SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 403

Raspoznavanje prometnih znakova neuronskim mrežama

Filip Krnjić

Zagreb, siječanj 2009.

Sadržaj

1. Uvod

Ljudi su od davnina težili učiniti svoje živote jednostavnijima tako da automatiziraju određene radnje. Tako su počeli izrađivati strojeve koji su im olakšavali rad, no radili su samo one stvari za koje su bili namijenjeni. Nakon toga ljudi su izmislili računala koja su mogla raditi mnogo složenije poslove no i ona su bila limitirana u pogledu da nisu mogla obavljati rad za koji nisu bili namjenjeni. Razvijanjem algoritama i programa koji će računalo od jednostavnih strojeva pretvoriti u strojeve kakve danas poznajemo došlo se do stvaranja nove grane u računarstvu nazvane umjetna inteligencija**.** Ona se bavila izrađivanjem algoritama i programa koji bi računalima pomogli da „uče“ i postaju sposobni za samostalno odlučivanje.

Nezavisno o tim tehničkim dostignućima, u području biologije i psihologije počela su istraživanja o ljudima i ljudskom mozgu. Došlo se do spoznaje kako je ljudski mozak puno napredniji i inteligentniji sustav od bilo kojeg računala do tada napravljenog, a i do danas. Daljnjim istraživanjima spoznalo se kako je ljudski mozak građen od velikog broja neurona koji povezivanjem čini neuronsku mrežu. Oponašanjem neuronske mreže iz mozga razvila se grana umjetne inteligencije koju nazivamo umjetne neuronske mreže (engl. Artificial Neural Network, ANN).

1940. godine McCulloch i Pitts [3] (Massachusetts Institute of Technology) su istraživajući neurofizičke karakteristike živih bića objavili matematički model neuronske mreže u okviru teorije automata. Međutim, procesna moć ondašnjih raučnala nije još bila dorasla implementaciji neuronske mreže. Tek su se u kasnim pedesetima, pojavom računala LSI (Large-Scale Integration), pojavila praktična ostvarenja. No ubrzo se dolazi do zastoja u istraživanju neuronskih mreža zbog problema oko učenja mreže i umalo dolazi do potpunog zaustavljanja. Krajem osamdesetih se neuronske mreže ponovno počinju istraživati, a značajan razlog tomu su dali Rumelhart, Hinton i McClelland [4] algoritmom backpropagation**,** koji se temelji na principu propagacije greške unazad u višeslojnim neuronskim mrežama. Danas su neuronske mreže često korišten koncept pri izgradnji i razvoju inteligentnih sustava, a posebno napreduju zbog same brzine današnjih računala.

Umjetna neuronska mreža je zbir umjetnih neurona koji su međusobno povezani i interaktivni kroz operacije obrade signala. Uređena je po uzoru na ljudski mozak te može imati više ulaza i izlaza. Postoji više vrsta neuronskih mreža koje se razlikuju po arhitekturi, načinu učenja i signalima: perceptron, backpropagation, SOM i Hopfieldova neuronska mreža [6]. U ovom radu detaljnije ćemo razmatrati višeslojnu acikličku mrežu te algoritam učenja backpropagation koji su korišteni i u eksperimentalnom dijelu.

2. Upoznavanje s problemom

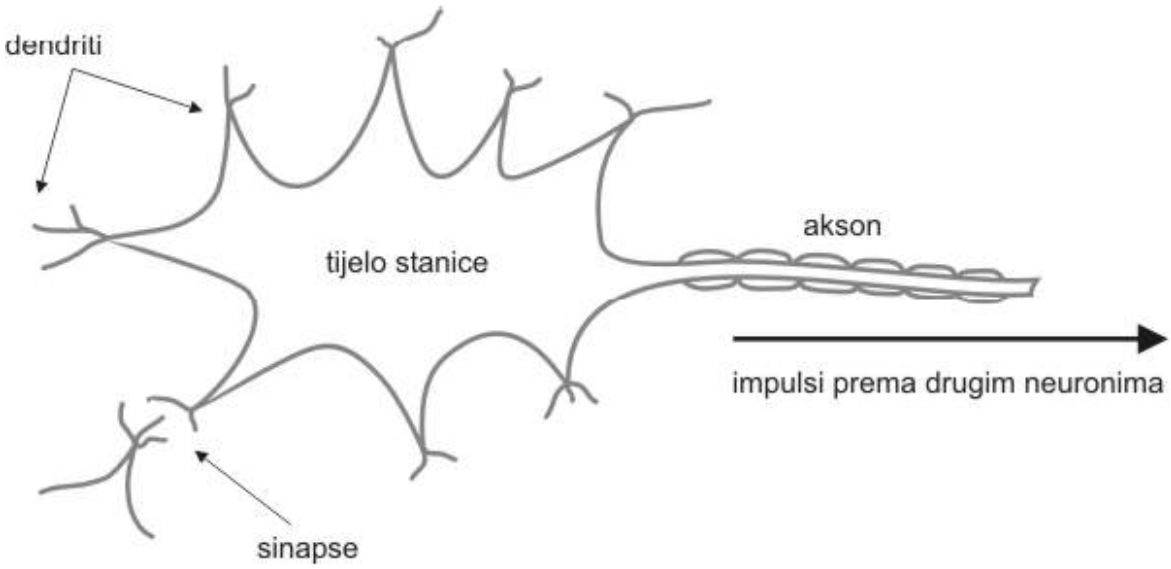
Širi kontekst zadatka je pronalaženje prometnih znakova u slijedu slika koje su pribavljene iz vozila u pokretu. Razmatra se raspoznavanje primjenom neuronskih mreža, pod pretpostavkom da su položaji znakova i njihovih veličina već određeni nekom drugom metodom.

Ovaj rad će se usko baviti problemom kako naučiti neuronsku mrežu da prepozna prometne znakove koristeći algoritam backpropagation, kako da ta ista neronska mreža prepozna nove, dotad neviđene znakove te će opisati koje su metode pritom korištene.

Na kraju treba ispitati program, analizirati rezultate i dobiti odgovor na pitanje može li se neuronskom mrežom postići raspoznavanje slučajnih prometnih znakova.

3. Neuron

Za razumijevanje neuronske mreže važno je upoznati građu njenog sastavnog dijela: neurona. Ljudski mozak je sastavljen od 10^{11} neurona od kojih je svaki povezan s oko 10^4 drugih neurona što ukupno daje $5 \cdot 10^{14}$ veza među neuronima. Pod pretpostavkom da za svaku vezu trebamo 4 bajta dolazi se do procjeni o memoriji ljudskog mozga od $2 \cdot 10^6$ Gb. Četiri su osnovna dijela neurona: tijelo stanice (soma), skup dendrita (ogranaka), akson (dugačke cijevčice koje prenose električne poruke) i niz završnih članaka (slika 3.1.)

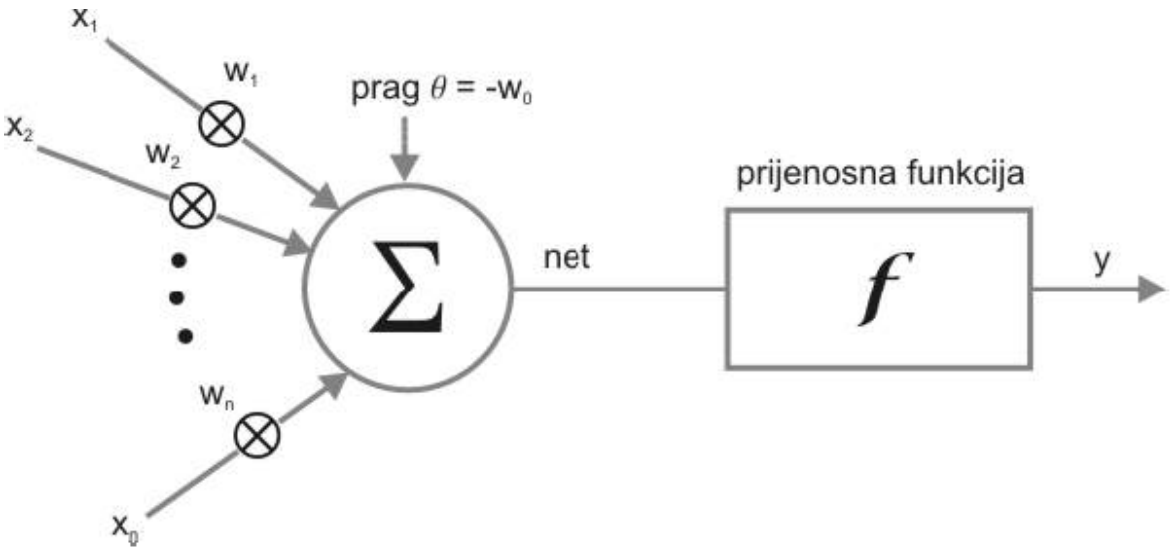


Slika 3.1. Građa neurona [3]

U tijelu stanice nalazi se informacija predstavljena električnim potencijalom između unutrašnjeg i vanjskog dijela stanice koja se s neurona na neuron prenosi sinapsom, spojnim sredstvom dva neurona preko dendrita. Prijenos se odvija kemijskim putem, a informacija koja dođe preko dendrita ispušta kemikaliju koje ako se dovoljno nakupi iz svih dendrita pređe određeni prag i neuron se aktivira tako da generira akcijski potencijal u trajanju od 1 ms. Taj se signal prenosi dalje na druge neurone sve dok se ne dođe do kraja gdje ovisno o tome je li se zadnji neuron aktivirao ili ne dolazi do „odluke mozga“.

Funkcionalnost biološkog neurona imitira model umjetnog neurona koji su predložili McCulloch-Pitts, poznatog pod nazivom Threshold Logic Unit (TLU). Model radi tako da se za ulazne signale koriste numerički iznosi koji množeni s težinskim faktorom veze opisuju jakost sinapse. Ulazi se zatim sumiraju što je analogno sumiranju u tijelu neurona i ako je ta suma viša od praga generira se izlaz.

Osim praga za kreiranje izlaza se može koristiti i prijenosna funkcija koja kao argument koristi sumu ulaza i rezultat te funkcije se šalje na izlaz (slika 3.2.).



Slika 3.2. Umjetni neuron [3]

Ulazi u neuron se označavaju nizom $x_1,x_2,...,x_n$, dok se težine označavaju nizom $w_1,w_2,...,w_n$. Prag se označava sa θ, a njihov ukupni zbroj *net*:

$$net = x_1 \cdot w_1 + x_2 \cdot w_2 + ... + x_n \cdot w_n - \theta \qquad (3.1.)$$

no zbog jednostavnosti se za prag koristi oznaka $-w_0$ te se množi s težinskim faktorom 1:

$$net = x_0 \cdot w_0 + x_1 \cdot w_1 + x_2 \cdot w_2 + ... + x_n \cdot w_n = \sum_{i=0}^{n} (x_i \cdot w_i),\ x_0=1,\ w_0= -\theta \qquad (3.2.)$$

Sada se izlaz neurona može izraziti:

$$y = f(\sum_{i=0}^{n} (x_i \cdot w_i)) = f(net) \qquad (3.3.)$$

3.1. Prijenosna funkcija

U jednadžbi (3.3.) vidjeli smo kako je izlaz *y* rezultat prijenosne funkcije *f* s parametrom sume *net*. Postoje tri glavne vrste prijenosnih funkcija: ADALINE, TLU i sigmoidalna funkcija.

3.1.1. Prijenosna funkcija ADALINE

ADALINE (Adaptive Linear Element) je najjednostavnija prijenosna funkcija. Izlaz funkcije je isti ulazu, tj. sumi ulaza:

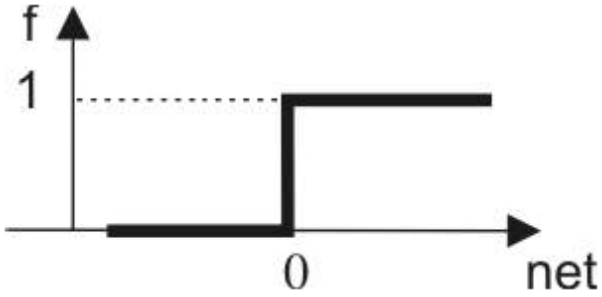
$$f(net) = net \qquad (3.4.)$$

Budući da je funkcija jednostavna, prigodna je za testiranje primjera kod neurona. Loša strana je ta što nema jasno mjesto odjeljenja izlaza.

3.1.2. Prijenosna funkcija TLU

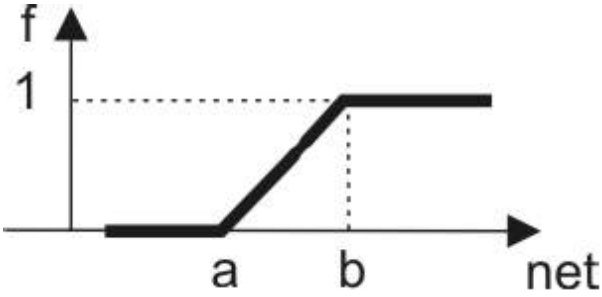
TLU za razliku od funkcije ADALINE ima jasno određenu granicu između dva stanja koja se zadaje definiranjem funkcije. Najčešći oblik funkcije TLU je onaj koji na izlazu daje 0 ako je suma ulaza manja od praga, a 1 ako je suma ulaza veća od praga.

$$f(net) = \begin{cases} 0, net < 0 \\ 1, inače \end{cases} \qquad (3.5.)$$



Također TLU prijenosna funkcija može može biti linearno određena na nekim dijelovima:

$$f(net) = \begin{cases} 0, net \leq a \\ net, a < net < b \\ 1, b \leq net \end{cases} \quad (3.6.)$$



Pošto se za TLU najčešće koristi funkcija 3.5. nastaje problem kod derivacije funkcije zato što funkcija nije derivabilna na intervalu $\{0^-,0^+\}$

3.1.3. Sigmoidalna prijenosna funkcija

Sigmoidalna funkcija je derivabilna verzija funkcije TLU. Kako bi ona to postala morala je izgubiti malo na nagnutosti vertikalnog skoka pošto pravi kut nije moguće u praksi izvesti. Više o važnosti derivacije prilikom procesa učenja neurona može se pogledati u [3]. Prikladniji oblik je eksponencijalna funkcija:

$$f(net) = \frac{1}{1+e^{-a\cdot net}} \qquad (3.7.)$$

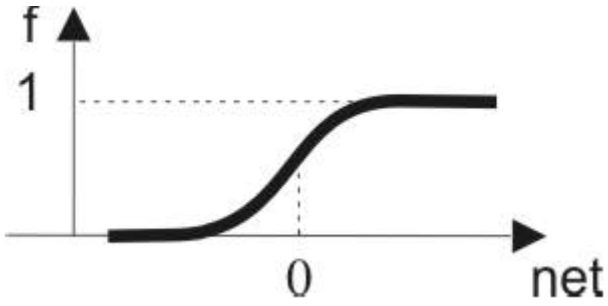
$$f'(net) = \frac{-(-a\cdot e^{-a\cdot net})}{(1+e^{-a\cdot net})^2}$$

$$f'(net) = a\cdot\frac{1}{1+e^{-a\cdot net}}\cdot\frac{e^{-a\cdot net}}{1+e^{-a\cdot net}}$$

$$f'(net) = a\cdot\frac{1}{1+e^{-a\cdot net}}\cdot\left(\frac{1+e^{-a\cdot net}}{1+e^{-a\cdot net}}-\frac{1}{1+e^{-a\cdot net}}\right)$$

$$f'(net) = a\cdot\frac{1}{1+e^{-a\cdot net}}\cdot\left(1-\frac{1}{1+e^{-a\cdot net}}\right)$$

$$f'(net) = a\cdot f(net)\cdot(1-f(net)) \text{ [2]} \qquad (3.8.)$$



U jednadžbama (3.7.) i (3.8.) parametar *a* određuje nagib i brzinu rasta funkcije. Uglavnom se za *a* uzima vrijednost 1 no ona se može promjeniti po želji korisnika.

Ovaj rad za prijenosnu funkciju koristi upravo sigmoidalnu funkciju, ali tijekom njenog korištenja nastao je problem oko odluke kada je izlaz 1, a kada 0, budući da je izlaz ove funkcije realni broj iz intervala {0,1} i vrlo teško je postići cijele brojeve. Zbog toga je u implementaciji uvrštena varijabla *PRAG* koja označava odstupanje od graničnih vrijednosti unutar kojega se uzima cijeli broj kao rješenje. Eksperimentalna vrijednost je postavljena na 0.05 no korisnik ju može promijeniti po potrebi i želji.

Primjer:

$$f(net) = 0.952 \Rightarrow y = 1$$

$$f(net) = 0.03 \Rightarrow y = 0$$

$$f(net) = 0.28 \Rightarrow y = 0.28$$

3.2. Implementacija neurona u jeziku C++

Neuron je kao klasa *Neuron* deklariran u datoteci *Neuron.h* (Slika 3.3.), dok se implementacije njegovih metoda nalaze u datoteci *Neuron.cpp*.

```
#include "Vektor.h"

class Neuron
{
public:
   int serijskiBrojNeurona;
   Real izlaz;
   Vektor ulazi;
   Vektor tezine;

   void init (int serialNumber, std::string dir,
   std::string bt, std::string mnm);
};
```

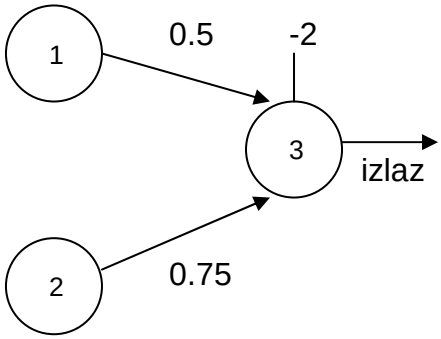
Slika 3.3. Neuron.h

Neuron sadrži slijedeće varijable i metode:

- *serijskiBrojNeurona*: identifikator neurona u mreži
- *izlaz*: izlaz koji neuron daje.
- *ulazi*: vektor identifikatora neurona čiji izlazi su ulazi u trenutni neuron.
- *tezine*: vektor težina veza koje ulaze u trenutni neuron.
- *init* (int *serialNumber*, std::string *dir*, std::string *bt*, std::string *mnm*): metoda inicijalizira neuron te mu dodijeljuje identifikator *serialNumber*. Ovisno o identifikatoru metoda prema podacima iz datoteke *mnm* (standardno zvana *ModelNeuronskeMreze.txt*) pronalazi koliko ulaza neuron ima te na temelju tog podatka kreira vektore *ulazi* i *tezine*. Metoda također ovisno o identifikatoru dodijeljuje vrijednosti vektorima *ulazi* iz datoteke *mnm* (standardno *ModelNeuronskeMreze.txt*) i *tezine* iz datoteke *bt* (standardno *BazaTezina.txt*). Očekuje se da se ove datoteke nalaze u tekućem direktoriju. U slučaju da se očekuju datoteke u drugom direktoriju to se treba navesti u varijabli *dir*. Za slijedeći zapis dobit ćemo rezultat mrežu kao što je prikazano na slici 3.4.:

```
BazaTezina.txt
1 0;
2 0;
3 -2,0.5,0.75;
```

```
ModelNeuronskeMreze.txt
1 0;
2 0;
3 1,2;
```



Neuron 1:
```
serijskiBrojNeurona=1
```

Neuron 2:
```
serijskiBrojNeurona=2
```

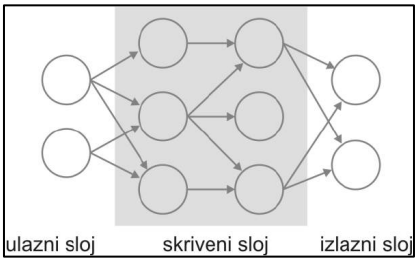
Neuron 3:
```
serijskiBrojNeurona=3
ulazi[1]=1
ulazi[2]=2
tezina[0]=-2
tezina[1]=0.5
tezina[2]=0.75
```

Slika 3.4. Rezultat metode init (int serialNumber)

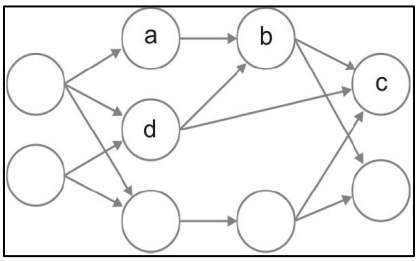
Vrijedi istaknuti kako klasa *Neuron* ne može sama računati izlaz već za nju to računa klasa *NeuronskaMreza*. Način na koji se to izvodi biti će naknadno objašnjen. Također valja istaknuti kako klasa *Neuron* koristi tipove podataka *Real* i *Vektor*. Tip podatka *Real* je skalar te je inicijalno postavljen na *float*. Tip podataka *Vektor* enkapsulira polje podataka tipa *Real*. Razred *Vektor* je definiran u datoteci *Vektor.h*, dok se implementacija nalazi u datoteci *Vektor.cpp*.

4. Neuronska mreža

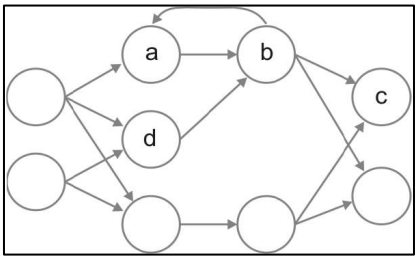
U prošlom poglavlju smo vidjeli od čega se sastoji i kako funkcionira neuron, sastavni dio neuronske mreže. U ovom dijelu upoznati ćemo se s vrstama neuronskih mreža, načinima na koje barataju podacima te na ono najvažnije, načinima na koje mreže uče. Ovisno o povezanosti mreže razlikujemo tri osnovne vrste mreža: acikličke slojevite, acikličke neslojevite i cikličke (Slika 4.1.).



a) aciklički slojevita – nema povratne veze unatrag i slojevi mreža su strogo definirani



b) aciklički neslojevita – nema povratne veze u natrag, ali ne postoji jasno definirani slojevi (primjer neuron c, ako idemo preko neurona b onda se neuron c nalazi u 4. sloju, a ako idemo preko neurona d onda se neuron c nalazi u 3. sloju)



c) ciklička – postoji povratna veza od neurona b do neurona a

Slika 4.1. Vrste neuronskih mreža [3]

Ovaj projekt se zasniva na acikličkoj slojevitoj mreži od tri sloja (ulazni, skriveni i izlazni) te će taj princip biti detaljnije objašnjen u nastavku. Kako u slojevitoj mreži nema povratka unazad, izlazi se dobivaju algoritmom feedforward. Učenje mreže se odvija već ranije spomenutim algoritmom backpropagation koji je revolucionarizirao korištenje neuronskih mreža.

4.1. Algoritam feedforward

Algoritam feedforward se koristi za dobivanje izlaza acikličke slojevite neuronske mreže. Princip je vrlo jednostavan: ulazi u mrežu prolaze kroz neurone u srednjem sloju čiji izlazi su ujedno ulazi u izlazni sloj. Izlazi izlaznog sloja čine izlaze neuronske mreže. Ovaj algoritam implementiran je u klasi *NeuronskaMreza* metodom `calculate (Vektor input);`.

Metoda radi tako da ide po serijskim brojevima neurona od prvog do posljednjeg i računa njegov izlaz pomoću sigmoidalne funkcije (3.7). Budući da je mreža aciklička slojevita, osigurano je da će prvo biti izračunati izlazi neurona ulaznog sloja, zatim srednjeg i na poslijetku završnog.

4.2. Algoritam backpropagation

Algoritam backpropagation (Slika 4.2.) koristi se kod postupka učenja acikličkih slojevitih neuronskih mreža. Algoritam je otkrio Paul Werbos 1974. godine, no kako je ranije spomenuto, pravi procvat algoritma bio je kada su ga primjenili 1986. godine Rumelhart, Hinton i Williams u svome radu. Algoritam je tipičan primjer učenja s učiteljem gdje „učitelj“, programer, koristi svoje znanje kako bi usmjeravao „učenika“, program, ka zadovoljavajućem rješenju. Pošto učitelj zna koji bi izlaz mreža trebala dati za svaki ispitni uzorak, on je usmjerava pravilom perceptrona i delta pravilom (objašnjeno u nastavku).

```
Inicijaliziraj težine u mreži (slučajnim odabirom)
Čini
  Za svaki uzorak e iz skupa za učenje
    σ = izlaz mreže za zadani uzorak
    t = izlaz učitelja
    Izračunaj grešku (t-σ)
    Izračunaj Δw_i za sve težine izlaznog sloja (delta pravilo)
    Izračunaj Δw_i za sve težine skrivenog sloja (delta pravilo)
    Obnovi sve težine u mreži (pravilo perceptrona)
Dok svi uzorci nisu pravilno raspoređeni
```

Slika 4.2. Algoritam backpropagation

4.2.1. Pravilo perceptrona

Već smo ustvrdili kako neuron računa izlaz (jednadžbe (3.2.) i (3.3.)), no što dalje s tim izlazom? Logično je da ako je izlaz odgovarajući nemamo nikakvih briga, ali što ako izlaz nije onakav kakav bi po učitelju trebao biti? Tu dolazi pravilo perceptrona (Slika 4.3.) koje kaže da ako izlaz nije jednak željenom izlazu moramo napraviti korekciju na težinskim faktorima ulaza. Metoda operira samo na jednom neuronu. Pravilo perceptrona, ako govorimo o učenju samo jednog neurona, ponavlja se dok god svi uzorci nisu ispravno klasificirani, isto kao što se algoritam backpropagation ponavlja dok god svi uzorci u mreži nisu ispravno klasificirani. Jednadžba po kojoj se radi korekcija težina je slijedeća:

$$w(k+1) = w(k) + c \cdot \delta \cdot x(k) \qquad (4.1.)$$

gdje $w(k+1)$ predstavlja vektor novih težina, $w(k)$ vektor starih težina, $x(k)$ vektor ulaza, c stopu učenja o kojoj ovisi brzina učenja (prevelika stopa može uzrokovati zaglavljenje u lokalnom minimumu, dok premala stopa može uzrokovati presporo učenje) te je uglavnom realni broj između 0 i 1, a δ predstavlja faktor korekcije koji se izračunava na slijedeći način:

$$\delta = a \cdot \sigma \cdot (1-\sigma) \cdot (t-\sigma) \qquad (4.2.)$$

gdje a predstavlja stopu porasta sigmoidalne funkcije, σ predstavlja izlaz mreže, a t predstavlja učiteljev izlaz. Faktor $(t - \sigma)$ računa grešku dok faktor $a \cdot \sigma \cdot (1 - \sigma)$ (3.8.) je derivacija funkcije 3.7. koja se koristi za dobivanje izlaza.

```
Čini
  Za svaki uzorak
    δ = a·σ·(1-σ)·(t-σ)
    w(k+1) = w(k) + c·δ·x(k)
Dok svi uzorci nisu pravilno raspoređeni
```

Slika 4.3. Algoritam pravila perceptrona

4.2.2. Delta pravilo

Vidjeli smo kod pravila perceptrona da se u jednadžbi (4.2.) koristi varijabla *t* koja označava učiteljev izlaz neurona. No kako je učitelju poznat samo krajnji rezultat, tj. izlazi neurona koji se nalaze u izlaznom sloju, ostaje nam misterij koji izlaz trebamo očekivati na izlazu neurona u skrivenom sloju. Tu koristimo delta pravilo koje se temelji na propagiranju greške vanjskog izlaza prema unazad, tj. skrivenom sloju. Odatle i cijeli naziv algoritma backpropagation (hrv. propagacija unazad) jer se on najviše upravo oslanja na delta pravilo.

```
Idi od zadnjeg neurona prema nazad za svaki neuron
  Ako je neuron u izlaznom sloju
    δ = (t-σ)·df(net)/dnet
  Inače
    Za svaku vezu koja izlazi iz tog neurona izračunaj
      δi = wij·δj
    δ = Σ_{i=1}^{n} (δi)·df(net)/dnet
Dok ne dođeš do prvog neurona
```

Slika 4.4. Delta pravilo

Kao što je prikazano u algoritmu (Slika 4.4.) delta pravilo se razlikuje za izlazni sloj neurona i skriveni i unutrašnji sloj neurona. Kada se radi o izlaznom sloju neurona tada neuronu možemo pristupiti kao samostalnom neuronu jer za njega nam je poznat izlaz te kako bi dobili δ koristimo jednadžbu (4.2.) gdje je faktor $a \cdot \sigma \cdot (1-\sigma)$ jednostavniji zapis za računanje od derivacije prijenosne funkcije $\frac{df(net)}{dnet}$. S druge strane, ako se radi o neuronu u skrivenom ili ulaznom sloju onda moramo uzeti u obzir sve veze koje iz tog neurona ulaze u druge neurone, njihove težine pomnožiti sa deltama neurona u koje ulaze:

$$\delta_i = w_{ij} \cdot \delta_j \qquad (4.3.)$$

gdje i označava broj neurona za koji računamo δ, j označava brojeve neurona u koje ulaze veze koje izlaze iz neurona i, a w_{ij} označava vezu od neurona i do neurona j. Na kraju nam ostaje da sve pojedinačne delte zbrojimo i pomnožimo sa derivacijom prijenosne funkcije (3.8.) neurona za koji računamo δ:

$$\delta = \sum_{i=1}^{n} (\delta_i) \cdot a \cdot \sigma \cdot (1-\sigma) \qquad (4.4.)$$

Nakon izračunavanja δ za svaki neuron ostaje nam uračunati taj faktor u jednadžbu (4.1.) kako bi mogli napraviti korekciju na neuronu. Detaljno objašnjenje može se vidjeti na http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html.

Ovim postupkom smo omogućili da program računa koliko je neki neuron kriv za krivi izlaz iz izlaznog sloja. No tu našim problemima nije kraj. Može se dogoditi situacija kada mreža upadne u „lokalni minimum“, stanje u kojem će se na izlazu pojaviti cijeli broj 0 ili 1. U tom slučaju će derivacija funkcije uvijek davati rezultat 0 koji će se manifestirati tako da nije potrebna promjena, tj. potrebna je promjena koeficijenta 0 što je daleko od onoga što želimo.

Primjer:

Ustvrdili smo kako derivacija sigmoidalne funkcije 3.7. jednostavnije napisano izgleda $a \cdot \sigma \cdot (1-\sigma)$, te smo također vidjeli kako δ iz jednadžbe (4.2.) (koji ovisi o derivaciji) u jednadžbi (4.1.) označava koeficijent korekcije.

$\sigma = 1 \Rightarrow 1-\sigma = 0 \Rightarrow \delta = 0 \Rightarrow c \cdot \delta \cdot x(k) = 0 \Rightarrow w(k+1) = w(k) \Rightarrow$ nema korekcije
$\sigma = 0 \Rightarrow \sigma = 0 \Rightarrow \delta = 0 \Rightarrow c \cdot \delta \cdot x(k) = 0 \Rightarrow w(k+1) = w(k) \Rightarrow$ nema korekcije

No zbog svojsta sigmoidalne funkcije da jako teško dostegne cijeli broj ovaj problem se rijetko pojavljuje. Ipak u slučaju da se upadne u „lokalni minimum“ najbolji način je ponovno pokretanje programa s novim slučajnim početnim vrijednostima što se primjenjivalo za riješavanje problema kod testiranja.

4.3. Implementacija neuronske mreže u jeziku C++

Neuronska mreža je kao klasa *NeuronskaMreza* deklarirana u datoteci *NeuronskaMreza.h* (Slika 4.5.), dok se implementacije njenih metoda nalaze u datoteci *NeuronskaMreza.cpp*.

```
#include "Neuron.h"

class NeuronskaMreza
{
public:
   NeuronskaMreza (int a_);
   NeuronskaMreza (std::string dir="", std::string bt="BazaTezina.txt", std::string
   mnm="ModelNeuronskeMreze.txt", std::string pzt="PrimjeriZaTestiranje.txt",
   std::string rp="RezultatiPrimjera.txt",int a_=1);

   void learn (Real c);
   int classify (Vektor input);
   void test ();

private:
   int a;
   Real e;
   Neuron neuron[BROJ_NEURONA];
   std::string fname;
   std::string fname1;
   std::string fname2;
   std::string fname3;
   std::string fname4;

   void init ();
   void calculate (Vektor input);
   void update ();
};
```

Slika 4.5. NeuronskaMreza.h

NeuronskaMreza sadrži slijedeće javne metode:

- *NeuronskaMreza* (): konstruktor klase NeuronskaMreza, kao argument može primiti varijablu *a* koja određuje nagib sigmoidalne funkcije. U slučaju da se ne navede vrijednosti, parametar se postavlja na vrijednost 1. Također se u konstruktoru mogu navesti imena datoteka u kojima se nalaze informacije koje neuronska mreža koristi za svoj rad, te put do tih datoteka. Ako se te vrijednosti ne navedu, program kao

standarni diretorij uzima tekući direktorij, a imena datoteka ona koja su navedena u poglavlju 5.2. Dodatne datoteke.

- *learn* (Real *c*): metoda koja obavlja fazu učenja mreže. Kao argument koristi varijablu *c* koja predstavlja stopu učenja iz jednadžbe (4.1.) Metoda uzima podatke o uzorcima iz datoteke *pzt* (standardno PrimjeriZaTestiranje.txt) te podatke o rezultatima koje bi ti uzorci trebali davati iz datoteke *rp* (standardno *RezultatiPrimjera.txt*). Metoda zatim koristi algoritam backpropagation kako bi podesila veze među neuronima tako da na izlazu daju željene rezultate.
- *classify* (Vektor *input*): metoda uspoređuje izlaze neurona koji se nalaze u izlaznom sloju i vraća vrijednost neurona koji na izlazu ima najveću vrijednost. Ta vrijednost služi za određivanje znaka čija slika se našla na ulazu mreže. U slučaju da niti jedan izlaz neurona nije veći od parametra *PRAG* metoda će to shvatiti kao da se na ulazu u mrežu pojavila slika prometnog znaka koji nije definiran u datoteci *Izlazi.txt* kao mogući izlaz mreže te će vratiti vrijednost 0.
- *test* (): metoda se koristi uz set uzoraka za testiranje i daje podatak koji uzorak je pravilno razvrstan, te ukupni postotak pravilno razvrstanih uzoraka. Podatci o uzorcima za testiranje se nalaze u datoteci *pzt* (standardno *PrimjeriZaTestiranje.txt*), a rezultati u datoteci *rp* (standardno *RezultatiPrimjera.txt*).

NeuronskaMreza sadrži slijedeće privatne varijable i metode:

- *a*: faktor nagiba sigmoidalne funkcije.
- *e*: konstanta 2.718281828.
- *neuron*[*BROJ_NEURONA*]: polje klase *Neuron*.
- *fname*: skup varijabla tipa string koje služe za spremanje podataka o imenima datoteka koje koristi neuronska mreža za svoj rad.
- *init* (): metoda kreira neuronsku mrežu kreiranjem neurona i veza među njima: Broj neurona je parametar koji je zapisan u datoteci *Konfiguracija.h*.
- *calculate* (Vektor *input*): metoda postavlja vrijednosti iz vektora *input* na ulaze u neuronsku mrežu i izračunava izlaze (algoritam feedforward).

- *update* (): metoda koja sprema trenutačno stanje težina veza iz memorije neurona u datoteku *bt* (standardno *BazaTezina.txt*).

5. Upute za korištenje

Prvi korak u korištenju ovog programa je prevesti ga u prikladnom programskom okruženju gdje će se program izvoditi. Pri izradi korišteno je programsko okruženje Visual C++ 2008 Express Edition. Kako pravilno prevesti program u okruženje detaljno je opisano u radu Prepoznavanje znakova, poglavlje 1 [1].

Drugi korak u korištenju programa je podešavanje parametara. U ovoj verziji projekta parametri su smješteni u datoteci *Konfiguracija.h* (Slika 5.1.) te podaci koje mreža koristi su standardno smješteni u datotekama:

- *BazaTezina.txt*
- *Izlazi.txt*
- *ModelNeuronskeMreze.txt*
- *PrimjeriZaTestiranje.txt*
- *RezultatiPrimjera.txt*

5.1. Konfiguracija.h

```
#define PRAG 0.1
#define MAX_BROJ_ITERACIJA 100
#define WIDTH 10
#define HEIGHT 10
#define BROJ_ULAZA WIDTH*HEIGHT
#define BROJ_SREDNJEG BROJ_ULAZA
#define BROJ_IZLAZA 9
#define BROJ_NEURONA BROJ_ULAZA+BROJ_SREDNJEG+BROJ_IZLAZA
#define BROJ_UZORAKA 606
```

Slika 5.1. Konfiguracija.h

U datoteci Konfiguracija.h nalaze se parametri koji se odnose na arhitekturu mreže te njeno ispitivanje. Parametri koji su ovdje definirani su *prag*, *max_broj_iteracija*, *width*, *height*, *broj_ulaza*, *broj_srednjeg*, *broj_izlaza*, *broj_neurona* i *broj_uzoraka*.

Prag se koristi, kako je opisano u odjeljku 3.1.3. za zaokruživanje vrijednosti na izlazu na cijeli broj zbog asimptotičnosti sigmoidalne funkcije.

Pošto učenje neuronske mreže može biti jako dugi postupak, unosi se parametar *max_broj_iteracija* koji određuje koliko se iteracija učenja mreže može obaviti prije nego se program nasilno prekine. Parametar može služiti u slučaju da želimo vidjeti kako težine u neuronskoj mreži izgledaju u n-tom koraku učenja ili kao osiguravatelj ako mreža upadne u „lokalni minimum“ da se algoritam ne vrti beskonačno.

Parametri *width* i *height* određuju broj piksela po dužini i širini slike. Budući da njihov umnožak daje broj piksela u slici, često se koristi za označavanje broja ulaza u mrežu.

Parametri *broj_ulaza*, *broj_srednjeg* i *broj_izlaza* označavaju broj neurona u pojedinom sloju. *Broj_ulaza* se dobiva množenjem parametara *width* i *height*, no također se može postaviti neovisno o tim parametrima ukoliko mrežu ne koristimo za rad sa slikama. *Broj_srednjeg* je proizvoljan broj neurona u srednjem sloju. *Broj_izlaza* se treba podudarati s brojem izlaza u datoteci *Izlazi.txt*. *Broj_neurona* označava ukupan broj neurona u mreži.

Broj_uzoraka se koristi kako bi algoritam backpropagation znao koliko uzoraka mora ispitati daju li zadovoljavajuću vrijednost i napraviti korekciju ako je potrebna prije nego krene u novu iteraciju.

5.2. Dodatne datoteke

Dodatne datoteke su tekstualne datoteke koje se koriste kako bi se trajno mogli pohraniti rezultati neuronske mreže te se kasnije ponovo koristiti. Program koristi pet dodatnih datoteka čija su standardizirana imena:

- *BazaTezina.txt*
- *Izlazi.txt*
- *ModelNeuronskeMreze.txt*
- *PrimjeriZaTestiranje.txt*
- *RezultatiPrimjera.txt*

BazaTezina.txt je najvažnija dodatna datoteka koja se koristi za zapis težina veza među neuronima. U datoteci su zapisi organizirani po redovima tako da na početku reda piše broj neurona, a zatim slijede iznosi težina veza koje ulaze u neuron. Prva težina u tom slijedu je w_0 koja predstavlja prag neurona, a dalje slijede w_1 koja predstavlja težinu prvog ulaza, w_2 i tako do kraja do w_n, ovisno o tome koliko ulaza neuron ima.

```
110 -0.697202,0.563037,-1.140217;
```

Ovim prikazom možemo vidjeti da se radi o neuronu broj 110 koji uz sebe ima tri težine, što znači kako u ovaj neuron ima dva ulaza s težinama 0.563037 i -1.140217 te prag u iznosu -0.697202.

Poseban opis imaju neuroni koji se nalaze u ulaznom sloju. Budući da njihova funkcija daje izlaz onda oni nemaju niti ulaz niti prag. Takvi neuroni označeni su tako da se pored broja neurona nalazi broj nula.

```
1 0;
```

Ovaj zapis nam govori kako se radi o prvom neuronu koji se nalazi u ulazom sloju. Kako neuron u drugom sloju mora imati minimalno dvije vrijednosti pored oznake broja neurona (minimalno jedan ulaz i prag) ovakav zapis jednoznačno određuje da je neuron u ulaznom sloju. Za kreiranje slučajne datoteke *BazaTezina.txt* postoji funkcija `makeBazaTezina(std::string dir="", std::string bt="BazaTezina.txt")` koja generira slučajne vrijednosti kao početne težine među neuronima. Slučajne vrijednosti mogu biti u intervalu [-1,1]. Funkcija kao argumente prima put do direktorija i ime datoteke no parametri mogu biti izostavljeni.

Izlazi.txt je datoteka u kojoj se nalaze oznake prometnih znakova koje mreža prepoznaje. Svaki znak koji se ne nalazi u ovoj datoteci, identificirat će se kao da nije znak. Oznaka znaka sastoji se od dva dijela: dio koji govori u koju skupinu znakova znak pripada te redni broj znaka u toj skupini. Oznake skupina su sljedeće:

- A: znakovi opasnosti
- B: znakovi izričite naredbe
- C: znakovi obavijesti
- D: putokazi
- E: dopunske ploče

Slika 5.3. Prometni znakovi a)A01, b) B31, c)C05

ModelNeuronskeMreze.txt opisuje strukturu neuronske mreže. Zapisi su organizirani kao i kod datoteke *BazaTezina.txt* tako da prvi broj u redu označava broj neurona o kojem se radi, dok ostatak informacije u redu označava ulazne neurone u trenutni neuron.

```
12 2,5,6,10;
```

U ovom slučaju 12 je identifikator neurona čiji ulazi su izlazi neurona 2,5,6 i 10. Budući da neuroni u prvom sloju nemaju ulaze njihov zapis je isti kao i u datoteci *BazaTezina.txt*.

```
2 0;
```

Ovdje vidimo da je neuron 2 ulazni jer ne postoji neuron s identifikatorom 0 budući da prvi neuron ima identifikator 1. Za kreiranje datoteke *ModelNeuronskeMreze.txt* postoji funkcija `makeModelNeuronskeMreze(std::string dir="", std::string mnm="ModelNeuronskeMreze.txt")`. Funkcija kao argumente prima put do direktorija i ime datoteke no parametri mogu biti izostavljeni.

Datoteka *PrimjeriZaTestiranje.txt* sadrži bitove slika koje se koriste za učenje mreže ili ispitivanje. Bitovi se dobivaju tako da se uzme slika na kojoj se znak nalazi, locira se slika znaka preko tekstualne datoteke koja sadrži koordinate znakova na slici, slika znaka se razvuće ili smanji na željenu rezoluciju te se preko funkcije `bits()` dohvate bitovi svakog piksela koji su u rasponu [0,255]. Kako neuronska mreža preferira ulaze u intervalu [0,1] pikseli se skaliraju na traženi interval te se kao takvi zapisuju u datoteku. U datoteci svaki uzorak predstavlja jedan red tako da se u jednom redu nalazi onoliko brojeva koliko neuronska mreža ima ulaza. Tako je uzorak koji ima 4 ulaza zapisan:

```
0.265625 0.421094 0.519531 0.499219
```

Datoteka *RezultatiPrimjera.txt* sadrži vrijednosti koje neuronska mreža očekuje na izlazu. Tijekom obrade slike iz datoteke koja sadrži koordinate znaka uzima se oznaka znaka koja se uspoređuje sa znakovima koji se očekuju na izlazu opisanima u *Izlazi.txt*, te se prema tome dodijeljuje 1 ako se na određenom neuronu očekuje taj znak ili 0 ako se ne očekuje. U jednom redu se nalazi onoliko zapisa koliko mreža ima izlaza, a svaki zapis određuje izlaz jednog neurona po redu. Dakle zapis:

```
0 1 0
```

uz datoteku *Izlazi.txt* gdje piše:

```
A01
B31
C05
```

će značiti da neuronska mreža očekuje na izlazu prvog i trećeg neurona 0, a na izlazu drugog neurona 1. Također znak kojeg će dati mreža preko funkcije `test()` iz klase *NeuronskaMreza* je B31, odnosno znak ograničenja brzine. U slučaju da se u jednom redu nalaze sve 0 to znači da se znak sa slike ne nalazi na popisu znakova kojima tražimo izlaz.

Također je potrebno napomenuti kako se za kreiranje dodatnih datoteka koristi funkcija `preprocessImage (std::string path, std::string name, std::string dir="", std::string bt="BazaTezina.txt", std::string iz="Izlazi.txt", std::string mnm="ModelNeuronskeMreze.txt", std::string pzt="PrimjeriZaTestiranje.txt", std::string rp="RezultatiPrimjera.txt")`. Funkcija prima parametre o putu do direktorija gdje se nalaze slike i ime datoteke koja sadrži informacije o položaju prometnih znakova na slikama. Dodatni parametri su imena dodatnih datoteka, koji ako se ne navedu, koriste se standardni nazivi koji su navedeni na početku ovog poglavlja. Funkcija vraća vrijednost 1 što znači da je uspjela izvršiti rad nad svim datotekama, odnosno 0 u slučaju da nije.

6. Ispitivanje programa i rezultati

Ispitivanje programa je vršeno sa slijedećim parametrima:

- prag zadovoljivosti izlaza: 0.05
- faktor nagiba sigmoidalne funkcije a: 1
- broj neurona u ulaznom sloju: 100 (rezolucija slike 10x10)
- broj neurona u skrivenom sloju: 100
- broj neurona u izlaznom sloju: 9
- izlazni znakovi: A01, A04, A10, A11, A12, A20, A22, A34, A44
- broj uzoraka za testiranje: 353
- broj uzoraka za testiranje po znaku:

A01	A04	A10	A11	A12	A20	A22	A34	A44	ostali
									-
26	8	71	68	15	37	6	12	49	61

Uz ove podatke uz slučajan odabir početnih vrijednosti za težine pokrenuto je učenje mreže koje je završeno nakon par tisuća iteracija kada su uspješno klasificirani svi uzorci.

Za testiranje su korišteni isti parameteri kao i za učenje. Za uzorke je korišten novi set od 72 znaka raspoređenih:

A01	A04	A10	A11	A12	A20	A22	A34	A44	ostali
4	4	7	15	4	8	8	8	3	11

Ostvareni su sljedeći rezultati s ukupnom uspješnošću od 53% (38/72):

A01	A04	A10	A11	A12	A20	A22	A34	A44	ostali
4/4	0/4	6/7	10/15	3/4	4/8	0/8	3/8	0/3	8/11
100%	0%	86%	67%	75%	50%	0%	38%	0%	73%

Rezultati variraju od zadovoljavajućih do nezadovoljavajućih, ovisno o znaku. Glavni razlozi za nezadovoljavajući rezultat nekih znakova prvenstveno su manjak uzoraka za učenje (A04 i A022) i uzoraka za ispitivanje (A44).

7. Zaključak

Tema ovog rada bila je osmišljavanje i implementacija neuronske mreže u jeziku C++ i korištenje te mreže kako bi se raspoznali prometni znakovi. Prometni znakovi su snimani kamerom te su se slike iz video zapisa na kojima se nalaze prometni znakovi izdvojile kako bi se njima lakše koristilo. Potom su na tim slikama označeni prometni znakovi čije su koordinate spremljene u tekstualnu datoteku. Korisnik neuronske mreže preko te datoteke dolazi do lokaliziranih slika prometnih znakova čije piksele koristi na ulazu u neuronsku mrežu. Mreža je izgrađena tako da koristi tri sloja: ulazni, skriveni i izlazni te prema izlazu zadnjeg sloja možemo očitati koji znak se našao na ulazu mreže.

Mreža koristi algoritam feedforward za izačunavanje izlaznih vrijednosti i algoritam backpropagation za izračunavanje faktora korekcije. Za korekciju težina ulaznih veza koristi se pravilo perceptrona.

Eksperimentalni postupak pokazao je kako je neuronska mreža solidno sredstvo za raspoznavanje uzoraka, no također je pokazao kako nije savršena te može biti mnogo propusta u njenom radu. Ukupni rezultat od 53% se može smatrati kao prolazna ocjena, ali s puno mjesta za poboljšanje performansi.

8. Literatura

[1] Babić T., Lukinić T., Kovač D., Popović K., Rojković D., Šegvić S., Šverko M.: Prepoznavanje znakova, Projekt iz programske podrške, Fakultet elektrotehnike i računarstva, Zagreb, 2008.

[2] Čupić M., Dalbelo Bašić B., Šnajder J.: Umjetne neuronske mreže, Službeni materijali s predavanja iz predmeta Umjetna inteligencija, Fakultet elektrotehnike i računarstva, Zagreb, 2008.

[3] Čupić M., Dalbelo Bašić B., Šnajder J.: Umjetne neuronske mreže, 2008.

[4] Seung S.: Multilayer perceptrons and backpropagation learning, 2002.

[5] Tou J.T., Gonzales R.C.: Pattern Recognition Principles, Addison-Wesley Publishing Company, Reading, Massachusetts, 1974.

[6] Neuronska mreža – Wikipedia, http://hr.wikipedia.org/wiki/Neuronska_mre%C5%BEa, 10.siječanj 2009.

[7] Backpropagation – Wikipedija, http://en.wikipedia.org/wiki/Backpropagation, 10.siječanj 2009.

[8] Backpropagation, http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html, 10. siječanj 2009.

9. Sažetak/Summary

Raspoznavanje prometnih znakova neuronskom mrežom

Ovaj rad razmatra raspoznavanje prometnih znakova prikladno učenom neuronskom mrežom. Objašnjava se princip rada neurona, algoritmi za učenje i klasificiranje, te arhitektura mreže. Mreža je učena algoritmom backpropagation koristeći ručno izlučene instance prometnih znakova. Mreža je implementirana i evaluirana unutar eksperimentalnog sustava za raspoznavanje koji je napisan u jeziku C++. Postignuti rezultati su prikazani i komentirani.

Ključne riječi: neuron, neuronska mreža, umjetna inteligencija, raspoznavanje prometnih znakova, feedforward, backpropagation, delta pravilo, pravilo perceptrona

Traffic signs recognition by neural network

This work considers traffic signs recognition by a suitably trained neural network. It explains principles of neuron, algorithms for learning and classifying, as well as network architecture. The network has been trained by the backpropagation algorithm using manually extracted instances of traffic signs. The network has been implemented and evaluated within an experimental recognition system written in C++. The obtained experimental results are presented and dsicussed.

Key words: neuron, neural network, artificial inteligence, traffic signs recongition, feedforward, backpropagation, delta rule, perceptron rule