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:

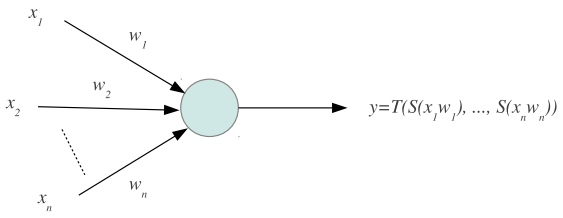
- ulazi i težine su realni brojevi iz intervala [0,1],

- 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.

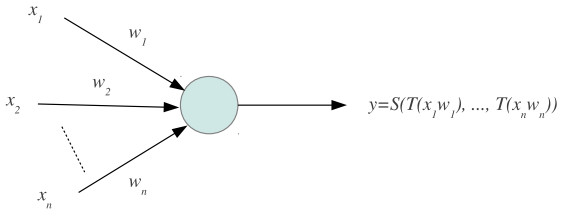
$$y = T(S(x_1, w_1), S(x_2, w_2), ...)$$



Slika 12.1: Neizraziti-I neuron

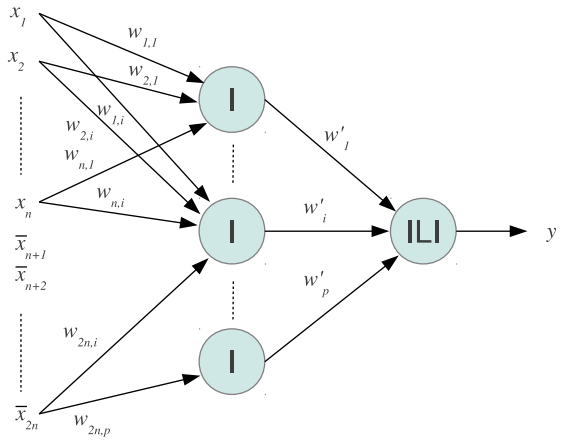
Neizraziti-ILI neuron (slika 12.2) je neuron kod kojeg se ulazi x_i i težine w_i kombiniraju se odabranom T-normom. Izlaz je agregacija ovih kombinacija uporabom S-norme.

$$y = S(T(x_1, w_1), T(x_2, w_2), ...)$$

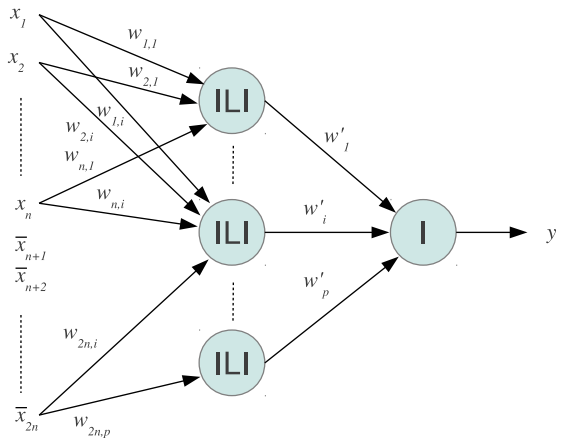


Slika 12.2: Neizraziti-ILI neuron

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

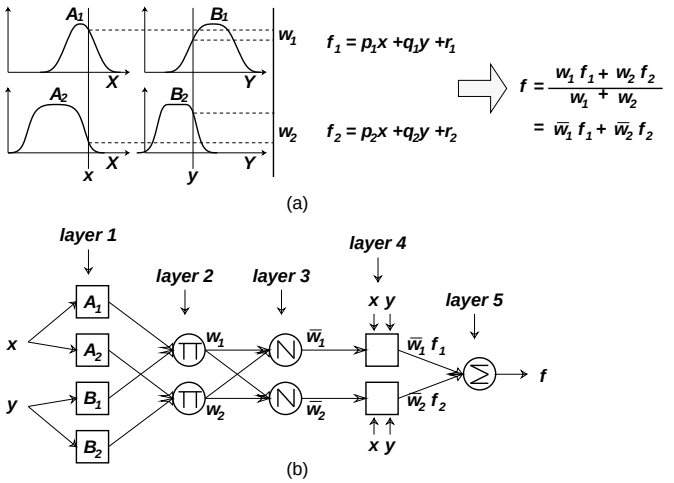
Sustav ANFIS (engl. *Adaptive Neuro-Fuzzy Inference System*) izvorno opisan u [Jang and Sun(1995)] pripada pod hibridni pristup kombiniranja neuronskih mreža i sustava neizrazitog zaključivanja. Kod ovog sustava neuronska mreža je istovremeno i sustav neizrazitog zaključivanja.

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).

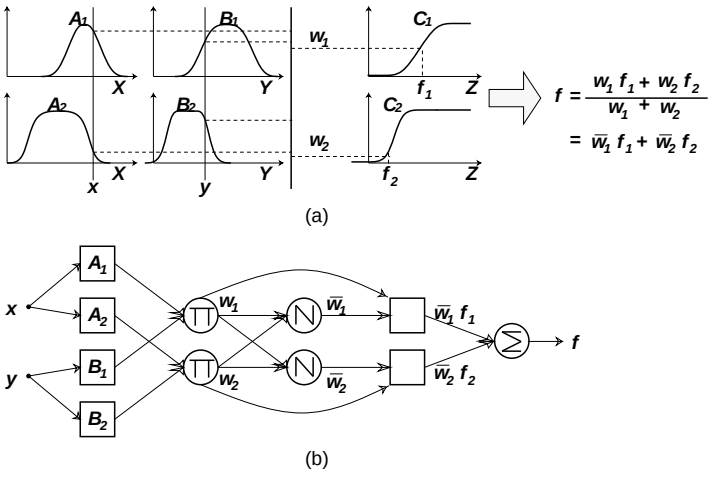
Drugi primjer neizrazitih pravila su pravila oblika:
`Ako` *brzina* `je` *velika*, `tada` *sila* `=` $k \cdot$ *brzina*^{2}.
Za razliku od prethodnog oblika pravila, ovdje se kao konsekvens nalazi izraz (formula) koja temeljem trenutnih jasnih (engl. *crisp*) ulaza određuje vrijednost izlaza koja je, 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.

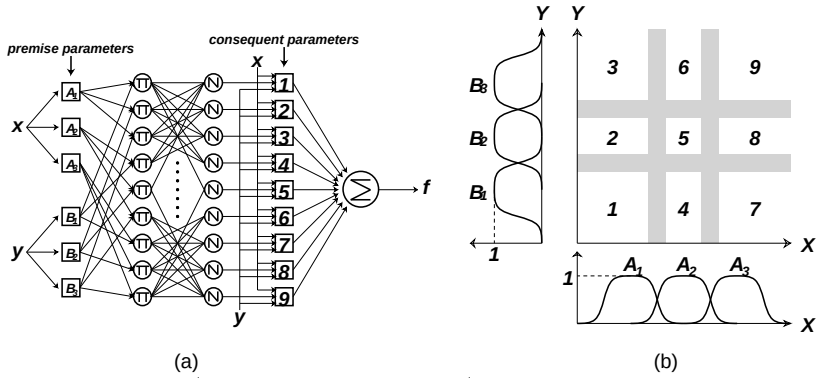
Svaki od ova tri oblika zaključivanja kao podlogu imaju općenitu strukturu sustava neizrazitog zaključivanja koja je prikazana na slici 12.8. Takav



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)

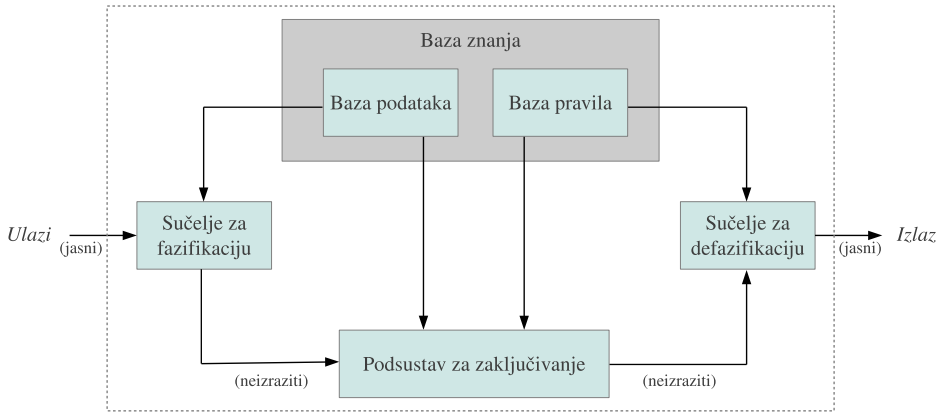
sustav sastoji se od sljedećih dijelova:

- *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 ψ 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 α_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 ξ_1, ξ_2, ..., ξ_r, ξ_{r+1}, ..., ξ_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 α_1, ..., α_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 ξ_1, ..., ξ_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 ξ_1, ..., ξ_r. Primjetimo, međutim, da to nisu globalno optimalne vrijednosti – to su optimalne vrijednosti za koje, uz fiksirane parametre ξ_{r+1}, ..., ξ_s funkcija f poprima minimum.

2. U drugom koraku algoritam gradijentnog spusta primjenjuje se samo na parametre ξ_{r+1}, ..., ξ_s dok se parametri ξ_1, ..., ξ_r drže fiksnima. Kako se time parametri ξ_{r+1}, ..., ξ_s mijenjaju, vrijednosti parametara ξ_1, ..., ξ_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.