PRESEK
List za mlade matematike, fizike, astronome in računalnikarje

ISSN 0351-6652
Letnik **19** (1991/1992)
Številka 3
Strani 154–158

Veselko Guštin:

RAZPOZNAVANJE CIFER – BRALNI POMNILNIK IN NEVRONSKA OMREŽJA

Ključne besede: računalništvo, pomnilniki.

Elektronska verzija: http://www.presek.si/19/1091-Gustin.pdf

RAZPOZNAVANJE CIFER - bralni pomnilnik in nevronska omrežja

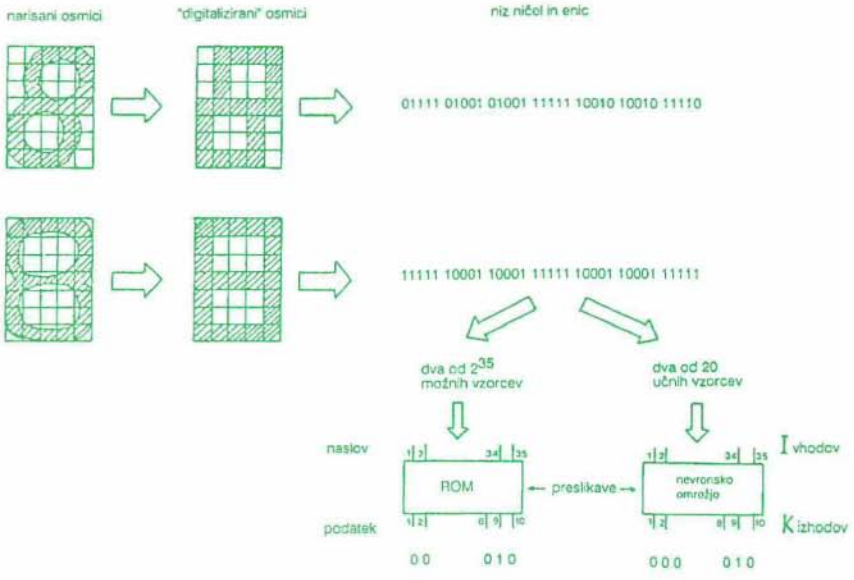
Vsak, ki pozna osnove računalništva, je gotovo že slišal za bralne pomnilnike ROM-e. Zanje lahko rečemo, da so v veliki meri pripomogli k hitremu razvoju centralnih procesnih enot, pa tudi k preprostejšemu hranjenju zagonske programske opreme. Kratica ROM prihaja iz angleščine, pomeni namreč Read Only Memory. Ime samo pove, da so to pomnilniki, ki so namenjeni trajnemu hranjenju informacije in iz katerih lahko med uporabo le beremo. Vsebino pa zapisujemo v ROM-e na naslednje načine: v postopku izdelave integriranega vezja (ROM), z enkratnim kasnejšim vpisovanjem (PROM) in z večkratnim vpisovanjem ter brisanjem z ultravijolično svetlobo (EPROM) ali električnim brisanjem (EEPROM). Navedenim načinom zapisovanja v bralni pomnilnik rečemo tudi **programiranje**.

Najbolj znani so ROM-i, ki jih lahko večkrat programiramo in brišemo. Uporabljamo jih predvsem kot stalne pomnilnike. Vhodi, njihovo število bomo označili z **I**, določajo naslov ene od 2^I možnih lokacij, njeno vrednost pa dobimo na **K** izhodih. Verjetno niste vedeli, da so ROM-i primerni tudi za realizacijo različnih logičnih funkcij. ROM lahko nadomešča poljuben sklop logičnih vrat ALI ter IN. To pa še ni vse! V splošnem je programirani ROM vezje, ki omogoča izračun poljubne preslikave 2^I vhodnih vrednosti v izhodne. To njegovo lastnost lahko s pridom uporabimo pri razpoznavanju cifer.

Posebno zanimive so preslikave črno-belih slik, tiskanih ali pisanih znakov, običajno cifer, lahko pa tudi črk. Predstavljajmo si, da čez povečano tiskano ali pisano cifro postavimo mrežo, sestavljeno na primer iz $5x7 = 35$ polj. Če izberemo vrednost "0" za vsa tista polja, ki so bolj bela kot črna, in "1" za tista, ki so bolj črna kot bela, dobimo niz dolžine 35, sestavljen iz samih "0" in "1". Takemu postopku pravimo **digitalizacija**. Za razpoznavanje takih slik potrebujemo le še ROM, ki ima 35 vhodov oziroma 2^{35} lokacij, kamor shranimo različne digitalizirane slike zapisanih cifer. Tu seveda upoštevamo še vse take slike, ki so delno pokvarjene in ne ustrezajo v celoti nobeni cifri. Na kratko, v takem ROM-u se nahajajo **prav vse** možne digitalizirane slike cifer na mreži velikosti 5 x 7 najrazličnejših oblik, velikosti in pisav. Če se omejimo samo na ločevanje cifer, tedaj zadošča 10 izhodov, za vsako cifro svoj. Naj prvi izhod predstavlja cifro "0", drugi izhod cifro "1", zadnji izhod pa naj ustreza cifri "9". Na sliki 1 vidimo pričakovani odziv, za cifro "8" vrednost "1" na devetem izhodu. Vse lepo in prav! Toda kaj kmalu ugotovimo, da tako velikega bralnega pomnilnika - nimamo. Za zapis vseh digitaliziranih slik cifer bi namreč potrebovali bralni pomnilnik približne

velikosti

$$2^{35}x10 \text{ bitov} = 3.436 \cdot 10^{10}x10 \text{ bitov} = 34360x10 \text{ Mbitov!}$$



Slika 1. Kako dobimo niz "0" in "1" iz cifre osem?

To pa je že tolikšen pomnilnik, kakršnega nimajo niti najboljši (osebni) računalniki skupaj z diskovnimi enotami. Vedimo tudi, da smo primer poenostavili, saj smo vzeli dokaj redko mrežo in zaradi tega vzorce tudi precej popačili. Kaj bi šele bilo, če bi na primer potrebovali mrežo velikosti 10 x 20 ali celo večjo. Z njo bi vsekakor dosegli neprimerno boljše pokrivanje napisane cifre z mrežo in s tem tudi natančnejše razpoznavanje. Toda pravkar opisani pristop za ta namen ni primeren, saj zahteva bralne pomnilnike skoraj nepredstavljive velikosti.

Torej tu smo! To je zadosten razlog, da si z bralnimi pomnilniki nimamo več kaj pomagati. Za razpoznavanje cifer je bilo razvitih veliko algoritmov, ki temeljijo na črno-beli sliki, iskanju posebnosti vsake cifre in ugotavljanju njene pripadnosti. Tovrstna programska oprema je ponavadi obsežna in največkrat neprimerna za uporabo v realnem času. Ker gre pri razpoznavanju vselej za preslikavo iz opazovane cifre v njeno ime in ker tovrstne preslikave z ROM-om

ne moremo narediti, se spomnimo prispevkov o nevronskih omrežjih, ki sta bila objavljena v tretji in četrti številki Preseka **17** (1989/90). Pokazali bomo, da želene preslikave hitro in enostavno naredimo z nevronskimi omrežji, če jih le omejimo na računanje z dvojiškimi vrednostmi. Le-te smo pri razpoznavanju tudi uporabljali! Zato bodo vhodi in izhodi omrežja imeli vrednosti ali 0 ali 1. Tudi zaviralne in vzbujevalne uteži bodo imele le dvoje različnih vrednosti 0 ali 1, a različnih po predznakih. Nevronsko omrežje z 200 vhodi in 10 izhodi ni nič posebnega. Za učenje omrežja potrebujemo le še spisek učnih vzorcev. Le-teh bo kakih sto, za vsako cifro po deset različnih.

Naše omrežje naj sestavlja J nevronov na prvem nivoju, K nevronov na drugem, vhodov v omrežje naj bo I, medtem ko je število izhodov K. Izhodne funkcije nevronov na drugem nivoju, označimo jih z y^k za $k = 1, 2, \ldots, K$, lahko enostavno izrazimo kot vsoto produktov izhodov prvega nivoja x_j in uteži $w_{j,k}$:

$$\text{če je } \sum_{j=1}^{J} w_{j,k} \cdot x_j \geq 1, \text{ tedaj je } y_k = 1, \text{ sicer pa je } y_k = 0.$$

Podobno izračunamo tudi izhode s prvega nivoja x_j. Med vhodi nevrona je potrebno upoštevati tudi pragovni vhod. Njegova vrednost je sicer 1, vrednost uteži pa izračunamo. S predznakom uteži $w_{i,k}$ povemo značaj povezave med nevronoma. Pozitivne uteži so na vzbujevalnih povezavah, negativne pa na zaviralnih.

Ker v našem primeru obdelujejo nevroni samo dvojiške vrednosti, se pri računanju izhoda nevrona izognemo realni aritmetiki. Če je $x_j \neq 0$, k delni vsoti prištejemo utež $w_{j,k}$, torej prištejemo ali odštejemo 1. Če pa je $x_j = 0$, tedaj j-ti vhod sploh ne vpliva na vrednost y_k, tako da lahko preidemo kar na naslednji vhod nevrona. Tak preprost dvojiški ali Boolov nevron lahko učimo tudi drugače, kot smo to prebrali v omenjenih prispevkih. Posluževali se bomo kar naključnega iskanja primernih uteži. Naključno izberemo utež in na omrežje postavimo učne vzorce, drugega za drugim. Če je rezultat preslikave ugodnejši od prejšnjega, tedaj novo utež zadržimo. Če pa se preverjanje kje zalomi in so rezultati preslikave slabši, tedaj naključno izberemo novo utež. Učenje je uspešno končano, ko najdemo take vrednosti uteži, da so izračunani izhodi enaki pričakovanim za vse učne vzorce. Če učenja ne moremo uspešno dokončati, tedaj ali dodamo nov nevron v prvem nivoju in s tem povečamo J ali pa povečamo število vhodov I in tako dodamo nov pragovni vhod.

Za lažje razumevanje bomo nevronsko omrežje s slike 2 učili z logično funkcijo EX-OR (ekskluzivni ali). Na sliki so uteži določene tako, da omrežje

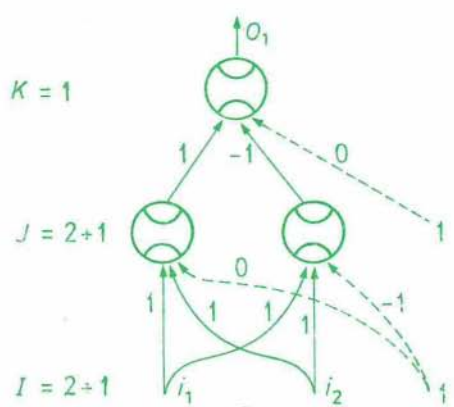
pravilno izračuna to funkcijo.

Učenje začnemo tako, da za vsem utežem damo naključne začetne vrednosti (-1, 0 ali +1) ter za prvega od štirih učnih vzorcev postavimo vhodni vrednosti i_1 in i_2 na omrežje in izračunamo odgovor o_1. Nato poskusimo še z drugimi učnimi vzorci. Če dobimo pri vseh štirih učnih vzorcih skupno več pravilnih rešitev, ali pa vsaj toliko kot pri predhodnih utežeh, uteži zadržimo, sicer pa vrnemo stare vrednosti. Če vsi odgovori niso bili pravilni, nato naključno izberemo vrednost nove uteži. Postopek ponavljamo tolikokrat, dokler za vse štiri učne kombinacije ne dobimo pravilnih izhodov.

Iz izkušenj vemo, da se tako omrežje kaj hitro nauči, recimo v tisoč ali več poizkusih, ki jih osebni računalnik naredi v nekaj sekundah. Preslikavo samo, ko je omrežje enkrat naučeno, pa izračunamo v trenutku.

Vrnimo se še enkrat k našim učnim vzorcem na mreži 5 x 7. Če primerjamo ugotovitve o razpoznavanju z vezjem ROM in z nevronskim omrežjem, opazimo naslednje:

- Vezje bralnega pomnilnika (če bi le-tega res imeli) zahteva tabeliranje **vseh** možnih digitaliziranih slik, ki jih lahko dobimo na mreži 5 x 7. Seveda je nekaj smiselnih, veliko pa je takih, ki nimajo pravega pomena, na primer same 0 ali same 1. Ko na vhode postavimo neko vrednost, nam vezje - glede na vsebino lokacije, ki je določena z vhodom - da odgovor. Le-tega spremenimo tako, da zapišemo novo vrednost na ustrezno lokacijo v ROM-u.

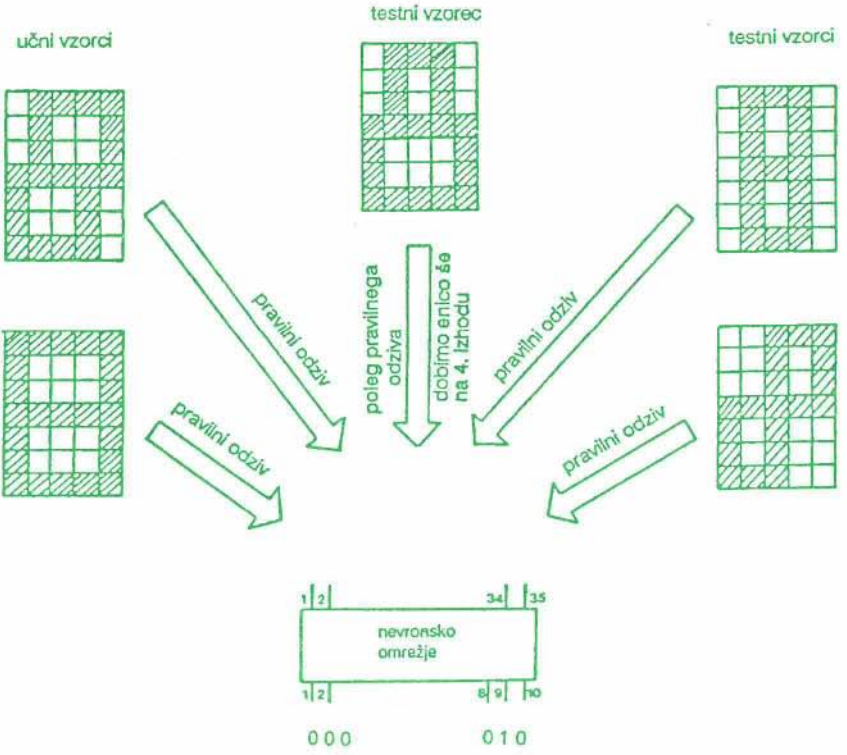


i_1	i_2	O_1
0	0	0
0	1	1
1	0	1
1	1	0

b

Slika 2a. Nevronsko omrežje

Slika 2b. Tabela logične funkcije EX-OR



Slika 3. Preslikava vzorcev nevronskega omrežja, učni vzorci in poljubni testni vzorci.

- Nevronsko omrežje, kot smo spoznali, pa naučimo le **nekaterih** učnih vzorcev. Ko je vezje naučeno, tedaj nam brez napake odgovarja na učne vzorce. Kaj pa na drugačne vzorce? Ugotovimo, da tudi na nekatere druge, neznane vzorce daje največkrat pravilne, ali pa vsaj smiselne odgovore. Nevronsko omrežje zna torej **samo** poiskati odgovor, pravimo, da zna posploševati.

To zanimivost bomo s pridom uporabljali predvsem pri obsežnih zgledih, takih z več sto nevroni v več nivojskem omrežju. Pomembno je tudi, da ni potrebno vezju podati prav vseh možnih vzorcev, pač pa le nekaj značilnih. Kadar so odgovori slabi, lahko omrežje še vedno dodatno poučimo o novih vzorcih.

Veselko Guštin