ažuriranja težinskog vektora, potrebno je još provesti njegovu normalizaciju.

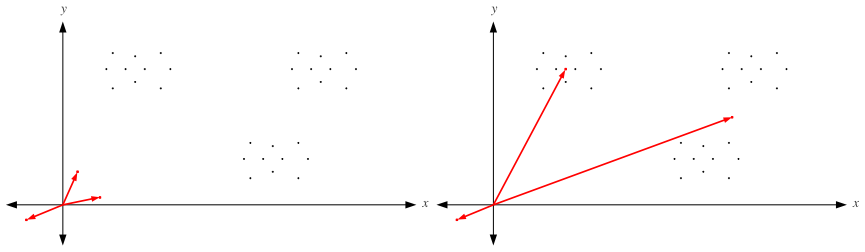
Umjesto uporabe skalarnog produkta, funkciju neurona u mreži *INSTAR* možemo redefinirati tako da svaki neuron umjesto skalarnog produkta uzorka i težinskog vektora računa korijen skalarnog produkta između razlike ovih dvaju vektora čime se na izlazu dobije euklidska udaljenost tih vektora:

$$d(\vec{X}, \vec{W}_i) = \sqrt{(\vec{X} - \vec{W}_i) \cdot (\vec{X} - \vec{W}_i)}.$$

U tom slučaju kao pobjednika treba izabrati neuron koji daje najmanji izlaz, i prilikom postupka učenja ulazni podatci kao niti težinski vektori ne trebaju biti normalizirani. Upravo je ovakva mreža pretpostavljena u početnom primjeru prikazanom na slici 9.4.

### 9.3.1 Problemi pri učenju mreže *INSTAR*

Kako se početne vrijednosti težinskih faktora biraju slučajno, moguće je da taj odabir bude nepovoljan. Scenarij je prikazan na slici 9.8. Neuron čije su težine takve da ga u primjeru na slici 9.8a smještaju u 3. kvadrant niti za jedan ulazni uzorak neće biti pobjednik natjecanja i nikada neće ažurirati svoje težine; takav se neuron nikada neće pomaknuti sa svoje pozicije. Takvi neuroni su *mrtvi* neuroni. Postupak učenja u tom će slučaju uzrokovati da jedan neuron bude određen kao predstavnik više od jednog razreda; situacija je prikazana na slici 9.8b.




(a) Početna situacija (b) Krajnje stanje

Slika 9.8: Nepovoljan odabir početnih težina

Kako bi se riješio ovaj problem, postoji nekoliko rješenja od kojih ćemo spomenuti dva. Najjednostavnije rješenje jest dopustiti svakom od neurona da obavi ažuriranje težinskih faktora. Pri tome se kod pobjednika (neuron $j$) korekcija množi s faktorom $\eta$ i obavlja prema uobičajenom izrazu:

$$\vec{W}_j \leftarrow \vec{W}_j + \eta(\vec{X} - \vec{W}_j)y_j$$

dok se kod svih ostalih neurona ($\forall i \neq j$) korekcija obavlja koristeći faktor $\lambda << \eta$:

$$\vec{W}_i \leftarrow \vec{W}_i + \lambda(\vec{X} - \vec{W}_i)y_i.$$

Ovakva modifikacija uzrokovat će da i mrtvi neuroni malo po malo izađu iz područja u kojem nikada nisu pobjednici i počnu povremeno pobjeđivati čime će im se omogućiti specijalizacija za određen dio ulaznog prostora.

Drugo predloženo rješenje zasniva se na uvođenju *savjesti*. Ideja je sljedeća. Svaki neuron procjenjuje postotak slučajeva u kojima je pobjednik. Što je taj iznos veći, kod neurona se javlja grižnja savjesti i zbog toga mu se udaljenosti do uzoraka koji se dovode na ulaz mreže prividno povećavaju te ih on doživljava većima no što stvarno jesu. S druge pak strane, neuroni koji malo pobjeđuju, žele pobjeđivati više pa im se udaljenosti do uzoraka koji se dovode na ulaz mreže prividno smanjuju. Ovo će dovesti do toga da nakon određenog vremena do tada mrtvi neuron počne pobjeđivati jer će njegova prividno umanjena udaljenost do ulaznog uzorka postati manja od prividno uvećanih udaljenosti do svih ostalih neurona, i takav će neuron početi korigirati svoje težinske faktore. Formalno, ovo možemo opisati na sljedeći način.

Neka je $d(\vec{W}_i, \vec{X})$ udaljenost između neurona $i$ i ulaznog uzorka $\vec{X}$. Ta se udaljenost mijenja prema izrazu:

$$d(\vec{W}_i, \vec{X}) \leftarrow d(\vec{W}_i, \vec{X}) - b_i$$

gdje je $b_i$ promjena duljine uslijed djelovanja savjesti za $i$-ti neuron. Iznos $b_i$ pri tome se računa temeljem procjene (oznaka $c_i$) postotka slučajeva u

kojima je $i$-ti neuron pobjednik. Naime, ako imamo $N$ neurona, očekivali bismo da u provednoj situaciji svaki neuron pobjeđuje u $\frac{1}{N}$ posto slučajeva, odnosno da je $c_i = \frac{1}{N}$. Stoga se duljina uslijed savjesti računa preko razlike očekivanog postotka pobjeda i procjene postotka pobjeda prema izrazu:

$$b_i = \gamma \cdot \left( \frac{1}{N} - c_i \right)$$

gdje $\gamma$ određuje maksimalni iznos promjene stvarne duljine uslijed savjesti pa time ovisi o skali prostora iz kojeg se podatci uzimaju. Primjerice, ako se uzorci po svim dimenzijama uzimaju iz raspona [-20,20], mogli bismo raditi s $\gamma = 10$ ili čak $\gamma = 20$.

Nakon što za sve neurone izračunamo prividne udaljenosti, na uobičajen se način dalje bira pobjednik i radi korekcija. Nakon što je odabran pobjednik, još je potrebno korigirati aktualnu procjenu postotka pobjeda za svaki neuron. Izraz koji se koristi je:

$$c_i \leftarrow c_i + \beta(y_i - c_i)$$

gdje je $\beta$ mala pozitivna konstanta (npr. 0.001). Uočimo: samo će neuron koji je pobjednik na svojem izlazu imati vrijednost $y_i = 1$ čime će iznos $y_i - c_i$ biti pozitivan i procjena postotka pobjeda će se malo povećati (to je razlog za razumno malu vrijednost za $\beta$). S druge pak strane, svi ostali neuroni koji nisu pobjednici imat će izlaz $y_i = 0$ čime će iznos $y_i - c_i$ biti negativan i procjena postotka pobjeda će se malo smanjiti.

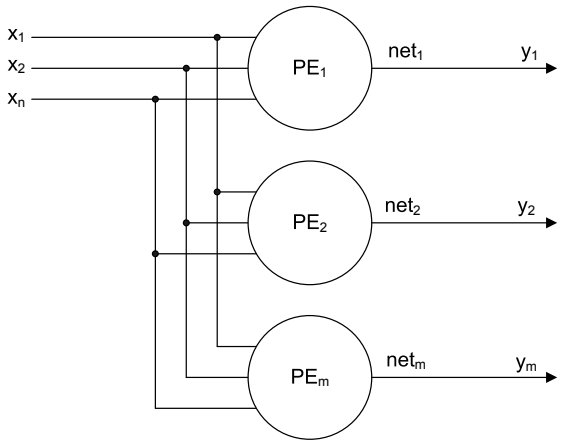
## 9.4 Mreža *OUTSTAR*

Mreža *OUTSTAR* radi upravo suprotno od mreže *INSTAR*. Dok je mreža *INSTAR* na ulaz mogla dobiti proizvoljan podatak a na izlazu je generirala na samo jednom neuronu vrijednosti 1 (svi ostali izlazi su bili 0), mreža *OUTSTAR* služi za repliciranje pohranjenih uzoraka: ulaz mora biti takav da ima na samo jednom mjestu vrijednost 1 a na izlaz se ne postavljaju ograničenja. Ideja mreže je sljedeća: mreža s $m$ neurona moći će zapamtiti $m$ proizvoljnih uzoraka (u određenom smislu vrsta memorije) i jedinicom na ulazu nekog neurona odredit ćemo koji će se od $m$ pohranjenih uzoraka pojaviti na izlazu mreže.

Postupak učenja ove mreže je takav da težinski faktori postaju slični zahtjevanim izlazima mreže a ne ulaznim uzorcima. Primjer ovakve mreže prikazan je na slici 9.9. Izlaz svakog neurona jednak je klasičnom skalarnom produktu ulaza i težina neurona, odnosno možemo pisati:

$$y_j = w_{1j}x_1 + w_{2j}x_2 + \ldots w_{nj}x_n.$$

Međutim, uzevši u obzir da se na ulazu mreže, dakle u $(x_1, \ldots, x_n)$ samo na jednom mjestu smije pojaviti vrijednost 1 (označimo poziciju tog ulaza s




Slika 9.9: Mreža *OUTSTAR*

$t$) dok svi ostali moraju biti 0, izlaz $j$-tog neurona će biti jednak:

$$y_j = w_{tj} x_t = w_{tj}.$$

Uz jedinicu na uzlazu $t$ izlaz mreže tada će dakle bili jednak ($w_{t1}$, $w_{t2}$, ..., $w_{tm}$). Označimo s ($y_1$, $y_2$, ..., $y_m$) željeni izlaz mreže za taj slučaj. Sada je evidentno što bi pravilo učenja trebalo postići: težina $w_{t1}$ bi trebala postati sličnija željenom izlazu $y_1$, $w_{t2}$ bi trebala postati sličnija željenom izlazu $y_2$, itd. Pravilo učenja koje se primjenjuje je tzv. *Grossbergovo učenje* i ono radi upravo opisano:

$$w_{ij} \leftarrow w_{ij} + \eta\,(y_j - w_{ij}) \cdot x_i$$

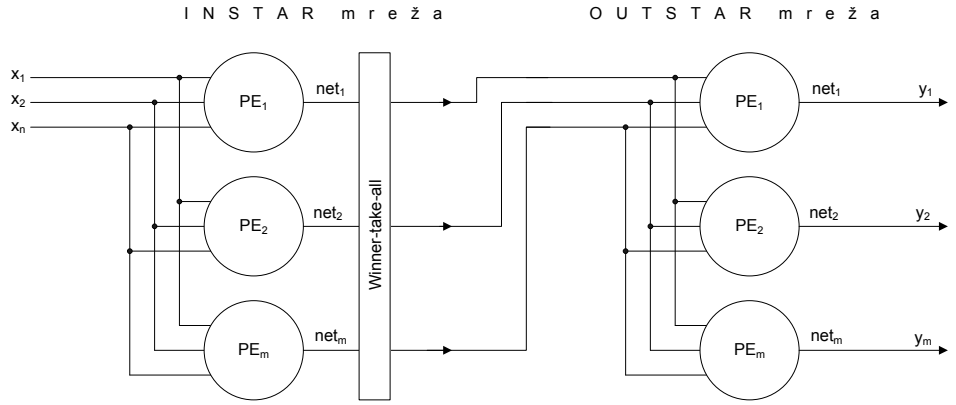
pri čemu je $j$ broj neurona, $i$ indeks ulaza a $\eta$ stopa učenja. Uočimo: za $i \neq t$ težine se ne mijenjaju jer su pripadni $x_i$ jednaki 0. Za $i = t$ slobodni indeks je samo $j$ koji ide po svim neuronima, i u svakom se promijeni samo težina kojom je neuron spojen s ulazom $t$ i to tako da postane sličnija zadanom izlazu tog neurona uz tu pobudu.

## 9.5 Mreža *Counterpropagation*

Najjednostavnija inačica mreže *Counterpropagation* je direktni spoj mreža *INSTAR* i *OUTSTAR*, kako je to prikazano na slici 9.10.

Ovakva mreža može imati vrlo interesantnu primjenu.

- Prvi dio mreže, odnosno mreža *INSTAR* iz šumovitih podataka prepoznaje o kojem se razredu radi i na izlazu daje jedinicu samo kod neurona koji je predstavnik tog razreda.
- Mreža *OUTSTAR* temeljem koda koji dobije na ulazu, na izlazu rekonstruira idealnog predstavnika tog razreda i postavlja bilo koji željeni podatak vezan uz taj razred.




Slika 9.10: Mreža *Counterpropagation*

## 9.6 Mreža *SOM*

Samoorganizirajuće mape, odnosno punim nazivom *Kohonenove samoorganizirajuće mape*, još su jedan primjer samoorganizirajućih neuronskih mreža. Za razliku od prethodno spomenutih samoorganizirajućih mreža, rad Kohonenove samoorganizirajuće mape zasniva se na uporabi *mekog natjecanja*. Temeljna ideja mekog natjecanja je omogućiti učenje pobjedniku te u manjoj mjeri i njegovom bliskom susjedstvu. Ovo je temeljna razlika naspram čvrstom natjecanju gdje uči samo i isključivo pobjednik.

Kako bismo mogli provoditi meko natjecanje, najprije je potrebno definirati pojam susjedstva, odnosno topologiju mreže. Najčešće topologije koje se koriste su:

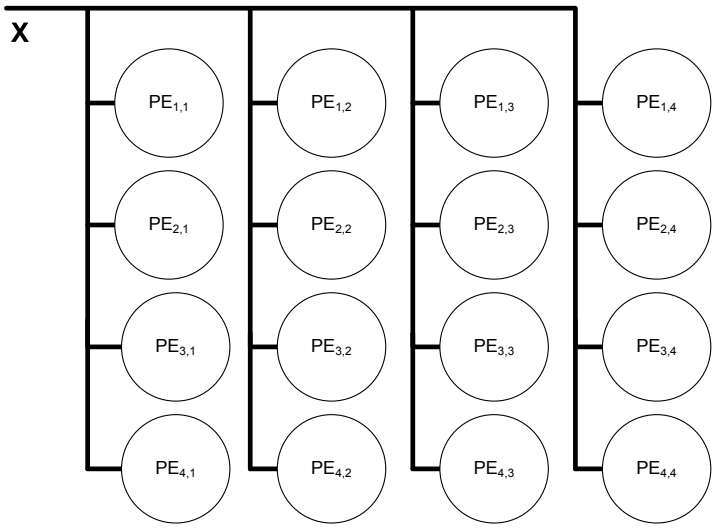
- jednodimenzijska, pri čemu su neuroni međusobno povezani u lanac pa svaki neuron osim krajnjih ima dva direktna susjeda (jednog lijevo i jednog desno) te

- dvodimenzijska rešetka, pri čemu svaki neuron osim rubnih ima četiri direktna susjeda (jednog iznad, jednog ispod, jednog lijevo i jednog desno).

Primjer dvodimenzijske rešetke prikazan je na slici 9.11 pri čemu je mreža dimenzija $4 \times 4$.

Kohonenove samoorganizirajuće mape iskazuju sljedeća svojstva:

1. očuvanje distribucije primjera za učenje – pretpostavimo da primjeri za učenje dolaze iz dva razreda te da primjera koji pripadaju prvom razredu ima dvostruko više; tada će i neurona koji predstavljaju reprezentante prvog razreda biti dvostruko više u odnosu na broj neurona koji će postati reprezentanti drugog razreda te




Slika 9.11: Mreža *SOM*

2. topološko preslikavanje – neuroni koji su u samoorganizirajućoj mapi međusobno blizu bit će predstavnici primjera koji su u ulaznom prostoru međusobno blizu (obrnuto ne mora vrijediti; moguće je da postoje primjeri koji su u ulaznom prostoru međusobno blizu a da neuroni koji ih predstavljaju u mapi budu međusobno daleko).

Susjedstvo se kod samoorganizirajuće mape uobičajeno definira funkcijom $\Lambda_{i,j}$ koja poprima vrijednost 1 ako su neuroni $i$ i $j$ "maksimalno" susjedni te pada prema nuli što su neuroni $i$ i $j$ u manjoj mjeri susjedi. Uobičajeno je funkciju susjednosti temeljiti na udaljenosti $d_{i,j}$ neurona $i$ i $j$ u mapi kojoj pripadaju. Primjerice, ako su neuroni razmješteni po dvodimenzijskoj rešetci, tada svakom neuronu $k$ možemo pridružiti njegove $(x_k, y_k)$ koordinate u toj rešetci. Udaljenost između dva neurona tada se najčešće definira kao:

- Manhattan udaljenost, odnosno $d_{i,j} = |x_i - x_j| + |y_i - y_j|$ ili pak
- Euklidsku udaljenost, odnosno $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ što je češći slučaj.

Naglasimo još jednom kako se udaljenost između neurona za potrebe izračuna funkcije $d_{i,j}$ ne promatra u ulaznom prostoru odnosno ne računa kao udaljenost između primjera koje ti neuroni predstavljaju – udaljenost se računa u prostoru same mape. Funkcija susjednosti uz ovako definiranu udaljenost uobičajeno se računa prema izrazu:

$$\Lambda_{i,j}(n) = e^{\frac{-d_{i,j}^2}{2\sigma^2(n)}}$$

gdje je s $n$ označena iteracija odnosno korak algoritma. Širina susjedstva određena je parametrom $\sigma(n)$ koji je funkcija od $n$ – prilikom učenja samoorganizirajuće mape taj će se parametar mijenjati ovisno o iteraciji čime će se i vrijednost funkcije susjednosti mijenjati.

Pravilo ažuriranja težina koje se koristi kod samoorganizirajuće mape glasi:

$$\vec{w}_i \leftarrow \vec{w}_i + \eta(n) \cdot \Lambda_{i,i^*}(n) \cdot (\vec{X} - \vec{w}_i)$$

gdje je $\vec{w}_i$ vektor težina $i$-tog neurona, $\eta(n)$ iznos stope učenja u $n$-toj iteraciji, $\vec{X}$ ulazni primjer za učenje, $i^*$ indeks neurona pobjednika (dakle neurona koji je najbliži ulaznom uzorku za učenje) te $\Lambda_{i,i^*}(n)$ iznos vrijednosti funkcije susjednosti između neurona $i$ i pobjedničkog neurona $i^*$ u $n$-toj iteraciji.

Algoritam učenja Kohonenove samoorganizirajuće mape sastoji se od dvije faze.

1. faza odnosno *inicijalna faza* osigurava grubo podešavanje samoorganizirajuće mape – služi za stvaranje grubog razmještaja neurona po prostoru primjera za učenje. U toj fazi kreće se od relativno velikog susjedstva (primjerice pola širine mape ili više) te se susjedstvo postupno smanjuje prema malom susjedstvu od sveka nekoliko (ili čak jednog) neurona. Stopa učenja također treba krenuti s relativno velikim iznosom i potom se postupno smanjivati (primjerice od 0.5 do 0.01). Ako za ovu fazu predvidimo izvođenje $N_0$ iteracija, za podeđavanje stope učenja može se koristiti izraz:

   $$\eta(n) = \eta_0 \left(1 - \frac{n}{N_0 + K}\right)$$

   gdje je $n$ iteracija, $\eta_0$ početni iznos stope učenja, $N_0$ broj iteracija koje će se obaviti tijekom inicijalne faze te $K$ konstanta koja omogućava podešavanje konačnog iznosa stope učenja kada $n$ postane jednak $N_0$. Susjedstvo se također smanjuje u skladu s izrazom:

   $$\sigma(n) = \sigma_0 \left(1 - \frac{n}{N_0}\right).$$

2. faza odnosno *faza finog podešavanja* koja se izvodi u $N_1$ koraka pri čemu je $N_1 >> N_0$ (tipično 10 do 100 puta ili čak i više). Tijekom ove faze susjedstvo se drži na vrlo malom (ili se čak ograničava na samog pobjednika) a stopa učenja je mala i još se smanjuje.

# Dio III

# Evolucijsko računanje

# Poglavlje 10

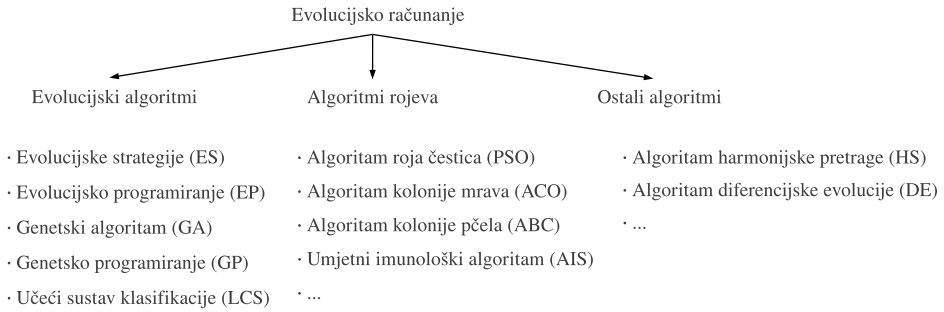
# Evolucijsko računanje

Što je to *evolucijsko računanje*? Na ovo naizgled jednostavno pitanje u današnje doba više nije baš jednostavno odgovoriti. Jedan kompromisni odgovor mogao bi glasiti ovako: evolucijsko računanje je područje računarske znanosti koje razmatra algoritme koji simuliraju evolucijski razvoj vrsta, život i ponašanje jedinke u društvu ili pak simuliraju različite aspekte umjetnog života. U pravilu, svi ovi algoritmi su populacijski algoritmi, iako se ponekad zna dogoditi i da populacija spadne na samo jednu jedinku. Također, najveći dio algoritama evolucijskog računanja nastao je modeliranjem procesa koji su opaženi u prirodi, tj. imaju jaku biološku motivaciju; dakako, niti ovo nije pravilo – postoje algoritmi koji nemaju analogiju s procesima u prirodi.

Zajednička obilježja većine algoritama ovog područja je prikladnost za rješavanje optimizacijskih problema. Uporabom prikladnih algoritama evolucijskog računanja moguće je napasti vrlo teške optimizacijske probleme koje je u praksi nemoguće rješiti iscrpnom pretragom. Pri tome su različiti algoritmi u različitoj mjeri prikladni za pojedine vrste optimizacijskih problema: neki je moguće direktno primjeniti samo na *kombinatorne optimizacijske probleme*, neke na *probleme optimizacije nad kontinuiranim domenama* a neke na obje vrste problema.

Razvoj ovog područja započeo je šezdesetih godina prošlog stoljeća. Područje evolucijskog računanja kako ga znamo danas dijeli se u tri velike skupine, kako to prikazuje slika 10.1. Područja su redom: evolucijski algoritmi, algoritmi rojeva te ostali algoritmi. U okviru ovog pregleda pobliže ćemo opisati samo granu evolucijskih algoritama.

Evolucijske strategije [Rechenberg(1971), Schwefel(1974)] – dominantni operator za generiranje potomstva je operator mutacije. Najjednostavnija izvedba, tzv. (1+1)-ES uvijek radi samo s jednim roditeljem; njegovom mutacijom nastaje točno jedno dijete. Ako je dijete lošije od roditelja, odbacuje se a u suprotnom dijete postaje roditelj za sljedeću generaciju (a stari roditelj se odbacuje). Kod druge varijante evolucijskih strategija, tzv. (1+$\lambda$)-ES, iz jednog se roditelja operatorom mutacije stvara $\lambda$ djece koja se natječu s




Slika 10.1: Pregled područja evolucijskog računanja

roditeljem za preživljavanje. Najbolja jedinka (bilo roditelj, bilo neko od stvorene djece) prenosi se u sljedeću generaciju i postaje roditelj. Kod evolucijske strategije (1,$\lambda$)-ES iz jednog roditelja operatorom mutacije stvara se $\lambda$ djece i najbolje dijete (čak ako je i lošije od trenutnog roditelja) prenosi se u sljedeću generaciju gdje postaje roditelj (stari se roditelj uvijek odbacuje). Osim navedenih, razvijen je još niz drugih podvrsta.

Evolucijsko programiranje [Fogel et al.(1966)Fogel, Owens, and Walsh] – dominantni operator za generiranje potomstva je operator mutacije; izvorno se koristila reprezentacija rješenja u obliku fiksnog stabla za koje su se tražili samo optimalni parametri; u današnje doba stablo više ne mora biti fiksne strukture.

Genetski algoritam [Holland(1975)] u određenom je smislu poopćenje evolucijskih strategija. Naime, iako postoji više izvedbi genetskih algoritama, svima je zajedničko da rade s populacijama roditelja, koriste različite operatore selekcije za odabir jedinki koje će generirati djecu te osim operatora mutacije koriste i operator križanja kojim se genetski materijal roditelja međusobno kombinira kako bi nastalo djete koje dio genetskog materijala dobiva od svakog roditelja.

Genetsko programiranje [Koza(1990),Koza(1992),Koza(1994),Koza et al.(1999)Koza, Andre, Bennett, and Keane] vrlo je slično genetskom algoritmu i predstavlja poopćenje evolucijskog programiranja. Genetsko programiranje za prikaz rješenja uobičajeno koristi stabla koja predstavljaju matematičke izraze (ako se evoluiraju funkcije) ili pak računalne programe. Genetsko programiranje pri tome koristi i operator mutacije i operator križanja.

Učeći sustavi klasifikacije [Holland(1976)] predstavljaju kombinaciju genetskih algoritama i podržanog učenja. Ideja je izgraditi sustav koji će za svaki ulaz odabrati tj. izvršiti onu akciju za koju će primiti maksimalnu nagradu. Zbog toga su jedinke s kojima rade ovakvi algoritmi upravo pravila. Danas su se uvriježile dvije vrste ovih sustava: sustavi tipa Pittsburgh i sustavi tipa Michigan. Kod sustava tipa Pittsburgh, svaka je jedinka genetskog algoritma skup pravila; operator križanja tada kombinira dva skupa pravila

i stvara novi rezultantni skup pravila koji postaje novo dijete. Kod sustava tipa Michigan u populaciji postoji samo jedan skup pravila; zadaća genetskog algoritma u takvom okruženju je odabrati podskup pravila na način da se dobije najbolja klasifikacija.

Zajednička svojstva algoritama zasnovanih na evolucijskom računanju navedena su u nastavku.

- *Algoritmi su zasnovani na populaciji rješenja.* Uobičajeno je da se algoritam inicijalizira tako da se slučajno generira početna populacija rješenja. U početnu populaciju se mogu usaditi i rješenja dobivena nekim drugim algoritmom kako bi se ubrzala konvergencija algoritma. Tijekom evolucijskog procesa broj jedinki u populaciji (veličina populacije) može se mijenjati, ali se najčešće ne mijenja, odnosno veličina populacije je konstantna.

- *Jedinke su međusobno usporedive prema dobroti.* Jedinka predstavlja točku u prostoru rješenja, odnosno predstavlja moguće rješenje problema. Primjerice, tražimo li maksimum funkcije $f(x)$, gdje je $x \in [dg, gg]$ tada jedinku predstavlja bilo koji broj između donje ($dg$) i gornje ranice ($gg$) i za svaka dva broja $x_1, x_2 \in [dg, gg]$ se može odrediti je li $f(x_1)$ manji, veći ili jednak $f(x_2)$. Treba napomenuti da ovo svojstvo imaju algoritmi evolucijskog računanja primijenjeni na probleme jednokriterijske optimizacije; kod višekriterijskih optimizacijskih problema ovo svojstvo općenito ne vrijedi.

- *Populacija jedinki se s vremenom mijenja, evoluira jer se provodi postupak selekcije jedinki.* U procesu selekcije bolje jedinke imaju veću vjerojatnost preživljavanja od onih lošijih. Pogrešno bi bilo reći da se selkcijom eliminiraju loše jedinke!

- *Svojstva jedinki se prenose s roditelja na djecu.* U stvaranju nove jedinke (novog rješenja) sudjeluju izabrane (u pravilu bolje) jedinke iz prošle populacije. Reći ćemo da nad jedinkama djeluju operatori u stvaranju nove populacije. Prema tome, prostor rješenja se usmjereno pretražuje na temelju već postignutih rješenja kao primjerice u slučaju operatora križanja kod genetskih algoritama.

- *Prostor rješenja se pretražuje osim usmjerenim pretraživanjem i slučajnim procesom.* Evolucijski proces u velikoj mjeri stohastičke prirode. Jedinke se slučajno odabiru, samo neke s većom, a neke s manjom vjerojatnošću. Tako i promjene koje unose operatori mogu biti slučajne kao u slučaju mutacije kod genetskog algoritma.

U ovom poglavlju nije namjera opisati i razraditi sve algorime zasnovane na evolucijskom računanju, već je cilj dati upute kako poboljšati njihovo ponašanje. Želimo odabrati takav algoritam da on s određenim parametrima uz

*prihvatljivo* trajanje izvođenja postigne *dovoljno dobro* rješenje. Postupak podešavanja ili optimiranja algoritma zasnovanog na evolucijskom računanju bit će opisan na primjeru genetskog algoritma jer genetski algoritam, osim što je u povijesti bio prvi predstavljen, je najrasprostranjeniji, odnosno primjenjuje se na širokom spektru optimizacijskih problema.

## 10.1 Osvrt na genetske algoritme

Genetski algoritam je heuristička metoda slučajnog i usmjerenog pretraživanja prostora rješenja koja imitira prirodni evolucijski proces. Genetski algoritam, kao uostalom i svi algoritmi zasnovani na evolucijskom računanju, služi za rješavanje težih optimizacijskih problema, za koje ne postoji egzaktna matematička metoda rješavanja ili su NP-teški pa se za veći broj nepoznanica ne mogu riješiti u zadanom vremenu.

Posao koji obavlja genetski algoritam može se opisati jednom rečenicom: nakon što se generira početna populacija, genetski algoritam ciklički obavlja selekciju boljih jedinki koje potom sudjeluju u reprodukciji, sve dok nije zadovoljen uvjet završetka evolucijskog procesa (slika 10.2). Reprodukcija stvara nove jedinke uz pomoć genetskih operatora križanja i mutacije. Križanje prenosi svojstva roditelja na djecu, a mutacija slučajno mijenja svojstva jedinke. Genetski algoritam ne specificira kako se križanjem prenose svojstva roditelja na djecu, kako se slučajno mijenjaju svojstva jedinki, kako se selektiraju bolje jedinke za reprodukciju, niti kako se generira početna populacija. Upravo je ta sloboda u odabiru vrste križanja, mutacije, selekcije i inicijalizacije otežavajuća okolnost u procesu izgradnje genetskog algoritma za rješavanje specifičnog optimizacijskog problema. Naime, pokazalo se da ne postoji takav skup genetskih operatora za koji bi GA, ako se primijeni za rješavanje proizvoljnog skupa optimizacijskih problema, davao superiorne rezultate u odnosu na GA s nekim drugim operatorima [Fogel(1999),Macready and Wolpert(1996),Wolpert and Macready(1996),Wolpert(1996)].

```
1 Genetski_algoritam(){
2     generiraj_pocetnu_populaciju();
3     dok (nije_zadovoljen_uvjet_zavrsetka_evolucijskog_procesa){
4         selektiraj_bolje_jedinke_za_reprodukciju();
5         reprodukcijom_generiraj_novu_populaciju();
6     }
7 }
```

Slika 10.2: Genetski algoritam

Genetski algoritam obavlja genetske operatore nad populacijom jedinki $Ji \in \mathbf{J}$, gdje je $\mathbf{J}$ skup svih mogućih rješenja. Primjerice, za binarni prikaz, gdje se jedinke sastoje od $b$ binarnih znamenaka, kardinalni broj skupa $\mathbf{J}$ je

broj svih mogućih rješenja i iznosi $\#\mathbf{J} = 2^b$. Jedinke se nazivaju još i potencijalna rješenja, jer genetski algoritam manipulirajući genetskim materijalom jedinki, postiže rješenje [Blickle and Thiele(1995)]. U računalnom žargonu, jedinka je nekakva struktura podataka koja se sastoji od kromosoma i vrijednosti funkcije cilja. Jedinka ili kromosom se sastoji od gena koji opisuju svojstva jedinke.

Populacija $P = \{J_1, J_2, \ldots, J_i, \ldots, J_N\} \in \mathbf{J^N}$ se sastoji od $N$ jedinki. Početna populacija se najčešće generira potpuno slučajno, ali može biti i uniformna (sve jedinke su jednake) ili se može sastojati od usađenih rješenja dobivenih nekim drugim optimizacijskim postupkom [Corcoran and Wainwright(1992)]. Početna populacija *P(0)* se s vremenom (iz generaciju u generaciju) mijenja i u trenutku (generaciji) $t$ ima oznaku *P(t)*. Obično je uvjet završetka evolucijskog procesa unaprijed zadan broj iteracija $I$.

Selekcijom se odabiru bolje jedinke za reprodukciju. Kvaliteta jedinki mjeri se s pomoću funkcije cilja $f : \mathbf{J} \longrightarrow \mathbb{R}$. Neki postupci selekcije zahtijevaju da funkcija cilja ne može biti negativna, a niti je poželjno da funkcija cilja poprima samo velike, približno jednake vrijednosti. Stoga se obično u svakom koraku obavlja translacija funkcije cilja, tj. u svakoj iteraciji oduzima se najmanja vrijednost funkcije cilja u cijeloj populaciji. Rezultat je funkcija dobrote $f : \mathbf{J} \to \mathbb{R}^+$ koja se računa prema izrazu:

$$d = f - f_{min}\,(t) \tag{10.1}$$

gdje je $f_{min}\,(t)$ najmanja vrijednost funkcije cilja u generaciji $t$. Jedinka $J_i$ je bolja od jedinke $J_k$ ako je $d_i > d_k$. Postupci skaliranja i translacije funkcije cilja navedeni su u [Golub(2004a)].

Genetski operatori: selekcija, križanje i mutacija u svakoj iteraciji modificiraju populaciju $P$. Stoga se može reći da GA pretražuje prostor rješenja mijenjajući populaciju u svakoj iteraciji uz pomoć nekakve složene funkcije $g : \mathbf{J^N} \to \mathbf{J^N}$. Mutacija i križanje pretražuju prostor rješenja, a selekcija koristi samo informaciju unutar populacije, odnosno ne traži nova rješenja, već favorizira bolje jedinke. Prostor rješenja $\mathbf{J^N}$ naziva se još i prostorom pretraživanja.

Danas još nije poznat postupak uz pomoć kojeg bi se za zadani problem automatski odredili genetski operatori i prikaz rješenja. Postupak izgradnje genetskog algoritma temelji se na vlastitom i tuđem iskustvu (obično iz literature) te na temelju eksperimentiranja i podešavanjprvom dijelu ovog radaa raznih kombinacija genetskih operatora (većinom se to obavlja metodom pokušaja i pogrešaka).

Postupak izgradnje genetskog algoritma može se u grubo razložiti u sljedećih nekoliko koraka:

- stvarni problem postaviti kao optimizacijski problem (primjerice, problem rasporeda se svodi na minimizaciju broja pogrešaka u rasporedu);

- odrediti prikaz i funkciju dobrote;
- odrediti pojedine genetske operatore;
- eksperimentalno podesiti parametre na jednostavnijem problemu za koji je poznato rješenje;
- eksperimentalno obaviti fino podešavanje parametara na stvarnom problemu;
- ukoliko postupak optimiranja stvarnog problema traje predugo, odabrati najpogodniji paralelni model GA i eksperimentalno podesiti dodatne parametre.

Detaljniji pregled svih genetskih operatora nalazi se u [Golub(2004b), Golub(2004a), Čupić(2012)].

## 10.2 Primjer generacijskog genetskog algoritma

U ovom poglavlju razradit ćemo konkretan primjer genetskog algoritma čiji će zadatak biti pronaći $\vec{x}^* \in \mathbb{R}^N$ koji maksimizira funkciju

$$f(\vec{x}) = \sum_{i=1}^{N} e^{\frac{-(x_i - i)^2}{0.1}},$$

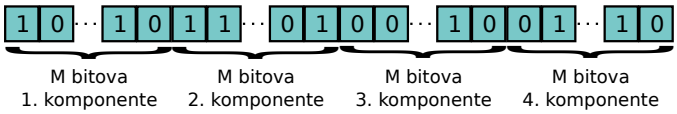
pri čemu je oznaka $x_i$ korištena za $i$-tu komponentu vektora $\vec{x}$. Lako je vidjeti da je za proizvoljan $N \geq 1$ vektor $\vec{x}^*$ upravo jednak $[1\ 2\ \ldots\ N]$ i da je tada $f(\vec{x}^*) = N$. Stoga ćemo tijekom rada genetskog algoritma uvijek znati koliko smo blizu (odnosno daleko) od optimalnog rješenja. Prostor pretraživanja bit će podskup $\mathbb{R}^N$ takav da je vrijednost koju svaka pojedinačna komponenta može poprimiti iz intervala $[-50,\ 50]$.

Eksperimente opisane u ovom poglavlju radit ćemo uz $n = 4$ pa je u tom slučaju $\vec{x}$ 4-dimenzijski vektor a funkcija postiže maksimum u $\vec{x}^* = [1\ 2\ 3\ 4]$ iznosa 4.

Populacija genetskog algoritma sastojat će se od *VelPop* jedinki pri čemu će svaka jedinka biti slijed of $M_{tot}$ bitova. U tom nizu prva četvrtina bitova kodirat će prvu komponentu rješenja, druga četvrtina bitova drugu komponentu rješenja i tako dalje. Ovo je prikazano na slici 10.3. Za svakoj komponenti rješenja pripada $M$ bitova pa je ukupna duljina binarnog niza jednaka $M_{tot} = 4 \cdot M$.

Rješenja optimizacijskog problema koji trenutno rješavamo nisu binarni nizovi – rješenja su vektori realnih brojeva. Stoga trebamo definirati način na koji ćemo svaki binarni uzorak preslikati u jedan vektor realnih brojeva: trebamo proceduru za *dekodiranje* rješenja.

Jedan od najjednostavnijih načina dekodiranja jest uporabom prirodnog binarnog koda. Ako je jedna komponenta predstavljena kao binarni uzorak




Slika 10.3: Kromosom kao niz bitova

od $M$ bitova, taj binarni uzorak možemo tretirati kao cijeli broj prikazan prirodnim binarnim kodom. U tom slučaju binarni uzorak `000...00` predstavlja broj 0, binarni uzorak `000...01` predstavlja broj 1, i tako redom sve do binarnog uzorka `111...11` koji predstavlja broj $2^M - 1$. Ove cijele brojeve možemo iskoristiti za linearnu interpolaciju raspona decimalnih brojeva koji pretražujemo (odnosno za uniformno uzorkovanje tog intervala). Neka je zadan neki niz bitova i neka taj niz bitova, kada ga se promatra kao cijeli broj kodiran prirodnim binarnim kodom, odgovara cijelom broju $m \in \{0, \ldots, 2^M - 1\}$. Ako algoritam pretražuje interval $[x_{min}, x_{max}]$, tada je decimalni broj koji odgovara danom binarnom nizu odnosno cijelom broju $m$ dan izrazom:

$$x = \frac{m}{2^M - 1} \cdot (x_{max} - x_{min}) + x_{min}. \tag{10.2}$$

Ako je rješenje vektor decimalnih brojeva, tada se izraz (10.2) primjenjuje na svaku pojedinačnu komponentu odnosno podniz binarnog uzorka koji odgovara promatranoj komponenti rješenja.

Evo u nastavku primjera koji to ilustrira.

---

**Primjer:** 29

Genetski algoritam za prikaz rješenja koristi niz od $M_{tot} = 16$ bitova pri čemu se dekodiranje radi uporabom prirodnog binarnog koda. Problem koji se rješava je optimizacija nad 4-dimenzijskim podprostorom decimalnih brojeva koji je po dimenzijama ograđen na $[-50, 50]$. Neka je jedan od kromosoma jednak `1101000110100000`. O kojem se rješenju radi?

**Rješenje:**

Kako su rješenja 4-dimenzijski vektori realnih brojeva, svakoj od 4 komponente pripada $M = \frac{M_{tot}}{4} = \frac{16}{4} = 4$ bita. Stoga ćemo kromosom razdijeliti u 4 grupe, počev od lijeve strane: `1101`, `0001`, `1010` i `0000`. Svaki od ova četiri binarna uzorka predstavlja jedan cijeli broj ako uzorke promatramo kao cijele brojeve zapisane prirodnim binarnim kodom. Ti brojevi su redom $m_1 = 1101_2 = 13_{10}$, $m_2 = 0001_2 = 1_{10}$, $m_3 = 1010_2 = 10_{10}$ te $m_4 = 0000_2 = 0_{10}$.

Ako označimo rješenje $\vec{x} = [x_1\ x_2\ x_3\ x_4]$, svaki $x_i$ dobit ćemo iz $m_i$ uporabom izraza (10.2) pri čemu je $x_{min} = -50$ a $x_{max} = 50$. Slijedi:

$$x_1 = \frac{m_1}{2^M - 1} \cdot (x_{max} - x_{min}) + x_{min} = \frac{13}{15} \cdot 100 - 50 = 36.666,$$

$$x_2 = \frac{m_2}{2^M - 1} \cdot (x_{max} - x_{min}) + x_{min} = \frac{1}{15} \cdot 100 - 50 = -43.333,$$

$$x_3 = \frac{m_3}{2^M - 1} \cdot (x_{max} - x_{min}) + x_{min} = \frac{10}{15} \cdot 100 - 50 = 16.666 \text{ te}$$

$$x_4 = \frac{m_4}{2^M - 1} \cdot (x_{max} - x_{min}) + x_{min} = \frac{0}{15} \cdot 100 - 50 = -50.$$

Pripadno rješenje tada je $\vec{x} = [36.666\ -43.333\ 16.666\ -50]$.

---

Uočimo da algoritam koji koristi binarni kromosom iz prethodnog primjera ne može raditi baš precizno pretraživanje. Svakoj komponenti u tom primjeru pripada jedan binarni uzorak duljine 4 bita. Taj binarni uzorak može poprimiti samo $2^4 = 16$ različitih kombinacija što znači da interval $[-50, 50]$ može uzorkovati u samo 16 točaka: dva uzorka su rubovi intervala i još se 14 uzoraka uzima uniformno iz unutrašnjosti intervala. Želimo li finije pretraživanje, potrebno je povećati broj bitova koji se koristi za svaku komponentu rješenja. Definirajmo stoga *preciznost pretraživanja* kao udaljenost između dva susjedna rješenja promatrane komponente. Ta je udaljenost dana izrazom:

$$\Delta = \left(\frac{m+1}{2^M - 1} \cdot (x_{max} - x_{min}) + x_{min}\right) - \left(\frac{m}{2^M - 1} \cdot (x_{max} - x_{min}) + x_{min}\right) \tag{10.3}$$

$$= \frac{x_{max} - x_{min}}{2^M - 1}. \tag{10.4}$$

Ako je pretraživanje potrebno obaviti uz zadanu preciznost $\Delta$, potreban broj bitova može se dobiti direktno iz izraza (10.4) i određen je izrazom:

$$M = ld\left(\frac{x_{max} - x_{min}}{\Delta} + 1\right) \tag{10.5}$$

gdje je $ld$ oznaka za dualni logaritam (tj. logaritam po bazi 2).

---

**Primjer:** 30

Pretraživanje opisano u prethodnom primjeru potrebno je obaviti uz preciznost $10^{-4}$. Koliko bitova trebamo za svaku komponentu i kolika će biti ukupna veličina kromosoma?

**Rješenje:**

Koristeći izraz (10.5) računamo:

$$M = ld\left(\frac{x_{max} - x_{min}}{\Delta} + 1\right) = ld\left(\frac{100}{10^{-4}} + 1\right) = ld\left(10^6 + 1\right) \approx 19.93$$

Slijedi da je potrebno uzeti $M = 20$ bitova po komponenti odnosno ukupno $M_{tot} = 4 \cdot M = 80$ bitova za 4-dimenzijski vektor realnih brojeva i traženu preciznost. Ovime je definiran prostor pretraživanja od $2^{80}$ unutar kojeg je potrebno pronaći onaj jedan binarni uzorak koji predstavlja optimalno rješenje.

---

Za implementaciju genetskog algoritma potrebni su nam još i *operator križanja* koji uzima dva rješenja (roditelje) i temeljem njih generira novo rješenje (dijete) koje sadrži dio genetskog materijala svakog od roditelja te *operator mutacije* koji će na rješenju napraviti nasumičnu promjenu.

Za križanje ćemo koristiti operator *uniformnog križanja* koji binarni niz djeteta konstruira tako da svaki bit sa vjerojatnošću od 50% preuzme od prvog roditelja odnosno s vjerojatnošću od 50% od drugog roditelja. Implementacija ovog operatora prikazana je u nastavku.

Ispis 10.1: Uniformno križanje.

```
1  public static BitvectorSolution uniformCrossover(
2      BitvectorSolution parent1, BitvectorSolution parent2,
3      Random rand) {
4
5    BitvectorSolution child = parent1.newLikeThis();
6    for(int i = 0; i < child.bits.length; i++) {
7      if(parent1.bits[i]==parent2.bits[i]) {
8        child.bits[i] = parent1.bits[i];
9      } else {
10       child.bits[i] = rand.nextFloat() < 0.5 ?
11         parent1.bits[i] : parent2.bits[i];
12     }
13   }
14   return child;
15
16 }
```

Operator mutacije izvest ćemo na sljedeći način. Kako koristimo niz bitova, operator mutacije invertirat će pojedine bitove. Neka je $p_m$ vjerojatnost promjene bita. Tada svaki od bitova binarnog uzorka invertiramo s tom vjerojatnošću. To pak znači da je moguće da mutacija ne promijeni ništa, promjeni nekoliko bitova ili ih promjeni sve; *u prosjeku*, ako mutaciju primjenjujemo nad binarnim uzorkom duljine $M_{tot}$, bit će invertirano $p_m \cdot M_{tot}$. Izvedba ovakve mutacije dana je u nastavku.

Ispis 10.2: Izvedba mutacije.

```
1  public static void binaryMutate(
2      BitvectorSolution solution, Random rand, double pm) {
3
4    float fpm = (float)pm;
5    for(int i = 0; i < solution.bits.length; i++) {
6      if(rand.nextFloat()<fpm) {
7        solution.bits[i] = (byte)(1 - solution.bits[i]);
8      }
9    }
10
11 }
```

Još je potrebno objasniti kako ćemo birati roditelje koji sudjeluju u križanju. Koristit ćemo proporcionalnu selekciju koja roditelja bira slučajno iz populacije ali na način da je vjerojatnost odabira jedinke proporcionalna njezinoj dobroti. Ako s $f_i$ označimo dobrotu $i$-te jedinke, s $p_i$ vjerojatnost odabira $i$-te jedinke te s $popSize$ broj jedinki u populaciji, tada treba vrijediti:

$$p_1 = k \cdot f_i$$

$$\ldots$$

$$p_{popSize} = k \cdot f_{popSize}$$

gdje je $k$ normalizacijska konstanta. Istovremeno, kako se radi o vjerojatnostima, mora biti zadovoljeno:

$$\sum_{i=1}^{popSize} p_i = 1.$$

Slijedi:

$$\sum_{i=1}^{popSize} (k \cdot f_i) = k \cdot \sum_{i=1}^{popSize} f_i = 1$$

što daje:

$$k = \frac{1}{\sum_{i=1}^{popSize} f_i}.$$

Stoga je vjerojatnost odabira $i$-te jedinke jednaka:

$$p_i = \frac{f_i}{\sum_{i=1}^{popSize} f_i}.$$

Jednostavna i poprilično neefikasna implementacija ovakve selekcije dana je u nastavku.

Ispis 10.3: Proporcionalna selekcija.

```
1  public static <T extends SingleObjectiveSolution> T[]
2    proportionalSimpleChoose(
3      T[] population, Random rand, int howMany) {
4
5    @SuppressWarnings("unchecked")
6    T[] parents = (T[]) Array.newInstance(
7      population.getClass().getComponentType(),
8      howMany);
9
10   // Izracunaj sumu svih dobrota:
11   double sum = 0;
12   for(int i = 0; i < population.length; i++) {
13     sum += population[i].fitness;
14   }
15
16   // Onoliko puta koliko biramo roditelje ponavljaj:
17   for(int parentIndex = 0; parentIndex < howMany; parentIndex++) {
18     // Generiraj slucajni broj iz [0, sum]:
19     double r = rand.nextDouble();
20     double limit = r * sum;
21     // Nadi jedinku koju smo time pogodili:
22     int chosen = 0;
23     double upperLimit = population[chosen].fitness;
24     while(limit > upperLimit && chosen < population.length-1) {
25       chosen++;
26       upperLimit += population[chosen].fitness;
27     }
```

```
28      parents[parentIndex] = population[chosen];
29    }
30    return parents;
31
32  }
```

Prkazani kod kao argument dobiva populaciju iz koje se biraju roditelji te broj roditelja koje treba odabrati.

Cjeloviti generacijski genetski algoritam napisan uporabom opisanih metoda (koje su izdvojene u novi razred) prikazan je u nastavku.

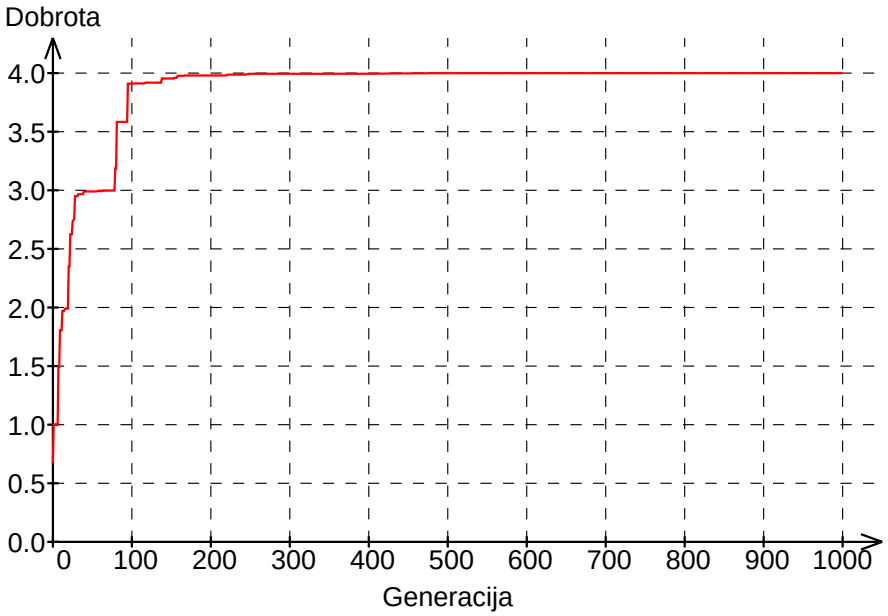
Ispis 10.4: Generacijski genetski algoritam.

```
1   private static double[] gaLoop(IFunction f,
2       BitvectorDecoder decoder, int popSize, double pm,
3       Random rand, int generationLimit) {
4
5     BitvectorSolution[] population =
6       EvoUtil.initializePopulation(popSize, rand, decoder);
7
8     for(int i = 0; i < population.length; i++) {
9       population[i].fitness = f.valueAt(
10        decoder.decode(population[i])
11      );
12    }
13
14    // Ponavljaj zadani broj generacija:
15    for(int generation = 1; generation <= generationLimit;
16      generation++) {
17
18      // Polje novih rjesenja:
19      BitvectorSolution[] nextPopulation =
20        new BitvectorSolution[popSize];
21
22      // U novu populaciju prenesi trenutno
23      // najbolje rjesenje (elitizam):
24      BitvectorSolution best = EvoUtil.findBest(population);
25      nextPopulation[0] = best;
26
27      // Preostalih popSize-1 rjesenja generiraj operatorima
28      for(int i = 1; i < popSize; i++) {
29        BitvectorSolution[] parents =
30          EvoUtil.proportionalSimpleChoose(population, rand, 2);
31        BitvectorSolution child =
32          EvoUtil.uniformCrossover(parents[0], parents[1], rand);
33        EvoUtil.binaryMutate(child, rand, pm);
34        child.fitness = f.valueAt(decoder.decode(child));
35        nextPopulation[i] = child;
36      }
37
38      // Ucini novu populaciju trenutnom
39      population = nextPopulation;
40    }
41
```

```
42    return decoder.decode(EvoUtil.findBest(population));
43  }
```

Algoritam kao parametar očekuje referencu na objekt koji radi dekodiranje rješenja, pa je u tom smislu neovisan o točnom načinu dekodiranja koje se želi koristiti. U prvom skupu eksperimenata koji će ovdje biti prikazanim kao dekoder ćemo koristiti objekt koji radi dekodiranje kod kojeg se niz bitova tumači kao cijeli broj zapisan u prirodnom binarnom kodu. Njegova metoda decode prima binarni uzorak a vraća polje decimalnih brojeva (tj. vektor koji predstavlja rješenje). Funkcija koja se optimira modelirana je sučeljem IFunction koje definira metodu **double** valueAt(**double**[] x); koja prima vektor $\vec{x}$ i vraća vrijednost funkcije u toj točki.

Tipičan tijek optimizacije prikazan je na slici 10.4. Graf prikazuje funkciju dobrote najboljeg rješenja kroz generacije, pri čemu je dobrota poistovjećena s iznosom funkcije koju se optimira, budući da je njezin minimum 0 i da radimo maksimizaciju. Rezultati prikazani na grafu dobiveni su uz sljedeće parametre: veličina populacije iznosi 50, vjerojatnost mutacije bita iznosi 0.05, broj generacija je 1000, broj bitova po komponenti je 20 čime se prostor uz raspon od $[-50\ 50]$ pretražuje s preciznošću od $10^{-4}$. Svaki kromosom sastavljen je od 80 bitova.

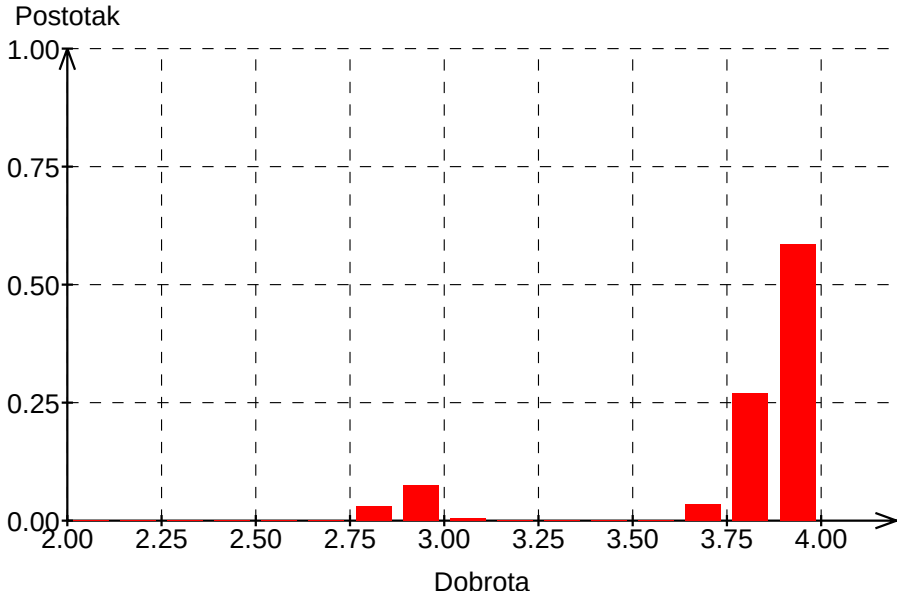



Slika 10.4: Kretanje dobrote najboljeg rješenja kroz generacije

Kako je početna generacija sastavljena od 50 nasumično generiranih binarnih nizova, njezina dobrota je bila oko 0.67. Potom, iz generacije u generaciju, dobrota najboljeg rješenja se malo po malo popravlja. Pri tome nije netipično vidjeti da se u više uzastopnih generacija dobrota ne mijenja da bi

se potom skokovito promijenila, potom kroz nekoliko generacija popravljala i potom opet jedno vrijeme stagnirala. Pogledate li malo bolje na kojim to dobrotama dolazi do stagnacije, uočit ćete da su to upravo situacije u kojima je algoritam pronašao jednu ispravnu komponenta rješenja, potom još jednu ispravnu komponentu rješenja i tako sve do optimalnog rješenja u kojem su pronađene sve četiri komponente.

Pokrenemo li više puta genetski algoritam, nećemo svaki puta dobiti kretanje dobrote koje je identično onome prikazanome na slici 10.4. Razlog tome leži u stohastičkoj naravi genetskog algoritma. Stoga je prirodno zapitati se kako izgleda distribucija najboljih rješenja koja se dobivaju ako se algoritam pokreće više puta? Na pitanje ćemo odgovoriti eksperimentom. Genetski algoritam pokrenut ćemo 200 puta uz sljedeće parametre: veličina populacije iznosi 20, vjerojatnost mutacije bita iznosi 0.05, broj generacija je 1000. Najbolje rješenje dobiveno nakon svakog pokretanja ćemo pohraniti, čime ćemo završiti s 200 rješenja. Njihova distribucija prikazana je na slici 10.5.




Slika 10.5: Distribucija najboljih rješenja

Prikazani histogram dobiven je podjelom 200 najboljih rješenja u grupe prema dobroti (prvu grupu čine rješenja dobrote od 2 do 2.125, drugu grupu čine rješenja dobrote od 2.125 do 2.25 i tako dalje sve do zadnje grupe koju čine rješenja dobrote od 3.875 do 4). Potom je izračunato koliko posto rješenja se nalazi u kojoj grupi i to je prikazano na grafikonu. Kako možemo vidjeti, razdioba rješenja nije normalna niti simetrična. U preko 50% pokretanja dobiveno je rješenje koje se nalazi u grupi 3.875–4. U nešto više

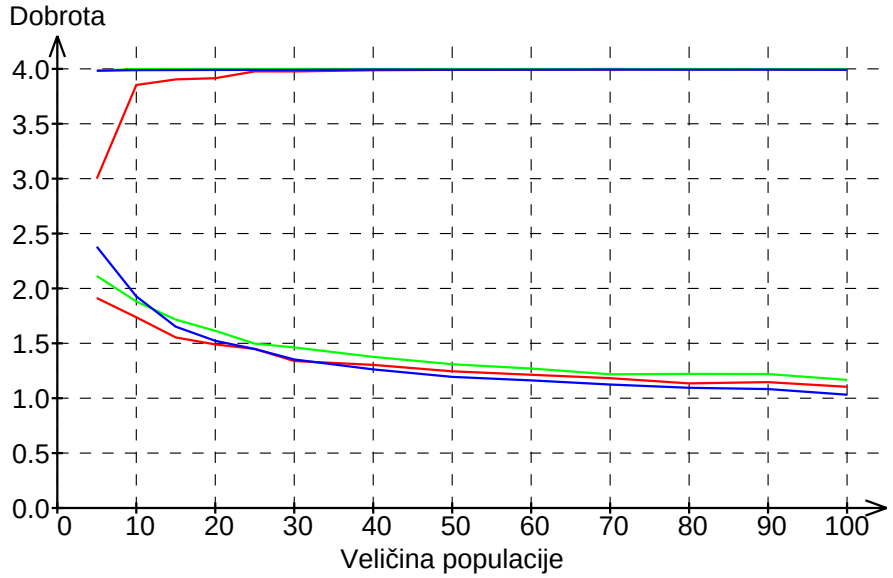
od 25% slučajeva rješenja su u prvoj lošijoj grupi. Takve grupe nema s desne strane najbolje grupe što je i jasno: nemoguće je da postoje bolja rješenja od optimalnih. Ovo je pak važan detalj jer kada krenemo govoriti o prosječnom ponašanju optimizacijskog algoritma, važno je razumjeti koji se statistički pokazatelj koristi i zašto. Primjerice, nakon napravljenih 200 pokretanja algoritma mogli bismo izračunati *prosječnu* dobrotu rješenja i istaknuti prosjek. Ovo međutim neće biti dobar pokazatelj: distribucija rješenja nije normalna, nije simetrična i ima (ponekad i dugačak) rep prema lijevoj strani. Taj će rep prosjek povući u lijevu stranu i stoga će biti manji no što bismo htjeli.

Bolji statistički pokazatelj u ovom je slučaju *medijan*: medijan niza brojeva je onaj broj od kojeg u nizu ima jednak broj manjih i većih brojeva. Ili drugačije, ako 200 dobivenih dobrota sortiramo i potom uzmemo onu koja se nalazi točno na polovici sortiranog niza, to je medijan. U slučaju optimizacijskih algoritama, medijan je puno realnija mjera: to je broj koji nam kaže da ćemo u pola slučajeva dobiti rješenje barem takve dobrote a moguće i bolje.

Pogledajmo sada nekoliko tipičnih eksperimenata: ima li veličina populacije, vjerojatnost mutacije ili pak broj generacija utjecaja na najbolje pronađeno rješenje? Ima li način prikaza rješenja utjecaja na najbolje pronađeno rješenje? Da bismo odgovorili na ova pitanja, napravljene su još dvije inačice opisanog genetskog algoritma: prva inačica umjesto dekodiranja prirodnim binarnim kodom pretpostavlja da su cijeli brojevi u binarne uzorke kodirani Grayevim kodom pa tako radi i dekodiranje. Druga inačica u cijelosti mijenja prikaz rješenja: umjesto da radi s nizom bitova, rješenje predstavlja direktno kao polje od 4 decimalna broja koje izravno predstavlja vektor $\vec{x}$. U tom slučaju mutacija je izvedena tako da u 50% slučajeva odabere jednu komponentu rješenja i doda joj slučajno generirani broj izvučen iz normalne distribucije $b\,\mathcal{N}(0,1)$ pa potom skaliran s 2 a u 50% slučajeva odabere jednu komponentu i potpuno je slučajno generira prema uniformnoj distribuciji $\mathcal{U}(-50,50)$. U ovoj implementaciji mutacija nikada ne mijenja više od jedne komponente rješenja. Križanje je izvedeno tako da u 50% slučajeva emulira uniformno križanje (dijete svaku od 4 komponente rješenja slučajno bira ili iz jednog roditelja ili iz drugog) a u 50% slučajeva svaku od komponenti rješenje djeteta postavi na aritmetičku sredinu odgovarajućih komponenti roditelja.

Prva serija eksperimenata ima zadaću istražiti što se događa s najboljim rješenjima kako variramo veličinu populacije. Eksperiment je proveden na sljedeći način. Vjerojatnost mutacije bita postavljena je na 0.05, broj generacija je 1000, broj bitova po komponenti je 20 čime se prostor uz raspon od $[-50\ 50]$ pretražuje s preciznošću od $10^{-4}$. Svaki kromosom sastavljen je od 80 bitova. Veličine populacije varirane su od 5 do 100 i za svaku fiksiranu veličinu populacije genetski je algoritam pokrenut 200 puta.

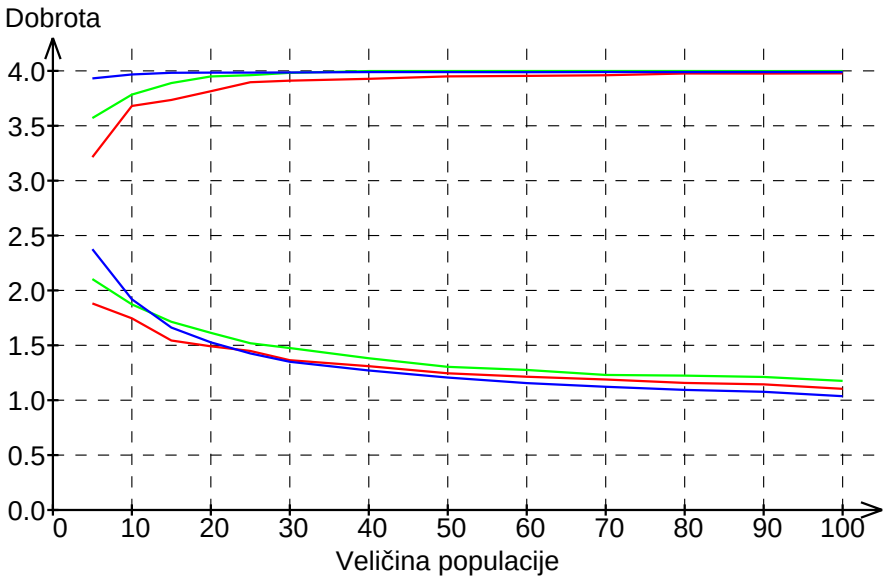
Slika 10.6 kao statistički pokazatelj koristi medijan. Za svaku fiksiranu




Slika 10.6: Ovisnost medijana najboljeg rješenja o veličini populacije. Gornji dio prikazuje statistiku najboljih rješenja a donji dio statistiku prosječnih rješenja. Crveno: prirodni binarni kod, zeleno: Grayev kod, plavo: decimalna reprezentacija.

veličinu populacije genetski algoritam je pokrenut 200 puta i za svako pokretanje je pohranjena dobrota najboljeg rješenja te prosječna dobrota konačne populacije. Potom su izračunati medijan dobrota najboljih rješenja te medijan prosječnih dobrota populacije. To je zatim ponovljeno za niz drugih veličina populacija i naravno za tri vrste prikaza.

Što možemo zaključiti sa slike? Na ovom konkretnom primjeru čak i s vrlo malim populacijama (najmanja isprobana populacija bila je veličine 5 jedinki) genetski algoritam uspjeva u 1000 generacija u barem pola pokretanja pronaći optimalno rješenje (medijan je 4 neovisno o veličini populacije). Inačica koja direktno radi s decimalnim brojevima s manjim populacijama ne uspjeva nedijan natjerati na 4; međutim, s porastom veličine populacije medijan brzo deseže vrijednost 4. Zanimljivo je pogledati donji dio grafa koji prikazuje medijan dobrota prosječnih rješenja 1000-te (tj. zadnje) populacije. Kada su populacije vrlo male, u 1000 generacija i prosječna dobrota populacije postaje relativno visoka. Naime, čim se pronađe bolje rješenje, kako je populacija mala, velika je vjerojatnost da će to rješenje imati direktan utjecaj i na preostale jedinke i da će se populacija brzo prilagoditi. Kako veličina populacije raste, promjene u populaciji postaju sve sporije i utjecaj najboljeg rješenja pada; to za posljedicu ima da proječna dobrota populacije sporije raste pa uz ograničen broj generacija medijan prikazan na slici pada.



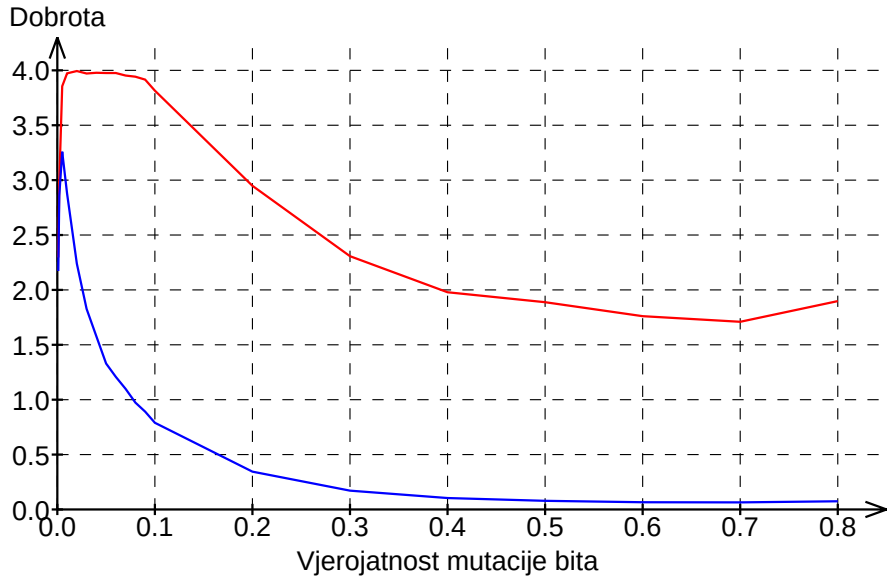

Slika 10.7: Ovisnost prosjeka najboljih rješenja o veličini populacije. Gornji dio prikazuje statistiku najboljih rješenja a donji dio statistiku prosječnih rješenja. Crveno: prirodni binarni kod, zeleno: Grayev kod, plavo: decimalna reprezentacija.

Iz prethodne analize čitatelj bi mogao zaključiti da je odlično mjesto upravo početak grafa: tamo gdje su veličine populacije najmanje prosječna veličina populacije je najveća. To je, međutim, vrlo opasno mjesto – mjesto gdje populacija lagano zaglavi u lokalnom optimumu iz kojeg se teško može izvući. Ovo je najbolje vidljivo na slučaju algoritma koji koristi polje decimalnih brojeva: uz veličinu populacije od 5 jedinki, medijan najboljih rješenja je 3. To znači da je algoritam uspio pronaći optimalne vrijednosti za 3 od 4 komponente, ali u populaciji nije bilo dovoljno genetske raznolikosti da bi se operatorima križanja omogućilo da "napipaju" optimalnu vrijednost i preostale četvrte komponente. S povećanjem veličine populacije medijan prosječnih dobrota populacije pada ali raste genetska raznolikost: već uz populaciju od 30 jedinki algoritam uspjeva konstruirati optimalne vrijednosti za sve četiri komponente i medijan se penje na 4.

Kako bismo još jednom istaknuli razliku između medijana i prosječne vrijednosti, rezultati istog eksperimenta obrađeni su koristeći prosjek a ne medijan kao statistički pokazatelj. Rezultat ove analize dan je na slici 10.7. Pogledajte prosječne vrijednosti dobrota najboljih rješenja i usporedite ih s medijanima dobrota najboljih rješenja prikazanih na slici 10.6. Za veličinu populacije od 5 jedinki prosječna dobrota najboljih rješenja je u svim slučajevima manja o 4; to se može zahvaliti velikom broju slučajeva u kojima je

algoritam zaglavio na nekom od nepovoljnijih lokalnih optimuma pa je taj rep prosjek povukao prema nižim vrijednostima.

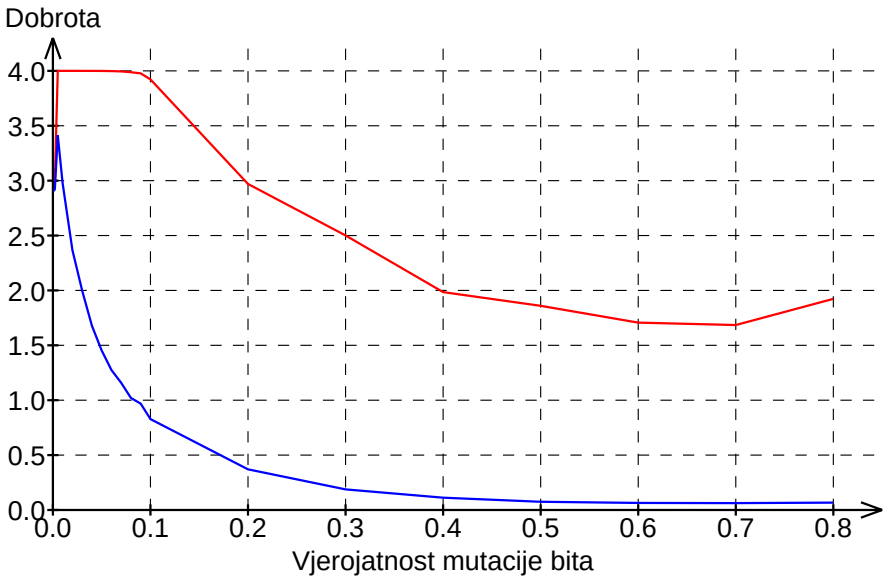
Sljedeće što ćemo istražiti jest utjecaj vjerojatnosti mutacije bita na medijan najboljih rješenja; eksperimente ćemo stoga napraviti za inačice koje koriste prirodni binarni kod i Grayev kod. Svi ostali parametri bit će fiksirani (veličina populacije 30, 1000 generacija).




Slika 10.8: Ovisnost medijana najboljih rješenja o vjerojatnosti mutacije bita kod inačice koja koristi prirodni binarni kod. Gornji dio prikazuje statistiku najboljih rješenja a donji dio statistiku prosječnih rješenja.

Slika 10.8 prikazuje kretanje medijana kod inačice koja koristi prirodni binarni kod dok slika 10.9 prikazuje kretanje medijana kod inačice koja koristi Grayev kod. Možemo uočiti da razlike postoje ali i pravilnosti. Uz premalu vjerojatnost mutacije bita dominantni mehanizam koji koristi genetski algoritam je križanje. Operator križanja je operator koji agregira populaciju – fokusira je na jedno područje i radi pretraživanje bliske okolice tog područja. Samo pod djelovanjem operatora križanja populacija će brzo izgubiti genetsku raznolikost i postupak pretraživanja će zaglaviti. To se na slici jasno vidi za vrlo male vjerojatnosti. Potom, kako vjerojatnost mutacije bita raste, dolazimo u područje u kojem je uspostavljen balans između sila uslijed kojih dolazi do sažimanja populacije te sila koje uzrokuju širenje prostora pretraživanja (za ovo posljednje u našem je slučaju zaslužan upravo operator mutacije). To je područje u kojem operator mutacije maksimalno doprinosi pronalasku dobrog rješenja.

Daljnjim porastom vjerojatnosti mutacije bita efekti mutacije postaju

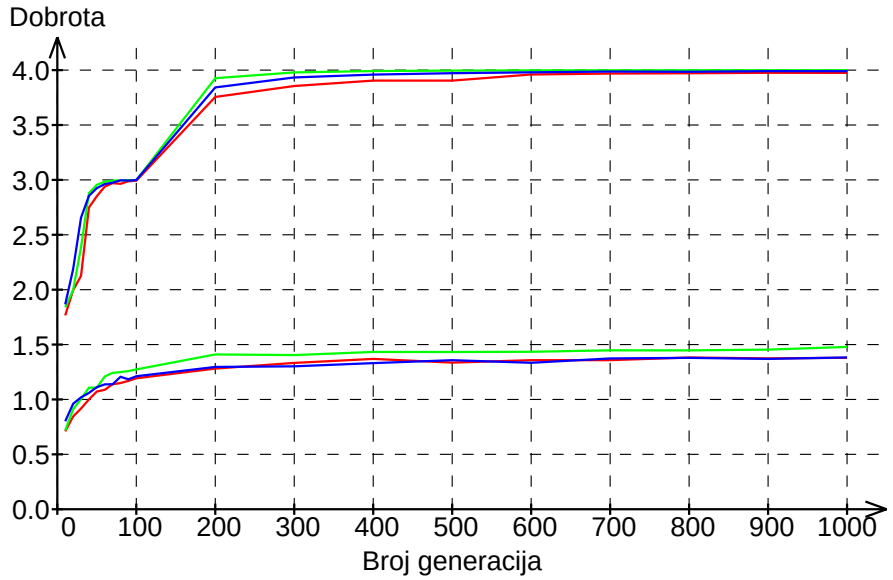



Slika 10.9: Ovisnost medijana najboljih rješenja o vjerojatnosti mutacije bita. Kod inačice koja koristi Grayev kod. Gornji dio prikazuje statistiku najboljih rješenja a donji dio statistiku prosječnih rješenja.

prejaki i kvaliteta rješenja polagano degradira. Uz velik iznos vjerojatnosti mutacije bita svi ostali mehanizmi pretraživanja koje koristi genetski algoritam bivaju potisnuti prevelikom količinom nasumičnih promjena koje radi mutacija; u krajnosti postupak pretraživanja svodi na nasumično generiranje novih binarnih nizova i njihovo vrednovanje, što nije efikasan način pretraživanja.

Konačno, možemo pogledati i kakav je utjecaj broja generacija koje smo spremni čekati i medijana dobrote najboljih rješenja koji možemo očekivati. Eksperimenti su rađeni uz veličinu populacije od 30 jedinki i vjerojatnost mutacije bita od 0.05. Rezultati eksperiment dani su na slici 10.10 i to za sve tri inačice algoritma. Očekivano, što se više generacija dopusti, algoritam se više približava području u kojem medijan postaje jednak 4.

Prethodni eksperimenti trebali su poslužiti tome da dobijete predodžbu kako genetski algoritam "diše". S obzirom da smo ustanovili da različite vrijednosti parametara itekako imaju utjecaj na rezultate koje algoritam daje, pitanje je: kako doći do optimalnih parametara? Pri tome treba uočiti da smo sve eksperimente radili tako da smo fiksirali sve parametre osim onog jednog koji smo istraživali. Nažalost, promjena bilo kojeg od parametara automatski mijenja i dobivene krivulje tako da ovaj odabir nije jednostavan, i o njemu će više riječi biti u sljedećem poglavlju.




Slika 10.10: Ovisnost medijana najboljih rješenja o broju generacija. Gornji dio prikazuje statistiku najboljih rješenja a donji dio statistiku prosječnih rješenja. Crveno: prirodni binarni kod, zeleno: Grayev kod, plavo: decimalna reprezentacija.

# Poglavlje 11

# Načini poboljšanja djelotvornosti genetskih algoritama

## 11.1 Uvod

Genetski algoritam je recept koji kazuje što treba raditi s genetskim materijalom kako bi se s određenom vjerojatnošću nakon određenog vremena postiglo zadovoljavajuće rješenje zadanog optimizacijskog problema. Genetski materijal je skup svojstava koji opisuju neku jedinku. Genetski algoritam ne određuje na koji način je genetski materijal pohranjen u radni spremnik, niti kako treba manipulirati genetskim materijalom, već samo kaže da se genetski materijal treba razmjenjivati, slučajno mijenjati, a bolje jedinke trebaju s većom vjerojatnošću preživljavati selekciju. Kako će se bolje jedinke selektirati za reprodukciju, te kako će se i s kolikim vjerojatnostima obaviti križanje i mutacija, određuje se na temelju iskustva i eksperimentalno, tj. heuristički. Stoga postoji velika sloboda u izradi genetskih algoritama. Cijena te slobode su loši ili nikakvi rezultati koji se najčešće dobivaju s genetskim algoritmom koji nema podešene parametre ili nema genetske operatore prilagođene problemu. U ovom poglavlju bavit ćemo se upravo optimiranjem samog genetskog algoritma.

Genetski algoritam je vremenski zahtjevan i troši najviše procesorskog vremena od bilo koje druge metode optimiranja. Cilj podešavanja, odnosno optimizacije genetskog algoritma je pronaći genetski algoritam koji u što je moguće kraćem roku pronalazi zadovoljavajuća rješenja. Jedan od mogućnosti načina ubrzanja genetskog algortma je, naravno, njegova paralelizacija. Mi se u ovom poglavlju nećemo baviti paralelizacijom genetskog algoritma, već ćemo se baviti poboljšavanju djelotvornosti sekvencijskog genetskog algoritma. Štoviše, kada podesimo sekvencijski genetski algoritam, njega je moguće paralelizirati i time postići još bolje rezultate.

| **CILJEVI** | **NAČINI** |
|---|---|
| Povećanje vjerojatnosti postizanja dobrih rješenja | Podešavanje parametara |
| Povećanje kvalitete dobivenih rješenja | |
| Skraćenje trajanja izvođenja programa: | |
| - povećanje brzine konvergencije i smanjenje broj iteracija | |
| - skraćenje trajanje izvođenja jedne iteracije | Optimiranje izvornog teksta programa |
| - paralelizacija | Paralelno izvođenje genetskih operacija |

Tablica 11.1: Ciljevi i načini poboljšanja djelotvornosti genetskih algoritama

Podešavanjem (optimiranjem) parametara genetskog algoritma mogu se postići zadovoljavajući rezultati. Međutim, sâm postupak podešavanja je dugotrajan proces. Primjerice, u postupku podešavanja samo tri parametra genetskog algoritma opisanog u ovom poglavlju trebalo je nekoliko tisuća puta pokrenuti genetski algoritam. Cijeli postupak trajao je oko 100 sati na današnjim PC računalima, unatoč tome što je jedan evolucijski proces trajao svega nekoliko minuta. Drugim riječima, postupak podešavanja parametara je vremenski vrlo zahtjevan. Srećom, postoje i neke pravilnosti u ponašanju genetskog algoritma koje se mogu iskoristiti za skračivnje pretrage za optimalnim parametrima algoritma.

Djelotvornost genetskog algoritma se može poboljšati na tri načina: povećanjem vjerojatnosti postizanja dobrih rješenja, povećanjem kvalitete dobivenih rješenja i skraćenjem trajanja izvođenja. Trajanje izvođenja se može skratiti također na tri načina: povećanjem brzine konvergencije čime se smanjuje broj iteracija, smanjenjem trajanja izvođenja jedne iteracije i paralelnim izvođenjem cijelog genetskog algoritma ili samo pojedinih genetskih operatora. Navedeni ciljevi mogu se ostvariti podešavanjem parametara, optimiranjem izvornog teksta programa i paralelizacijom genetskog algoritma (tablica 11.1).

Povećanje vjerojatnosti postizanja dobrih rješenja, povećanje kvalitete rješenja i smanjenje broja iteracija može se postići podešavanjem parametara genetskog algoritma. Podešavanje parametara je dugotrajan posao, jer se obavlja isključivo eksperimentalno. Mnogi autori se bave tom problematikom, pa se u literaturi mogu pronaći neki skupovi parametara, koji su podešeni za određeni genetski algoritam primijenjen za rješavanje određenog optimizacijskog problema. Ta tuđa iskustva mogu poslužiti kao dobre smjernice ili početne vrijednosti prilikom eksperimentalnog podešavanja pa-

rametara za specifičan genetski algoritam i specifičan optimizacijski problem. S obzirom da je proces podešavanja parametara dugotrajan, prilikom izgradnje genetskog algoritma treba voditi računa da broj parametara bude što je moguće manji.

Trajanje izvođenja jedne iteracije može se skratiti optimiranjem izvornog koda. Time su iscrpljene sve mogućnosti skraćenja trajanja izvođenja sekvencijskog GA, odnosno genetskog algoritma koji se izvodi na jednoprocesorskom računalu. Paraleliziranjem algoritma dodatno se skraćuje trajanje izvođenja. Paralelizirati se mogu svi ili samo neki genetski operatori. Ako se već ne obavljaju svi genetski operatori paralelno, poželjno je da se paralelno obavljaju oni operatori, koji troše najviše procesorskog vremena.

## 11.2 Odabir prikladnog prikaza rješenja

Za genetski algoritam jedinke su potencijalna rješenja, a okolinu predstavlja funkcija cilja. Želi li se implementirati genetski algoritam kao metoda optimiranja, potrebno je definirati proces dekodiranja i preslikavanja podataka zapisanih u kromosomu te odrediti funkciju dobrote. U konačnici svaka će jedinka, čije su osobine zapisane u kromosomu, biti u računalu zapisana kao niz jeinica i nula. Međutim, ono što predstavljaju te jedinice i nule naziva se prikazom rješenja. Tako primjerice postoje sljedeći načini prikaza rješenja:

- binarni prikaz,
- prikaz brojevima s pomičnom točkom,
- permutirani niz cijelih brojeva,
- niz brojeva,
- matrica,
- program,
- stablo te drugi.

Nisu svi genetski operatori primijenjivi za sve vrste prikaza rješenja. Zapravo, genetski operatori su prilagođeni upravo vrsti prikaza rješenja. Primjerice, primijeni li se križanje s jednom točkom prekida na dva permutirana niza cijelih brojeva dobit će se niz koji najvjerojatnije sadrži više istih cjelobrojnih vrijednosti. Radi se o nemogućem rješenju, odnosno rezultat križanja je niz cijelih brojeva, ali nije permutirani niz cijelih brojeva.

## 11.3 Vrednovanje jedinki

Postupak vrednovanja jedinki, odnosno postupak evaluacije provodi se izračunavanjem vrijednosti funkcije dobrote, odnosno funkcije kazne (ovisno o tome koristi li se generacijska ili eliminacijska selekcija, tj, izabire li se jedinka za reprodukciju u idućoj generaciji ili za brisanje iz populacije). Pravilnim izborom funkcije dobrote (ili kazne) uvelike se može utjecati na djelotvornost genetskog algoritma. Funkcija dobrote mora *dobro* oslikavati optimizacijski problem. Funkcija dobrote mora biti što je moguće jednostavnija, jer se u praksi pokazalo na mnoštvu primjera da genetski algoritam troši najviše vremena upravo za evaluaciju.

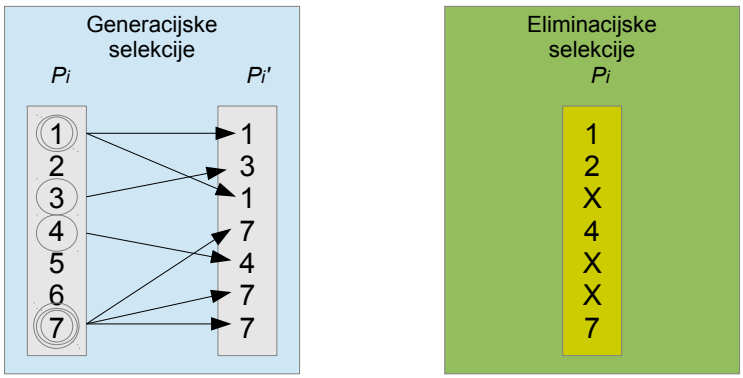
## 11.4 Selekcije

### 11.4.1 Podjela

Genetski algoritmi koriste selekcijski mehanizam za odabir jedinki koje će sudjelovati u reprodukciji. Selekcijom se omogućava prenošenje boljeg genetskog materijala iz generacije u generaciju. Zajedničko svojstvo svih vrsta selekcija je veća vjerojatnost odabira boljih jedinki za reprodukciju. Postupci selekcije se međusobno razlikuju po načinu odabira boljih jedinki. Prema načinu prenošenja genetskog materijala boljih jedinki u sljedeću iteraciju postupci selekcije se dijele na (slika 11.1):

- generacijske selekcije - selekcijom se direktno biraju bolje jedinke čiji će se genetski materijal prenijeti u sljedeću iteraciju i
- eliminacijske selekcije - biraju se loše jedinke za eliminaciju, a bolje jedinke prežive postupak selekcije.

Generacijskom selekcijom se odabiru bolje jedinke koje će sudjelovati u reprodukciji. Između jedinki iz populacije iz prethodnog koraka biraju se bolje jedinke i kopiraju u novu populaciju. Ta nova populacija, koja sudjeluje u reprodukciji, naziva se međupopulacijom. Na taj način se priprema nova generacija jedinki za postupak reprodukcije. To je ujedno prvi nedostatak generacijskih selekcija, jer se u radnom spremniku odjednom nalaze dvije populacije jedinki. Broj jedinki koje prežive selekciju je manji od veličine populacije. Najčešće se ta razlika u broju preživjelih jedinki i veličine populacije popunjava duplikatima preživjelih jedinki. Pojava duplikata u sljedećoj iteraciji je drugi nedostatak generacijskih selekcija, jer duplikati nikako ne doprinose kvaliteti dobivenog rješenja, već samo usporavaju evolucijski proces. Generacijska selekcija postavlja granice među generacijama. Svaka jedinka egzistira točno samo jednu generaciju. Izuzetak su jedino najbolje jedinke koje žive duže od jedne generacije i to samo ako se primijeni elitizam.

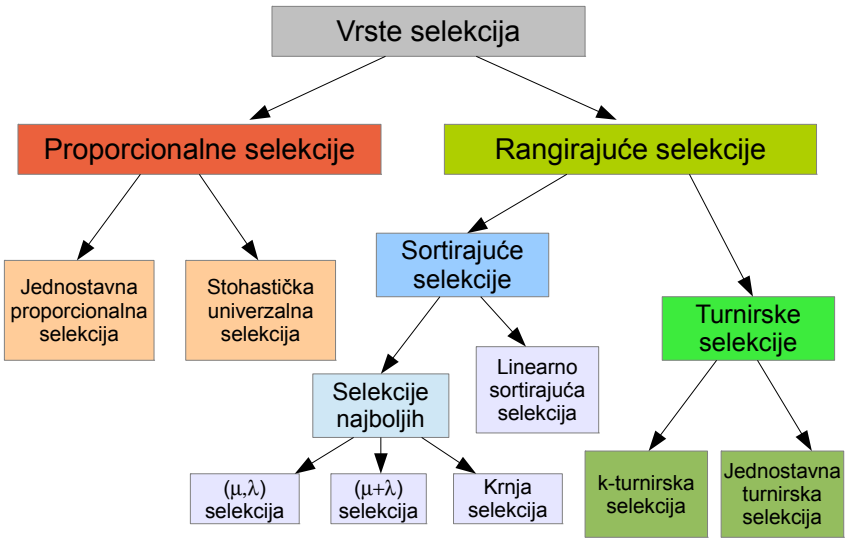



Slika 11.1: Podjela postupaka selekcije prema načinu prenošenja genetskog materijala boljih jedinki u novu iteraciju

Kod eliminacijskih selekcija bolje jedinke preživljavaju mnoge iteracije pa stroge granice između generacija nema, kao što je to slučaj kod generacijskih selekcija. Eliminacijska selekcija briše $M$ jedinki. Parametar $M$ naziva se mortalitetom ili generacijskim jazom. Izbrisane jedinke bivaju nadomještene jedinkama koje se dobivaju reprodukcijom. Eliminacijskom selekcijom se otklanjaju oba nedostatka generacijske selekcije: nema dvije populacije u istoj iteraciji i u postupku selekcije se ne stvaraju duplikati. Na prvi pogled je uvođenje novog parametra $M$ nedostatak eliminacijske selekcije, jer se svakim dodatnim parametrom značajno produžava postupak podešavanja algoritma. Međutim, uvođenjem novog parametra, nestaje parametar vjerojatnost križanja, jer križanje treba obaviti točno onoliko puta koliko je potrebno da se nadopuni populacija do početne veličine $N$. Što je veća vjerojatnost križanja, treba biti veći mortalitet i obrnuto. Primjerice, ako se koristi uniformno križanje, koje producira jednu novu jedinku, tada se točno $M$ puta treba ponoviti križanje kako bi se nadopunila populacija, odnosno zamijenile eliminirane jedinke. Selekcijom se osigurava prenošenje boljeg genetskog materijala s većom vjerojatnošću u sljedeću iteraciju. Ovisno o metodi odabira boljih jedinki kod generacijskih selekcija, odnosno loših jedinki kod eliminacijskih selekcija, postupci selekcije se dijele na proporcionalne i rangirajuće. Važno je napomenuti da je zajedničko obilježje svim selekcijama veća vjerojatnost preživljavanja bolje jedinke od bilo koje druge lošije jedinke. Selekcije se razlikuju prema načinu određivanja vrijednosti vjerojatnosti selekcije određene jedinke. Proporcionalne selekcije odabiru jedinke s vjerojatnošću koja je proporcionalna dobroti jedinke, odnosno vjerojatnost selekcije ovisi o kvocijentu dobrote jedinke i prosječne dobrote populacije. Broj jedinki s određenim svojstvom u sljedećoj iteraciji je proporcionalan

kvocijentu prosječne dobrote tih jedinki i prosječne dobrote populacije. Tako glasi dio teorema sheme koji se odnosi na selekciju.

Rangirajuće selekcije odabiru jedinke s vjerojatnošću koja ovisi o položaju jedinke u poretku jedinki sortiranih po dobroti. Rangirajuće selekcije se dijele na sortirajuće i turnirske selekcije. Sortirajuće sbilaelekcije su: linearno sortirajuća selekcija, $\mu - \lambda$ selekcije i krnja selekcija. Turnirske selekcije se dijele prema broju jedinki koje sudjeluju u turniru. Na slici 11.2 prikazana je podjela selekcija na proporcionalne i rangirajuće, te na njihove podvrste.




Slika 11.2: Vrste selekcija

Genetski algoritam eksplicitno ne definira genetske operatore (slika 10.2), tj. genetski algoritam ne kaže na koji način, odnosno kako genetski operatori obavljaju svoj posao, već što trebaju učiniti. Selekcija odabire bolje jedinke za reprodukciju. Križanjem se svojstva roditelja prenose na djecu, a mutacijom se slučajno mijenja genetski materijal. Kako će križanje osigurati prenošenje genetskog materijala na djecu i kako će mutacija izmijeniti genetski materijal, ovisi o samom problemu koji se rješava i o vrsti prikaza rješenja. Tako, primjerice, križanje s jednom točkom prekida nije pogodno za rješavanje problema trgovačkog putnika, ali se zato može primijeniti za optimiranje funkcija realne varijable i binarni prikaz. Jednako tako nije važno kako će selekcija odabrati bolje jedinke za reprodukciju, već je važno da bolje jedinke imaju veću vjerojatnost preživljavanja od lošijih. Osim toga, važno je i koliko veću vjerojatnost odabira imaju bolje jedinke od lošijih.

### 11.4.2 Selekcijski pritisak

Selekcijski pritisak je odnos vjerojatnosti preživljavanja dobrih i loših jedinki. Drugim riječima, selekcijski pritisak je sposobnost preživljavanja jedinki. Neka vjerojatnosti selekcije dvije jedinke s oznakama $i$ i $j$ iznose $p(i)$ i $p(j)$ respektivno. Neka je jedinka s oznakom $i$ bolja od one s oznakom $j$, tj. neka je $d(i) > d(j)$, odnosno $p(i) > p(j)$. Selekcijski pritisak je veći što je kvocijent $p(i)/p(j)$ veći ili što je veća razlika $p(i) - p(j)$. Selekcija ima veliki selekcijski pritisak ukoliko s velikom vjerojatnošću prenosi bolje jedinke u iduću iteraciju, odnosno s velikom vjerojatnošću eliminira jedinke ispodprosječne dobrote. Selekcijski pritisak utječe na kvalitetu rješenja. Genetski algoritmi s velikim selekcijskim pritiskom pronalaze prije rješenje, tj. brže konvergiraju žrtvujući kvalitetu dobivenog rješenja, jer brža konvergencija uzrokuje veću vjerojatnost zaglavljivanja u lokalnom optimumu. S druge strane, ako je selekcijski pritisak premali, nepotrebno se troši vrijeme na beskorisne iteracije jer je konvergencija u tom slučaju prespora.

Parametri selekcije utječu na selekcijski pritisak, odnosno na brzinu konvergencije algoritma. Na konvergenciju algoritma ne utječe samo selekcija, nego i ostali operatori, pa brzina konvergencije ne može biti mjera za selekcijski pritisak. Selekcijski pritisak se posredno određuje (kvantificira) uz pomoć primjerice *trajanja preuzimanja*, *selekcijske razlike* ili *selekcijskog intenziteta*.

**Trajanje preuzimanja** je prosječan broj generacija nakon kojih se populacija sastoji od $N$ duplikata najbolje jedinke, ukoliko se u svakoj iteraciji uporabi samo operator selekcije bez križanja i mutacije. Drugim riječima, mjeri se broj iteracija nakon kojih se eliminiraju sve jedinke osim najbolje. Što je trajanje preuzimanja manje, selekcijski pritisak je veći i obrnuto.

Trajanje preuzimanja poprima najmanju vrijednost kada je vjerojatnost selekcije svih jedinki jednaka i iznosi $p = \frac{1}{N}$ (slučajno pretraživanje). Osim o vrsti selekcije i njezinim kontrolnim parametrima, trajanje preuzimanja za proporcionalne selekcije ovisi i o funkciji cilja.

Selekcijom preživljavaju bolje jedinke i očekuje se da prosječna vrijednost preživjelih jedinki bude veća od prosječne vrijednosti cijele populacije. Posljedica selekcijskog pritiska je **selekcijska razlika** između preživjelih jedinki i cijele populacije [Cantú-Paz(2001)]. Selekcijska razlika $s$ je razlika prosječne vrijednosti dobrote preživjelih jedinki $\overline{d_p}$ i prosječne vrijednosti dobrote svih jedinki $\overline{d}$ u svakoj iteraciji $t$:

$$s(t) = \overline{d_p}(t) - \overline{d}(t) \tag{11.1}$$

Prosječna vrijednost dobrote svih jedinki koje čine populaciju računa se prema izrazu:

$$\overline{d}(t) = \frac{1}{N}\sum_{i=1}^{N} d_i(t) \tag{11.2}$$

Nadalje, neka je $\sigma(t)$standardna devijacija svih dobrota u populaciji u

generaciji $t$:

$$\sigma(t) = \sqrt{\frac{\sum_{i=1}^{N} \left[\overline{d_i}(t) - \overline{d}(t)\right]^2}{N}} \tag{11.3}$$

Uz pretpostavku da dobrota populacije $d_i$ ima normalnu razdiobu $N\left[\overline{d}(t), \sigma(t)\right]$ u generaciji $t$, selekcijska razlika je proporcionalna standardnoj devijaciji populacije:

$$s(t) = s_I \cdot \sigma(t) \tag{11.4}$$

pri čemu se faktor $s_I = s(t)/\sigma(t)$ naziva **selekcijskim intenzitetom** [Bäck(1995)].

Naravno, dobrota nema normalnu razdiobu za sve funkcije cilja i u svakom trenutku $t$. Međutim, za većinu funkcija cilja pred kraj evolucijskog procesa ($t >>$) razdioba vrijednosti funkcije dobrote populacije može se dobro aproksimirati normalnom razdiobom [Bäck(1995)].

Selekcijski pritisak je veći što je veća selekcijska razlika, odnosno selekcijski intenzitet. Selekcijska razlika ovisi o standardnoj devijaciji populacije koja se mijenja iz generacije u generaciju, a selekcijski intenzitet je konstantna vrijednost koja ovisi samo o vrsti selekcije i njezinim parametrima. Stoga se selekcijski intenzitet može koristiti kao mjera za selekcijski pritisak uz ograničenje da vrijednosti funkcije dobrote populacije imaju normalnu razdiobu.

Poželjno je da selekcijski mehanizam bude takav da se na jednostavan način može kontrolirati selekcijski pritisak. Dodatni zahtjevi nad postupkom selekcije prema Bäcku [Bäck(1994)] su:

- promjena kontrolnih parametara selekcije mora biti jednostavna, a posljedice te promjene moraju biti predvidive;
- poželjno je da selekcija ima samo jedan kontrolni parametar i
- raspon selekcijskog pritiska treba biti što veći.

Nažalost većina postupaka selekcije ne zadovoljava niti prvi zahtjev [Bäck(1994)].

### 11.4.3 Izbor selekcije

U postupku selekcije nije važan način odabira jedinki, već vjerojatnost odabira, odnosno selekcijski pritisak. Prilikom izbora selekcije favoriziraju se postupci selekcije koji su lako upravljivi (selekcijski pritisak ne ovisi o optimizacijskom problemu i kontrolira se jednim parametrom), jednostavni su za implementaciju, ne troše mnogo procesorskog vremena i pogodni su za paralelizaciju.

Analizirajmo prvo proporcionalne selekcije kroz njihove prednosti i nedostatke kako bi smo ih mogli usporediti sa rangirajućim selekcijama.

Nedostaci proporcionalnih selekcija pobrojani su u nastavku.

- Nad funkcijom cilja, ako se ona izravno koristi za izračunavanje vjerojatnosti selekcije, postavljena su sljedeća ograničenja: ne smije poprimati negativne vrijednosti i ne smije poprimati samo približno jednake velike vrijednosti. U prvom bi slučaju vjerojatnost selekcije bila negativna, a u drugom slučaju bi se genetski algoritam pretvorio u postupak slučajnog pretraživanja, jer sve jedinke imaju približno istu vjerojatnost selekcije. Rješenje tih problema je funkcija dobrote koja se dobiva translacijom funkcije cilja. Translacija funkcije cilja se mora obaviti u svakom koraku i to usporava rad genetskog algoritma.

- Nemoguće je paralelizirati postupak selekcije bez sinkronizacije dretvi, jer se u svakoj iteraciji mora izračunati ukupna dobrota populacije koja mora biti dostupna svim dretvama.

- Eliminacijska proporcionalna selekcija se mora obavljati sekvencijski, jer svaka eliminacija jedinke ima za posljedicu promjenu vjerojatnosti eliminacije svih ostalih (preživjelih) jedinki.

- Troše znatno više procesorskog vremena od rangirajućih selekcija.

- Nemaju mogućnosti podešavanja selekcijskog pritiska (nema parametra s kojim bi se mogao podešavati selekcijski pritisak).

- Selekcijski pritisak ovisi o funkciji dobrote. To znači da je za svaku funkciju cilja drugačiji selekcijski pritisak, a time i brzina konvergencije koja direktno utječe na kvalitetu dobivenih rješenja.

Nasuprot navedenim nedostacima, postoji samo jedna prednost proporcionalnih selekcija.

- Analiza rada genetskog algoritma je olakšana, jer gotovo sva teorijska istraživanja se temelje na proporcionalnim selekcijama. Stoga se, unatoč svim navedenim nedostacima, još uvijek nerjetko koristi u praksi.

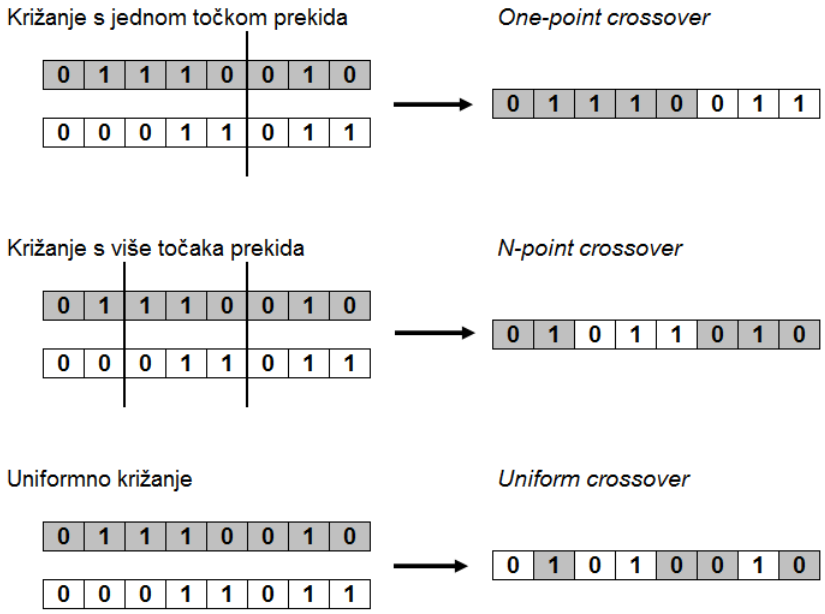
Kako je to već navedeno, svaki od postupaka selekcije na slici 11.2 može biti generacijski ili eliminacijski. Generacijske selekcije nisu pogodne za paraleliziranje, jer se reprodukcija mora obavljati odvojeno od selekcije. Reprodukcija može početi tek kada generacijska selekcija formira novu populaciju. Generacijska selekcija troši više spremničkog prostora jer algoritam obavlja posao nad dvije populacije. Povrh toga, generacijskom selekcijom se generiraju duplikati. Duplikate je poželjno nakon reprodukcije eliminirati, što dodatno usporava rad genetskog algoritma. Stoga je eliminacijska selekcija pogodnija za paralelno izvođenje te na jednostavan način je moguće ugraditi elitizam tako da se naprosto ne eliminiraju najbolje jedinke.

Postupci selekcije se dijele i prema tome kako funkcija cilja utječe na vjerojatnost selekcije. Ako funkcija cilja izravno utječe na vjerojatnost selekcije, tako da je vjerojatnost selekcije proporcionalna vrijednosti funkcije cilja, radi

se o proporcionalnim selekcijama. Između brojnih navedenih nedostataka proporcionalnih selekcija ističu se: nemogućnost paralelizacije eliminacijske proporcionalne selekcije bez sinkronizacije, nemogućnost jednostavne kontrole selekcijskog pritiska i prevelika ovisnost ponašanja genetskog algoritma o funkciji cilja. Dakle, selekciju pogodnu za paralelno izvođenje treba tražiti među rangirajućim selekcijama. Sortirajuće selekcije su nepogodne za paralelizaciju upravo zbog postupka sortiranja jedinki koji se mora obaviti u svakoj iteraciji. Osim toga, jednostavan eksperiment u kojem se mjerilo vrijeme trajanja eliminacijske proporcionalne i eliminacijske 3-turnirske selekcije dao je sljedeći rezultat: eliminacijska proporcionalna selekcija troši 3.4 puta više procesorskog vremena od eliminacijske 3-turnirske selekcije. Prema tome, eliminacijske turnirske selekcije su najpogodnije za paralelno izvođenje.

Turnirske selekcije obavljaju selekciju samo nad dijelom populacije slično kao i distribuirani genetski algoritam. Distribuirani genetski algoritam koristi izolirane subpopulacije nad kojima se paralelno obavljaju svi genetski operatori. Osim toga, selekcijski pritisak ne ovisi o funkciji cilja, već samo o jednom parametru: veličini turnira $k$. Prema tome, ostaje još samo odabir parametra $k$. $k$ je bilo koji cijeli broj u intervalu [2, VEL_POP].

Selekcijski pritisak se može na vrlo jednostavan način podešavati kod genetskih algoritama s $k$-turnirskom selekcijom i to uz pomoć parametra $k$. Što je veći $k$, veći je i selekcijski pritisak nad slabijim jedinkama, odnosno, bolje jedinke se više favoriziraju [Miller and Goldberg(1995)]. Preveliki selekcijski pritisak uzrokuje preranu konvergenciju k lokalnom optimumu, a premali pretvara algoritam u slučajnu pretragu. Optimalni selekcijski pritisak je, naravno, nešto između ta dva ekstrema. Prevelika vrijednost parametra $k$ uzrokuje preveliki selekcijski pritisak. Primjerice, ako se populacija sastoji od 30 jedinki i koristi se generacijska 10-turnirska selekcija 9 najlošijih jedinki nikada neće biti izabrane za iduću populaciju. Ovakav, preveliki selekcijski pritisak pogotovo nema smisla u početku evolucijskog procesa kada su sve jedinke otprilike jednako loše pa postoji velika opasnost da se dobar genetski materijal u lošijim jedinkama eliminira odmah na početku optimiranja. Pokazalo se [Golub(2004a)] da u većini slučajeva $k$-turnirska selekcija ima veći selekcijski pritisak od primjerice jednostavne proporcionalne selekcije. Prema tome, treba odabrati što je moguće manji $k$. Najmanji $k$ je 2, međutim, treba voditi računa da je selekcija jednostavna za implementaciju i paralelizaciju. Pokazalo se da je jednostavno paralelizirati genetski algoritam s $k$-turnirskom selekcijom u kojoj se nakon svakog turnira obavljaju i ostali genetski operatori (križanje i mutacija). Najmanji turnir u kojem je to moguće je veličine 3: slučajnim postupokom se odabiru 3 jedinke od VEL_POP jedinki u populaciji, odabira se najlošija od 3 slučajno odabrane jedinke i nadomješćuje se jedinkom koja je rezultat reprodukcije preživjele dvije jedinke. Ta se selekcija naziva eliminacijska 3-turnirska selekcija. Treba naglasiti da ova selekcija ima još jedno dobro svojstvo: inherentno je ugra-



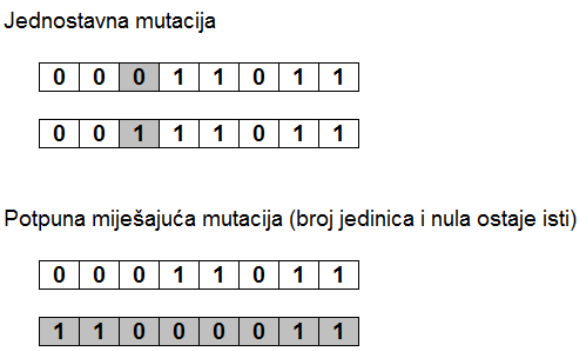

Slika 11.3: Operatori križanja u slučaju binarnog prikaza

đen elitizam. Dakle, da bi se očuvale dvije najbolje jedinke u populaciji nije potrebno pisati nikakvog dodatnog programskog koda jer je naprosto vjerojatnost eliminacije dvije najbolje jedinke jednaka nuli.

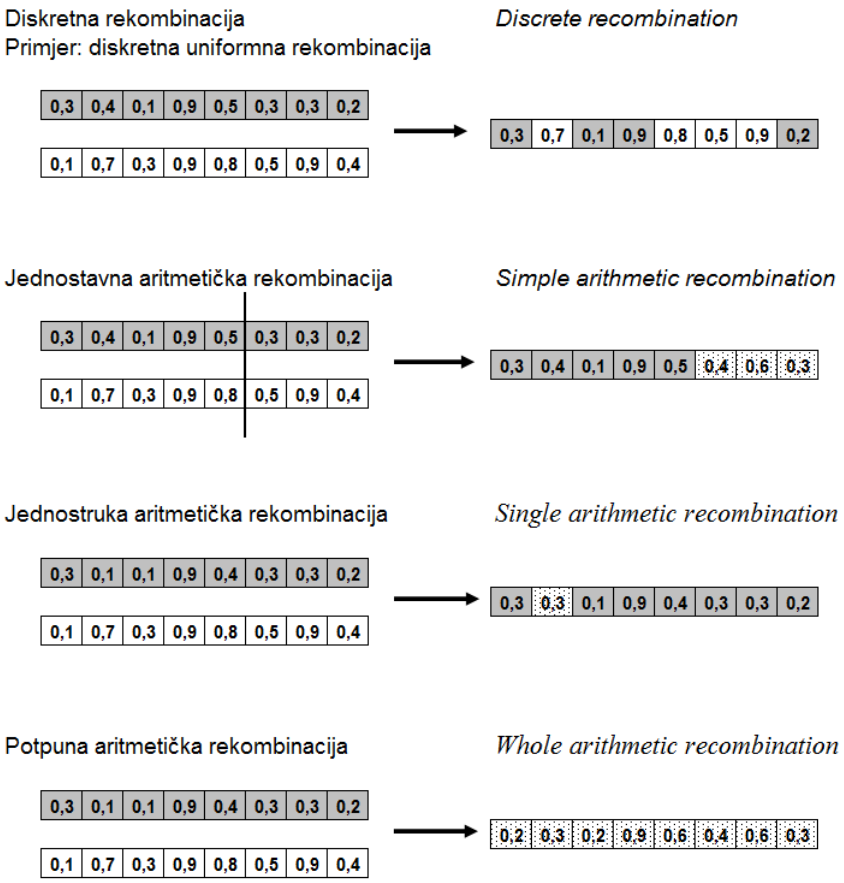
## 11.5 Reprodukcija

Reprodukcija je postupak stvaranja nove populacije nakon selekcije. Prema tome, reprodukcija kod genetskog algoritma obuhvaća križanje i mutaciju. Pod pojmom reprodukcija se nerjetko u literaturi podrazumijeva samo križanje pa treba pripaziti na što se taj pojam odnosi. U ovoj knjizi ćemo pod pojmom reprodukcije podrazumijevati i križanje i mutaciju. U pravilu križanje se obavlja prije mutacije i mutiraju se jedinke dobivene križanjem. Operatori križanja i mutacije ovise o vrsti prikaza rješenja. Tako su primjerice za binarni prikaz pogodni operatori križanja s jednom ili više točaka prekida i uniformno križanje (slika 11.3) te operatori mutacije jednostavna mutacija i potpuna miješajuća mutacija (slika 11.4).

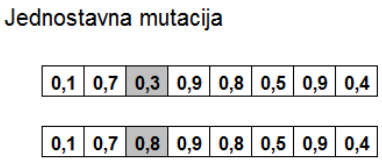
Ukoliko se primjerice koriste polje realnih brojeva za prikaz rješenja, prikladni operatori križanja su: diskretna rekombinacija, jednostavna i jednostruka aritmetička rekombinacija (slika 11.5) a prikladni operator mutacije je jednostavna mutacija (slika 11.6) slično kao i u slučaju binarnog prikaza.




Slika 11.4: Operatori mutacije prikladni za binarni prikaz




Slika 11.5: Operatori križanja prikladni za prikaz poljem realnih brojeva




Slika 11.6: Operatori mutacije prikladni za prikaz poljem realnih brojeva

Prikaz permutiranim nizom cijelih brojeva zahtjeva svoj skup specifičnih operatora križanja i mutacije. Tako su, primjerice, prikladni u ovom slučaju sljedeći operatori križanja: djelomično mapirano križanje, križanje u poretku i kružno križanje (slika 11.7). Prikladni operatori mutacije su mutacije zamjenom, ubacivanjem, premetanjem, obrtanjem i posmakom (slika 11.8).

## 11.6 Teorem sheme i hipoteza blokova

### 11.6.1 Pretpostavke

Teorijske osnove genetskog algoritma (teorem sheme i hipoteza blokova) odnose se na genetski algoritam s binarnim prikazom.

Da bi vrijedili teorem sheme i hipoteza blokova, potrebno je zadovoljiti sljedeće pretpostavke:

- populacija je neograničena,
- funkcija dobrote vjerno odražava problem i
- geni u kromosomu su međusobno nezavisni.

### 11.6.2 Teorem sheme

Shema je uzorak ili skup rješenja koja su međusobno slična. Sličnost se odnosi na jednakost gena na pojedinim mjestima kromosoma. Ostali geni mogu biti ili isti ili različiti i označavaju se zvjezdicom: "`*`". Primjerice, shema `*101*1` predstavlja skup od četiri rješenja:

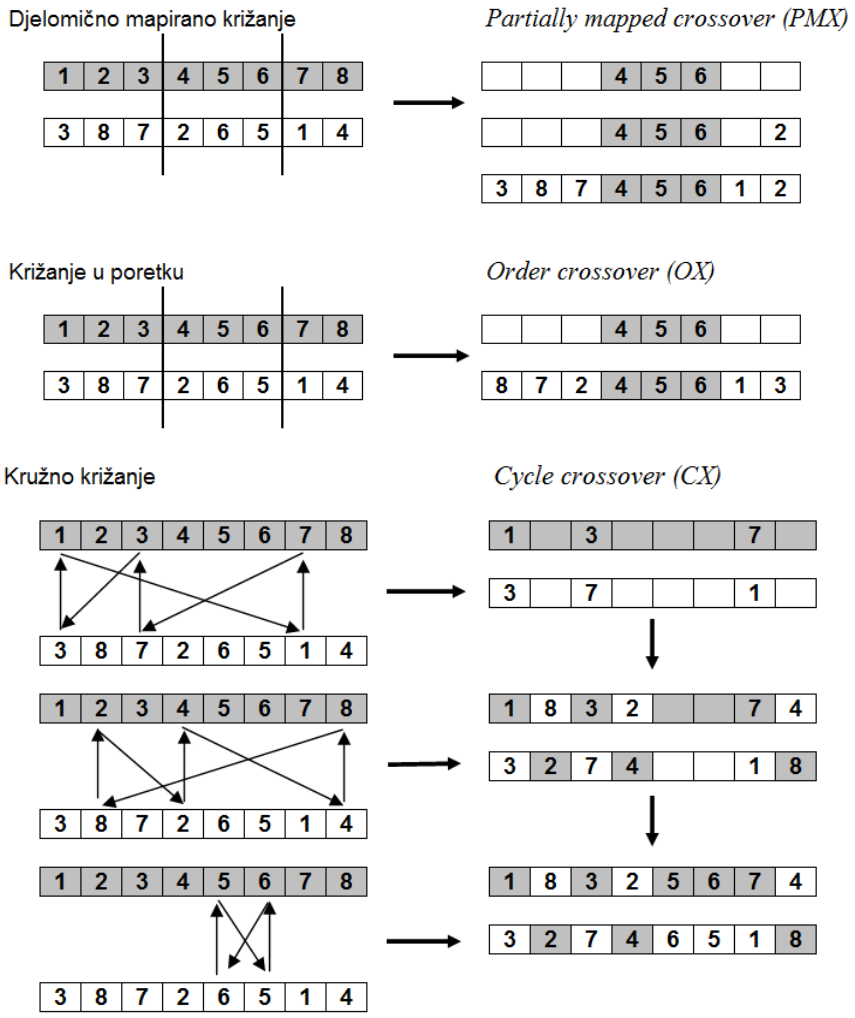
- `010101`,
- `010111`,
- `110101`,
- `110111`.

Zvjezdica "`*`" označava "bilo što" ili "bilo koji znak", a u ovom slučaju to je `0` ili `1`.

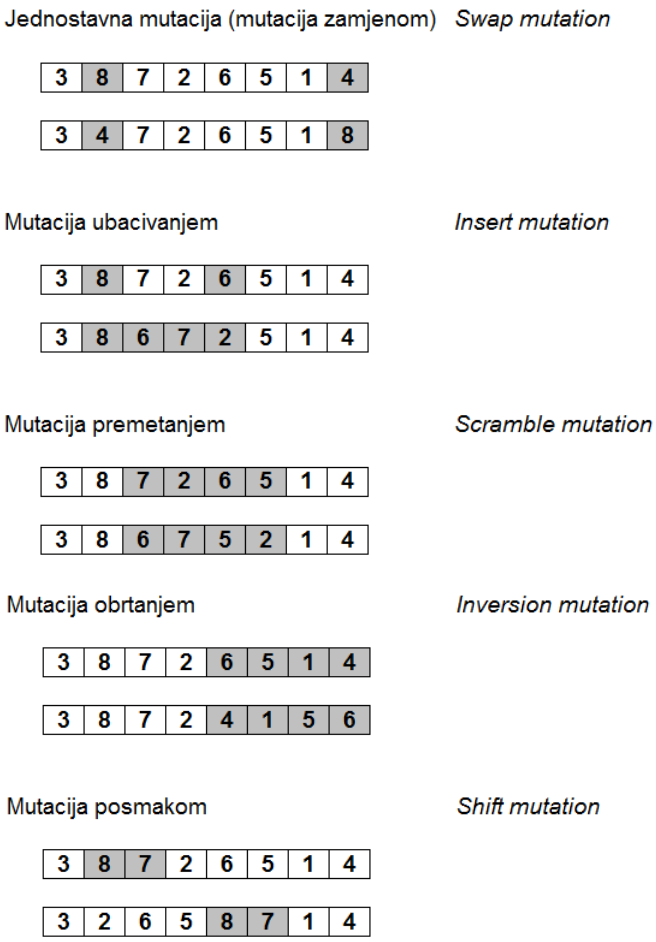
Neka je kromosom duljine $n$, a shema neka sadrži $r$ znakova "`*`". Takva shema predstavlja $2r$ rješenja. S druge strane, jedno rješenje može biti predstavljeno s $2n$ shema. Ukupan broj shema kod binarnog prikaza rješenja iznosi $3n$, jer jedan znak sheme može biti "`*`", "`0`" ili "`1`".

**Red sheme** $S$ (oznaka $o(S)$) je broj nepromjenjivih (fiksnih) gena, odnosno broj svih onih znakova sheme S koji nisu "`*`":

$$o(s) = n - r. \tag{11.5}$$




Slika 11.7: Operatori križanja za prikaz permutiranim nizom cijelih brojeva




Slika 11.8: Operatori mutacije za prikaz permutiranim nizom cijelih brojeva

Na primjer, *o(*`*1*001`*)=4* i o*(*`******11******`*)=2*.

**Definirana dužina sheme** ili definirajući razmak (oznaka $\delta(S)$) je udaljenost između prve i zadnje nepromjenjive znamenke. Na primjer, *δ(*`*01110011*`*)=9-2=7*, *δ(*`10**0`*)=5-1=4* i *δ(*`**1**`*)=3-3=0*.

**Definicija 49** (Teorem sheme)**.** *Broj jedinki koje sadrže shemu niskog reda, kratke definirane dužine i iznadprosječne dobrote raste eksponencijalno.*

Formalni opis teorema sheme prikazan je sljedećom nejednakošću:

$$N(S,t+1) \geq N(S,t)\,\frac{\overline{D_S}}{\overline{D}}\left[1-\frac{\delta(S)}{n-1}p_c - o(S)\,p_m\right] \tag{11.6}$$

gdje su:

| | |
|---|---|
| $N(S,t+1)$ | očekivani broj jedinki koje su podskup sheme $S$ u generaciji $t$ |
| $\overline{D_S}$ | prosječna dobrota sheme |
| $\overline{D}$ | prosječna dobrota populacije |
| $\delta(S)$ | definirana dužina sheme $S$ |
| $o(S)$ | red sheme $S$ |
| $p_c$ | vjerojatnost križanja |
| $p_m$ | vjerojatnost mutacije |
| $n$ | dužina kromosoma |

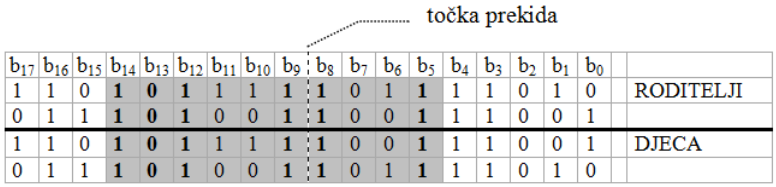
Vjerojatnost gubitka sheme zbog operatora križanja je proporcionalna vjerojatnosti križanja i definiranoj dužini sheme, a obrnuto je proporcionalna broju mogućih prekidnih točaka ($n$-1) i iznosi: $p_{gc} = \frac{\delta(S)}{n-1} \cdot p_c$.

Točnije, vjerojatnost gubitka sheme zbog operatora križanja je manja ili jednaka $p_{gc}$, jer postoji mogućnost da oba roditelja sadrže shemu $S$. Slučajnim odabirom točke prekida shema $S$ ostaje sačuvana, jer križanjem nastaje ista shema iz istih dijelova koje sadrže roditelji. Na primjer, promotrimo shemu `101**11**1` koju sadrže oba roditelja. Neka je točka prekida između dvije jedinice unutar sheme. Nakon križanja shema je ostala sačuvana (slika 11.9).

Svako križanje, osim miješajućeg križanja, (npr. uniformno križanje ili križanje s više točaka prekida) čuva shemu ako je sadrže oba roditelja.

Vjerojatnost gubitka sheme zbog mutacije je proporcionalna vjerojatnosti mutacije i redu sheme: $p_{gm} = o(S)\,p_m$. Treba spomenuti da u izrazu 11.6 u uglatoj zagradi nedostaje četvrti član koji se zanemaruje. Ispravno, spomenuti izraz glasi:

$$N(S,t+1) \geq N(S,t)\,\frac{\overline{D_S}}{\overline{D}}\left[1-\frac{\delta(S)}{n-1}p_c - o(S)\,p_m + \frac{\delta(S)}{n-1}o(S)\,p_c p_m\right] \tag{11.7}$$




| $b_{17}$ | $b_{16}$ | $b_{15}$ | $b_{14}$ | $b_{13}$ | $b_{12}$ | $b_{11}$ | $b_{10}$ | $b_9$ | $b_8$ | $b_7$ | $b_6$ | $b_5$ | $b_4$ | $b_3$ | $b_2$ | $b_1$ | $b_0$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | **1** | **0** | **1** | 1 | 1 | **1** | **1** | 0 | 1 | **1** | 1 | 1 | 0 | 1 | 0 | RODITELJI |
| 0 | 1 | 1 | **1** | **0** | **1** | 0 | 0 | **1** | **1** | 0 | 0 | **1** | 1 | 1 | 0 | 0 | 1 | |
| 1 | 1 | 0 | **1** | **0** | **1** | 1 | 1 | **1** | **1** | 0 | 0 | **1** | 1 | 1 | 0 | 0 | 1 | DJECA |
| 0 | 1 | 1 | **1** | **0** | **1** | 0 | 0 | **1** | **1** | 0 | 1 | **1** | 1 | 1 | 0 | 1 | 0 | |

Slika 11.9: Shema je sačuvana nakon križanja s jednom točkom prekida ako oba roditelja sadrže istu shemu

### 11.6.3 Hipoteza građevnih blokova

Rad genetskog algoritma može se predstaviti kao natjecanje između shema gdje kratke sheme niskog reda s nadprosječnom dobrotom pobjeđuju. Sastavljajući i nizajući takve sheme algoritam dolazi do rješenja. Kratke sheme niskog reda s iznadprosječnom dobrotom nazivaju se građevni blokovi.

**Definicija 50** (Hipoteza građevnih blokova)**.** *Genetski algoritam pretražuje prostor rješenja nizajući sheme niskog reda, kratke definirane dužine i iznadprosječne dobrote, zvane građevni blokovi.*

U slučaju da su geni u kromosomu međusobno zavisni, hipoteza građevnih blokova će i tada vrijediti ako su ti zavisni geni *blizu*.

S druge strane, građevni blokovi mogu navesti genetski algoritam na krivo rješenje, tj. da konvergira lokalnom optimumu. Ta se pojava naziva *decepcija* , a funkcija koja izaziva tu pojavu *decepcijska* ili *varajuća* funkcija.

Najjednostavniji primjer decepcijske funkcije je tzv. minimalni decepcijski problem. Dvobitna funkcija koja izaziva minimalni decepcijski problem je određena nejednakostima:

$f(11)>f(00)$,
$f(11)>f(01)$,
$f(11)>f(10)$,
$f(*0)>f(*1)$,
$f(0*)>f(1*)$.

$0*$ i $*0$ su kratke sheme niskog reda i iznadprosječne dobrote: $f(*0)>f(*1)$ i $f(0*)>f(1*)$, ali ne sadrže niz $11$ za koji se postiže optimum.

Potpun decepcijski problem je takav problem gdje sve kratke sheme niskog reda koje sadrži optimalno rješenje imaju ispodprosječnu dobrotu u odnosu na ostale sheme s istim nepromjenjivim genima.

Primjer potpunog decepcijskog problema. Neka se kromosom sastoji od niza jedinica i nula duljine $b$. Funkcija $f$ definirana nad kromosomom

vraća broj jedinica, a ako su sve nule vraća vrijednost *2b*. Globalni optimum se postiže upravo s kromosomom koji sadrži sve nule i niti jednu jedinicu: *f(0000..0000)=2b*, a lokalni optimum je kromosom sa svim jedinicama: *f(1111....1111)=b*. Svaka kratka shema niskog reda koju sadrži optimalno rješenje (npr. *00*0* ili *0*0*) ima ispodprosječnu dobrotu u odnosu na ostale sheme s istim fiksiranim genima.

## 11.7 Podešavanje parametara algoritma

### 11.7.1 Parametri genetskog algoritma

Djelotvoran evolucijski algoritam ima uravnoteženo slučajno i usmjereno pretraživanje prostora rješenja i uravnoteženu selekciju roditelja i selekciju jedinki za novu populaciju. Tu ravnotežu je moguće postići podešavanjem parametara algoritma.

Osnovni parametri genetskog algoritma su: veličina populacije ($N{=}VEL\_POP$), duljina kromosoma ($b$), vjerojatnost mutacije ($pm$), vjerojatnost križanja ($pc$), selekcijski pritisak i broj iteracija ($I$). U ovisnosti o modelu genetskog algoritma i vrsti selekcije koja se koristi, ovisi i broj parametara genetskog algoritma. Primjerice, genetski algoritam s eliminacijskom selekcijom umjesto parametra vjerojatnost križanja ima parametar generacijski jaz ili mortalitet ($M$). Drugi primjer je distribuirani genetski algoritam koji ima, osim osnovnih, čak sedam dodatnih parametara koji utječu na djelotvornost paralelnog genetskog algoritma.

Veličina populacije utječe na kvalitetu rješenja i na vrijeme trajanja genetskog algoritma. Ako se koristi $k$-turnirska selekcija, veličina populacije izravno ne utječe na vrijeme trajanja optimizacije, već utječe na kvalitetu rješenja. Želi li se postići rješenje iste kvalitete s većom populacijom, genetski algoritam treba odraditi više iteracija. Za više iteracija treba, naravno, više vremena. Bez obzira na vrstu selekcije, veličina populacije znatno utječe na kvalitetu rješenja, a time izravno ili neizravno (ovisno o vrsti selekcije) na trajanje optimizacijskog postupka.

Duljina kromosoma, primjerice kod binarnog prikaza, jednaka je umnošku dimenzije problema (broja nepoznanica) i broju bitova. Broj bitova utječe na preciznost rješenja, koje je unaprijed zadano. Taj je parametar unaprijed zadan i ne treba ga podešavati. Ono što treba učiniti jest podesiti ostale parametre genetskog algoritma tako da genetski algoritam postiže u određenom vremenu dovoljno kvalitetno rješenje za unaprijed zadanu veličinu kromosoma. Kako bi genetski algoritam bio što je moguće djelotvorniji, ne treba pretjerivati s prevelikom preciznošću algoritma jer za veliku preciznost treba više bitova čime je prostor pretraživanja veći.

Genetski algoritam je naročito osjetljiv na vjerojatnost mutacije pa treba posvetiti posebnu pažnju prilikom podešavanja tog parametra. Vjerojatnost križanja ne utječe značajnije na svojstva genetskog algoritma. Štoviše, ge-

| oznaka | parametar | preporučene vrijednosti |
|---|---|---|
| $pm$ | vjerojatnost mutacije | 0,1%-3% (najčešća vrijednost je 1%) |
| $N$ | veličina populacije | 20-100 |
| $I$ | broj iteracija | $10^3 - 10^7$ |

Tablica 11.2: Preporučeni intervali vrijednosti parametara

netski algoritam koji u istom koraku obavlja i selekciju i reprodukciju nema tog parametra, jer se križanje obavlja u svakoj iteraciji.

Selekcijski pritisak se mjeri uz pomoć selekcijske razlike $s$, selekcijskog intenziteta $sI$ ili uz pomoć trajanja preuzimanja $T$. Selekcijski pritisak se kod GA s turnirskom selekcijom podešava uz pomoć parametra $k$, gdje je $k = 2, 3, \ldots, N$. Što je veći $k$, veći je i selekcijski pritisak. Veći selekcijski pritisak ima za posljedicu i bržu konvergenciju zbog čega algoritam kraće traje, ali ima i veću vjerojatnost zaglavljivanja u lokalnom optimumu. Turnirska selekcija, i s najmanje mogućom vrijednošću parametra $k$ ($k_{min} = 2$), ima veći selekcijski pritisak od proporcionalnih selekcija i rangirajućih selekcija [Bäck(1994)]. Stoga, zapravo i nema izbora prilikom odabira parametra $k$: parametar $k$ mora biti što je moguće manji. U slučaju da se u istom koraku obavlja i selekcija i reprodukcija, tri ili više jedinki trebaju sudjelovati u turnirskoj selekciji (jedna jedinka se eliminira i nadomjesti djetetom dvaju slučajno odabranih roditelja od $k - 1$ preživjelih jedinki). Dakle, želimo li u istom koraku obavljati i turnirsku selekciju i reprodukciju, minimalna vrijednost parametra $k$ je tri.

Posljednji parametar iz osnovnog skupa parametara genetskog algoritma je broj iteracija. Što je veći broj iteracija, očekuje se bolje rješenje. Trajanje izvođenja algoritma proporcionalno je broju iteracija. Za kontradiktorne zahtjeve: postići što je moguće kvalitetnije rješenje u što je moguće kraćem vremenu, treba učiniti kompromis: genetski algoritam treba obaviti dovoljan broj iteracija da postigne zadovoljavajuće rješenje.

Prema tome, parametri genetskog algoritma s 3-turnirskom eliminacijskom selekcijom koje treba odrediti su: veličina populacije, vjerojatnost mutacije i broj iteracija. Parametri se u pravilu određuju eksperimentalno. Međutim, dobro je znati preporučene intervale vrijednosti parametara kako bi se smanjio prostor pretrage u potrazi za optimalnim parametrima (tablica 11.2).

### 11.7.2 Karakteristične krivulje

Kvaliteta dobivenog rješenja za dva slična skupa parametara može biti bitno različita, jer je genetski algoritam osjetljiv na svaku promjenu parametara. Rezultati optimiranja se mogu znatno poboljšati finim podešavanjem parametara. Dakle, optimalni skupovi parametara iz literature mogu poslužiti

samo kao smjernice gdje treba tražiti optimalno skup parametara za određen genetski algoritam primijenjen na određenom optimizacijskom problemu. Jednako tako, da bi se izbjeglo dugotrajno eksperimentalno određivanje parametara, treba iskoristiti sljedeće pravilnosti u ponašanju genetskog algoritma:

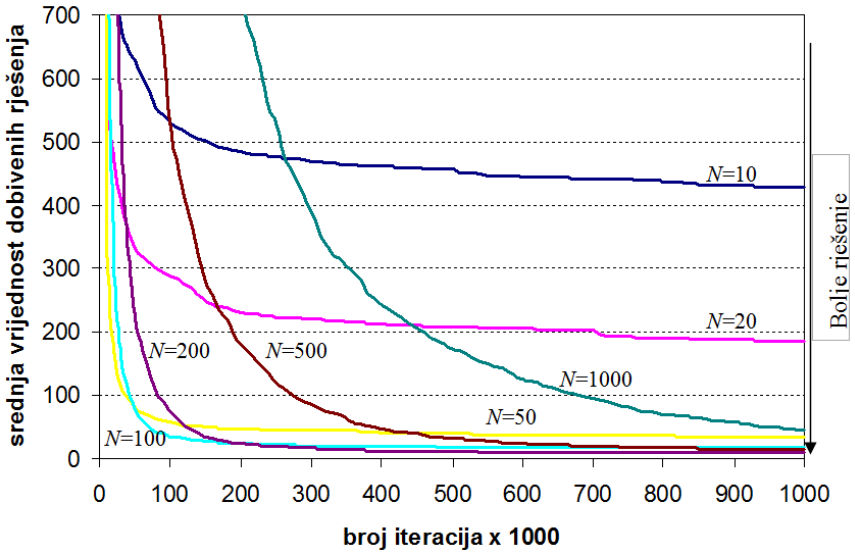
- povećavanjem broja iteracija se u pravilu postiže bolje rješenje;
- za veće populacije, treba obaviti više iteracija, a vjerojatnost mutacije treba smanjiti;
- jednako kvalitetna rješenja se mogu postići za više skupova parametara.

Poželjno je unaprijed znati broj iteracija i za poznati broj iteracija eksperimentalno odrediti veličinu populacije i vjerojatnost mutacije. Obično je unaprijed poznato raspoloživo vrijeme za optimiranje, a time i broj iteracija. Ukoliko se s tako određenim skupom parametara ne postiže zadovoljavajuće rješenje, broj iteracija se mora povećati. S druge strane, ako se s dobivenim skupom parametara postiže zadovoljavajuće rješenje, treba pokušati i s manjim brojem iteracija. Tako sve dok se ne dobije rješenje koje ne zadovoljava. Na taj način je moguće odrediti gornju i donju granicu vremena izvođenja, odnosno broja iteracija.

Za primjer osjetljivosti genetskog algoritma na vrijednosti parametara ostavren je 3-turnirski eliminacijski genetski algoritam koji se koristi za rješavanje jednog aproksimacijskog problema. Na slici 11.10 je prikazano nekoliko pokretanja evolucijskog procesa za različite vrijednosti veličina populacije. Za očekivati je da će genetski algoritam doći do boljeg rješenja ako djeluje nad većom populacijom. Međutim, za veće populacije, treba odraditi i veći broj iteracija. Kako su svi ostali parametri u ovom primjeru, osim veličine populacije, bili fiksirani, za prevelike vrijednosti veličina populacije za jednak broj iteracija se postiglo lošije rješenje.

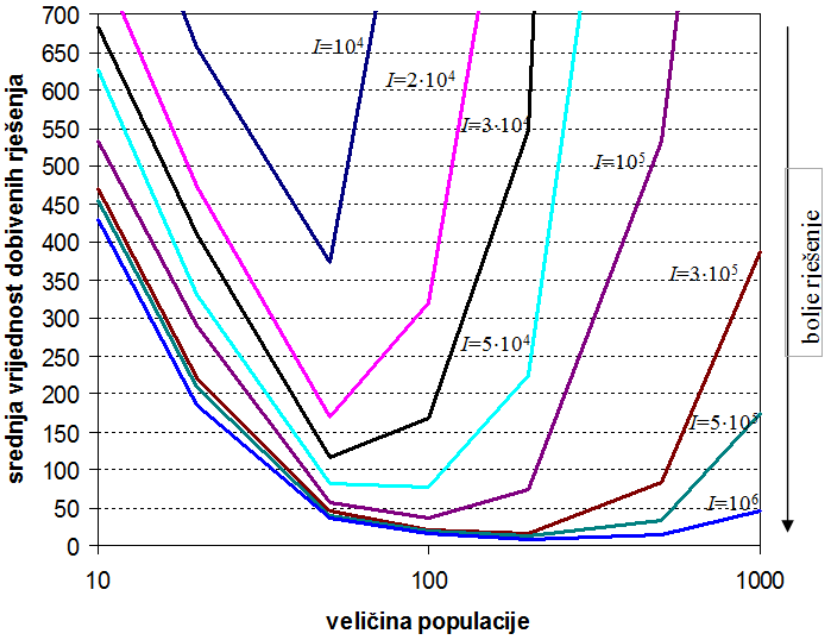
Na istom primjeru provedena su ispitivanja što se zbiva ako je unaprijed postavljen samo parametar vjeroatnost mutacije. Na slici 11.11 se dobro vidi kako veličinu populacije treba *pratiti* parametar broj iteracija: uzalud velika populacija ako je broj iteracija premali.

Na temelju iskustava stečenih na nizu eksperimenata moguće je skicirati očekivano ponašanje genetskih, odnosno općenitije, evolucijski algoritama. Na slici 11.12 prikazano je šest karakterističnih krivulja ovisnosti rješenja optimiranja o tri osnovna parametra: veličini populacije, broju iteracija i vjerojatnosti mutacije. Crtkane linije predstavljaju novu krivulju ako se promijeni neki od parametara. Strelica na slici označava smjer promjene krivulje ukoliko se poveća naznačeni parametar. Primjerice, na krivulji ovisnosti rješenja o veličini populacije, u slučaju da se traži minimum funkcije $f$ (krivulja 5 na slici 11.12), povećanjem broja iteracija krivulja se širi, a dno krivulje se pomiče dolje desno. Krivulje se odnose na genetski algoritam s ugrađenim elitizmom.

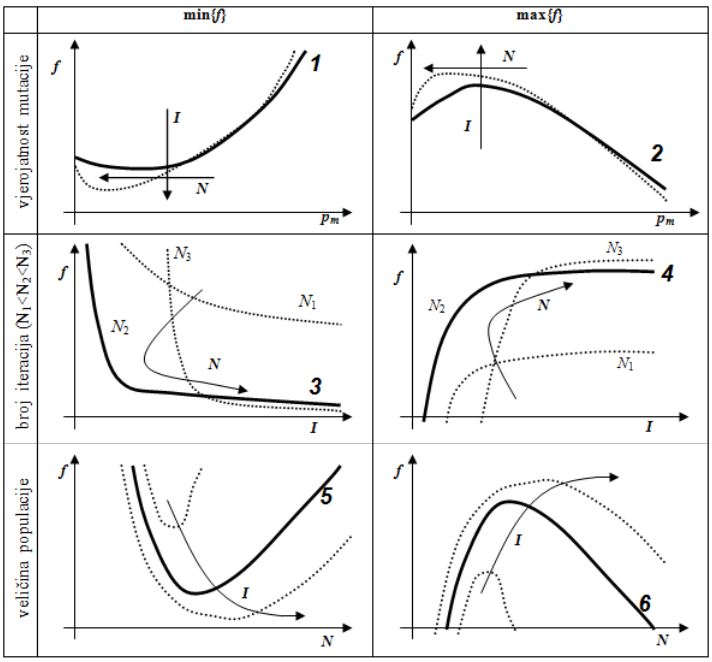



Slika 11.10: Utjecaj veličine populacije na evolucijski proces

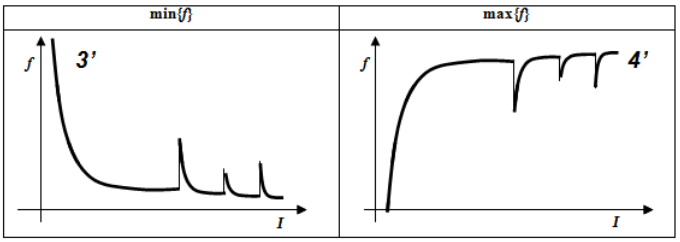
Ukoliko elitizam nije ugrađen, krivulje ovisnosti rješenja o broju iteracija imaju specifičan zupčasti oblik koji je posljedica gubljenja pronađenog najboljeg rješenja, ukoliko se ono ne zaštiti elitizmom (slika 11.13).




Slika 11.11: Utjecaj veličine populacije i broj iteracija na kvalitetu rješenja




Slika 11.12: Karakteristične krivulje ovisnosti kvalitete rješenja o tri parametra




Slika 11.13: Karakteristična krivulja rješenja u ovisnosti o broju iteracija za genetski algoritam bez elitizma

# Dio IV

# Kombiniranje pristupa mekog računarstva

# Poglavlje 12

# Kombiniranje neuronskih mreža i sustava neizrazitog upravljanja

Neuronske mreže i sustavi neizrazitog zaključivanja, svaki za sebe, imaju niz prednosti ali i niz mana. Tako primjerice, sustave neizrazitog upravljanja gradimo uporabom jednostavnih i jasnih pravila. Takvi sustavi mogu raditi s nepreciznim podatcima, mogu relativno jednostavno aproksimirati ponašanje složenih sustava, i čitavo vrijeme pri tome zadržavaju mogućnost interpretacije onoga što rade. Izgradnja ovakvih sustava tipično zahtjeva da na raspolaganju imamo eksperta za problem koji se rješava, i takav ekspert svoje domensko znanje modelira neizrazitim pravilima. Nedostatak ovih sustava pak leži u nemogućnosti učenja.

Za razliku od sustava neizrazitih upravljanja, neuronske mreže omogućavaju nam samostalno učenje iz podataka. Ne zahtjeva se prisustvo eksperta, gradi se model koji je tipično vrlo robustan, koji također može raditi s nepreciznim podatcima i koji lagano modelira visokonelinearne sustave. Praktički sve što je potrebno pripremiti je samo velika količina podataka na temelju koje će se mreža trenirati. Karakteristike oba ova pristupa sažeta su u tablici 12.1.

Tablica 12.1: Usporedba karakteristika sustava neizrazitog upravljanja te neuronskih mreža

| **Neuronske mreže** | **Neizraziti sustavi** |
|---|---|
| nije potreban matematički model o problemu koji se rješava | nije potreban matematički model o problemu koji se rješava |
| učenje temeljem podataka | zahtjeva apriorno znanje o problemu |
| različiti algoritmi učenja | nemaju mogućnost učenja |
| model *crne kutije* | jednostavna interpretacija pravila |

U najopćenitijem smislu, kada govorimo bilo o uporabi neuronskih mreža bilo o sustavima neizrazitog upravljanja, najčešće govorimo o aproksimaciji nekog nepoznatog funkcijskog ponašanja. Polazeći od te pretpostavke, u [Hayashi and Buckley(1994)] je pokazano da su oba pristupa zapravo ekvivalentna i jednako ekspresivna. Stoga je upravo prirodno razmišljati o tome kako se ovi pristupi mogu međusobno kombinirati kako bi se dobio novi pristup koji zadržava pozitivne osobine oba pristupa a poništava njihove mane.

Danas se u literaturi uobičajeno nalazi nekoliko načina na koji se neuronske mreže i sustavi neizrazitog upravljanja mogu kombinirati. Ugrubo, pristupi se mogu podijeliti u tri kategorije.

1. *Kooperativni pristup* pri čemu se neuronska mreža koristi za učenje bilo funkcija pripadnosti, bilo pravila ili oboje. Dvije su mogućnosti za ovakvu izvedbu. Jedna mogućnost jest dopustiti neuronskoj mreži da proces učenja napravi samo jednom (na početku) i potom se temeljem naučenoga gradi sustav neizrazitog upravljanja. Druga mogućnost je neuronsku mrežu učiti kontinuirano tijekom rada čitavog sustava i temeljem rezultata učenja dinamički mijenjati sustav neizrazitog zaključivanja.

2. *Konkurentni pristup* pri čemu se obrada obavlja tako da se slijedno koristi najprije neuronska mreža pa potom sustav neizrazitog zaključivanja (ili obrnuto).

3. *Hibridni pristup* pri čemu se sustav neizrazitog upravljanja direktno prikazuje u obliku neuronske mreže određene strukture čime je omogućena uporaba algoritama za učenje neuronskih mreža kako bi se naučila pravila i potrebne funkcije pripadnosti za sustav neizrazitog upravljanja. Kod ovog se pristupa, međutim, sustav neizrazitog upravljanja ne izvodi zasebno, već je neuronska mreža istovremeno i sustav neizrazitog upravljanja.

U nastavku ovog poglavlja pogledat ćemo dva načina kombiniranja neizrazitih mreža i neizrazite logike: *neizrazite neuronske mreže* (koristi se još i naziv *hibridne neuronske mreže*) te *ANFIS*.

## 12.1 Neizrazite neuronske mreže

Jedan od načina kombiniranja neizrazite logike i neuronskih mreža jest uvođenja neizrazitih koncepata na razini neurona, kako je to predloženo u [Pedrycz and Rocha(1993)].

*Hibridna neuronska mreža* je mreža kod koje su signali, težine i prijenosne funkcije klasične (engl. *crisp*). Međutim, tražimo da vrijedi:

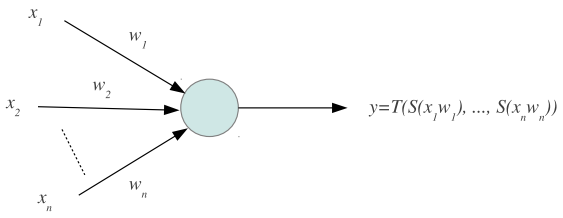
- ulazi i težine su realni brojevi iz intervala [0,1],

- ulaze i težine kombiniramo $t$-normama ili $s$-normama ili sličnim kontinuiranim operatorima,
- agregacije ulaza i težina opet kombiniramo $s$-normama ili $t$-normama te
- prijenosna funkcija može biti bilo kakva kontinuirana funkcija.

Procesni element hibridne neuronske mreže naziva se *neizraziti neuron*. Dvije su osnovne vrste neizrazitih neurona i one su opisane u nastavku.

*Neizraziti-I neuron* (slika 12.1) je neuron kod kojeg se ulazi $x_i$ i težine $w_i$ kombiniraju se odabranom $S$-normom. Izlaz je agregacija ovih kombinacija uporabom $T$-norme.

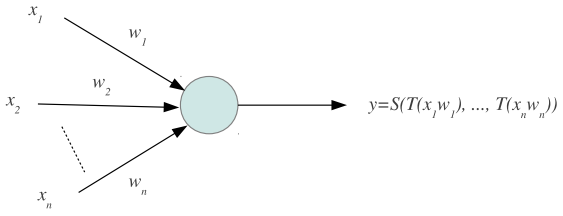
$$y = T(S(x_1, w_1), S(x_2, w_2), ...)$$




Slika 12.1: Neizraziti-I neuron

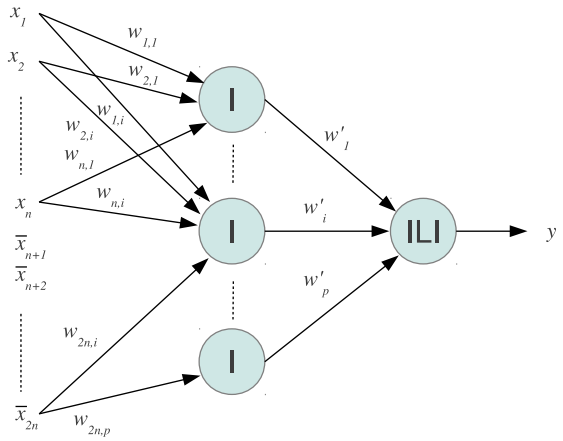
*Neizraziti-ILI neuron* (slika 12.2) je neuron kod kojeg se ulazi $x_i$ i težine $w_i$ kombiniraju se odabranom $T$-normom. Izlaz je agregacija ovih kombinacija uporabom $S$-norme.

$$y = S(T(x_1, w_1), T(x_2, w_2), ...)$$

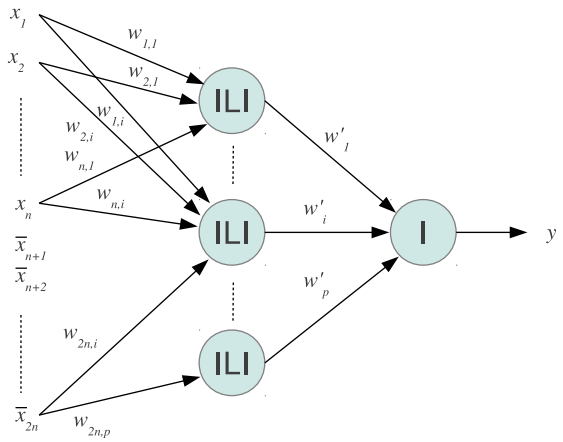



Slika 12.2: Neizraziti-ILI neuron

Uočimo da uslijed želje da model zadrži interpretabilnost na razini neizrazite logike nismo u mogućnosti koristiti negativne težine. Stoga se u ovakvom modelu taj problem rješava proširivanjem ulaznog vektora $\vec{x} = (x_1, x_2, \ldots)$ s komplementarnim ulazima tako da se dobiva novi ulazni vektor $\vec{x} = (x_1, x_2, \ldots, \bar{x}_1, \bar{x}_2, \ldots)$. Komplementi se pri tome računaju bilo na




Slika 12.3: Primjer neizrazite neuronske mreže tipa ILI-od-I




Slika 12.4: Primjer neizrazite neuronske mreže tipa I-od-ILI

način kako je to definirao Zadeh: $\bar{x}_i = 1 - x_i$ bilo uporabom nekog generaliziranog operatora komplementa.

Uporabom ovako definiranih neizrazitih neurona sada se mogu graditi kompleksnije višeslojne neuronske mreže; primjeri takvih dviju mreža prikazani su na slikama 12.4 i 12.3. Neizrazita neuronska mreža prikazana na slici 12.3 sastoji se od skrivenog sloja sastavljenog od $p$ neizrazitih-I neurona te izlaznog sloja u kojem se nalazi neizraziti-ILI neuron. Neizrazita neuronska mreža prikazana na slici 12.4 sastoji se od skrivenog sloja sastavljenog od $p$ neizrazitih-ILI neurona te izlaznog sloja u kojem se nalazi neizraziti-I neuron. Restrikcijom ulaznih vrijednosti na $\{0, 1\}$ prva će se mreža pretvoriti u sustav koji računa sumu minterma a druga u sustav koji računa produkt maksterma poznatih iz Booleove logike.

Ovako definiranu neizrazitu neuronsku mrežu moguće je učiti odgovara-

jućim algoritmima kako bi se uz unaprijed zadan niz parova za učenje $(\vec{X}_i, y_i)$ pronašao optimalni skup težina uz koje će mreža raditi najmanju pogrešku. Prilikom učenja potrebno je pripaziti da težine ne smiju izaći izvan intervala $[0, 1]$.

Više o ovoj vrsti neurona i mreža moguće je pronaći u [Pedrycz and Rocha(1993), Pedrycz(1993), Hirota and Pedrycz(1994)].

## 12.2 Sustav ANFIS

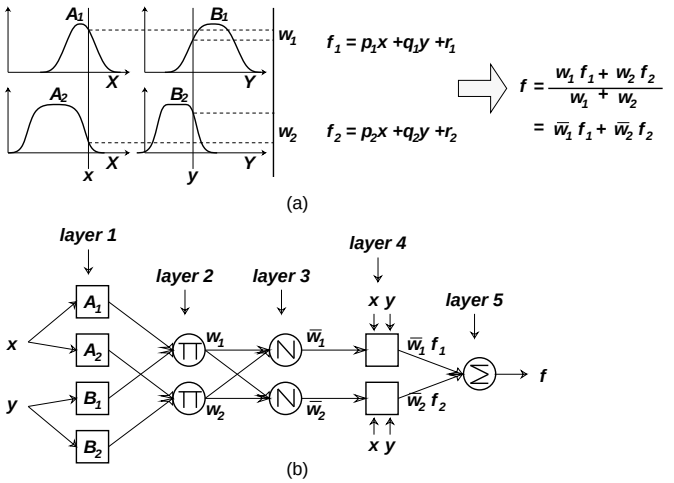
Sustav ANFIS (engl. *Adaptive Neuro-Fuzzy Inference System*) izvorno opisan u [Jang and Sun(1995)] pripada pod hibridni pristup kombiniranja neuronskih mreža i sustava neizrazitog zaključivanja. Kod ovog sustava neuronska mreža je istovremeno i sustav neizrazitog zaključivanja.

Da bismo pojasnili građu sustava ANFIS, prisjetimo se najprije tipičnih oblika neizrazitih pravila koja se koriste u sustavima neizrazitog upravljanja. Primjerice,
`Ako` *tlak* `je` *visok*, `tada` *volumen* `je` *mali*.
U ovom primjeru *tlak* i *volumen* su jezične varijable a *visok* i *mali* su jezični izrazi (tj. neizraziti skupovi) koje je moguće pridijeliti tim jezičnim varijablama. Jezičnim izrazima pridijeljene su odgovarajuće funkcije pripadnosti. Uporabom ovakvih pravila moguće je izgraditi sustave koji koriste zaključivanje tipa 1 (Tsukamotovo zaključivanje) ili zaključivanje tipa 2 (Mamdanijevo zaključivanje).

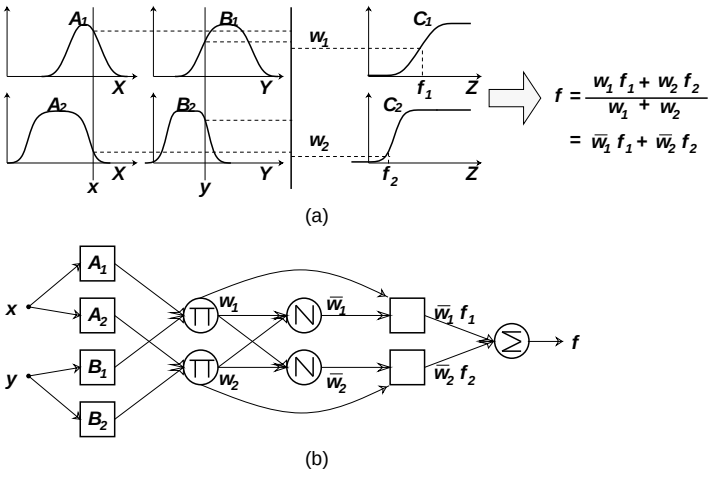
Drugi primjer neizrazitih pravila su pravila oblika:
`Ako` *brzina* `je` *velika*, `tada` *sila* `=` $k \cdot$ *brzina*$^2$.
Za razliku od prethodnog oblika pravila, ovdje se kao konsekvens nalazi izraz (formula) koja temeljem trenutnih jasnih (engl. *crisp*) ulaza određuje vrijednost izlaza koja je, formalno govoreći, singleton neizraziti skup (tj. jednočlani neizraziti skup). Ovakva pravila vode nas na zaključivanje tipa 3 (TSK zaključivanje).

Za svaki od tri uobičajena načina izvedbe sustava neizrazitog upravljanja definiran je po jedan oblik sustava ANFIS s prikladnom neuronskom mrežom. Sustav ANFIS prikazan na slici 12.5 prikazuje sustav koji implementira zaključivanje Takagi-Sugeno-Kang (TSK); prisjetimo se, kod tog tipa konsekventi su klasične funkcije koje su uobičajeno definirane kao linearne kombinacije ulaznih varijabli a dekodiranje neizrazitosti obavlja se uporabom težinske sume gdje težine odgovaraju jakostima paljenja antecedenata. Na slici 12.6 prikazano je zaključivanje kako je definirao Tsukamoto – konsekvent je neizraziti skup definiran monotonom funkcijom pripadnosti. Konačno, slika 12.7 prikazuje sustav ANFIS koji obavlja klasično Mamdanijevo zaključivanje.

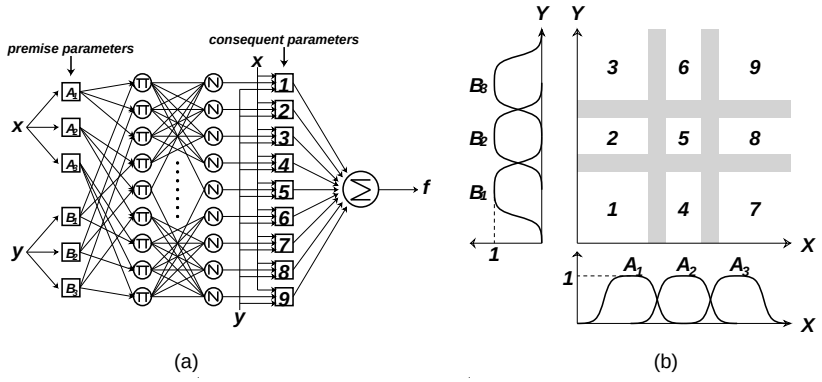
Svaki od ova tri oblika zaključivanja kao podlogu imaju općenitu strukturu sustava neizrazitog zaključivanja koja je prikazana na slici 12.8. Takav




Slika 12.5: ANFIS mreža koja ostvaruje neizraziti sustav (zaključivanje tipa 3)




Slika 12.6: ANFIS mreža koja ostvaruje neizraziti sustav (zaključivanje tipa 1)




Slika 12.7: ANFIS mreža koja ostvaruje neizraziti sustav (zaključivanje tipa 2)

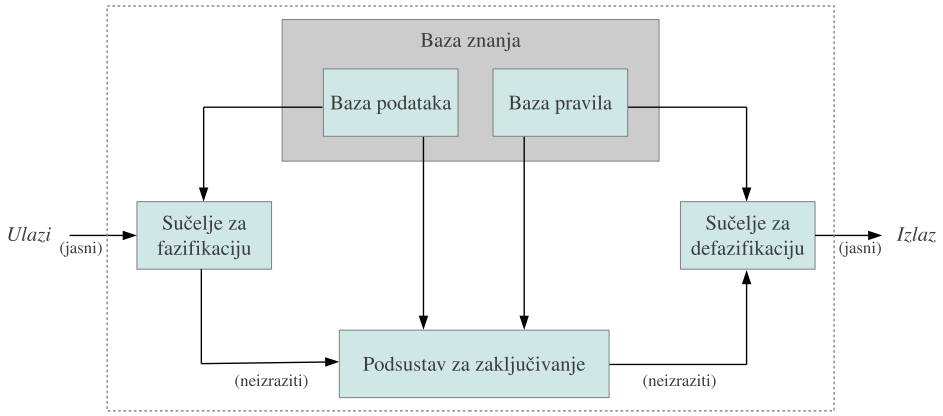
sustav sastoji se od sljedećih dijelova:

- *baza pravila* koja se sastoji od niza neizrazitih *ako-onda* pravila,
- *baza podataka* koja sadrži definicije jezičnih varijabli i jezičnih izraza koji se koriste u neizrazitim *ako-onda* pravilima (sadrži primjerice točne definicije funkcija pripadnosti koje su pridružene svakom jezičnom izrazu),
- *podsustav za zaključivanje* koji temeljem dobivenih neizrazitih ulaza, neizrazitih *ako-onda* pravila te definicija jezičnih varijabli obavlja zaključivanje i generira neizraziti zaključak,
- *sučelje za fazifikaciju* koje jasne ulazne podatke preslikava u neizrazite ulaze te
- *sučelje za defazifikaciju* koje neizraziti zaključak pretvara u jasni izlaz.

Kod sustava ANFIS, cjelokupna prikazana struktura implementirana je u obliku neuronske mreže.

### 12.2.1 Radni primjer

Primjer učenja sustava ANFIS prikazat ćemo na pojednostavljenom modelu koji približno odgovara sustavu tipa 3. Potrebno je naučiti sustav koji obavlje preslikavanje $x \to y$ kojim se modelira neko nepoznato funkcijsko preslikavanje. Pri tome ćemo za potrebe izvoda algoritma promatrati *pojednostavljen* sustav kod kojeg su konsekvensi **konstante**, a premise ispituju samo jednu varijablu. Primjer takvih pravila je:
$\mathbb{R}_1$ : Ako $x$ je $A_1$ tada $y$ je $z_1$
$\mathbb{R}_2$ : Ako $x$ je $A_2$ tada $y$ je $z_2$
...




Slika 12.8: Općenita građa sustava neizrazitog zaključivanja

$\mathbb{R}_m$ : Ako $x$ je $A_m$ tada $y$ je $z_m$

Pretpostavit ćemo također da su funkcije pripadnosti neizrazitih skupova $A_1 \ldots A_m$ modelirane sigmoidalnim funkcijama oblika:

$$A_i(x) \equiv \mu_{A_i}(x) = \frac{1}{1 + e^{b_i(x-a_i)}}$$

gdje su $a_i$ i $b_i$ parametri. Konsekventi pravila su $z_1, \ldots, z_m$ – konstante (realni brojevi). Algoritam učenja ovakvog sustava ima zadaću pronaći optimalne vrijednosti svih parametara sustava: $a_i$, $b_i$ te $z_i$.

Neka nam je dostupan niz od $N$ uzoraka za učenje: $\{(x_1,y_1), \ldots, (x_N,y_N)\}$. Potrebno je izvesti *online* verziju algoritma koju ćemo temeljiti na gradijentnom spustu. Neka je izlaz sustava neizrazitog upravljanja za predočeni $k$-ti uzorak označen s $o_k$. Funkciju pogreške za taj uzorak tada možemo definirati na sljedeći način:

$$E_k = \frac{1}{2}(y_k - o_k)^2$$

Ako parametre ažuriramo u skladu s algoritmom gradijentnog spusta, za ažuriranje nekog proizvoljnog parametra $\psi$ pri koristi se izraz:

$$\psi(t+1) = \psi(t) - \eta \frac{\partial E_k}{\partial \psi}$$

Naš je zadatak stoga utvrditi parcijalne derivacije funkcije $E_k$ po svim parametrima ($a_i$, $b_i$, $z_i$) o kojima ovisi rad promatranog sustava ANFIS.

### 12.2.2 Izvod pravila za ažuriranje parametara $z_i$

Izlaz sustava neizrazitog upravljanja definiran je kao težinska suma:

$$o = \frac{\sum_{i=1}^{m} \alpha_i z_i}{\sum_{i=1}^{m} \alpha_i}$$

pri čemu je $\alpha_i$ jakost paljenja $i$-tog pravila i u promatranom slučaju ona je jednaka:

$$\alpha_i = A_i(x) = \frac{1}{1 + e^{b_i(x-a_i)}}.$$

Za potrebe izračuna parcijalne derivacije funkcije pogreške po parametru $z_i$ poslužit ćemo se pravilom ulančavanja:

$$\frac{\partial E_k}{\partial z_i} = \frac{\partial E_k}{\partial o_k}\frac{\partial o_k}{\partial z_i}.$$

Svaku od pojednih parcijalnih derivacija sada je jednostavno izračunati:

$$\begin{aligned}\frac{\partial E_k}{\partial o_k} &= \frac{\partial}{\partial z_i}\left(\frac{1}{2}(y_k - o_k)^2\right)\\ &= -(y_k - o_k)\end{aligned}$$

$$\begin{aligned}\frac{\partial o_k}{\partial z_i} &= \frac{\partial}{\partial z_i}\left(\frac{\sum_{j=1}^{m}\alpha_j z_j}{\sum_{j=1}^{m}\alpha_j}\right)\\ &= \frac{\alpha_i}{\sum_{j=1}^{m}\alpha_j}\end{aligned}$$

Uvrštavanjem izvedenoga u početni izraz dolazi se do konačnog izraza za traženu parcijalnu derivaciju:

$$\frac{\partial E_k}{\partial z_i} = -(y_k - o_k)\frac{\alpha_i}{\sum_{j=1}^{m}\alpha_j}.$$

Temeljem ovog izraza imamo i konačni izraz za ažiriranje parametara $z_i$:

$$z_i(t+1) = z_i(t) + \eta(y_k - o_k)\frac{\alpha_i}{\sum_{j=1}^{m}\alpha_j}.$$

### 12.2.3 Izvod pravila za ažuriranje parametara $a_i$

Postupit ćemo na isti način kao i u prethodnom slučaju. Koristimo pravilo ulančavanja:

$$\frac{\partial E_k}{\partial a_i} = \frac{\partial E_k}{\partial o_k}\frac{\partial o_k}{\partial \alpha_i}\frac{\partial \alpha_i}{\partial a_i}.$$

Svaka od komponenti tada je:

$$\frac{\partial E_k}{\partial o_k} = -(y_k - o_k)$$

$$\frac{\partial o_k}{\partial \alpha_i} = \frac{\sum_{j=1, j\neq i}^{m}\alpha_j(z_i - z_j)}{\left(\sum_{j=1}^{m}\alpha_j\right)^2}$$

$$\frac{\partial \alpha_i}{\partial a_i} = b_i \alpha_i (1 - \alpha_i)$$

Uvrštavanjem izvedenoga u početni izraz dolazi se do konačnog izraza za traženu parcijalnu derivaciju:

$$\frac{\partial E_k}{\partial a_i} = -(y_k - o_k) \frac{\sum_{j=1, j \neq i}^{m} \alpha_j (z_i - z_j)}{\left( \sum_{j=1}^{m} \alpha_j \right)^2} b_i \alpha_i (1 - \alpha_i).$$

Temeljem ovog izraza imamo i konačni izraz za ažiriranje parametara $a_i$:

$$a_i(t+1) = a_i(t) + \eta (y_k - o_k) \frac{\sum_{j=1, j \neq i}^{m} \alpha_j (z_i - z_j)}{\left( \sum_{j=1}^{m} \alpha_j \right)^2} b_i \alpha_i (1 - \alpha_i).$$

### 12.2.4 Izvod pravila za ažuriranje parametara $b_i$

I po treći puta iskoristit ćemo pravilo o ulančavanju:

$$\frac{\partial E_k}{\partial b_i} = \frac{\partial E_k}{\partial o_k} \frac{\partial o_k}{\partial \alpha_i} \frac{\partial \alpha_i}{\partial b_i}.$$

Svaka od komponenti tada je:

$$\frac{\partial E_k}{\partial o_k} = -(y_k - o_k)$$

$$\frac{\partial o_k}{\partial \alpha_i} = \frac{\sum_{j=1, j \neq i}^{m} \alpha_j (z_i - z_j)}{\left( \sum_{j=1}^{m} \alpha_j \right)^2}$$

$$\frac{\partial \alpha_i}{\partial b_i} = -(x - a_i) \alpha_i (1 - \alpha_i)$$

Uvrštavanjem izvedenoga u početni izraz dolazi se do konačnog izraza za traženu parcijalnu derivaciju:

$$\frac{\partial E_k}{\partial a_i} = (y_k - o_k) \frac{\sum_{j=1, j \neq i}^{m} \alpha_j (z_i - z_j)}{\left( \sum_{j=1}^{m} \alpha_j \right)^2} (x - a_i) \alpha_i (1 - \alpha_i).$$

Temeljem ovog izraza imamo i konačni izraz za ažiriranje parametara $b_i$:

$$b_i(t+1) = b_i(t) - \eta (y_k - o_k) \frac{\sum_{j=1, j \neq i}^{m} \alpha_j (z_i - z_j)}{\left( \sum_{j=1}^{m} \alpha_j \right)^2} (x - a_i) \alpha_i (1 - \alpha_i).$$

### 12.2.5 Konačni algoritam

Čitav algoritam učenja sada možemo sažeti u tri puta po $m$ izraza koje primjenjujemo iterativno.

$$z_i(t+1) = z_i(t) + \eta(y_k - o_k)\frac{\alpha_i}{\sum_{j=1}^{m}\alpha_j}$$

$$a_i(t+1) = a_i(t) + \eta(y_k - o_k)\frac{\sum_{j=1,j\neq i}^{m}\alpha_j(z_i - z_j)}{\left(\sum_{j=1}^{m}\alpha_j\right)^2}b_i\alpha_i(1-\alpha_i)$$

$$b_i(t+1) = b_i(t) - \eta(y_k - o_k)\frac{\sum_{j=1,j\neq i}^{m}\alpha_j(z_i - z_j)}{\left(\sum_{j=1}^{m}\alpha_j\right)^2}(x - a_i)\alpha_i(1-\alpha_i)$$

### 12.2.6 Općenita verzija ANFIS-a

Za potrebe izvedbe algortima učenja napravili odlučili smo se analizirati jednu pojednostavljenu verziju sustava ANFIS. Zbog kompletnosti ovog pregleda potrebno je spomenuti varijacije u odnosu na opisani sustav koje se također koriste.

Umjesto sigmoidalnih funkcija pripadnosti često se za modeliranje funkcija pripadnosti koriste i različite "zvonolike" funkcije poput:

- $$\mu_{A_i}(x) = \frac{1}{1 + \left[\left(\frac{x-c_i}{a_i}\right)^2\right]^{b_i}} ili$$
- $$\mu_{A_i}(x) = e^{-\left[\left(\frac{x-c_i}{a_i}\right)^2\right]^{b_i}}.$$

U konsekvens dijelu najčešće se koriste funkcije koje su linearne kombinacije ulaza; npr. ako imamo ulaze $x$ i $y$, funkcije su oblika:

$$f_i = p_i x + q_i y + r_i$$

gdje su $p_i$, $q_i$ i $r_i$ parametri koje je moguće podešavati algoritmom učenja.

I konačno, umjesto antecedenta koji ispituje samo jednu varijablu, uobičajeno je da antecedent ispituje sve varijable koje su ulaz u sustav.

### 12.2.7 Poboljšanje algoritma učenja

Izvedeni algoritam učenja, odnosno opisani postupak za njegov izvod može se modificirati na način koji će rezultirati algoritmom koji brže konvergira; primjer je detaljno opisan u [Jang and Sun(1995)]. Ovdje ćemo se samo ukratko osvrnuti na generalnu ideju.

Algoritam gradijentnog spusta koji smo koristili za izvod pravila učenja jednostavan je algoritam koji pretpostavlja relativno malo o funkciji na koju se primjenjuje i stoga je relativno popularan. Međutim, upravo zbog toga što ne koristi sve potencijalno dostupne informacije, postupak učenja je relativno spor.

Pretpostavimo, za potrebe ovog razmatranja, da je funkcija koju je potrebno minimizirati funkcija nad parametrima $\xi_1$, $\xi_2$, ..., $\xi_r$, $\xi_{r+1}$, ..., $\xi_s$, pri čemu o prvih $r$ parametara funkcija ovisi linearno, a o preostalih $s-r$ parametara nelinearno. U tom slučaju tu funkciju možemo zapisati u sljedećem obliku:

$$f(\xi_1, \ldots, \xi_s) = \alpha_1\xi_1 + \ldots + \alpha_r\xi_r + g(\xi_{r+1}, \ldots, \xi_s),$$

gdje su $\alpha_1$, ..., $\alpha_r$ poznati koeficijenti a $g$ nelinearna funkcija preostalih parametara.

Ideja je sljedeća: postupak učenja provodit će se u ciklusima koji se sastoje od dva koraka.

1. U prvom koraku parametri o kojima funkcija $f$ ovisi nelinearno drže se na fiksnim vrijednostima (koje su, primjerice, na samom početku odabrane sasvim slučajno) i funkcija $f(\xi_1, \ldots, \xi_s)$ se promatra kao funkcija $f(\xi_1, \ldots, \xi_r)$. Traži se optimalan skup parametara $\xi_1$, ..., $\xi_r$ o kojima funkcija $f(\xi_1, \ldots, \xi_r)$ ovisi linearno. To je klasični problem linearne regresije za koji postoje puno efikasniji algoritmi u odnosu na gradijentni spust; primjerice, jedan od takvih je LSE (engl. *Least-Squares-Estimate*). Primjenom takvog algoritma brzo se pronađu optimalne vrijednosti $\xi_1$, ..., $\xi_r$. Primjetimo, međutim, da to nisu globalno optimalne vrijednosti – to su optimalne vrijednosti za koje, uz fiksirane parametre $\xi_{r+1}$, ..., $\xi_s$ funkcija $f$ poprima minimum.

2. U drugom koraku algoritam gradijentnog spusta primjenjuje se samo na parametre $\xi_{r+1}$, ..., $\xi_s$ dok se parametri $\xi_1$, ..., $\xi_r$ drže fiksnima. Kako se time parametri $\xi_{r+1}$, ..., $\xi_s$ mijenjaju, vrijednosti parametara $\xi_1$, ..., $\xi_r$ pronađenih u prvom koraku više nisu globalno optimalne. Stoga se po završetku postupak ciklički ponavlja ponovno od prvog koraka.

Ovakav način stoga kombinira brz pronalazak optimalnih vrijednosti parametara o kojima funkcija ovisi linearno s gradijentnim spustom koji je bitno sporiji ali kojim se mogu tražiti optimalne vrijednosti onih parametara o kojima funkcija ovisi nelinearno. Takvom kombinacijom dobiva se algoritam koji ima bržu konvergenciju od algoritma koji gradijentni spust primjenjuje za pronalazak svih parametara.

Važno je napomenuti da izračunska kompleksnost LSE algoritma i gradijentnog spusta nije niti približno jednaka. Naime, jedan korak algoritma gradijentnog spusta bitno je jednostavniji u odnosu na LSE algoritam; međutim, gradijentni spust je iterativna metoda kojoj treba puno iteracija kako

bi pronašla dobre vrijednosti parametara. Stoga je ove algoritme moguće kombinirati na sljedeće načine.

- *Uporaba samo algoritma gradijentnog spusta.* Kod ovakve izvedbe algoritma učenja za podešavanje svih parametara koristi se samo algoritam gradijentnog spusta koji je računski vrlo jednostavan no zahtjeva puno iteracija.
- *Uporaba algoritma LSE samo u prvom koraku pa potom daljnja uporaba algoritma gradijentnog spusta.* Kod ovakve izvedbe algoritam LSE iskoristi se na početku postupka učenja kako bi pronašao početne optimalne vrijednosti linearnih parametara. Potom se, krenuvši od tako postavljenih parametara dalje primjenjuje samo algoritam gradijentnog spusta.
- *Ciklička uporaba algoritama LSE i gradijentnog spusta.* Oba algoritma koriste se jedan za drugim; najprije se pomoću LSE pronađu optimalne vrijednosti linearnih parametara pa potom gradijentnim spustom optimalne vrijednosti nelinearnih parametara, nakon čega se postupak ciklički ponavlja.

### 12.2.8 Detaljniji pregled neuro-fuzzy sustava

Za dosta detaljniji pregled integracije neuronskih mreža te sustava neizrazitog učenja zainteresirane se čitatelje upućuje na pregledni rad [Abraham(2005)], u kojem je moguće pronaći daljnje reference na sustave poput *Fuzzy Adaptive Learning Control Network* (FALCON), *Generalized Approximate Reasoning Based Intelligent Control* (GARIC), *Neuro-Fuzzy Controller* (NEFCON), *Neuro-Fuzzy Classification* (NEFCLASS), *Neuro-Fuzzy Function Approximation* (NEFPROX), *Fuzzy Inference Environment Software with Tuning* (FINEST), *Self Constructing Neural Fuzzy Inference Network* (SONFIN), *Fuzzy Net* (FUN) i druge.

# Poglavlje 13

# Oblikovanje i učenje neuronske mreže algoritmima evolucijskog računanja

Kroz prethodna poglavlja upoznali smo se s pojmom umjetne neuronske mreže te smo naveli dvije tipične arhitekture – unaprijedne slojevite neuronske mreže te samoorganizirajuće neuronske mreže. Od mnoštva različitih arhitektura mreža izabrali smo ta dva kako bismo ilustrirali mogućnosti neuronskih mreža. Prisjetimo se, unaprijedne slojevite neuronske mreže mogli smo primijeniti na probleme funkcijske regresije i aproksimacije kao i klasifikacije. S druge pak strane, samoorganizirajuća neuronska mreža, poput Kohonenovog SOM-a, omogućavala je rješavanje zadataka poput grupiranja podataka.

Ako se fokusiramo na samo jedan od tih tipova, primjerice, unaprijedne slojevite mreže, ili još općenitije, unaprijedne mreže, postavlja se sljedeće pitanje: za zadatak koji mreža treba riješiti, koja je optimalna arhitektura neuronske mreže te koji su optimalni parametri takve mreže a uz koje neuronska mreža dobro rješava zadani problem, i pri tome zadržava dobro svojstvo generalizacije? U današnjem trenutku egzaktan odgovor na to pitanje ne postoji i odabir i učenje neuronske mreže svodi se na niz pokušaja i pogrešaka. Stoga je upravo prirodno razmotriti mogućnost kombiniranja evolucijskog procesa i razvoja neuronskih mreža.

Istraživanja na temu uporabe evolucijskih procesa za razvoj neuronskih mreža traju već preko dvadesetak godina, pri čemu su intenzivnija istraživanja započela ranih devedesetih. Primjena evolucijskih procesa na razvoj neuronskih mreža može se kategorizirati u tri područja.

1. *Evolucija težinskih faktora.* Klasični algoritmi učenja izvedeni su za pojedine arhitekture neuronskih mreža; primjerice, samoorganizirajuće neuronske mreže ne možemo učiti algoritmom *Backpropagation*. Evolucijski algoritmi kao univerzalni optimizacijski algoritmi primjenjivi

su na evoluciju parametara praktički neograničenog broja različitih arhitektura neuronskih mreža čime se izbjegava potreba za izvođenjem algoritama prilagođenih pojedinim arhitekturama.

2. *Evolucija arhitektura.* Poznato je da će prejednostavna neuronska mreža koju se primijeni na rješavanje nekog zadatka pronaći neprikladno rješenje – rješenje koje ima veliku pogrešku. S druge pak strane, preveliku neuronsku mrežu teško ćemo naučiti da kvalitetno rješava zadatak a nakon postupka učenja mreža će ispoljavati svojstvo pretreniranosti i vrlo loše generalizirati. Evolucijske algoritme moguće je stoga primijeniti na razvoj arhitekture neuronske mreže koja će biti jednostavna, a opet dovoljno složena kako bi rješavala zadani problem i zadržala svojstvo generalizacije.

3. *Evolucija pravila učenja.* Proučavanjem unaprijedne slojevite neuronske mreže u poglavlju 8 izveli smo algoritam *Backpropagation*, odnosno pokazali na koji se način računa korekcija svakog težinskog faktora. Međutim, izraz koji smo dobili nije jedini koji se može koristiti za postupak učenja. Evolucijske algoritme stoga je moguće primijeniti upravo na razvoj pravila učenja uz koje će neuronska mreža učiti brzo i efikasno. Uprabom evolucijskih algoritama pokušava se naučiti kako treba učiti.

## 13.1 Evolucija težinskih faktora

Unaprijedne neuronske mreže najčešće se koriste tako da se fiksira arhitektura mreže (broj ulaznih, izlaznih te skrivenih neurona; između koji neurona postoje veze te koje se prijenosne funkcije koriste) te se temeljem zadanog skupa uzoraka za učenje algoritmom učenja pokušava pronaći optimalan skup težinskih faktora uz koje mreža radi minimalnu pogrešku (sve drugo je fiksirano). Ovdje govorimo o učenju s učiteljem, a najpoznatiji algoritam koji se koristi za postupak učenja je *Backpropagation* [Rumelhart et al.(1986)Rumelhart, Hinton, and Williams] koji se zasniva na uporabi gradijenta pogreške. Osim tog algoritma i niza njegovih modifikacija, postoji još niz drugih koji se koriste, poput algoritma QuickProp [Fahlman(1989), Craig Veitch and Holmes(1991)], algoritma konjugiranog gradijenta [Moller(1990)] te Levenberg-Marquardt algoritma [Hagan and Menhaj(1994), Suratgar et al.(2005)Suratgar, Tavakoli, and Hoseinabadi].

Zajednička karakteristika svih ovih algoritama je njihova utemeljenost na gradijentu – takvi algoritmi često zapinju u lokalnim optimumima; algoritmi koji se temelje samo na informaciji o gradijentu također su vrlo neefikasni odnosno zahtijevaju velik broj iteracija. Dodatno, neprimjenjivi su na probleme minimizacije funkcije pogreške koja nije derivabilna.

Evolucijski algoritmi, primjerice genetski algoritam stohastički su optimizacijski algoritmi koji su poprilično robusni na lokalne optimume te ne zahtijevaju da funkcija koja se optimira bude derivabilna. Stoga je jedan od načina zaobilaženja problema koje sa sobom donosi algoritam *Backpropagation* uporaba evolucijskog algoritma primijenjenom na pronalazak optimalnog skupa težinskih faktora u fiksiranoj mreži. U literaturi se može pronaći dosta radova na temu usporedbe algoritma *Backpropagation* i *genetskog algoritma* kojima je trenirana ista mreža i rezultati nisu jasni – dio literature pokazuje da je algoritam *Backpropagation* efikasniji od genetskog algoritma, drugi dio literature pokazuje upravo suprotno. Međutim, treba napomenuti da su usporedbe uvijek rađene nad različitim problemima te između različitih vrsta algoritma *Backpropagation* i genetskih algoritama, što u konačnici samo potvrđuje *No-free-lunch* teorem: različiti algoritmi nad različitim problemima ponašaju se različito.

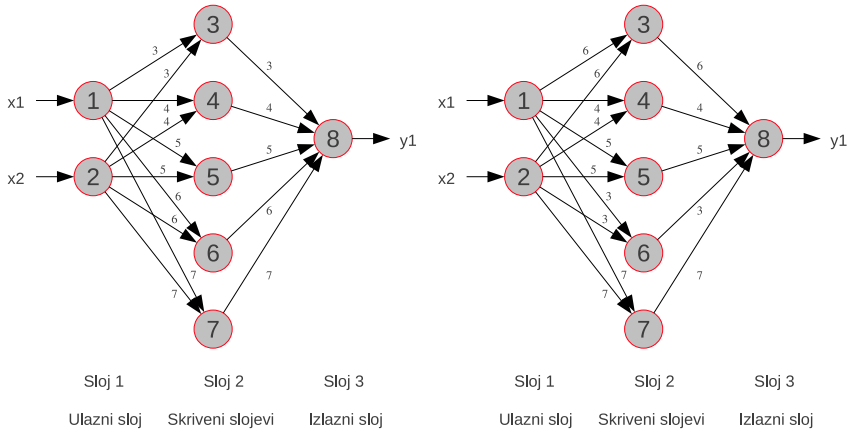
### 13.1.1 Primjena genetskog algoritma na evoluciju težinskih faktora

U fiksnoj arhitekturi neuronske mreže broj težinskih faktora koje treba naučiti je fiksan i ne mijenja se tijekom postupka učenja. Stoga se u praksi koriste dvije reprezentacije rješenja s kojima radi genetski algoritam: binarni prikaz te prikaz vektorom decimalnih brojeva.

Kod binarnog prikaza sve se težine numeriraju (primjerice od 0 do $n-1$), za svaku se težinu uzme $k$ bitova i potom se izgradi kromosom od $n \cdot k$ bitova. Da bi se omogućilo dekodiranje težina, potrebno je još utvrditi minimalni i maksimalni iznos težine te kod koji se koristi (primjerice, binarni kod, Grayev kod i slično). Prednost ovakvog prikaza je izostanak potrebe za definiranjem posebnih operatora križanja i mutacije. Jedan od nedostataka ovog prikaza je ograničen prostor pretraživanja – naime, za težine je unaprije potrebno zadati raspon unutar kojeg se obavlja pretraga. Prisjetimo se, kod algoritma *Backpropagation* to nije bio slučaj – težine smo tretirali kao realne brojeve i nismo (svjesno) postavljali nikakva ograničenja jer često ne znamo unutar kojeg raspona će se težine kretati.

Prikaz s vektorom decimalnih brojeva u ovom slučaju je nešto prirodniji iako sa sobom donosi niz problema: više ne možemo koristiti uobičajene binarne operatore već je potrebno razviti skup operatora koji rade nad decimalnim brojevima.

Relativno opsežno istraživanje na temu uporabe genetskog algoritma za treniranje neuronske mreže prikazano je u [Montana and Davis(1989)], gdje autori definiraju niz različitih operatora mutacije i križanja od kojih su neki prikazani u nastavku. Međutim, prije detaljnijeg pregleda tih operatora potrebno je spomenuti i problem koji se javlja prilikom učenja neuronskih mreža populacijskim algoritmima – radi se o problemu permutacija kod neuronskih mreža (engl. *Permutation problem*, također poznat i pod nazivom




(a) Originalna mreža (b) Drugačija mreža za isto preslikavanje

Slika 13.1: Ilustracija simetričnosti unutar neuronske mreže – problem permutacija

engl. *Competing conventions problem*) [Hancock(1992)]. Kako smo do sada govorili samo o algoritmu *Backpropagation* koji nije populacijski, nismo ga do sada spominjali. A radi se o sljedećem: neuronske mreže pravilnih struktura (primjerice troslojna unaprijedna potpuno-povezana neuronska mreža) izuzetno su simetrične, odnosno postoji mnoštvo načina raspodjele težina uz koje mreža obavlja identično preslikavanje. Da je tome tako, možemo se uvjeriti jednostavnim primjerom koji prikazuje slika 13.1.

Obje prikazane neuronske mreže su troslojne neuronske mreže s dva ulaza, pet skrivenih neurona i jednim izlazom. Svi neuroni su numerirani; neuroni 1 i 2 su ulazni i ništa ne računaju, neuroni 3 do 7 su skriveni i neuron 8 je izlazni. Mreža prikazana na slici 13.1b dobivena je direktno iz mreže prikazane na slici 13.1a tako da su sve težine na ulazima i izlazima neurona 3 i 6 zamijenjene. Uočimo, time neuron 6 sada računa ono što je u mreži prikazanoj na slici 13.1a računao neuron 3, a neuron 3 sada računa ono što je prije računao neuron 6. Kako su prilikom zamjene zamijenjene i težine kojima izračunati izlazi utječu na neuron 8, slijedi da obje mreže rade identično funkcijsko preslikavanje iz $\mathbb{R}^2 \to \mathbb{R}$.

Posljedica ove simetričnosti jest postojanje više globalnih optimuma funkcije pogreške koja se minimizira. U prikazanom primjeru na slici 13.1 u skrivenom sloju se nalazi 5 neurona; to znači težine u mreži možemo raspodijeliti na 5! različitih načina a da pri tome imamo identično ponašanje neuronske mreže. Već uz jedan skriveni sloj, broj globalnih optimuma jednak je dakle faktorijeli broja neurona u tom skrivenom sloju. Kako za iole veći broj neurona u tom skrivenom sloju broj rješenja raste vrlo brzo (a da ne spominjeno neuronske mreže s dva ili više skrivenih slojeva), jasno je da će

populacijski optimizacijski postupci koji međusobno kombiniraju dva ili više rješenja imati težak posao. Iz tog razloga dosta istraživača u ovom području pokušava izbjegavati takve operatore – kod genetskog algoritma to je upravo operator križanja, i koristiti samo operatore koji rade direktno nad jednim rješenjem (kod genetsko algoritma to bi bio operator mutacije).

Pogledajmo i na dva primjera zašto je križanje problematično u ovom slučaju. Pretpostavimo da su rješenja vektori decimalnih brojeva. Križajmo dvije mreže prikazane na slikama 13.1a i 13.1b najprije na način da za svaku težinu slučajno biramo hoćemo li je uzeti iz jednog djeteta ili iz drugog. Kako su težine kod svih neurona osim neurona 3 i 6 identične, dijete će ih naslijediti. Pretpostavimo sada da u djetetu neuron 3 dobije jednu težinu od jednog roditelja a drugu od drugog (dakle dobije težine 3 i 6) a to isto nastupi i kod težina neurona 6. Evo u čemu je problem: dotadašnji postupak evolucije neurone 3 i 6 u oba roditelja specijalizirao je za određeno područje ulaznog prostora i pronašao im prikladne težine; križanjem smo međutim dobili dva neurona od kojih niti jedan više nije niti sličan onome za što su polazni neuroni bili specijalizirani, i takva će mreža tipično raditi bitno lošije od roditelja; u tom smislu, operator križanja je naprosto previše destruktivan.

Pretpostavimo sada da smo izveli drugačije križanje: svjesni smo da u mreži postoji struktura pa operatorom mutacije mijenjamo težine, a operatorom križanja u dijete prenosimo kompletne neurone s njihovim vezama. Razmotrimo opet križanje dviju mreža prikazanih na slikama 13.1a i 13.1b. Neka dijete od prve mreže pokupi neurone 3, 4 i 5 a od druge mreže neurone 6 i 7. Tada će kod djeteta neuron 3 imati težine (3,3), neuron 4 težine (4,4), neuron 5 težine (5,5), neuron 6 težine (3,3) i neuron 7 težine (7,7). Uočimo sada: neuron koji je bio specijaliziran za ulazni dio prostora pokriven težinama (6,6) je kod djeteta izgubljen, a umjesto njega, dijete je dobilo dvije kopije neurona koji ima težine (3,3). Takvo dijete opet će vrlo vjerojatno biti lošije od roditelja pa križanje niti na koji način neće pomoći u evoluciji rješenja.

Opisani problem kod manjih mreža ne dolazi toliko do izražaja kao kod onih većih. Dapače, dio istraživača pokazao je da genetski algoritam u određenim slučajevima može biti dovoljno robustan da se nosi s ovim problemima i da opet pronalazi zadovoljavajuća rješenja. Međutim, ukazani problem jasno ukazuje i na potrebu da se, ako se već koriste operatori koji kombiniraju više rješenja, izvedbi takvih operatora posveti posebna pažnja.

U nastavku ćemo dati još i prikaz različitih izvedbi operatora mutacije i križanja za genetski algoritam koji traži optimalne težinske faktore neuronske mreže, kako su definirani u [Montana and Davis(1989)]. Operatori su izvedeni uz pretpostavku da se za prikaz rješenja koristi vektor decimalnih brojeva. Krenimo najprije s operatorima mutacije.

- *Mutacija bez pristranosti.* Za svaki element vektora operator s vjerojat-

nošću $p_m$ mijenja tu vrijednost nekom drugom nasumično generiranom vrijednosti.

- *Mutacija s pristranošću.* Za svaki element vektora operator s vjerojatnošću $p_m$ toj vrijednosti nadodaje neku drugu nasumično generiranu vrijednost.

- *Mutacija neurona.* Operator odabire $n$ neurona koji nisu ulazni. Potom svim dolaznim težinama odabranih neurona nadodaje neku nasumično generiranu vrijednost. Ovim se promjene lokaliziraju na razini pojedinih neurona.

- *Mutacija najslabijeg neurona.* Jakost neurona definira se kao razlika između izlaza neuronske mreže u kojoj promatrani neuron normalno funkcionira i izlaza lobotomizirane neuronske mreže u kojoj je izlaz promatranog neurona pritegnut na vrijednost 0. Neuron koji je najslabiji očito nije specijaliziran niti za jedno korisno područje i ne obavlja važnu ulogu u mreži – stoga se takav neuron mutacijom pokušava aktivirati.

Od operatora križanja, spomenut ćemo 3 opisana u nastavku.

- *Križanje težina.* Vektor težina djeteta tvori se tako da se svaka komponenta vektora nasumično preuzme od jednog ili drugog roditelja.

- *Križanje neurona.* Za svaki se neuron koji nije ulazni nasumično odabere jedan od roditelja i od njega se preuzme kompletan skup ulaznih težina tog neurona; naime, kako je pretpostavka da je svaki neuron u određenom smislu specijaliziran za određeno područje, u dijete se prenosi kompletan skup težina za neuron od izabranog roditelja čime se ta specijalizacija prenosi u dijete.

- *Križanje značajki.* Za svaki neuron u prvom roditelju pokušava se pronaći neuron koji obavlja istu ili najsličniju ulogu tog neurona u drugom roditelju. To se radi tako da se na ulaze obje mreže dovodi niz različitih ulaza i potom se promatra odziv fiksiranog neurona iz prve mreže i kandidata neurona iz druge mreže. Jednom kada se za svaki neuron iz prvog roditelja pronađe njegov par, poredak neurona u drugom roditelju se presloži tako da mreža i dalje obavlja istu funkciju ali da poredak neurona u obje mreže bude jednak u funkcijskom smislu. Tada se roditelji križaju na prethodno opisan način ("križaj neurone").

Za detalje se zainteresirane čitatelje upućuje na rad [Montana and Davis(1989)]. Spomenimo i da se u određenim slučajevima rad genetskog algoritma može dosta ubrzati ugradnjom lokalne pretrage. Lokalna pretraga je postupak koji se pokreće nakon generiranja djece (kada, nad kojoj točno djecom i koliko dugo dodatni su parametri koje tada treba podesiti) a čiji je cilj

pokušati popraviti stvoreno rješenje. Kao postupak lokalne pretrage moguće je implementirati upravo algoritam *Backpropagation* (pa odraditi nekoliko iteracija) ili neki drugi postupak. Algoritam *Backpropagation* izuzetno je osjetljiv na inicijalni odabir težina pa je česta situacija u praksi da se algoritam mora pokrenuti više puta kako bi pronašao neko dovoljno dobro rješenje, a ne zapeo u neprihvatljivo lošem lokalnom optimumu. Genetski algoritam je pak algoritam koji dosta grubo pretražuje prostor i koji je upravo sposoban pronaći dobar skup težina od kojih algoritam lokalne pretrage može dalje generirati dobra rješenja. Stoga ovakva kombinacija u praksi može davati dobre rezultate.

## 13.2 Evolucija arhitektura

Kako bi neuronska mreža zadani problem rješavala optimalno dobro, potrebno je prije postupka učenja težina odabrati mrežu prikladne složenosti. Naime, prejednostavna mreža neće biti sposobna zadatak rješavati uz malu pogrešku. S druge strane, kod presložene mreža postupak učenja će trajati vrlo dugo a dobivena mreža, iako će raditi malu pogrešku na skupu za učenje, generalno govoreći iskazivat će svojstvo vrlo slabe generalizacije. Pitanje je, stoga, kako odabrati optimalnu arhitekturu koja je prikladna za rješavanje postavljenog zadatka. Ovaj "meta-zadatak" – pronalazak optimalne arhitekture mreže – može se definirati kao zaseban optimizacijski problem koji je potrebno riješiti. Kriterijska funkcija koja se pri tome definira može uključivati faktore poput minimalne složenosti mreže, brzine učenja mreže i slično. Dva su uobičajena načina rješavanja tog problema: *konstruktivnim algoritmima* te *destruktivnim algoritmima*. Konstruktivni algoritmi počinju s neuronskom mrežom minimalne složenosti i po potrebi dodaju nove neurone, nove veze i nove slojeve kako bi mrežu učinili prikladnijom za rješavanje zadanog problema. Destruktivni algoritmi kreću od vrlo općenite i složene arhitekture mreže i potom pokušavaju uklanjati veze, neurone i slojeve pazeći da pri tome mreža i dalje dobro rješava zadani problem. Međutim, i kod jedih i kod drugih vrsta algoritama nema jasnog pravila (i opravdanja) na koji način treba točno raditi već su takvi algoritmi vođeni heurističkim pravilima.

Umjesto uporabe konstruktivnih i destruktivnih algoritama, problem pronalaska optimalne arhitekture može se rješavati evolucijskim algoritmima koji su prema [Miller et al.(1989)Miller, Todd, and Hedge] posebno pogodni za ovaj zadatak iz nekoliko razloga.

- Površina koju razapinje kriterijska funkcija a koju pretražuje optimizacijski algoritam beskonačno je velika jer broj mogućih arhitektura nije ograničen.

- Površina koju razapinje kriterijska funkcija nije derivabilna jer postoji

niz promjena arhitektura koje su diskretne (primjerice, dodavanje ili uklanjanje jedne veze, jednog neurona ili jednog sloja neurona ili pak promjena prijenosne funkcije u neuronu) .

- Površina koju razapinje kriterijska funkcija je kompleksna i sadrži veliku količinu šuma jer nemamo način egzaktne procjene kvalitete pojedine arhitekture već se ta ocjena računa indirektno – primjerice, mjereći vrijeme potrebno da mreža nauči, što je izuzetno podložno utjecaju slučajnosti.

- Površina koju razapinje kriterijska funkcija je decepcijska, budući da slične arhitekture mogu imati bitno različite iznose kriterijske funkcije.

- Površina koju razapinje kriterijska funkcija je višemodalna, budući da različite arhitekture mogu imati slične iznose kriterijske funkcije.

Prilikom dizajniranja evolucijskog algoritma prikladnog za rješavanje problema pronalaska optimalne arhitekture potrebno je odlučiti se *koliko će informacija biti kodirano u kromosom* s kojim evolucijski algoritam radi. Postoje dvije mogućnosti: svaki kromosom može do u posljednji detalj specificirati sve detalje neuronske mreže – sve neurone, sve veze među njima, iznose težina, odabrane prijenosne funkcije i slično. U tom slučaju govorimo o *direktnom kodiranju neuronske mreže*. Alternativa je u kromosom pohraniti samo najvažnije informacije, primjerice, koliko slojeva mreža treba imati i koliko neurona u svakom sloju (ako razvijamo slojevite mreže). U tom slučaju govorimo o *indirektnom kodiranju*.

Jednom kada je odabran način reprezentacije neuronske mreže, razvoj arhitektura može se obavljati prema pseudokodu prikazanom u nastavku.

1. Svaki kromosom dekodiraj u pripadnu neuronsku mrežu. Ako se koristi indirektno kodiranje, parametre koji nedostaju postavi na neki način (primjerice, dodatnim pravilom koje se koristi ili naprosto postupkom učenja težina).

2. Svaku neuronsku mrežu treniraj puno puta, svaki puta počevši od različitih početnih težina, različitih parametara u pravilu učenja (npr. ako se koristi *Backpropagation*, pokušavati s više različitih stopa učenja i faktora inercije).

3. Svakoj mreži (kromosomu) dodijeli pripadni iznos funkcije dobrote koji je izračunat temeljem obavljenih pokušaja učenja iz koraka 2 i u skladu s definiranom kriterijskom funkcijom.

4. Obavi križanje roditelja i mutaciju djece u skladu s odabranom vrstom evolucijskog algoritma a sukladno dodijeljenim iznosima funkcije dobrote te primijeni i eventualno neke druge operatore čime se u konačnici generira sljedeća generacija roditelja.

U nastavku ćemo se još osvrnuti na načine kodiranja neuronskih mreža.

Tablica 13.1: Prikaz arhitekture neuronske mreže matricom veza

| | **1** | **2** | **3** | **4** |
|---|---|---|---|---|
| **1** | $c_{1,1}$ | $c_{1,2}$ | $c_{1,3}$ | $c_{1,4}$ |
| **2** | $c_{2,1}$ | $c_{2,2}$ | $c_{2,3}$ | $c_{2,4}$ |
| **3** | $c_{3,1}$ | $c_{3,2}$ | $c_{3,3}$ | $c_{3,4}$ |
| **4** | $c_{4,1}$ | $c_{4,2}$ | $c_{4,3}$ | $c_{4,4}$ |

### 13.2.1 Direktno kodiranje neuronske mreže

Najjednostavniji primjer direktnog kodiranja neuronske mreže koja sadrži $N$ neurona jest matricom dimenzija $N \times N$. Primjer takvog zapisa za neuronsku mrežu koja se sastoji od ukupno 4 neurona dan je u tablici 13.1. Koeficijenti $c_{i,j} \in \{0, 1\}$ određuju postoji li veza između izlaza neurona $i$ i ulaza neurona $j$ (tj. da li neuron $i$ pobuđuje neuron $j$); ako je $c_{i,j} = 1$ tada neuron $i$ pobuđuje neuron $j$; u suprotnom neuron $i$ ne pobuđuje neuron $j$. Ako ovakva mreža obavlja preslikavanje iz $\mathbb{R}^m \to \mathbb{R}^n$ (tj. ima $m$ ulaza i $n$ izlaza, možemo se pridržavati konvencije da su prvih $m$ neurona ulazni (neuroni 1, 2, ..., $m$) a posljednjih $n$ neurona izlazni (neuroni $4 - n + 1$, $4 - n + 2$, ..., 4).

Umjesto restrikcije elemenata matrice na binarne zastavice, matrični prikaz možemo koristiti i direktno za pohranu težina (vidi tablicu 13.2). U tom slučaju elemente matrice ćemo označavati s $w_{i,j}$ i držat ćemo se sljedeće konvencije: ako je $w_{i,j} = 0$ tada ne postoji veza kojom neuron $i$ pobuđuje neuron $j$; u suprotnom, neuron $i$ pobuđuje neuron $j$ i pri tome je iznos težinskog faktora te veze upravo $w_{i,j}$. Primjetimo još da svaki neuron u neuronskoj mreži osim težinskih faktora koji se nalaze na njegovim ulazima i skaliraju pobude koje neuron dobiva od drugih neurona neuron sadrži i slobodnu težinu koju smo obično označavali s $w_0$. Kako bismo mogli pamtiti i ove težine, matrični prikaz koji je dan u tablici 13.2 uvodi virtualni neuron indeksiran s 0 koji nema nikakvih ulaza i na svojem izlazu generira fiksnu jedinicu. Tada za sve ostale neurone vrijednost slobodnog težinskog faktora možemo zapisati kao težinu između tog virtualnog neurona i svakog od preostalih neurona (prvi redak tablice). Primijetimo također da, ako se radi o unaprijednim mrežama, pripadna matrica ne može biti skroz popunjena. Primjerice, ako je neuron 1 ulazni neuron, tada njega ne pobuđuje niti jedan drugi neuron što znači da $\forall i \; w_{i,1} = 0$. Naš virtualni neuron također nitko ne pobuđuje, pa mora vrijediti $\forall i \; w_{i,0} = 0$. Ako je neuron 4 izlazni, tada on nikoga ne pobuđuje, a to znači da mora vrijediti $\forall i \; w_{4,i} = 0$. I konačno, da bi mreža bila unaprijedna, dozvoljeno je postojanje veza od neurona $i$ prema neuronu $j$ ako je $i < j$ ali ne i obratno. Naime, ako se dozvoli da neuron $i$ pobuđuje neuron $j$ i da neuron $j$ pobuđuje neuron $i$, tada izlaz svakog od neurona ovisi o izlazu upravo onog drugog, pa više nemamo unaprijednu mrežu već smo dobili mrežu koja ima vremenski promjenjiv odziv (sjetite se primjerice mreže *Winner-Takes-All* koja je upravo takva). Iz istog razloga,

Tablica 13.2: Prikaz arhitekture neuronske mreže matricom težina

| | **0** | **1** | **2** | **3** | **4** |
|---|---|---|---|---|---|
| **0** | $w_{0,0}$ | $w_{0,1}$ | $w_{0,2}$ | $w_{0,3}$ | $w_{0,4}$ |
| **1** | $w_{1,0}$ | $w_{1,1}$ | $w_{1,2}$ | $w_{1,3}$ | $w_{1,4}$ |
| **2** | $w_{2,0}$ | $w_{2,1}$ | $w_{2,2}$ | $w_{2,3}$ | $w_{2,4}$ |
| **3** | $w_{3,0}$ | $w_{3,1}$ | $w_{3,2}$ | $w_{3,3}$ | $w_{3,4}$ |
| **4** | $w_{4,0}$ | $w_{4,1}$ | $w_{4,2}$ | $w_{4,3}$ | $w_{4,4}$ |

mora vrijediti $\forall i \; w_{i,i} = 0$. Međutim, uočimo da ova ograničenja postoje ako želimo dobiti unaprijednu mrežu; ako to nije slučaj, matrični je prikaz dovoljno ekspresivan za dobivanje bilo kakve arhitekture.

### Genetski algoritam za rješavanje problema XOR uz direktno kodiranje

Problem XOR definirat ćemo kao problem pronalaska neuronske mreže koja korektno radi na sljedećem skupu uzoraka za učenje: {(-1,-1)→-1, (-1,1)→1, (1,-1)→1, (1,1)→-1}.

Kako bismo ilustrirali genetski algoritam koji koristi direktno kodiranje, napisana je implementacija troturnirskog genetskog algoritma koji trenira neuronsku mrežu koja se sastoji od neurona kod kojih se kao prijenosna funkcija koristi funkcija skoka oblika:

$$f(net) = \begin{cases} 1, & net > 0 \\ -1, & \text{inače.} \end{cases}$$

**Pokušaj 1** Kako je poznato da neuronska mreža koja se sastoji od jednog neurona (ne računajući ulazne neurone) ne može obaviti taj posao, genetskim algoritmom pokušali smo pronaći slojevitu potpuno-povezanu troslojnu neuronsku mrežu koja u skrivenom sloju ima dva neurona i koja rješava taj problem, tj. mrežu oblika $2 \times 2 \times 1$. Pripadna matrica težina prikazana je u tablici 13.3. S obzirom na postavljeni zahtjev slojevitosti, neuronska mreža ima relativno malo težina koje se mogu postavljati. Neuron 0 je virtualni neuron koji na izlazu generira fiksnu vrijednost 1 i služi za podešavanje slobodnih težinskih faktora, neuroni 1 i 2 su ulazni neuroni koji na svoj izlaz preslikavaju dovedeni uzorak, odnosno komponente $x_1$ i $x_2$, neuroni 3 i 4 su neuroni skrivenog sloja i prvi koji doista nešto računaju i neuron 5 je izlazni neuron.

Operatori križanja izveden je tako da dijete svaki element matrice s 50% šanse preuzme od jednog roditelja odnosno s 50% šanse od drugog roditelja. Mutacija je izvedena na način da se svakom elementu matrice s vjerojatnošću $p_m$ nadoda slučajno generirani broj iz normalne distribucije s parametrima

Tablica 13.3: Matrični prikaz arhitekture neuronske mreže za problem XOR

| | **0** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|---|
| **0** | 0.0 | 0.0 | 0.0 | $w_{0,3}$ | $w_{0,4}$ | $w_{0,5}$ |
| **1** | 0.0 | 0.0 | 0.0 | $w_{1,3}$ | $w_{1,4}$ | 0.0 |
| **2** | 0.0 | 0.0 | 0.0 | $w_{2,3}$ | $w_{2,4}$ | 0.0 |
| **3** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | $w_{3,5}$ |
| **4** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | $w_{4,5}$ |
| **5** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

$(0, \sigma)$. Nakon križanja i mutacije, nad matricom je pokrenut dodatni postupak koji je sve težine za koje je $c_{i,j} = 0$ resetirao ponovno na 0 kako bi se osiguralo da je mreža i dalje unaprijedna slojevita.

Konkretno, genetski algoritam je pokrenut uz sljedeće parametre:

- veličina populacije: 50,
- $p_m = 0.33$ te
- $\sigma = 0.25$.

Inicijalno, težine su birane uniformno iz intervala $[-5,5]$. Algoritam je pronašao rješenje koje problem rješava ispravno već u 4. generaciji nakon čega je ostatak vremena proveo popravljajući njegovu kvalitetu s obzirom na zadanu kriterijsku funkciju. Kriterijska funkcija pri tome je definirana kao negativan iznos funkcije kazne. Funkcija kazne računana je na sljedeći način:

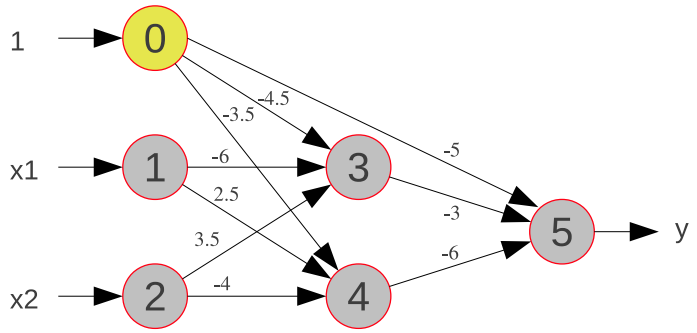
- Označimo s $K_1$ iznos sume kvadratnog odstupanja izračunatog na čitavom skupu za učenje. U našem slučaju minimalni iznos ove komponente je 0 a maksimalni $4 \cdot 2^2 = 16$, koji se postiže ako mreža sva 4 uzorka pogrešno klasificira.
- Označimo s $K_2$ iznos sume kazni zbog neprikladnog oblika težina. Naime, htjeli bismo da sve težine budu ili cijeli brojevi, ili decimalni brojevi oblika $x.5$. Komponenta $K_2$ je suma apsolutnog odstupanja svakog elementa matrice od željenog oblika težine. Primjerice, ako je neka težina jednaka 3.4, najbliži "poželjni" broj je 3.5 pa će za tu težinu u $K_2$ biti nadodan iznos $|3.4 - 3.5| = 0.1$.
- Označimo s $K_3$ iznos sume kazni zbog veličine težina. Naime, htjeli bismo postići da je što više težina jednako 0 ili da su vrlo male. Stoga komponentu $K_3$ računamo kao sumu kvadata svih težina u matrici.
- Ukupna kazna tada se računa kao:

$$K_1 + \frac{1}{2}\phi(K_2) + \frac{1}{2}\phi(K_3)$$

gdje je funkcija $\phi(x) = \frac{2}{\pi} \cdot arctan(x)$.

Tablica 13.4: Matrični prikaz arhitekture neuronske mreže za problem XOR

| | **0** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|---|
| **0** | 0.0 | 0.0 | 0.0 | -4.5 | -3.5 | -5.0 |
| **1** | 0.0 | 0.0 | 0.0 | -6.0 | 2.5 | 0.0 |
| **2** | 0.0 | 0.0 | 0.0 | 3.5 | -4.0 | 0.0 |
| **3** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | -3.0 |
| **4** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | -6.0 |
| **5** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |




Slika 13.2: MLP s dva neurona u skrivenom sloju za problem XOR-a naučen genetskim algoritmom

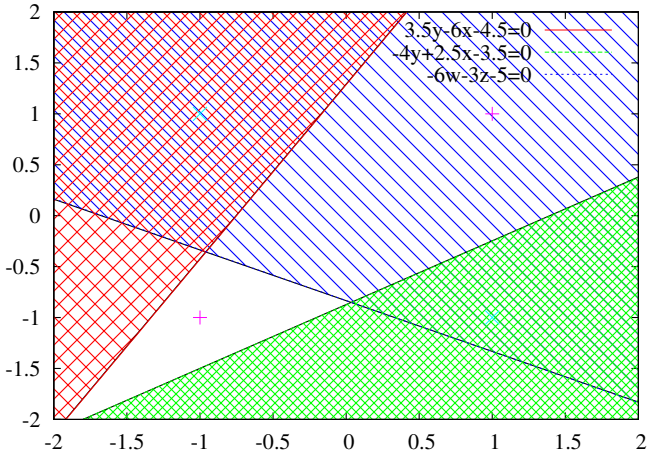
Razvijeni genetski algoritam prilikom rada pokušava maksimizirati dobrotu generiranih neuronskih mreža odnosno pronaći mrežu koja ima minimalni iznos funkcije kazne. Dobiveni rezultat prikazan je u tablici 13.4 a dekodirana neuronska mreža prikazana je na slci 13.2.

Interesantno je pogledati na koji je način genetski algoritam podesio težine, odnosno što točno pronađena neuronska mreža radi. Za potrebe pojašnjenja, uvedimo oznaku $z$ za izlaz neurona 3 te oznaku $w$ za izlaz neurona 4. Uz pronađeni skup težina vrijedi:

$$z = step(-6x_1 + 3.5x_2 - 4.5)$$

$$w = step(2.5x_1 - 4x_2 - 3.5).$$

Decizijske funkcije koje se dobiju za $z = 0$ (crvena linija) i $w = 0$ (zelena linija) prikazane su na slici 13.3. Dodatno, čitav podprostor u kojem je *net* koji računa neuron 3 pozitivan (pa stoga izlaz $z = 1$) obojan je crvenim uzorkom dok je čitav podprostor u kojem je *net* koji računa neuron 4 pozitivan (pa stoga izlaz $w = 1$) obojan je zelenim uzorkom. Sada vidimo da je neuron 3 specijaliziran za detekciju uzorka $(x_1, x_2) = (-1, 1)$ i samo za njega daje na izlazu vrijednost 1 a neuron 4 je specijaliziran za detekciju uzorka $(x_1, x_2) = (1, -1)$ i samo za njega daje na izlazu vrijednost 1. Čitava neuronska mreža sada treba na izlazu dati $-1$ samo ako nije prepoznat




Slika 13.3: Decizijske funkcije naučene za svaki od tri neurona u MLP-u za problem XOR-a

uzorak (-1,1) niti uzorak (1,-1); drugim riječima, samo kada su oba izlaza $w$ i $z$ jednaka $-1$; ako je prepoznat bilo koji od navedena dva uzorka, mreža treba dati izlaz 1. No taj je posao rješiv jednim neuronom – onim izlaznim. Naime, taj neuron treba obaviti sljedeće preslikavanje:{(w,z)$\rightarrow \phi(w,z)$}: {(-1,-1)$\rightarrow$-1, (-1,1)$\rightarrow$1, (1,-1)$\rightarrow$1, (1,1)$\rightarrow$1} što je najobičnija funkcija ILI i ona je linearno razdvojiva. Taj posao obavlja upravo neuron 5 čija je decizijska funkcija:

$$y = step(-3z - 6w - 5)$$

koja je na slici 13.3 prikazana plavom linijom a područje za koje pripadni *net* poprima pozitivnu vrijednost (pa je tada $y = 1$) obojano je plavim uzorkom.

Na koji je dakle način genetski algoritam podesio težine neuronske mreže? Neurone skrivenog sloja specijalizirao je tako da iz prostora uzoraka detektiraju određene zanimljive uzorke (ili općenito kombinacije uzoraka); potom je iskoristio sljedeći sloj neurona (kod nas i posljednji, izlazni) kako bi temeljem aktivacija tih detektora (a ne više direktnim promatranjem ulaznih uzoraka) generirao konačnu klasifikaciju. Sada je jasno zašto neuronske mreže s više slojeva i malim brojem neurona mogu naučiti obavljati komplicirane klasifikacije: svaki sljedeći sloj neurona postaje detektor koji svoj izlaz gradi temeljem prethodnog sloja detektora i time gradi sve kompliciranije decizijske funkcije.

**Pokušaj 2** Umjesto troslojne potpuno povezane neuronske mreže tipa $2 \times 2 \times 1$, genetskim algoritmom smo pokušali naučiti jednostavniju troslojnu potpuno-povezanu neuronsku mrežu: $2 \times 1 \times 1$. Međutim, postupak učenja nije mogao pronaći niti jedan skup težina uz koje bi mreža korektno klasificirala sve uzorke – razmislite je li to krivica algoritma učenja? Potom

Tablica 13.5: Matrični prikaz arhitekture neuronske mreže s dva operativna neurona za problem XOR

| | **0** | **1** | **2** | **3** | **4** |
|---|---|---|---|---|---|
| **0** | 0.0 | 0.0 | 0.0 | $w_{0,3}$ | $w_{0,4}$ |
| **1** | 0.0 | 0.0 | 0.0 | $w_{1,3}$ | $w_{1,4}$ |
| **2** | 0.0 | 0.0 | 0.0 | $w_{2,3}$ | $w_{2,4}$ |
| **3** | 0.0 | 0.0 | 0.0 | 0.0 | $w_{3,4}$ |
| **4** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

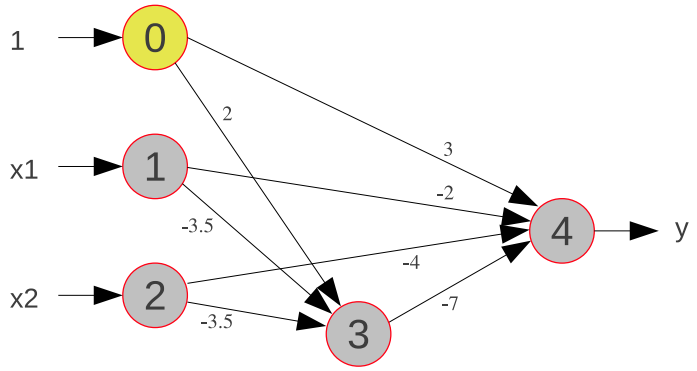
Tablica 13.6: Matrični prikaz arhitekture neuronske mreže za problem XOR

| | **0** | **1** | **2** | **3** | **4** |
|---|---|---|---|---|---|
| **0** | 0.0 | 0.0 | 0.0 | 2.0 | 3.0 |
| **1** | 0.0 | 0.0 | 0.0 | -3.5 | -2.0 |
| **2** | 0.0 | 0.0 | 0.0 | -3.5 | -4.0 |
| **3** | 0.0 | 0.0 | 0.0 | 0.0 | -7.0 |
| **4** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

smo odustali od zahtjeva da mreža bude slojevita, i pokušali naučiti neuronsku mrežu koja je unaprijedna i koja ima dva ulaza te još dva neurona na raspolaganju (od koji je jedan ujedno i izlazni). Opći oblik matrice koja predstavlja ovakvu mrežu prikazan je u tablici 13.5.

Genetski algoritam pokrenuli smo s istim skupom parametara; jedina je razlika u dozvoljenom obliku matrice koja se evoluira. I opet, genetski algoritam je uspješno pronašao rješenje. Evoluirana matrica te dekodirana neuronska mreža prikazane su u tablici 13.6 i na slici 13.4.

Prokomentirajmo i ovo rješenje. Kako sada imamo nedovoljno neurona kako bismo direktno razvili detektore za pojedine dijelove ulaznog prostora




Slika 13.4: Unaprijedna (ali neslojevita) neuronska mreža s ukupno 2 operativna neurona za problem XOR-a naučen genetskim algoritmom

pa klasifikaciju radili temeljem njih, genetski algoritam je evoluirao drugačiju strategiju rješavanja problema: umjesto da se problem pokuša razriješiti direktno u dvodimenzijskom ulaznom prostoru gdje problem nije linearno razdvojiv, evolucijski proces je najprije napravio preslikavanje iz dvodimenzijskog ulaznog prostora u trodimenzijski prostor i potom klasifikaciju pokušao napraviti u tom trodimenzijskom prostoru. Kako to vidimo? Izlazni neuron dobiva kao ulaze $x_1$, $x_2$ i $z$; stoga on pokušava napraviti klasifikaciju u trodimenzijskom prostoru. Spomenuto preslikavanje radi se tako da se dvije komponente iz ulaznog prostora direktno proslijede kao dvije komponente za trodimenzijski prostor dok se treća komponenta računa uporabom za to predviđenog neurona 3. Ako izlaz tog neurona 3 označimo sa $z$, tada uzorke $(x_1,x_2) \in \mathbb{R}^2$ preslikavamo u uzorke $(x_1,x_2,z) \in \mathbb{R}^3$. Uz težine prikazane na slici 13.4, izlazni neuron pokušava naučiti sljedeće preslikavanje: {(-1,-1,1)→-1, (-1,1,1)→1, (1,-1,1)→1, (1,1,-1)→-1}. Lako se je uvjeriti da je taj zadatak rješiv jer je ovako definirani ulazni skup linearno razdvojiv, i taj posao upravo uspješno odrađuje izlazni neuron.

**Zaključak** Direktno kodiranje neuronske mreže najjednostavniji je oblik zapisa neuronske mreže; činjenica da se u zapisu nalaze sve potrebne informacije koje u potpunosti određuju neuronsku mrežu bitno olakšavaju postupak evaluacije kvalitete razvijene arhitekture neuronske mreže. Međutim, ovakav zapis, nažalost, ne funkcionira dobro ako se pokušaju razvijati velike neuronske mreže jer broj elemenata matrice raste kvadratno s brojem korištenih neurona. Također, iako neka istraživanja pokazuju da operatori koji kombiniraju više neuronskih mreža prilikom stvaranja potomaka mogu pozitivno utjecati na evolucijski proces, činjenica je da je direktno kodiranje izravno podložno prethodno opisanome problemu permutacija. Indirektno kodiranje pokušava jednim dijelom riješiti upravo te probleme.

### 13.2.2 Indirektno kodiranje neuronske mreže

Za razliku od direktnog kodiranja, indirektno kodiranje u kromosom ne pohranjuje sve detalje o neuronskoj mreži već samo određen dio. Nakon što se izgradi mreža koja je u skladu s tim informacijama, sve što nedostaje, definira se nekim unaprijed zadanim pravilom ili se pak uči. Postoji više načina indirektnog kodiranja od kojih ćemo spomenuti samo neke.

#### Parametarski prikaz

Pretpostavimo da razvijamo unaprijednu slojevitu potpuno-povezanu neuronsku mrežu. Takva se mreža tada može opisati definiranjem broja neurona u svakom od slojeva; primjerice, radi li se o mreži tipa $2 \times 2 \times 3$ ili $2 \times 5 \times 3$ ili $2 \times 3 \times 4 \times 3$ i slično. Kod ovakvog prikaza u kromosom se upisuju samo informacije o broju slojeva te o broju neurona unutar svakog

sloja. Uočimo da se sada više ne pamte težine koje odgovaraju svakoj od veza između međusobno povezanih neurona. Operatori križanja i mutacije u ovom slučaju rade operacije nad ovako zapisanim parametrima.

Problem koji se sada javlja jest kako utvrditi dobrotu arhitekture koju predstavlja pojedini kromosom. Da bi to bilo moguće, potrebno je više puta uz različite početne vrijednosti težina obaviti postupak učenja takve neuronske mreže i temeljem performansi postupka učenja donijeti ocjenu kvalitete ove arhitekture. Ovakav pristup ocjenjivanju u sebi inherentno sadrži veću količinu šuma nego što je to slučaj kod direktnog kodiranja – moguće je da naprosto nismo imali sreće pa smo puno puta odabrali nepovoljne početne težine uz koje je postupak učenja bio dugotrajan i uz koje je zapeo u nepovoljnom lokalnom optimumu, no osim toga ta je arhitektura inače vrlo dobra za problem koji rješavamo.

Kod ovakvog prikaza osim parametara koji određuju mrežu moguće je istovremeno evoluirati i parametre koji su prisutni u pravilu učenja; primjerice, ako mrežu učimo algoritmom *Backpropagation* uz moment inercije, u kromosom se mogu upakirati i parametri poput stope učenja te faktora inercije čime će evolucijski algoritam evoluirati istovremeno i prikladnu arhitekturu mreže i njoj i problemu za koji se mreža koristi prilagođen algoritam učenja.

### Prikaz uporabom pravila razvoja

Razvojna pravila su pravila koja govore na koji će se način izgraditi neuronska mreža. Zamislimo to ovako: umjesto da evoluiramo samu mrežu, mi evoluiramo program koji kada pokrenemo generira određenu neuronsku mrežu. Kromosomi su tada različita razvojna pravila. U literaturi je zabilježen niz prednosti ovakvog pristupa pred direktnim kodiranjem: bolja skalabilnost te imunost na problem permutacija budući da kromosomi ne predstavljaju direktno neuronske mreže.

Razvojna pravila najčešće dolaze u dva oblika: ili kao rekurzivne jednadžbe [Mjolsness et al.(1989)Mjolsness, Sharp, and Alpert] ili kao produkcijska pravila [Kitano(1990)] koja generiraju konačnu neuronsku mrežu. U oba slučaja, ono što se gradi jest matrica težina koja u konačnici određuje oblik neuronske mreže.

Primjer uporabe produkcijskih pravila prikazan je na slikama 13.5 i 13.6 gdje se pravilima gradi matrica veza.

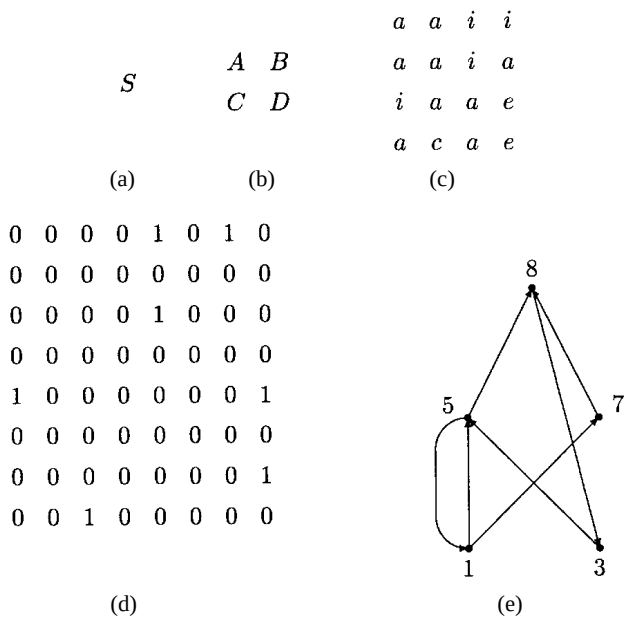
Produkcijska pravila prikazana na slici 13.5 sastoje se od nezavršnih simbola (velika slova i mala slova) te završnih simbola (znamenke). Pravila su podijeljena u dvije grupe: pravila prve razine kod kojih se s lijeve strane nalazi veliko slovo te pravila druge razine kod kojih se s lijeve strane nalazi malo slovo. Početni simbol od kojeg se kreće je `S`. Pravila prve razine s desne strane imaju matrice dimenzija $2 \times 2$ čiji su elementi mala slova. Pravila druge razine s desne strane imaju matrice dimenzija $2 \times 2$ čiji su

$$S \longrightarrow \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

$$A \longrightarrow \begin{pmatrix} a & a \\ a & a \end{pmatrix} \quad B \longrightarrow \begin{pmatrix} i & i \\ i & a \end{pmatrix} \quad C \longrightarrow \begin{pmatrix} i & a \\ a & c \end{pmatrix} \quad D \longrightarrow \begin{pmatrix} a & e \\ a & e \end{pmatrix} \quad \cdots$$

$$a \longrightarrow \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \quad c \longrightarrow \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \quad e \longrightarrow \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \quad i \longrightarrow \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad \cdots$$

Slika 13.5: Primjer produkcijskih pravila za izgradnju matrice veza neuronske mreže




Slika 13.6: Primjer uporabe produkcijskih pravila za izgradnju matrice veza neuronske mreže

elementi znamenke 0 ili 1.

Primjer izgradnje konkretne neuronske mreže prikazan je na slici 13.6. Kreće se od simbola `S` (a) koji se potom ekspandira u matricu $2 \times 2$ (b). Potom se svaki od elemenata te matrice mijenja podmatricama $2 \times 2$ (c) čime se dobiva matrica $4 \times 4$. Konačno, svaki se nezavršni element u toj matrici supstituira novom matricom $2 \times 2$ (d) čime se dobiva konačna matrica veza dimenzija $8 \times 8$. Ova matrica ne definira vrijednosti težinskih faktora već samo određuje između kojih neurona postoje veze. Točne iznose težina trebat će utvrditi posebno, primjerice algoritmom učenja nad konkretnim problemom koji se rješava. Konačna neuronska mreža prikazana je pod (e).

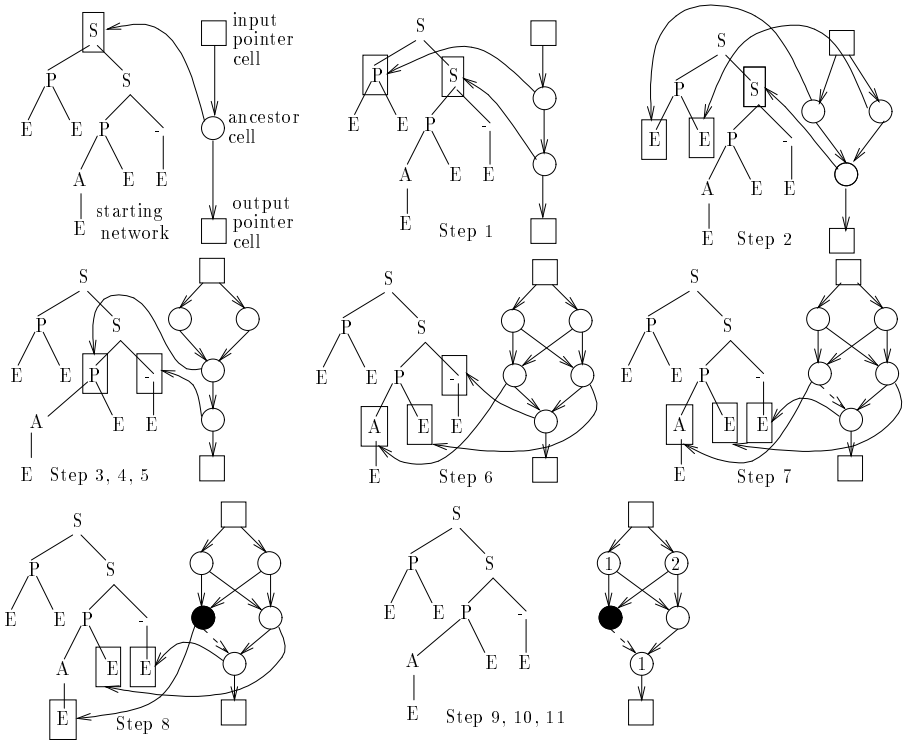
Pretpostavimo sada da su pravila druge grupe zadana unaprijed (pravila koja s lijeve strane imaju mala slova) i pretpostavimo da ih ima 16 (od `a` do `p`). Evolucijskim algoritmom uče se samo pravila prve grupe. Ako koristimo uobičajen skup slova, onda takvih pravila može biti ukupno 26. Kako svako pravilo s desne strane sadrži matricu od 4 elementa, slijedi da jedan kromosom koji specificira kompletan skup pravila treba sadržavati ukupno $26 \cdot 4 = 104$ simbola. Primjerice, početak kromosoma koji predstavlja upravo skup pravila prikazan na slici 13.5 bio bi `ABCDaaaaiiiaiaacaeae` (prva 4 simbola predstavljaju matricu pravila `S`, sljedeća 4 matricu pravila `A` itd.).

Zadaća evolucijskog algoritma primijenjenog na ovakav zapis neuronske mreže jest pronaći produkcijski sustav koji će izgraditi optimalnu neuronsku mrežu.

Pristup koji umjesto produkcijskih pravila koristi rekurzivne jednadžbe prikazan je u [Mjolsness et al.(1989)Mjolsness, Sharp, and Alpert]. Rekurzivne jednadžbe definirane su nad cjelobrojnim matricama i one određuju kako se iz manje matrice u sljedećem koraku gradi veća matrica. Na taj način iteraciju po iteraciju dolazi se do konačne matrice koja predstavlja neuronsku mrežu. Zadaća algoritma učenja je tada pronaći optimalne vrijednosti koeficijenata u rekurzivnim jednadžbama. Ovaj pristup dalje nećemo pojašnjavati.

Konačno, zanimljiv pristup uporabom staničnog razvoja opisan je u [Gruau(1992)]. Svaki neuron ima pridruženu kopiju genetskog koda koji upravlja njegovim razvojem. Genetski kod je stablo koje određuje kako se neuron transformira. Stablo se sastoji od niza simbola koje neuron izvršava. Svaki simbol predstavlja određenu operaciju koja modificira neuron koji je izvede. Razvoj počinje od najjednostavnije moguće neuronske mreže koja se sastoji samo od ulaznih neurona, izlaznih neurona te jednog *početnog* neurona koji je spojen na sve ulaze i sve izlaze i koji jedini dobiva genetski kod koji počinje izvršavati; ulazni i izlazni neuroni već su oformljeni neuroni koji se dalje ne razvijaju.

Simbol `S` označava da se neuron treba podijeliti u dva neurona. Prvi neuron pri tome preuzima sve ulazne veze početnog neurona a drugi neuron preuzima sve izlazne veze početnog neurona. Dodatno izlaz prvog neurona spaja se kao jedini ulaz drugog neurona. U stablu, simbol `S` ujedno je i točka




Slika 13.7: Primjer staničnog razvoja neuronske mreže

grananja: prvi stvoreni neuron koji je nastao ovom podjelom nastavit će izvoditi lijevu granu pridruženu simbolu `S` a drugi desnu granu (vidi sliku 13.7).

Simbol `P` predstavlja paralelno dijeljenje. Kad neuron obavi instrukciju `P`, podijelit će se u dva neurona pri čemu oba novonastala neurona preuzimaju sve ulazne i izlazne veze od neurona koji se je podijelio. U stablu, i simbol `P` predstavlja točku grananja: prvi stvoreni neuron koji je nastao ovom podjelom nastavit će izvoditi lijevu granu pridruženu simbolu `P` a drugi desnu granu.

Simbol `E` označava kraj programa; kad neuron pročita i izvrši taj simbol, neuron postaje nepromjenjiv i gubi pridruženi genetski kod. Simbol `A` povećava vrijednost praga u neuronu (težina $w_0$), a simbol `-` obrće predznak ulazne težine. Osim ovih, definiran je još niz simbola koji obavljaju različite akcije, uključivo i rekurzivne pozive i koje dalje nećemo opisivati. Primjer razvoja jedne mreže detaljnije je prikazan na slici 13.7.

Primjenom evolucijskog algoritma na ovakvu vrstu pravila razvoja pokušava se razviti program prikazan u obliku stabla koji će generirati neuronsku mrežu optimalnih karakteristika.

U praksi se osim navedenih primjera dosta radi i na evoluciji prijenosnih funkcija kao i na istovremenoj evoluciji i arhitektura i pripadnih težinskih faktora. Međutim, u okviru ovog poglavlja nećemo razmatrati ta područja.

## 13.3 Evolucija pravila učenja

Algoritam *Backpropagation* jedan je od primjera algoritama za učenje težina neuronske mreže; naravno – postoje i drugi. Stoga se postavlja pitanje možemo li evolucijski proces iskoristiti za pronalaženje novih algoritama učenja? Kada govorimo o pravilima učenja, evolucijski postupci se primjenjuju na dva mjesta.

### 13.3.1 Evolucija parametara algoritma učenja

Koji god algoritam učenja koristili, uobičajeno je da postoji jedan ili više parametara koji upravljaju radom tog algoritma. Kod algoritma *Backpropagation* to su, primjerice, stopa učenja i faktor inercije. Uz fiksirani algoritam učenja, evolucijski se proces može upotrijebiti kako bi pronašao optimalni skup tih parametara uz koje će postupak učenja biti maksimalno djelotvoran i učinkovit.

### 13.3.2 Evolucija novih algoritama učenja

Prilikom razvoja algoritama učenja, uobičajeno se kreće od sljedećih pretpostavki:

1. ažuriranje težine ovisi samo o lokalno dostupnim informacijama poput aktivacije dolaznog neurona, aktivacije promatranog neurona i slično te

2. algoritam učenja (odnosno pravilo) jednako je za sve težine u mreži.

Pravilo učenja tada se formulira kao linearna kombinacija dostupnih lokalnih varijabli i njihovih produkata:

$$\Delta w(t) = \sum_{k=1}^{n} \sum_{i_1,i_2,\ldots,i_k=1}^{n} \left( \theta_{i_1,i_2,\ldots,i_k} \prod_{j=1}^{k} x_{i,j}(t-1) \right)$$

gdje je $t$ vrijeme odnosno iteracija, $\Delta w(t)$ promjena težine koju treba napraviti $x_1, \ldots, x_n$ su varijable koje čuvaju lokalno dostupne informacije, a $\theta_{i_1,i_2,\ldots,i_k}$ koeficijenti. Uz različite vrijednosti koeficijenata $\theta_{i_1,i_2,\ldots,i_k}$ dobivaju se različita pravila učenja, i zadaća evolucijskog procesa je utvrditi optimalne iznose ovih koeficijenata kako bi se dobio maksimalno učinkovit i djelotvoran algoritam učenja.

# Poglavlje 14

# Kombiniranje neuro-fuzzy sustava i evolucijskog računanja

Primjena algoritama evolucijskog računanja na neuro-fuzzy sustave postoji čitav niz. U okviru ovog poglavlja opisat ćemo nekoliko takvih primjena (radovi [Lin et al.(2011)Lin, Peng, and Lee, Ghomsheh et al.(2007)Ghomsheh, Shoorehdeli, and Teshnehlab]).

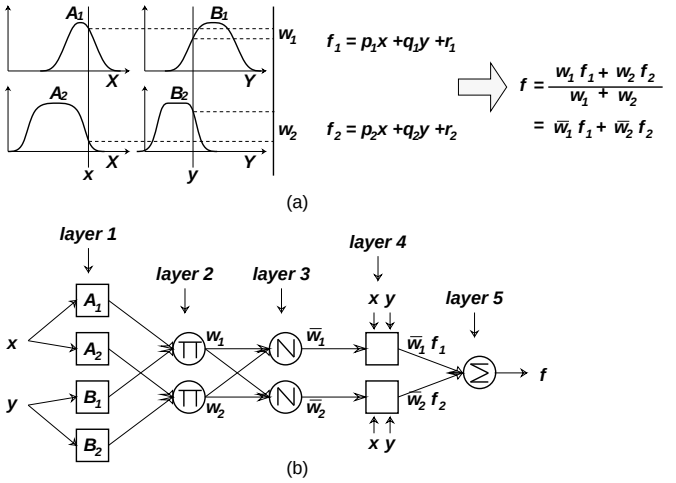
## 14.1 Primjena algoritma PSO na otkrivanje optimalnih parametara ANFIS-a

U radu [Ghomsheh et al.(2007)Ghomsheh, Shoorehdeli, and Teshnehlab] opisana je primjena algoritma PSO na treniranje neuro-fuzzy sustava ANFIS. U okviru ove knjige već smo opisali sustav ANFIS u poglavlju 12.2. Algoritam za treniranje tog sustava razvili smo polazeći od gradijentnog spusta koji zahtjeva da su funkcije koje se njime optimiraju derivabilne te da možemo relativno jednostavno i efikasno računati njihov gradijent.

Kako bi se zaobišla ta ograničenja, a ujedno i povećala otpornost algoritma učenja na lokalne optimume, u radu [Ghomsheh et al.(2007)Ghomsheh, Shoorehdeli, and Teshnehlab] opisana je primjena algoritma roja čestica. Primjena je ilustrirana na primjeru sustava ANFIS koji koristi zaključivanje tipa 3 (TSK-zaključivanje); takav sustav još je jednom prikazan na slici 14.1.

Sustav ANFIS za koji je razvijen algoritam treniranja opisan je sljedećim parametrima.

- $I$ - broj ulaznih varijabli odnosno dimenzionalnost ulaza (na slici 14.1 $I = 2$).
- $N^*$ - broj jezičnih izraza po jezičnoj varijabli, odnosno broj funkcija




Slika 14.1: ANFIS mreža koja ostvaruje neizraziti sustav (zaključivanje tipa 3)

pripadnosti za svaku od varijabli (na slici 14.1 $N^* = 2$ – i za $x$ i za $y$ imamo po dvije funkcije pripadnosti koje treba naučiti).

- $N$ - ukupni broj jezičnih izraza odnosno funkcija pripadnosti koje treba naučiti u antecedent dijelu pravila za čitav sustav; $N = N^* \cdot I$ (na slici 14.1 $N = 4$).

- $R$ - broj pravila kojima raspolaže sustav; pretpostavka je da vrijedi $R = N^*$ (tako na slici 14.1 postoje upravo dva pravila; prvo gleda neizrazite skupove $A_1$ i $B_1$, drugo neizrazite skupove $A_2$ i $B_2$). Alternativna izvedba mogla bi definirati po jedno pravilo za svaku moguću kombinaciju neizrazitih skupova na ulazu čime bi nastalo $R = N^{*I}$ pravila.

Također, pretpostavka je da se kao funkcije pripadnosti neizrazitih skupova na ulazu koriste funkcije pripadnosti oblika:

$$\mu_{A_i}(x) = e^{-\left[\left(\frac{x-c_i}{a_i}\right)^2\right]^{b_i}}$$

čiji su parametri $a_i$, $b_i$ i $c_i$ dok se u konsekvens dijelu koriste linearne kombinacije ulaza pa za pravilo $r$ imamo izlaz definiran kao:

$$f_r(x_1, x_2, \ldots, x_I) = \xi_{r,1}x_1 + \xi_{r,2}x_2 + \ldots + \xi_{r,I}x_I + \xi_{r,I+1}$$

gdje je $\vec{x} = (x_1, \ldots, x_I)$ ulazni podatak a $\xi_{r,1}$, …, $\xi_{r,I+1}$ skup nepoznatih parametara pravila $r$ koje treba naučiti. Ukupno postoji $N$ parova parametara ($a_i$,$b_i$,$c_i$) te $R \cdot (I + 1)$ parametara u konsekvens dijelu koje treba naučiti.

Struktura čestice PSO algoritma opisana u ovom radu sastoji se od nekoliko skupova parametara. Prvi skup parametara čine svi $a_i$-ovi; sljedeći skup parametara čine svi $b_i$-ovi, sljedeći skup parametara čine svi $c_i$-ovi (svaki od ova tri skupa ima po $N$ elemenata). Potom slijedi još $I+1$ skup pri čemu $j$-ti skup sadrži sve parametre $\xi_{r,j}$ ($r = \{1, \ldots, R\}$). Svaka različita vrsta parametara smještena je u zaseban skup kojih ukupno ima $I+4$.

### 14.1.1 Općenita struktura algoritma PSO

Algoritam PSO razvijen je u nekoliko inačica, i ovdje ćemo opisati samo onu koja se koristi u spomenutome radu.

1. Inicijaliziraj sve čestice u roju $P(t)$ na slučajno odabrane pozicije $\vec{x}_i(t)$ (na početku $t = 0$, $P_i \in P(t)$).

2. Izračunaj dobrotu $F(\vec{x}_i(t))$ svih čestica koristeći njihovu trenutnu poziciju $\vec{x}_i(t)$.

3. Usporedi trenutnu dobrotu svake jedinke s njezinim najboljim rješenjem; ako je trenutna dobrota bolja, ažuriraj njezino najbolje rješenje tako da je $pbest_i = F(\vec{x}_i(t))$, $\vec{x}_{pbest_i} = \vec{x}_i(t)$.

4. Usporedi trenutnu dobrotu svake jedinke s globalno najboljim rješenjem; ako je trenutna dobrota bolja, ažuriraj globalno najbolje rješenje tako da je $gbest = F(\vec{x}_i(t))$, $\vec{x}_{gbest} = \vec{x}_i(t)$.

5. Korigiraj brzinu svake čestice prema izrazu:

$$\vec{v}_i(t) = \vec{v}_i(t-1) + \vec{r}_1 \cdot C_1 \cdot (\vec{x}_{pbest_i} - \vec{x}_i(t)) + \vec{r}_2 \cdot C_2 \cdot (\vec{x}_{gbest} - \vec{x}_i(t))$$

   gdje su $\vec{r}_1$ i $\vec{r}_2$ vektori jednake dimenzionalnosti kao i $\vec{x}_i$ pri čemu su im sve komponente generirane nasumično iz intervala $[0, 1]$ (vektori se generiraju svaki puta nanovo); $C_1$ je komponenta individualnosti čestice a $C_2$ socijalna komponenta i uobičajeno se postavljaju na vrijednost 2.

6. Pomakni čestice prema izrazu:

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t), \quad t = t+1.$$

7. Ponavljaj postupak od koraka 2 do konvergencije.

### 14.1.2 Modifikacija algoritma PSO

Za potrebe učenja opisane mreže ANFIS autori rada predlažu modifikaciju opisanog algoritma kako bi se ubrzala konvergencija. Ideja ove modifikacije je spojiti operatore genetskog algoritma i opisani algoritam PSO na sljedeći način. Nakon svake iteracije algoritma PSO odabire se $i$-ta čestica koja ima

najgoru vrijednost $pbest_i$ i ona se izbacuje iz roja te se za nju traži zamjena. Iz ostatka roja slučajno se odabiru dvije čestice i uporabom operatora križanja stvaraju se dva djeteta koja se potom vrednuju. Dijete koje je lošije se odbacuje a preostalo dijete nadomješta najgoru česticu koja je prethodno izbačena iz roja.

### 14.1.3 Postupak učenja mreže ANFIS

Postupak učenja mreže provodi se ciklički, uporabom modificiranog algoritma PSO. Pri tome se u svakoj iteraciji uče parametri iz samo jednog skupa dok se svi ostali parametri drže fiksnima. Primjerice, u prvoj iteraciji dozvoljava se mijenjanje samo parametara $a_i$; u drugoj iteraciji samo mijenjanje parametara $b_i$ i tako redom.

Rad opisanog algoritma ispitan je na dva primjera od kojih jedan prenosimo u nastavku (slika 14.2). Slika 14.2a prikazuje funkciju koju je sustav trebao naučiti aproksimirati (puna linija) te odziv naučenog sustava ANFIS (točkana linija) – kako se može primjetiti sa slike, sustav ANFIS vrlo dobro aproksimira zadanu funkciju. Radi se o funkciji od 5 varijabli. Za aproksimaciju je korišten sustav ANFIS uz sljedeće parametre: $I = 5$, $N^* = 2$, $N = 5 \times 2 = 10$. Korišteno je 1000 uzoraka za učenje, 15 čestica u roju te 100 epoha za svaku populaciju.

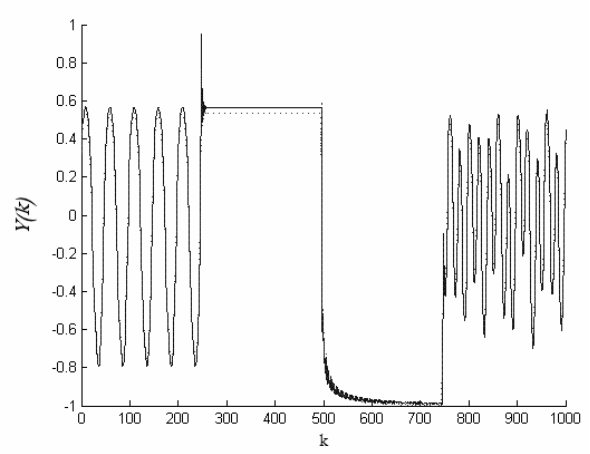
Slika 14.2b pokazuje kretanje pogreške prilikom učenja opisanim algoritmom PSO dok je slika 14.2c dana samo kao usporedba pogreške koja se dobije učenjem klasičnim gradijentnim spustom. Kako se može vidjeti, ukupna pogreška učenja veća je kod sustava učenog gradijentnim spustom i to za više redova veličine što ukazuje da je gradijentni spust zapeo u relativno nepovoljnom lokalnom optimumu dok je PSO algoritam uspio pronaći puno bolje rješenje.

## 14.2 Učenje sustava ANFIS uporabom simbiotskog adaptivnog algoritma PSO

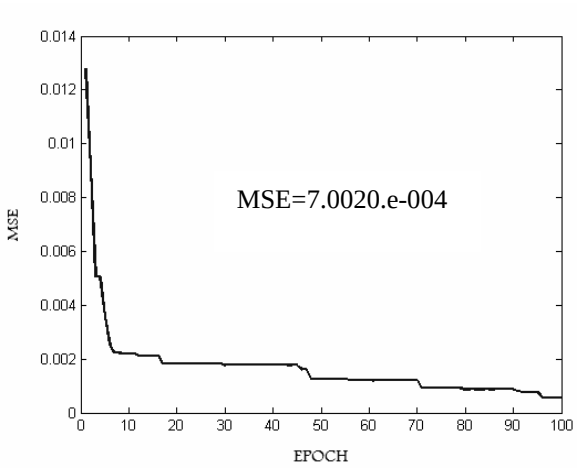
I u okviru ovog podpoglavlja opet ćemo pogledati primjenu algoritma PSO na učenje sustava ANFIS; međutim, pristup koji je izvorno iznesen u [Lin et al.(2011)Lin, Peng, and Lee] dovoljno je različit i interesantan da ga se isplati pobliže istražiti.

U podpoglavlju 14.1 opisani algoritam PSO koristio je roj čestica gdje je svaka čestica predstavljala jedan kompletan sustav ANFIS. Čestice su bile vrednovane na način da se provjerio rad sustava ANFIS kojeg čestica predstavlja i temeljem tih podataka obavljalo se ažuriranje čestica.

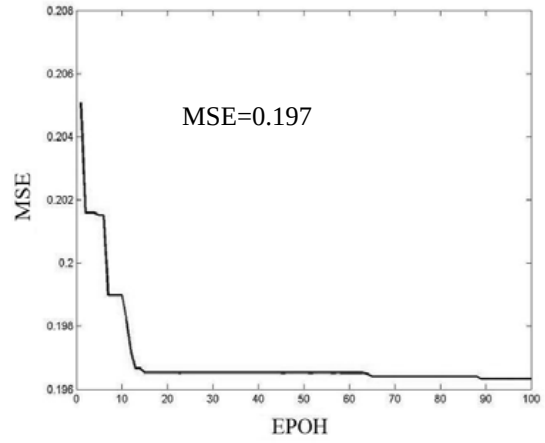
Za razliku od takvog pristupa, kod algoritma *Symbiotic adaptive particle swarm optimization* (SAPSO) koriste se pojmovi podčestice i čestice. Svaka podčestica predstavlja jedno pravilo sustava. Primjerice, roj koji se sastoji




(a) Funkcija




(b) Kretanje pogreške pri učenju s PSO




(c) Kretanje pogreške pri učenju s GS

Slika 14.2: Rezultat učenja sustava ANFIS

od ukupno 20 podčestica predstavlja 20 različitih pravila. Pretpostavimo sada da je zadaća ovog algoritma pronaći parametre sustava ANFIS koji se sastoji od ukupno $R$ pravila (npr. neka je $R = 8$). Također pretpostavimo da algoritam radi s $N$ čestica (npr. $N = 100$). Algoritam SAPSO iterativno gradi svaku od 100 čestica tako da nasumice bira po 8 podčestica (odnosno pravila) iz roja i takav skup od 8 podčestica tretira kao jedan sustav (česticu) koji potom vrednuje.

### 14.2.1 Specifikacija sustava ANFIS

I ovaj algoritam primijenjen je na učenje sustava ANFIS koji koristi zaključivanje tipa 3 (TSK). Algoritam pretpostavlja da se gradi sustav koji dobiva $n$-dimenzijski ulaz te koristi pravila oblika:

`Ako` $x_1$ `je` $A_{1j}(m_{1j}, \sigma_{1j})$ `i ... i ako` $x_n$ `je` $A_{nj}(m_{nj}, \sigma_{nj})$
`tada` $y_j = w_{0j} + x_1 w_{1j} + \ldots + x_n w_{nj}$.

Neizraziti skupovi su pri tome zvonolike funkcije čija je funkcija pripadnosti definirana sljedećim izrazom:

$$\mu_{A_{ij}} = exp\left(-\frac{(x_i - m_{ij})^2}{\sigma_{ij}^2}\right).$$

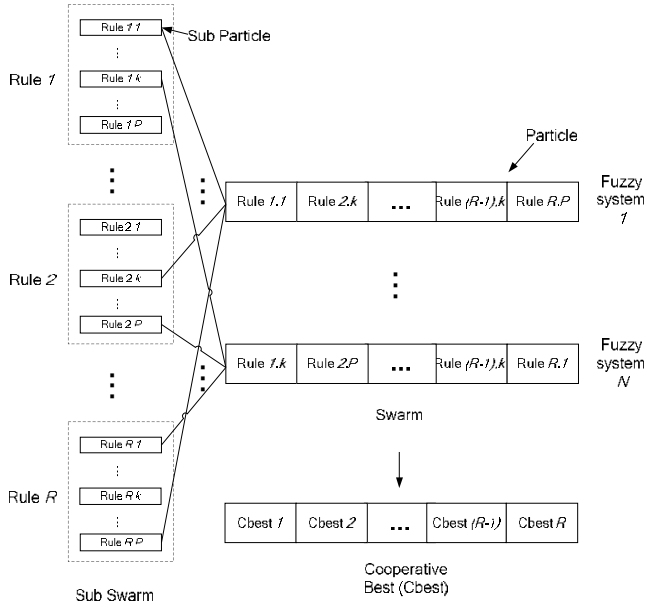
Svako pravilo stoga u antecedent dijelu sadrži ukupno $2n$ parametara ($m_{1j}$,$\sigma_{1j}$)-($m_{nj}$,$\sigma_{nj}$) te u konsekvens dijelu još $n+1$ parametar ($w_{1j}$, ..., $w_{nj}$ te $w_{0j}$). Jedna podčestica stoga se prikazuje kao $2n + (n+1) = 3n+1$ dimenzijski vektor, kako je to prikazano na slici 14.3.

### 14.2.2 Opis primijenjenog algoritma PSO

Algoritam SAPSO, za razliku od klasičnog algoritma PSO ne koristi samo jedan roj. Ako je zadaća algoritma izgraditi sustav ANFIS koji se sastoji od $R$ pravila, algoritam SAPSO koristit će $R$ podrojeva (engl. *subswarms*). Svaki podroj sadržavat će pri tome $P$ podčestica, kako je to prikazano na slici 14.4. Ideja ovog pristupa je, krenuvši od činjenice da algoritma mora izgraditi sustav ANFIS koji se sastoji od $R$ pravila, osigurati upravo $R$ podrojeva čija je zadaća da svaki zasebno pronađe najbolje moguće pravilo koje će biti uključeno u konačni sustav.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $m_{1j}$ | $\sigma_{1j}$ | $m_{2j}$ | $\sigma_{2j}$ | ... | $m_{nj}$ | $\sigma_{nj}$ | ... | $w_{0j}$ | $w_{1j}$ | ... | $w_{nj}$ |

Slika 14.3: Struktura podčestice koja odgovara jednom pravilu sustava ANFIS




Slika 14.4: Struktura rojeva algoritma SAPSO

Opisana situacija jasno je prikazana na slici 14.4: u gornjem lijevom dijelu prikazan je podroj koji se sastoji od $P$ podčestica koje su kandidati za pravilo 1, ispod njega je podroj koji se opet sastoji od $P$ podčestica koje su kandidati za pravilo 2, itd. Čitav algoritam SAPSO radi s ukupno $R \times P$ podčestica.

Kako svaka podčestica za sebe ne predstavlja kompletan sustav, sljedeći korak je temeljem kandidata za svako od $R$ pravila nasumice odabrati po jedno pravilo iz svakog podroja čime se gradi jedna čestica odnosno jedan kompletan sustav ANFIS; ovo se ponavlja $N$ puta kako bi se izgradilo $N$ čestica. Svaka se čestica potom vrednuje i usporeduje s najboljom ikad pronađenom česticom *cbest*; ako je pronađeni sustav bolji, *cbest* se ažurira.

Pitanje na koje još treba odgovoriti jest kako se radi ažuriranje čestica? Odgovor je – nikako. Čestice su umjetna tvorevina koja se gradi kombiniranjem podčestica iz podrojeva, i upravo su podčestice ono što treba ažurirati; problem koji stoga treba riješiti je kako vrednovati dobrotu svake podčestice kad podčestica sama za sebe ne predstavlja kompletan sustav koji je moguće vrednovati. Da bismo to riješili, treba uočiti da svaka podčestica sudjeluje u jednoj ili više čestica koje su izgrađene slučajnim odabirom (odabir treba raditi tako da svaka čestica sigurno bude uključena u nekoliko čestica). Dobrota svake podčestice stoga se računa kao prosječna dobrota čestica koje uključuju tu podčesticu, i temeljem te informacije radi se ažuriranje podčestica. Detaljniji opis algoritma dan je u nastavku.

1. korak: kodiranje podčestica. Podčestice treba izgraditi na način da predstavljaju jedno pravilo, tj. da uključuju sve potrebne parametre kako bi se mogle protumačiti kao jedno pravilo. To uključuje sve parametre odabranih funkcija pripadnosti te eventualno parametre $t$-norme koja se koristi za izračun jakosti paljenja pravila u antecedentu, te sve parametre koji određuju konsekvens.

2. korak: inicijalizacija. Podčestice u svim podrojevima se inicijaliziraju na slučajne vrijednosti unutar prostora koji se pretražuje.

3. korak: dodjela dobrote podčesticama.

   (a) Postavi akumulator dobrote za svaku podčesticu na 0.

   (b) Posredstvom slučajnog mehanizma odaberi po jednu podčesticu iz svakog podroja i sve ih spoji u česticu koja tada predstavlja jedan kompletan sustav ANFIS.

   (c) Vrednuj tako nastalu česticu (tj. sustave ANFIS). Primjerice, ako se sustav trenira nad $T$ uzoraka za učenje, funkcija pogreške može se definirati kao:

   $$E = \frac{1}{T}\sum_{i=1}^{T}(y_i - o_i)^2$$

   gdje je $y_i$ željena vrijednost izlaza za uzorak $i$ a $o_i$ dobivena vrijednost izlaza promatranog sustava ANFIS. Dobrota sustava tada se može definirati kao:

   $$fittnes = \frac{1}{1+\sqrt{E}}.$$

   (d) Za svaku podčesticu koja je uključena u česticu dodaj dobrotu čestice u njezin akumulator dobrote.

   (e) Ponavljaj postupak stvaranja čestica dok svaka čestica nije uključena dovoljan broj puta u čestice, i za svaku podčesticu pamti u koliko je čestica bila uključena.

   (f) Za svaku podčesticu podijeli vrijednost u njezinom akumulatoru dobrote s brojem čestica u koji je ta podčestica bila uključena.

4. korak: ažuriranje podčestica. Jednom kada je svakoj podčestici dodijeljena njezina dobrota, istovemeno se za svaku podčesticu ažurira njezin *pbest* te za pripadni podroj *gbest*.

5. korak: ažuriranje kooperativnog najboljeg rješenja *cbest*: nakon što se izračuna dobrota svake čestice (tj. pripadnog sustava ANFIS), ta se dobrota uspoređuje s najboljim kooperativnim rješenjem *cbest* i ako je dobrota čestice bolja, ažurira se *cbest* i pamti pridružena najbolja čestica.

Ovi koraci ponavljaju se sve dok nije zadovoljen uvjet za prestanak rada algoritma, kada se kao pronađeno rješenje vraća *cbest*.

Ažuriranje brzine svake podčestice radi se prema izrazu:

$$\begin{aligned}\vec{v}_i(k+1) = \omega \cdot \vec{v}_i(k) \\ + \phi_1 \cdot rand() \cdot (pbest - \vec{x}_i(k)) \\ + \phi_2 \cdot rand() \cdot (gbest - \vec{x}_i(k)) \\ + \phi_3 \cdot rand() \cdot (cbest - \vec{x}_i(k)).\end{aligned}$$

gdje je $\omega$ inercijski faktor, $\phi_1$ faktor individualnosti, $\phi_2$ socijalni faktor a $\phi_3$ faktor zajednice. Nakon toga, pozicije se ažuriraju prema izrazu:

$$\vec{x}_i(k+1) = \vec{x}_i(k) + \vec{v}_i(k+1).$$

U radu [Lin et al.(2011)Lin, Peng, and Lee] predlaže se da se umjesto *pbest* za svaku podčesticu prati *vbest* vrijednost – najbolje pronađeno rješenje susjedstva te podčestice. Susjedstvo pri tome nije definirano na uobičajen način kao topološko susjedstvo, već se za svaku podčesticu u svakom koraku susjedstvo utvrđuje dinamički direktno u prostoru pretraživanja na način da se traže sve podčestice koje su u prostoru pretraživanja dovoljno blizu promatranoj podčestici. Takav pristup relativno je skup jer mu je za $P$ podčestica složenost $o(P^2)$. Za svaku se podčesticu gledaju udaljenosti do svih drugih podčestica i pronalazi se $dist_{max}$ kao najveća udaljenost. Potom se, da bi se odlučilo pripada li podčestica $j$ u susjedstvo podčestice $i$ gleda omjer udaljenosti $d_{ij} = \frac{dist(i,j)}{dist_{max}}$. Definira se prag $l = 0.5 + 0.5 \cdot \frac{iter}{iter_{max}}$ gdje je $iter$ trenutna iteracija algoritma a $iter_{max}$ maksimalni broj iteracija koliko će se dopustiti algoritmu da radi. Uočimo, $l$ se stoga mijenja linearno od 0.5 na početku pa sve do 1 na samom kraju. Uzima se da podčestica $j$ pripada u susjedstvo podčestice $i$ ako je $d_{ij} < l$. Ovakav način definiranja susjedstva osigurat će da u ranim fazama rada algoritma podčestice komuniciraju samo s bliskim podčesticama, a prema kraju rada algoritma susjedstvo će se širiti ka globalnom susjedstvu gdje će svaka podčestica u podroju imati pristup najboljem rješenju čitavog podroja.

Detalji o stabilnosti rada ovog algoritma kao i o provedenim eksperimentima mogu se pogledati direktno u [Lin et al.(2011)Lin, Peng, and Lee].

# Indeks

# Bibliografija


[Abraham(2005)] A. Abraham. Adaptation of fuzzy inference system using neural learning. *Studies in Fuzziness and Soft Computing*, 181(13):53–83, 2005.

[Battiti(1989)] R. Battiti. Accelerated backpropagation learning: Two optimization methods. *Complex systems*, 3(4):331–342, 1989.

[Blickle and Thiele(1995)] T. Blickle and L. Thiele. A mathematical analisys of tournament selection. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 2–8, San Francisco, CA, 1995.

[Bäck(1994)] T. Bäck. Selective pressure in evolutionary algorithms: A characterization of selection mechanisms. In *International Conference on Evolutionary Computation*, pages 57–62, 1994. URL `http://dblp.uni-trier.de/db/conf/icec/icec1994-1.html#Back94`.

[Bäck(1995)] T. Bäck. Generalized convergence models for tournament- and (mu, lambda)-selection. In L. J. Eshelman, editor, *ICGA*, pages 2–8. Morgan Kaufmann, 1995. ISBN 1-55860-370-0. URL `http://dblp.uni-trier.de/db/conf/icga/icga1995.html#Back95`.

[Cantú-Paz(2001)] E. Cantú-Paz. Migration policies, selection pressure, and parallel evolutionary algorithms. *J. Heuristics*, 7(4):311–334, 2001. URL `http://dblp.uni-trier.de/db/journals/heuristics/heuristics7.html#Cantu-Paz01`.

[Corcoran and Wainwright(1992)] A. Corcoran and R. Wainwright. A genetic algorithm for packing in three dimensions. *Applied computing*, 2: 1021–1030, 1992. URL `ftp://ftp.santafe.edu/pub/wgm/hard.ps`.

[Craig Veitch and Holmes(1991)] A. Craig Veitch and G. Holmes. A modified quickprop algorithm. *Neural Computing*, 3(3):310–311, 1991.

[Fahlman(1988a)] S. E. Fahlman. An empirical study of learning speed in back-propagation networks. 1988a.

[Fahlman(1988b)] S. E. Fahlman. Faster-learning variations on back-propagation: An empirical study. 1988b.

[Fahlman(1989)] S. E. Fahlman. Faster-learning variations on back-propagation: An empirical study. In D. S. Touretzky, G. E. Hinton, and T. J. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 38–51. San Francisco, CA: Morgan Kaufmann, 1989.

[Fogel(1999)] D. Fogel. Some recent important foundational results in evolutionary computation. In K. Miettinen, P. Neittaanmaki, M. Makela, and J. Periaux, editors, *Evolutionary Algorithms in Engineering and Computer Science*. John Wiley & sons, LTD, 1999.

[Fogel et al.(1966)Fogel, Owens, and Walsh] L. Fogel, A. Owens, and M. Walsh. *Artificial intelligence through simulated evolution*. Wiley, Chichester, WS, UK, 1966.

[Ghomsheh et al.(2007)Ghomsheh, Shoorehdeli, and Teshnehlab] V. S. Ghomsheh, M. A. Shoorehdeli, and M. Teshnehlab. Training anfis structure with modified pso algorithm. In *Proceedings of the 2007 Mediterranean Conference on Control Automation*, pages 1–6. IEEE, 2007.

[Golub(2004a)] M. Golub. Genetski algoritam, drugi dio, 2004a. URL `http://www.zemris.fer.hr/\textasciitilde{}golub/ga/ga\_skripta1.pdf`.

[Golub(2004b)] M. Golub. Genetski algoritam, prvi dio, 2004b. URL `http://www.zemris.fer.hr/\textasciitilde{}golub/ga/ga\_skripta2.pdf`.

[Gruau(1992)] F. Gruau. Genetic synthesis of boolean neural networks with a cell rewriting developmental process. *Combinations of Genetic Algorithms and Neural Networks, 1992., COGANN-92. International Workshop on*, pages 55–74, 6 Jun 1992. doi: 10.1109/COGANN.1992.273948.

[Hagan and Menhaj(1994)] M. Hagan and M. Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, 5:989–993, 1994. ISSN 1045-9227.

[Hancock(1992)] P. J. B. Hancock. Genetic algorithms and permutation problems: A comparison of recombination operators for neural net structure specification. In D. Whitley and J. Schaffer, editors, *International Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN-92)*. IEEE Computer Society Press, 1992.

[Hayashi and Buckley(1994)] Y. Hayashi and J. J. Buckley. Approximations between fuzzy expert systems and neural networks. *Int. J. Approx. Reasoning*, 10(1):63–73, 1994.

[Hirota and Pedrycz(1994)] K. Hirota and W. Pedrycz. Or/and neuron in modeling fuzzy set connectives. *IEEE Transactions on Fuzzy Systems*, 2(2):151–161, 1994.

[Holland(1975)] J. Holland. *Adaptation in natural and artificial systems.* The University of Michigan Press, Ann Arbor, 1975.

[Holland(1976)] J. H. Holland. Adaptation. In R. Rosen and F. Snell, editors, *Progress in theoretical biology*, volume 4, pages 263–293. Academic Press, New York, 1976.

[Igel and Hüsken(2000)] C. Igel and M. Hüsken. Improving the rprop learning algorithm. In *Proceedings of the second international ICSC symposium on neural computation (NC 2000)*, pages 115–121. Citeseer, 2000.

[Jacobs(1988)] R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural networks*, 1(4):295–307, 1988.

[Jang and Sun(1995)] J.-S. R. Jang and C.-T. Sun. Neuro-fuzzy modeling and control. In *PROCEEDINGS OF THE IEEE*, volume 83, pages 378–406. IEEE, 1995.

[Kitano(1990)] H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4:461–476, 1990.

[Koza(1990)] J. Koza. Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. Technical Report STAN-CS-90-1314, Dept. of Computer Science, Stanford University, June 1990. URL `ftp://reports.stanford.edu/pub/cstr/reports/cs/tr/90/1314/CS-TR-90-1314.pdf`.

[Koza(1994)] J. Koza. *Genetic programming II.* Complex adaptive systems. MIT Press, Cambridge, Mass, 1994. ISBN 0262111896. URL `http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+186785119&sourceid=fbw_bibsonomy`.

[Koza(1992)] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, USA, 1992. ISBN 0-262-11170-5.

[Koza et al.(1999)Koza, Andre, Bennett, and Keane] J. R. Koza, D. Andre, F. H. Bennett, and M. A. Keane. *Genetic Programming III: Darwinian Invention & Problem Solving.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999. ISBN 1558605436.

[Lin et al.(2011)Lin, Peng, and Lee] C.-J. Lin, C.-C. Peng, and C.-Y. Lee. Identification and prediction using neuro-fuzzy networks with symbiotic adaptive particle swarm optimization. *Informatica*, 35:113–122, 2011.

[Macready and Wolpert(1996)] W. G. Macready and D. H. Wolpert. What makes an optimization problem hard? *Complexity*, 5:40–46, 1996. URL `ftp://ftp.santafe.edu/pub/wgm/hard.ps`.

[Miller and Goldberg(1995)] B. Miller and D. Goldberg. Gas tournament selection and the effects of noise, 1995. URL `http://www.dai.ed.ac.uk/groups/evalg/Local\_Copies\_of\_Papers/Miller.Goldberg.GAs\_Tournament\_Selection\_and\_the\_Effects\_of\_Noise.ps.gz`.

[Miller et al.(1989)Miller, Todd, and Hedge] G. Miller, P. Todd, and S. Hedge. Designing neural networks using genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 379–384. Morgan Kauffman, 1989.

[Mjolsness et al.(1989)Mjolsness, Sharp, and Alpert] E. Mjolsness, D. H. Sharp, and B. K. Alpert. Scaling, machine learning, and genetic neural nets. *Advances in Applied Math.*, 10:137–163, 1989.

[Moller(1990)] M. F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. Technical Report PB-339, Computer Science Department, University of Aarhus, Aarhus, Denmark, 1990.

[Montana and Davis(1989)] D. J. Montana and L. Davis. Training feedforward neural networks using genetic algorithms. In *Proceedings of the 11th international joint conference on Artificial intelligence - Volume 1*, pages 762–767, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc. URL `http://dl.acm.org/citation.cfm?id=1623755.1623876`.

[Pedrycz(1993)] W. Pedrycz. *Fuzzy Control and Fuzzy Systems*. John Wiley & Sons, New York, 1993.

[Pedrycz and Rocha(1993)] W. Pedrycz and A. Rocha. Fuzzy-set based models of neurons and knowledge-based networks. *IEEE Transactions on Fuzzy Systems*, 1(4):254–266, 1993.

[Rechenberg(1971)] I. Rechenberg. *Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. PhD thesis, TU Berlin, 1971.

[Riedmiller and Braun(1993)] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Neural Networks, 1993., IEEE International Conference on*, pages 586–591. IEEE, 1993.

[Rumelhart et al.(1986)Rumelhart, Hinton, and Williams] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland,

and P. R. Group, editors, *Parallel distributed processing: explorations in the microstructure of cognition*, volume 1: foundations, pages 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X. URL `http://portal.acm.org/citation.cfm?id=104293`.

[Schwefel(1974)] H.-P. Schwefel. *Numerische Optimierung von Computer-Modellen*. PhD thesis, TU Berlin, 1974.

[Suratgar et al.(2005)Suratgar, Tavakoli, and Hoseinabadi] A. A. Suratgar, M. B. Tavakoli, and A. Hoseinabadi. Modified levenberg-marquardt method for neural networks training, 2005.

[Tollenaere(1990)] T. Tollenaere. Supersab: fast adaptive back propagation with good scaling properties. *Neural networks*, 3(5):561–573, 1990.

[Wolpert(1996)] D. H. Wolpert. No free lunch theorems for optimization, 1996. URL `ftp://ftp.santafe.edu/pub/dhw\_ftp/nfl.search.published.ps.Z`.

[Wolpert and Macready(1996)] D. H. Wolpert and W. G. Macready. No free lunch theorems for search. Tech. Rep. No. SFI-TR-95-02-010, Santa Fe Institute, Santa Fe, NM, 1996. URL `ftp://ftp.santafe.edu/pub/wgm/nfl.ps`.

[Čupić(2012)] M. Čupić. Prirodom inspirirani optimizacijski algoritmi. metaheuristike., 2012. URL `http://java.zemris.fer.hr/nastava/pioa`.