Umjetne neuronske mreže

Paris, Stella

SVEUČILIŠTE U RIJECI
FILOZOFSKI FAKULTET U RIJECI
Odsjek za politehniku

Stella Paris

UMJETNE NEURONSKE MREŽE

(završni rad)

Rijeka, 2017. godine

SVEUČILIŠTE U RIJECI

FILOZOFSKI FAKULTET U RIJECI

Studijski program: sveučilišni preddiplomski studij politehnike

Stella Paris
mat. broj: 0009066497

UMJETNE NEURONSKE MREŽE

- završni rad -

Mentor : mr. sc. Gordan Đurović

Rijeka, 2017. godine

FILOZOFSKI FAKULTET U RIJECI

Odsjek za Politehniku

U Rijeci, 20. siječnja 2017. godine

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Stella Paris**

Studij: **Sveučilišni preddiplomski studij politehnike**

Naslov završnog rada: **Umjetne neuronske mreže / Artificial neural networks**

Znanstveno područje: **2. Tehničke znanosti**
Znanstveno polje: **2.09. Računarstvo**
Znanstvena grana: **2.09.04 umjetna inteligencija**

Kratak opis zadatka: Opisati povijest razvoja umjetnih neuronskih mreža te sličnosti i razlike između neurona u ljudskom mozgu i umjetnih neuronskih mreža. Detaljno prikazati Perceptron, njegove dijelove i način rada, princip treniranja te njegova ograničenja. Prikazati organizacijsku strukturu višeslojnih neuronskiuh mreža, podjelu na nivoe te kroz primjere osnovne metode njihovog učenja i rada. Istrenirati perceptron da funkcionira kao osnovni logički elementi AND i OR te prikazati razlike između dobivenih vrijednosti u mreži iste strukture.

Zadatak uručen pristupniku: **1. ožujka 2017. godine**

Ovjera prihvaćanja završnog rada od strane mentora: _____

Završni rad predan: ____ _____

Datum obrane završnog rada: _____

Članovi ispitnog povjerenstva: 1. predsjednik - _____

2. mentor - _____

3. član - _____

Konačna ocjena: _____

Mentor

_____

mr. sc. Gordan Đurović

Sažetak

Ovaj završni rad predstaviti će nam osnove umjetnih neuronskih mreža, osnovnu ideju strojnog učenja, učenja i pravila koja čine umjetne neuronske mreže, te mehanizme koji omogućuju računalima naučiti iz iskustva. Predstaviti ćemo koncept perceptrona kao najjednostavnijeg oblika neuronske mreže i njegovo učenje. Također ćemo istražili višeslojne neuronske mreže čije se učenje odvija na isti način kao u perceptronu i prikazati ćemo kako poboljšati računalnu učinkovitost algoritma učenja povratnog širenja. Spomenuti ćemo i Hopfield mreže, algoritam treninga sa dvije osnovne faze i dvosmjernu asociativnu memoriju. Predstaviti ćemo i samoorganizacijske neuronske mreže i istražiti Hebbian i konkurentno učenje. Na kraju, prikazati ćemo primjer treninga perceptrona.

Ključne riječi: Neuroni, mreža, učenje, mehanizam, perceptron

Summary

In this esay, we will present the basics of artificial neural networks, the basic idea of machine learning, learning and rules that make up artificial neural networks, and mechanisms that enable computers to learn from experience. We will present the concept perceptron as the simplest form of neural networks and their learning. We will also investigate multilayer neural networks whose learning takes place in the same way as in the perceptron and we will show how to improve the computer performance of the backward learning algorithm. We will also mention the Hopfield network, training algorithm with two basic phases and two-way associative memory. We will also present self-organizing neural networks and explore Hebbian's and competitive learning. Finally, we will present an example of perceptron training.

Keywords: Neurons, Networks, Learning, Mechanism, Perceptron

IZJAVA

Izjavljujem da sam završni rad izradila samostalno, isključivo znanjem stečenim na Odsjeku za politehniku Filozofskoga fakulteta u Rijeci, služeći se navedenim izvorima podataka i uz stručno vodstvo mentora mr.sc. Gordana Đurovića.

U Rijeci, rujan 2017.

Sadržaj

1. Uvod

„Računalo nije pokazalo još ništa“, izjavio je Garry Kasparov, svjetski prvak u šahu, nakon poraza u New Yorku u svibnju 1997. godine.“Da smo igrali pravi natjecateljski meč, ja bi srušio„Deep Blue“ u komadiće.“Ali Kasparov trud, kako bi umanjio značenje poraza u šest igara utakmice, bio je uzaludan. Činjenica da je Kasparov - vjerojatno najveći šahist svijeta - bio poražen od strane računala, bila je prekretnica u potrazi za inteligentnim strojevima. IBM-ovo superračunalo po imenu „Deep Blue“ je sposobno za analizu 200 milijuna pozicija u sekundi, i čini se da pokazuje inteligentne misli. U jednoj je fazi Kasparov čak optužio stroj za varanje. Bilo je mnogo, mnogo otkrića u ovoj utakmici, a jedan od njih je da ponekad računalo igra poteze koji su vrlo slični ljudskima. To je duboko razumijevanje pozicijskih faktora i izvanredan znanstveni uspjeh, te je to polako dovelo do toga da se počne više istraživati i razmišljati o umjetnoj inteligencji i pitanju njenog napretka.

U drugom poglavlju proći ćemo kroz osnovne pojmove umjetnih neuronskih mreža kao što je strojno učenje, te predstaviti građu neurona, kako bi u 3. poglavlju lakše razumijeli sam neuron u kontekstu računalnog elementa, te ćemo predstaviti perceptron kao najjednostavniji oblik neuronske mreže i pokušati objasniti kako on uči. U četvrtom poglavlju istražiti ćemo višeslojne neuronske mreže, te kako one uče. Peto poglavlje nas upoznaje sa Hopfieldovim mrežama, te algoritmom za trening istih, a šesto sa dvosmjernom asocijativnom memorijom, te njenim radom. U sedmom poglavlju, predstaviti ćemo samoorganiziracijske neuronske mreže i istražiti Hebbian i konkurentno učenje, nakon čega ćmo u osmom poglavlju prikazati primjer treniranja perceptrona za obavljanje logičkih operacija AND i OR. Na kraju rada je dan zaključak.

2. Osnovni pojmovi

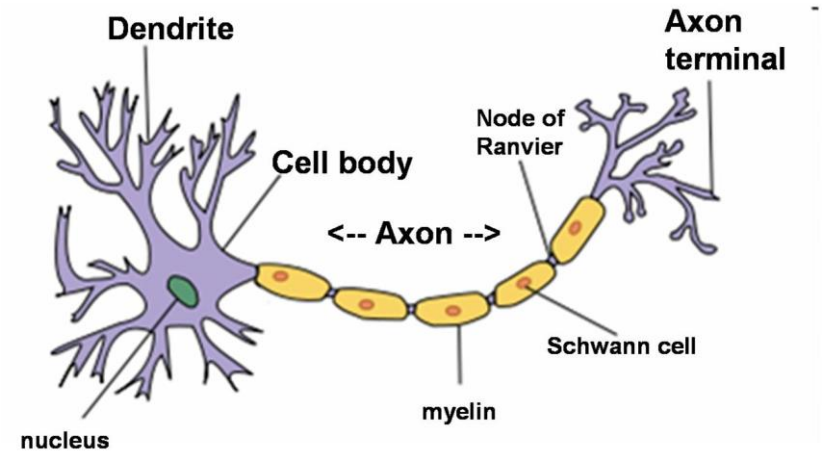
2.1. Strojno učenje

Općenito, stroj za učenje uključuje adaptivne mehanizme koji omogućuju računalima naučiti iz iskustva, tj. uče primjerom i analogijom. Učenje daje mogućnosti da mogu poboljšati performanse inteligentnog sustava tijekom vremena. Mehanizmi učenja strojeva čine osnovu za prilagodljive sustave. Najpopularniji pristupi strojnog učenja su umjetne neuronske mreže i genetike algoritama.

2.2. Osnove neuronskih mreža

Umjetna neuronska mreža može se definirati kao model zaključivanja na temelju ljudskog mozga. Mozak se sastoji od gusto međusobno povezanih skupa živčanih stanica ili osnovnih jedinica za obradu podataka - neurona. Ljudski mozak sadrži približno 10 milijardi neurona i 60 bilijuna sinapsi (veza) između njih. Pomoću istovremenog korištenja više neurona, mozak može obavljati svoje funkcije mnogo brže od najbržih računala u današnjem životu.

Iako svaki neuron ima vrlo jednostavnu strukturu, vojska takvih elemenata predstavlja ogromnu procesorsku snagu. Neuron se sastoji od tijela stanice - soma, vlakana zvanih dendriti i jednog dužeg vlakna - aksona. Dok se dendriti granaju u mreži oko soma, akson se proteže na dendritima i somama drugih neurona. Signali se prenose iz jednog u drugi neuron složenim elektro-kemijskim reakcijama. Kemijske tvari koje se ispuštaju iz sinapsi uzrokuju promjenu električnog potencijala tijela stanica. Kada potencijal dosegne prag, električni impuls šalje akcijski potencijal preko aksona. Puls se prostire i na kraju dođe do sinapse, te poveća ili smanji njen potencijal.



Slika 2.1. Građa neurona.

Ipak, najzanimljivije je otkriće to da neuronska mreža pokazuje plastičnost. Kao odgovor na simulacijski uzorak, neuroni pokazuju dugoročne promjene u snazi njihovih veza. Neuroni također mogu formirati nove veze s drugim neuronima. Čak i cijele kolekcije neurona ponekad mogu seliti s jednog mjesta na drugo. Ovi mehanizmi čine osnovu za učenje u mozgu. Naš se mozak može smatrati vrlo složenim, nelinearnim i paralelnim sustavom obrade informacija. Podaci se pohranjuju i obrađuju u neuronske mreže istovremeno kroz cijelu mrežu, a ne na pojedinim lokacijama. Drugim riječima, u neuronskim mrežama, podaci i njihova obrada su globalni, a ne lokalni. Zahvaljujući plastičnosti, jačaju veze između neurona koji dovode do "pravog odgovora", dok oni koji vode do "pogrešnog odgovora" slabe. Kao rezultat toga, neuronske mreže imaju sposobnost učenja kroz iskustvo.

Učenje je osnovna i bitna karakteristika bioloških neuronskih mreža. Jednostavnost i prirodnost s kojom se može učiti dovelo je do pokušaja da se natječe s neuronskim mrežama u računalu. Iako današnje umjetne neuronske mreže podsjećaju na ljudski mozak koliko papirnati avion podsjeća na Supersonic zrakoplov, to je veliki korak unaprijed. Umjetne neuronske mreže su sposobne za „učenje", to jest, one koriste iskustva kako bi poboljšale svoje preformanse. Mogu prepoznati rukom pisane znakove, prepoznavati riječi u ljudskom govoru, i otkrivati eksplozive na aerodromima. Štoviše, umjetne neuronske mreže mogu promatrati uzorke koje ljudski stručnjaci ne prepoznaju. Na primjer, banka Chase Manhattan koristi neuronske mreže kako bi ispitali niz informacija o korištenju ukradenih kreditnih kartica.

2.3. Modeliranje mozga pomoću umjetnih neuronskih mreža

Umjetna neuronska mreža sastoji se od nekoliko vrlo jednostavnih i međusobno povezanih procesora, koji se nazivaju neuroni, koji su analogni biološkim neuronima u mozgu. Neuroni su povezan s vezama čiji signali prelaze s jednog neurona na drugi. Svaki neuron prima određeni broj ulaznih signala putem svojih veza; međutim to nikada ne daje više od jednog izlaznog signala. Izlazni signal se prenosi preko neurona izlaznog spoja (koji odgovara biološkom aksonu). Odlazna se veza dijeli u nekoliko grana koje emitiraju isti signal (signal nije podijeljen među tim granama na bilo koji način). Izlazne grane završavaju na dolaznim priključcima drugih neurona u mreži.

2.4. Podučavanje umjetne neuronske mreže

Neuroni su povezani vezama, a svaka veza ima numeričku težinu povezanu s njom. Težine su osnovni načini dugoročnog pamćenja u umjetnim neuronskim mrežama. Oni izražavaju snagu, ili drugim riječima važnost, za svaki ulaz neurona. Umjetna neuronska mreža „uči" kroz ponovljene prilagodbe tih težina.

Biološka neuronska mreža	Umjetna neuronska mreža
Soma	Neuron
Dendriti	Ulaz
Akson	Izlaz
Sinapse	Težine

Tablica 2.1. Analogija između bioloških i umjetnih neuronskih mreža

2.5. Podešavanje važnosti u neuronskim mrežama

Tipična umjetna neuronska mreža sastoji se od hijerarhije slojeva, a neuroni u mrežama su raspoređeni uzduž tih slojeva. Neuroni povezani s vanjskom okolinom čine ulazne i izlazne slojeve. Važnost modificiranja je u usklađivanju ulaznog i izlaznog ponašanja mreže s vanjskom okolinom. Svaki neuron je osnovna informacijska procesorska jedinica. To je sredstvo za računanje razine aktivacije s obzirom na ulazne i numeričke težine.

Za izgradnju umjetne neuronske mreže, moramo odlučiti prvo koliko ćemo neurona koristiti i kako se neuroni trebaju povezati da bi stvorili mrežu. Drugim riječima, mi prvo moramo odabrati mrežne arhitekture, a onda ćemo odlučiti koji algoritam koristiti za učenje. Na kraju obučavamo neuronsku mrežu, odnosno, mi inicijaliziramo težinu mreže i ažuriramo težinu iz skupa primjera treninga. Počnimo s neuronom koji je osnovni građevni element umjetnih neuronskih mreža.

3. Neuron kao jednostavan računalni element

Neuron prima nekoliko signala od svojih ulaznih veza, izračunava novu razinu aktivacije i šalje ga kao izlazni signal kroz izlazne veze. Ulazni signal može biti neobrađeni podatak ili izlaz drugih neurona. Izlazni signal može biti ili konačno rješenje za problem ili ulaz prema drugim neuronima.

3.1. Određivanje izlaza pomoću neurona

Godine 1943., Warren McCulloch i Walter Pitts predložili su vrlo jednostavnu ideju koja je još uvijek osnova za većinu umjetnih neuronskih mreža. Neuron izračunava prosječni zbroj ulaznih signala i uspoređuje rezultate s vrijednošću praga, θ. Ako je mrežni ulaz manji od praga, izlazni neuron je -1. Ali, ako je mrežni ulaz veći ili jednak pragu, neuron se aktivira a izlaz dosegne vrijednost +1. Drugim riječima, neuron koristi sljedeći prijenos ili aktivacijsku funkciju:

$$X = \sum_{i=1}^{n} X_i \, W_i$$

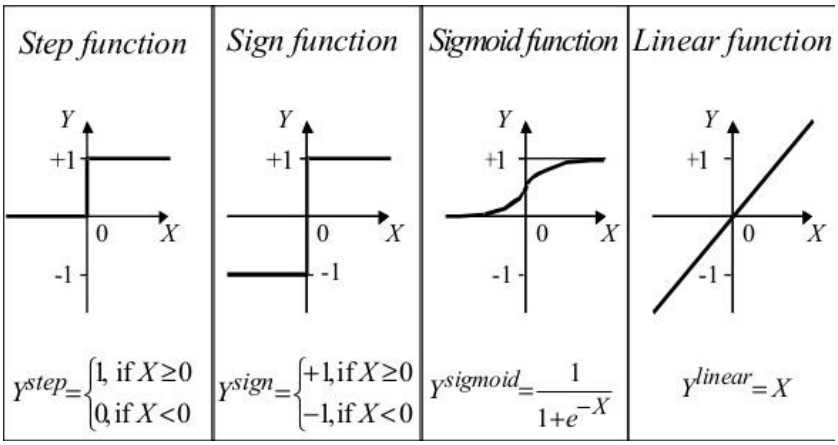
$$Y = \begin{cases} +1 & if \; X \geq \theta \\ -1 & if \; X < \theta \end{cases}$$

gdje je *X* mrežni prosječni ulaz u neuron, X_i je vrijednost unosa i, W_i je težina unosa, n je broj neurona na ulazu, a *Y* je izlaz neurona. Ova vrsta funkcije za aktivaciju se naziva znak funkcije. Tako se stvarni izlaz neurona s funkcijom aktivacije može se prikazati kao:

$$Y = sign \left[\sum_{i=1}^{n} X_i \, W_i - \theta \right]$$

3.2. Znak funkcije

Mnoge funkcije aktivacije su testirane, ali samo nekoliko njih našlo se u praktičnim primjenama. Četiri zajednička izbora - korak, znak, linearna i sigmoidna funkcija - prikazani su na slici 3.1. Korak i znak aktivacije funkcije, koji se nazivaju hard limit funkcijama, često se koriste u neuronima odlučivanja za razvrstavanje i raspoznavanje uzoraka zadataka.

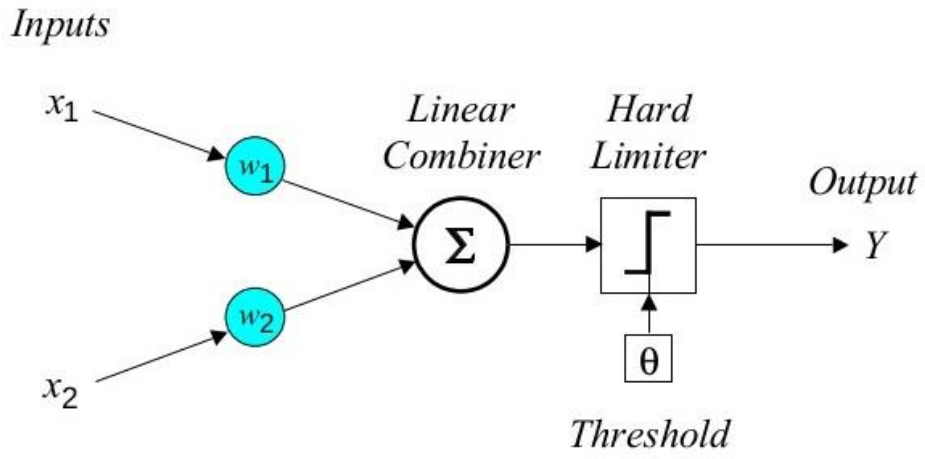


Slika 3.1. Funkcije aktivacije neurona.

Sigmoidna funkcija pretvara ulaz, koji može imati bilo koju vrijednost između plus i minus beskonačno, u razumnu vrijednosti u rasponu između 0 i 1. Neuroni s tom funkcijom koriste se u mrežama povratnog širenja. Linearna aktivacija funkcije daje izlaz koji je jednak neuronskom prosječnom ulazu. Neuroni s linearnom funkcijom često se koriste za linearne aproksimacije.

3.3. Učenje neurona

Godine 1958. Frank Rosenblatt predstavio je algoritam za trening koji je osigurao prvi postupak za obuku jednostavne umjetne neuronske mreže: Perceptron. Perceptron je najjednostavniji oblik neuronske mreže. Sastoji se od jednog neurona s podesivim sinaptičkim težinama i tvrdim graničnicima. Jednoslojni dvo-ulazni perceptron je prikazan na slici 3.2.



Slika 3.2. Jednoslojni dvo-ulazni perceptron.

3.4. Perceptron

Rad Rosenblattovog perceptrona se temelji na McCulloch-ovom and Pitts-ovom neuronskom modelu. Model se sastoji od linearnog kombinatora praćenog tvrdim graničnikom. Određeni zbroj ulaza primjenjuje se na tvrdi graničnik, koji daje izlaz jednak +1 ako je njegov unos pozitivan i -1 ako je negativan. Cilj perceptrona je klasificirati ulaze, drugim riječima, izvana primijenjene podražaje x_1, x_2, ..., x_n, sortirati u jednu od dvije klase, A1 ili A2. Dakle, u slučaju elementarnog perceptrona, N-dimenzionalni prostor podijeljen je s hiperplanom u dvije odlučujuće regije. Hiperplan je definiran linearno odvojivom funkcijom:

$$\sum_{i=1}^{n} X_i W_i - \theta = 0$$

3.5. Perceptron i razvrstavanje

To se postiže malim prilagodbama u težinama kako bi se smanjila razlika između stvarnih i željenih izlaza perceptrona. Početne težine su nasumično podijeljene, obično u rasponu (-0.5, 0.5), a zatim ažurirane kako bi dobili rezultat sukladan primjeni obuke. Za perceptron proces ažuriranja težine je posebno jednostavan. Ako je *p* iteracija, stvarni izlaz je Y(p) i željeni je izlaz $Y_d(p)$, a zatimje pogreška dana:

$e(p) = Y_d(p) – Y(p)$ gdje je p=1,2,3...

Iteracija *p* ovdje se odnosi na *p*-ti trening s primjerom prikazanim na perceptronu.Ako je pogreška, e(p), pozitivna, moramo povećati izlaz perceptrona Y(p), ali ako je negativan, moramo ga smanjiti. Uzimajući u obzir da svaki perceptronski ulaz doprinosi $x_i(p)$ x $w_i(p)$ ukupnom izlazu X(p), zaključujemo da ako je ulazna vrijednost pozitivna, povećanje njezine težine povećava perceptronski izlaz, dok ako je negativna, povećanje teži smanjenju. Dakle, sljedeće pravilo učenja perceptrona može se utvrditi:

$$w_i(p+1) = w_i(p) + \alpha \times x_i(p) \times e(p),$$

gdje je stopa učenja α, pozitivna konstanta.

Pravilo učenja perceptrona prvi je iznio Rosenblatt 1960. Koristeći ovo pravilo možemo izvesti perceptronski algoritam obuke za klasifikaciju zadataka.

1.korak: Inicijalizacija

Postavljanje početne težine w_1, w_2, ..., w_n i prag θ, za nasumične brojeve u rasponu [-0.5, 0.5].

2.korak: Aktivacija

Aktiviranje perceptrona primjenom ulaza $x_1(p)$, $x_2(p)$, ..., $x_n(p)$ i željenim rezultatom $Y_d(p)$. Izračunti stvarni izlaz za iteraciju p = 1

$$Y(p) = step\left[\sum_{i=1}^{n} X_i(p)w_i(p) - \theta\right]$$

gdje je *n* broj perceptronskih ulaza, a *step* je korak aktivacijske funkcije.

3.korak: Trening sa promjenama težina

Ažurirati težine perceptrona

$$w_i(p+1) = w_i(p) + \Delta w_i(p)$$

gdje je $\Delta w_i(p)$ težina točne iteracije *p*. Ispravak težine izračunava se po pravilu delta:

$$\Delta W_i(p) = \alpha * X_i(p) * e(p)$$

4.korak: Ponavljanje

Povećati ponavljanje *p* za jedan, vratiti se na korak 2. i ponoviti postupak do usklađivanja.

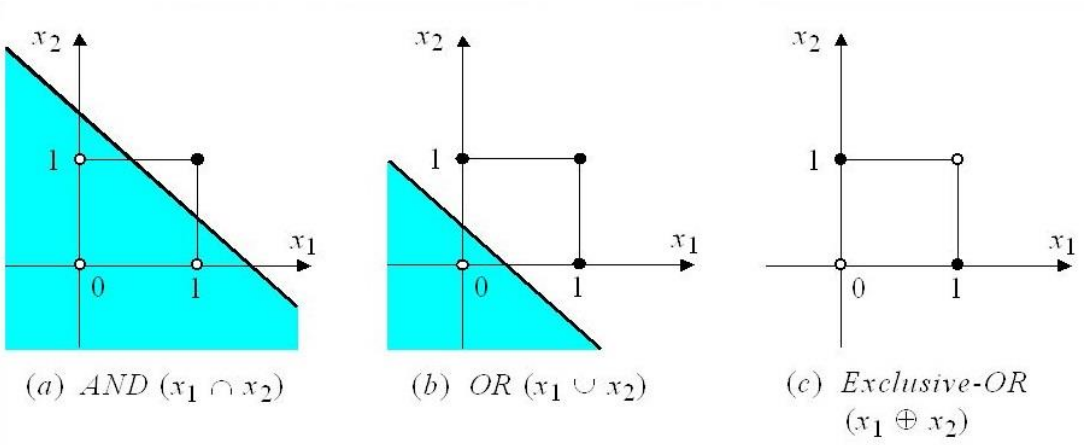
3.6. Perceptron i osnovne logičke operacije (I, ILI ili ekskluzivni-ILI)

Tablica istinitosti operacija I, ILI i Ekskluzivni-ILI prikazani su u tablici 3.1. Tablica predstavlja sve moguće kombinacije vrijednosti za dvije varijable, X1 i X2. Perceptroni moraju biti osposobljeni za klasifikaciju ulaznih uzoraka. Prvo ćemo razmotriti operaciju I. Nakon završetka koraka inicijalizacije, perceptron se aktivira slijedom od četiri uzorka ulaza koji predstavljaju epohu. Preceptronske težine se ažuriraju nakon svakog aktiviranja. Ovaj postupak se ponavlja dok se sve težine usklađuju na jedinstvenom nizu vrijednosti.

Ulazne variable		I	ILI	Ekskluzivni ILI
X1	X2	$X1 \cap X2$	$X1 \cup X2$	$X1\ E-OR\ X2$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Tablica 3.2. Tablica istinitosti osnovnih logičkih operacija.

Na sličan način, perceptron može naučiti operaciju ILI. Međutim, jednoslojni perceptron ne može se obučavati za izvođenje operacije Ekskluzivno-ILI. Malo geometrije može nam pomoći da razumijemo zašto je to tako. Slika 3.3 predstavlja I, ILI i Ekskluzivno-ILI kao dvodimenzionalna polja na temelju vrijednosti dva ulaza. Točke u ulaznom prostoru, gdje je izlaz funkcije 1, označene su crnim točkama, a točke gdje je izlaz 0, označene su bijelim točkicama.



Slika 3.3. Dvodimenzonalna polja osmovnih logičkih operacija.

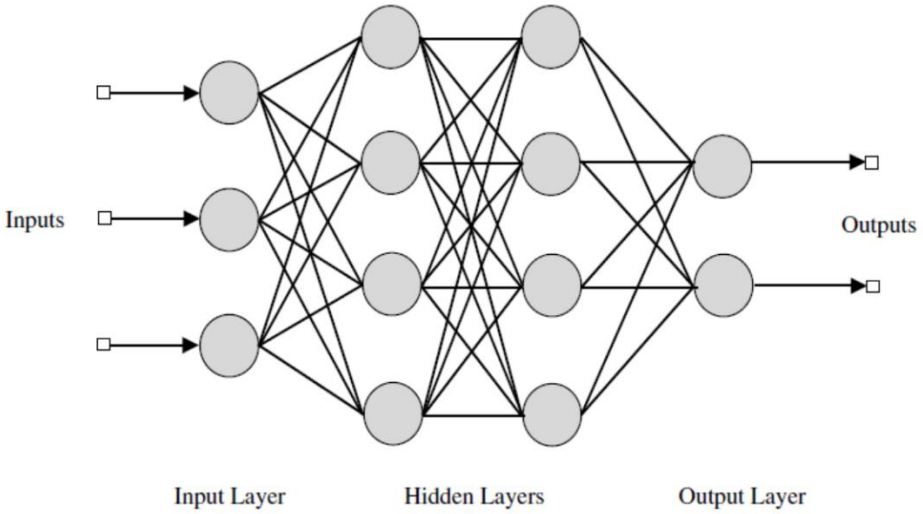
Na slikama 3.3. (a) i (b) možemo povući crtu, tako da su crne točkice na jednoj strani i bijele točkice na drugoj strani, ali točkice koje su prikazane na slici 3.3. (c) nisu odvojive od jednog retka. Perceptron može predstavljati funkciju samo ako postoji neka linija koja razdvaja sve crne točkice od svih bijelih točkica. Takve funkcije nazivaju se linearno odvojive. Dakle, perceptron može naučiti operacije I i ILI, ali ne Ekskluzivno-ILI. No zašto je to tako? Perceptronski izlaz Y je 1 samo ako je ukupni ulaz X veća ili jednaka vrijednosti praga θ. To znači da je cijeli ulazni prostor podijeljen na dva dijela duž granice definirane x = 0. Na primjer, linija deljenja za operaciju I definira se jednadžbom:

$$X_1W_1 + X_2W_2 = \theta$$

Činjenica da perceptron može naučiti samo linearno odvojive funkcije je prilično loša vijest, jer nema puno takvih funkcija. Postavlja se pitanje, kako se nositi s problemima koji nisu linearno odvojivi? Za takve probleme trebamo višeslojne neuronske mreže. U stvari, povijest je pokazala da se ograničenja Rosenblattova perceptrona mogu prevladati naprednim oblicima neuronskih mreža, primjerice višeslojni perceptroni s algoritmom povratnog širenja.

4. Višeslojne neuronske mreže

Višeslojni perceptron je preuranjena neuronska mreža s jednim ili više skrivenih slojeva. Tipično, mreža se sastoji od ulaznog sloja izvora neurona, barem jednog srednjeg ili skrivenog sloja računalnih neurona i izlaznog sloja računalnih neurona. Ulazni signali se prenose u smjeru prema naprijed na „sloj po sloj“ osnovi.



Slika 4.1. Višeslojni perceptron sa dva skrivena sloja.

4.1. Skriveni sloj

Svaki sloj višeslojne neuronske mreže ima svoju specifičnu funkciju. Ulazni sloj prihvaća ulazne signale iz vanjskog svijeta i redistribuira te signale na sve neuone u skrivenom sloju. Zapravo, ulazni sloj rijetko uključuje računalne neurone, i stoga ne obrađuje ulazne uzorke. Izlazni sloj prihvaća izlazne signale, ili drugim riječima, poticajni uzorak iz skrivenog sloja i uspostavlja izlazni uzorak cijele mreže.

Neuroni u skrivenom sloju otkrivaju značajke; težine neurona predstavljaju značajke skrivene u ulaznim uzorcima. Te se značajke koriste u izlaznom sloju u određivanju izlazne sheme. S jednim skrivenim slojem možemo predstavljati svaku kontinuiranu funkciju ulaznih signala, a s dva skrivena sloja mogu se prikazati i diskontinuirane funkcije.

Komercijalne umjetne neuronske mreže uključuju tri, a ponekad i četiri sloja, uključujući jedan ili dva skrivena sloja. Svaki sloj može sadržavati od 10 do 1000 neurona. Eksperimentalne neuronske mreže mogu imati pet ili čak šest slojeva, uključujući tri ili četiri skrivena sloja, a koriste milijune neurona, ali većina praktičnih aplikacija koriste samo tri sloja, jer svaki dodatni sloj eksponencijalno povećava računalnu memoriju.

Skriveni sloj skriva svoj željeni izlaz. Neuroni u skrivenom sloju se ne mogu promatrati kroz ulazno/ izlazno ponašanje mreže. Ne postoji očigledan način da se sazna što željeni izlaz skrivenog sloja treba biti. Drugim riječima, željeni izlaz skrivenog sloja određuje sam sloj.

4.2. Učenje višeslojnih neuronskih mreža

Dostupno je više od stotinu različitih algoritama učenja, ali najpopularnija metoda je metoda povratnog širenja. Ova metoda je prvi put predložena 1969. godine, ali je zanemarena zbog svojih

zahtjevnih računanja. Tek sredinom osamdesetih godina ponovno je otkriven algoritam učenja povratnog širenja.

Učenje u višeslojnim mrežama odvija se na isti način kao i za perceptron. Na mreži je predstavljen set obuke obrazaca. Mreža izračunava svoju izlaznu shemu, a ako postoji pogreška ili razlika između stvarnih i željenih izlaznih uzoraka, težine se podešavaju kako bi se smanjila pogreška. U perceptronu postoji samo jedna težina za svaki ulaz i samo jedan izlaz. No, u višeslojnim mrežama, postoji mnogo težina, od kojih svaki doprinosi više no jednom izlazu.

U povratnom širenju neuronskih mreža, algoritam učenja ima dvije faze. Prvo, obrazac unosa obuke prikazan je mrežnim slojem unosa. Mreža se zatim širi ulaznim uzorkom od sloja do sloja dok izlazni uzorak generira izlazni sloj. Ako se ovaj uzorak razlikuje od željenog izlaza, greška se računa, a zatim širi unatrag kroz mrežu od izlaznog do ulaznog sloja. Vrijednosti se modificiranju kako se greška propagira. Kao i kod bilo koje druge neuronske mreže, povratno širenje određuje se vezama između neurona, aktivacijskom funkcijomkoju koriste neuroni i algoritmom učenja koji određuje postupak podešavanja težine. Mreža povratnog širenja je višeslojna mreža koja ima tri ili četiri sloja. Slojevi su potpuno povezani, to jest, svaki je neuron u svakom sloju spojen na svaki drugi neuron u susjednom prosljeđenom sloju.

Neuron određuje svoj izlaz na način sličan Rosenblattovu perceptronu. Prvo, ona izračunava neto količinu ulaznih podataka kao i prije:

$$X = \sum_{i=1}^{n} xiwi - \theta$$

pri čemu je *n* broj ulaza, a θ je prag koji se primjenjuje na neuronu. Zatim te ulazne vrijednosti prolaze kroz funkciju aktivacije. Međutim, za razliku od perceptrona, neuroni u povratnim mrežama mogu imati sigmoidnu funkciju aktivacije:

$$Y^{sigmoid} = \frac{1}{1 + e^{-X}}$$

Derivaciju ove funkcije je lako izračunati. Ona također jamči da je izlaz neurona ograničen između 0 i 1.

4.3. Gradijent pogreška

Gradijent pogreške određen je kao derivat funkcije aktivacije pomnožen s pogreškom na izlazu neurona. Za neuron *k* u izlaznom sloju, imamo

$$\delta_k(p) = \frac{\vartheta y_k(p)}{\vartheta X_k(p)} * \ e_k(p)$$

gdje je $y_k(p)$ izlaz *k* neurona u iteraciju *p* i $X_k(p)$ je eto ponderirani ulaz neurona *k* na istu iteraciju.

4.4. Određivanje korekcije težine za neuron u skrivenom sloju

Za izračun korekcije težine u skrivenom sloju, možemo primijeniti iste jednadžbe kao za izlazni sloj:

$$\Delta W_{ij}(p) = \alpha * x_i(p) * \delta_j(p)$$

gdje $\delta_j(p)$ predstavlja gradijent pogreške na neuronu *j* u skrivenom sloju:

$$\delta_j(p) = y_j(p) * \left[1 - y_j(p)\right] * \sum_{k=1}^{l} \delta_k(p) W_{jk}(p)$$

gdje je *l* broj neurona izlaznog sloja;

$$y_j(p) = \frac{1}{1 + e^{-X_j(p)}}$$

$$X_j(p) = \sum_{i=1}^{n} x_i(p) * W_{ij}(p) - \theta_j$$

i *n* je broj neurona u ulazni sloj. Sada možemo izvesti algoritam povratnog širenja.

Korak 1: Inicijalizacija

Postaviti sve težine i razine praga mreže za slučajne brojeve ravnomjerno raspoređene unutar malog raspona:

$$\left(-\frac{2.4}{F_i}, +\frac{2.4}{F_i}\right)$$

gdje F_i je ukupan broj ulaza neurona u mreži. Inicijalizacija težine obavlja se u osnovi, neuron po neuron.

Korak 2: Aktivacija

Aktivirati neuronske mreže povratnog širenja s primjenom ulaza $x_1(p)$, $x_2(p)$, ..., $x_n(p)$ i željenim rezultatima.

(a) Izračunati stvarne izlaze neurona u skrivenom sloju:

$$y_j(p) = sigmoid\left[\sum_{i=1}^{n} x_i(p) * W_{ij}(p) - \theta_j\right]$$

kada je *n* jednak broju ulaza u neuron *j* skrivenog sloja i *sigmoid* je funkcija aktivacije sigmoida.

(b) Izračunati stvarne izlaze neurona u izlaznom sloju:

$$y_j(p) = sigmoid\left[\sum_{j=1}^{m} x_{jk}(p) * W_{jk}(p) - \theta_k\right]$$

gdje je *m* broj ulaza neurona *k* u izlaznom sloju.

Korak 3: Trening s težinama

Ažurirati težine u mreži povratnog širenja koji propagiraju pogreške vezane uz izlazne neurone.

(a) Izračunati gradijent pogreške za neurone u izlaznom sloju.
(b) Izračunati gradijent pogreške za neurone u skrivenom sloju.

Korak 4: Ponavljanje

Povećati iteraciju p za jedan, vratiti se na korak 2 i ponavljati postupak dok se ne zadovolji odabrani kriterij pogreške.

4.5. Sažimanje kvadriranih pogrešaka

Zbroj kvadrata grešaka je koristan pokazatelj uspješnosti mreže. Algoritam uvježbavanja povratnog širenja pogreške pokušava minimalizirati taj kriterij. Kada je vrijednost zbroja kvadratnih pogrešaka u cijelom prolazu, kroz sve sate ili epohe vježbanja, dovoljno mali, mreže se smatraju konvergentne.

Zanimljo je pitanje: Da li ista mreža može pronaći ista rješenja kada su početne težine i pragovi postavljeni nasumično? Mreža dobiva različite težine i vrijednosti praga kad se počne s različitim početnim uvjetima. Uvijek ćemo riješiti problem, iako koristimo različiti broj ponavljanja.

4.6. Metoda povratnog širenja za strojno učenje

Iako je široko korištena, metoda povratnog širenja pogreške nije otporna na probleme. Na primjer, algoritam učenja povratnog prostiranja pogreške ne funkcionira u biološkom svijetu. Biološki neuroni ne rade unatrag za podešavanje snage međusobnim vezama, sinapsama, a time se i učenje povratnog širenja pogreške ne može promatrati kao proces koji oponaša učenje nalik mozgu.

Drugi problem je da su izračuni opsežni i kao rezultat toga, a obuka je spora. Čisti algoritam povratnog širenja se rijetko koristi u praktičnoj primjeni. Postoji nekoliko mogućih načina kako poboljšati računalne učinkovitost algoritma širenja.

4.7. Ubrzano učenje u višeslojnim neuronskim mrežama

Višeslojna mreža, općenito, uči mnogo brže kada sigmodalna funkcija aktivacije predstavlja hiperbolički tangens,

$$Y^{\tan h} = \frac{2a}{1 + e^{-bX}} - a$$

gdje su *a* i *b* konstante. Prikladne vrijednosti za a i b su: a = 1,716, a b = 0.667. Mi također možemo ubrzati obuku, uključujući generalizirano delta pravilo:

$$\Delta w_{jk}(p) = \beta * \Delta w_{jk}(p-1) + \alpha * y_j(p) * \delta_k(p)$$

gdje β je pozitivan broj ($0 <= \beta < 1$) naziva se konstanta zamaha. Konstanta zamaha postavljena je na 0,95.

4.8. Konstanta zamaha

Prema zapažanjima Watrous-a i Jacobs-a, uključivanje zamaha u povratnim algoritmima ima stabilizirajući učinak na vježbama. Drugim riječima, uključivanje zamaha ima tendenciju ubrzati silazak u ravnomjernom smjeru spusta i usporiti proces kada površina učenja ima vrhove i doline.

4.9. Parametar stope učenja (α)

Jedan od najučinkovitijih načina za ubrzavanje konvergencije učenja povratnog širenja je podešavanje parametra brzine učenja tijekom obuke. Parametar male brzine učenja,α, uzrokuje male promjene težine u mreži od jedne iteracije do druge i tako dovodi do glatke krivulje učenja. S druge strane, ako je parametar stope učenja, α, postao veći kako bi se ubrzao proces obuke, što je rezultiralo većim promjenama u težinama, može doći do nestabilnosti i kao rezultat toga, mreža može postati promijenjiva. Kako bi se ubrzala konvergencija, a da se ipak izbjegne opasnost od nestabilnosti, mogu se primijeniti dvije heuristike.

- **Heuristika 1**. Ako promjena zbroja kvadrata grešaka ima isti predznak za nekoliko posljedičnih epoha, onda treba povećati parametar stope učenja.
- **Heuristika 2.** Ako je predznak za promjenu zbroja kvadrata grešaka alternativa za nekoliko posljedičnih epoha, onda parametar stope učenja treba smaniti.

Prilagođavanje brzine učenja zahtijeva neke promjene u povratnim algoritmima. Prvo, mrežni izlaz i pogreške izračunate su iz prvog parametra stope učenja. Ako zbroj kvadrata pogrešaka na sadašnjoj epohi premašuje prethodne vrijednosti za više od unaprijed definiranog omjera (obično 1,04) parametar stope učenja se smanjuje (obično množenjem s 0,7) i nove težine i prag su izračunate. Međutim, ako je greška manja od prethodne, stopa učenja se povećava (obično množenjem 1,05).

Prilagodba stope učenja može se koristiti zajedno sa zamahom. Korištenje zamaha i prilagođavanje brzine učenja značajno poboljšava karakteristike višeslojnih povratnih neuronskih mreža i smanjuje šanse da mreža može oscilirati.Neuronske mreže su dizajnirane analogno s mozgom. Memorija mozga, međutim, djeluje prema asocijacijama. Na primjer, možemo prepoznati poznato lice i u nepoznatom okruženju unutar 100 - 200 ms. Također se možemo prisjetiti potpunih osjetilni doživljaja, uključujući zvukove i prizore, te kada čujemo samo nekoliko taktova glazbe. Mozak rutinski povezuje jedno s drugim.

4.10. Simulacija asocijativne osobine ljudske memorije pomoću euronske mreže

Višeslojne neuronske mreže obučene algoritmom povratnog širenja, koriste se za probleme prepoznavanja uzoraka. No, kao što smo naglasili, takve mreže nisu istinski inteligentne.Za oponašanje ljudskih sjećanja, asocijativne karakteristike, potrebna nam je druga vrsta mreže: povratna neuronska mreža.

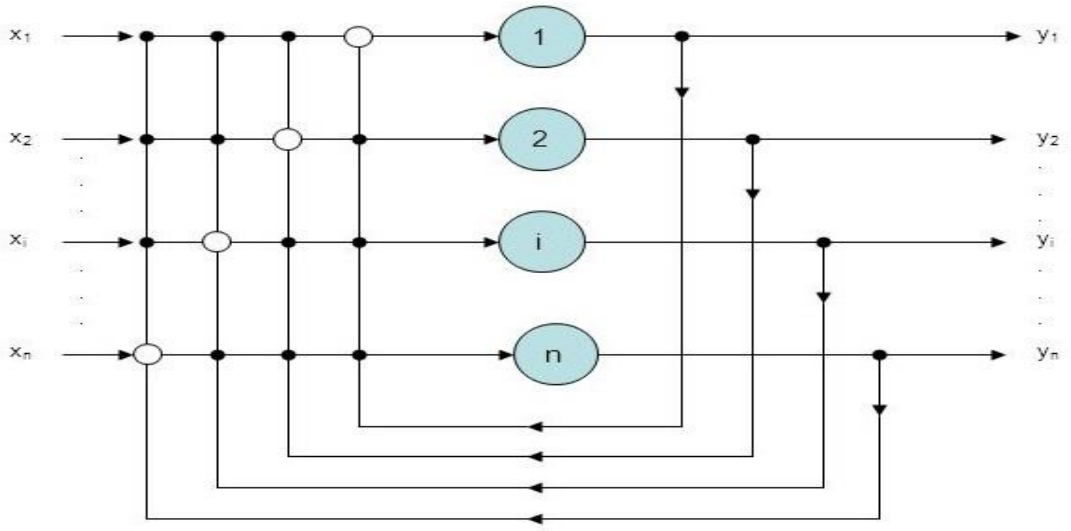
5. Hopfield mreže

Povratna neuronska mreža ima petlje povratne veze od svojih izlaza do ulaza. Prisutnost takvih petlji ima značajan utjecaj na sposobnost učenja mreže.

5.1. Učenje povratne mreže

Nakon primjene novog ulaza, mrežni izlaz se izračunava i vrati natrag za podešavanje ulaza. Onda se izlaz izračunava ponovno, a postupak se ponavlja sve dok izlaz ne postane konstantan.Uzastopna iteracija ne proizvodi uvijek sve manje i manje izlazne promjene, već naprotiv može dovesti do kaotičnog ponašanja. U takvom slučaju, mrežni izlaz nikada ne postaje konstanta, a za mrežu se kaže da je nestabilna.

Stabilnost povratnih mreža zaintrigirala je nekoliko istraživača 1960-ih i 1970-ih. Međutim, nitko nije mogao predvidjeti koje mreže će biti stabilne, a neki istraživači su bili pesimistični oko pronalaženja rješenja na sve. Problem je riješen tek 1982., kada je John Hopfield formulirao fizički princip pohranjivanja informacija u dinamički stabilnu mrežu. Slika 5.1. prikazuje jednoslojnu Hopfieldovu mrežu koja se sastoji od n neurona. Hopfieldova mreža obično koristi McCullochove i Pittove neurone s funkcijom aktiviranja znaka kao računalnim elementom.



Slika 5.1. Jednoslojna Hopfieldova mreža.

5.2. Funkcija aktivacije

Djeluje na sličan način kao znak funkcije. Ako je vagani neuronski ulazi manji od nule, izlaz je -1; ako su ulazi veći od nule, izlaz je 1. Međutim, ako je vagani neuronski ulaz točno nula, njegov izlaz ostaje nepromijenjen, drugim riječima, neuron ostaje u prethodnom stanju, bez obzira da li je 1 ili -1.

$$Y^{sign} = \begin{cases} +1, & if\ X > 0 \\ -1, & if\ X < 0 \\ Y, & if\ X = 0 \end{cases}$$

Znak aktivacije funkcije može biti zamijenjen sa zasićenom linearnom funkcijom, koja djeluje kao čista linearna funkcija u regiji [-1,1] i kao znak funkcije izvan ovog područja. U Hopfieldovim mrežama, sinaptičke težine između neurona obično su zastupljene u matrici kao što slijedi:

$$W = \sum_{m=1}^{M} Y_m Y_m^T - MI$$

gdje je *M* broj stanja koje mreža mora memorirati, Y_m je n-dimenzionalni binarni vektor, I nxn matrica identiteta i eksponent *T* označava transpoziciju matrice. Operacije Hopfieldovih mreža mogu biti geometrijski zastupljeni. Općenito, mreža sa n neurona ima 2^n mogućih stanja i povezana je s n-hiperkocki.

5.3. Testiranje Hopfieldove mreže

Aktiviramo ju primjenom ulaznog vektora *X*. Zatim izračunamo stvarni izlazni vektor *Y*, a na kraju uspoređujemo rezultat s početnim ulazni vektorom *X*.

$$Y_m = sign(WX_m - \theta)$$
$$m = 1,2,\dots,M$$

gdje je θ prag matrice. U našem primjeru, možemo pretpostaviti da su svi pragovi nula. Dakle,

$$Y_1 = sign\left\{\begin{bmatrix}0 & 2 & 2\\ 2 & 0 & 2\\ 2 & 2 & 0\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix} - \begin{bmatrix}0\\0\\0\end{bmatrix}\right\} = \begin{bmatrix}1\\1\\1\end{bmatrix}$$

i

$$Y_2 = sign\left\{\begin{bmatrix}0 & 2 & 2\\ 2 & 0 & 2\\ 2 & 2 & 0\end{bmatrix}\begin{bmatrix}-1\\-1\\-1\end{bmatrix} - \begin{bmatrix}0\\0\\0\end{bmatrix}\right\} = \begin{bmatrix}-1\\-1\\-1\end{bmatrix}$$

Kao što vidimo, $Y_1 = X_1$ i $Y_2 = X_2$. Tako se za oba stanja, (1,1,1) i (-1, -1, 1) su, kaže da su stabilna.

A što je sa drugim stanjima? Sa tri neurona u mreži, imamo osam mogućih stanja. Preostalih šest stanja su nestabilna. Međutim, stabilna stanja (nazivamo ih temeljnim sjećanjima) su sposobna privlačiti stanja koja su im bliska. Svaki od tih nestabilnih stanja predstavlja jednu pogrešku, u odnosu na temeljnu memoriju (1,1,1). S druge strane, temeljna memorija (-1, -1, -1) privlači nestabilna stanja (-1, -1,1), (-1,1, -1) i (1,-1,-1). Ovdje opet, svaka od nestabilnih stanja predstavlja jednu pogrešku, u usporedbi s temeljnom memorijom. Na taj način, Hopfieldova mreža može doista djelovati kao mreža za ispravljanje pogrešaka. Sada sumiramo Hopfieldov mrežni algoritam treninga.

Korak 1: Skladištenje

N-neurona Hopfieldove mreže potrebno je pohraniti u niz M temeljnih sjećanja, Y_1, Y_2, ..., Y_M. Sinaptičke težine od neurona *i* do neurona *j* izračunavaju se kao

$$w_{ij} = \begin{cases} \sum_{m=1}^{M} y_{m,i} y_{m,j}, & i \neq j \\ 0, & i = j \end{cases}$$

gdje $Y_{m,i}$ i $Y_{m,j}$ su i-ti i j-ti elementi temeljne memorije Y_m. U matričnoj fomi, sinaptičke težine između neurona su zastupljene kao:

$$W = \sum_{m=1}^{M} Y_m Y_m^T - MI$$

Hopfield mreža može pohraniti niz temeljnih uspomena ako je težina matrice simetrična s nulama u svojoj glavnoj dijagonali. To je,

$$W = \begin{bmatrix} 0 & w_{12} & \dots & w_{1i} & \dots & w_{1n} \\ w_{21} & 0 & \dots & w_{2i} & \dots & w_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ w_{i1} & w_{i2} & \dots & 0 & \dots & w_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{ni} & \dots & 0 \end{bmatrix}$$

gdje je $w_{ij}=w_{ji}$. Nakon što su težine izračunate, ostaju fiksne.

Korak 2: Testiranje

Moramo potvrditi da je Hopfieldova mreža je sposobna za podsjećanje na sve temeljne uspomene. Drugim riječima, mreža se mora podsjetiti na sve temeljne uspomene Y_m kada je predstavljena kao ulaz. To je,

$$x_{m,i} = y_{m,i}, \quad i = 1,2, \dots, n; \quad m = 1,2, \dots, M$$

$$y_{m,i} = sign\left(\sum_{j=i}^{n} w_{ij} x_{m,j} - \theta_i \right)$$

gdje je $Y_{m,i}$ i-ti element stvarnog izlaznog vektora Y_mi $x_{m,j}$ je j-ti element ulaznog vektora X_m. U matričnom obliku,

$$X_m = Y_m, \quad m = 1,2, \dots, M$$
$$Y_m = sign(WX_m - \theta)$$

Ako su sva temeljna sjećanja savršeno dohvaćena možemo nastaviti na sljedeći korak.

Korak 3: Vađenje

Predstaviti nepoznati n-dimenzionalni vektor (sonda), X, na mrežu i dohvatiti stabilno stanje. Tipično, Sonda predstavlja oštećenu ili nepotpunu verziju temeljne memorije, koji je

$$X \neq Y_m, \quad m = 1,2, \dots, M$$

(a) Inicijalizirati algoritam za pronalazak Hopfieldovei izračunati početno stanje za svaki neuron.

(b) Ažurirati elemenat stanja vektora, Y(p). Neuroni za ažuriranje biraju se nesinkronizirano, slučajno i jedan po jedan. Ponavlja se inačicu dok vektor stanja ne postane nepromijenjen ili drugim riječima, postigne stabilno stanje. Uvjet za stabilnost može se definirati kao:

$$y_i(p+1) = sign\left(\sum_{j=1}^{n} w_{ij}x_j(p) - \theta_i\right), \qquad i = 1,2,\dots,n$$

Ili u matričnom obliku

$$Y(p+1) = sign[WY(p) - \theta]$$

Hopfieldova mreža uvijek će konvergirati u stabilnom stanju, ako se dohvat obavlja asinkronizirano. Međutim, to stabilno stanje ne mora nužno predstavljati jednu od temeljnih sjećanja i ako temeljna memorija nije nužno najbliža. Pretpostavimo, na primjer, želimo pohraniti tri temeljne uspomene u pet neurona Hopfieldove mreže:

$$X_1 = (+1,+1,+1,+1,+1)$$
$$X_2 = (+1,-1,+1,-1,+1)$$
$$X_3 = (-1,+1,-1,+1,-1)$$

Matrica težine,

$$W = \begin{bmatrix} 0 & -1 & 3 & -1 & 3 \\ -1 & 0 & -1 & 3 & -1 \\ 3 & -1 & 0 & -1 & 3 \\ -1 & 3 & -1 & 0 & -1 \\ 3 & -1 & 3 & -1 & 0 \end{bmatrix}$$

Pretpostavimo da je sada sonda vektora predstavljena X=(+1, +1, -1, +1, +1). Ako usporedimo ovu sondu sa temeljnom memorijom X_1, možemo vidjeti da se ta dva vektora razlikuju samo malo. Dakle, možemo očekivati da će sonda X konvergirati u temeljnu memoriju X_1. Kada primijenimo opisani algoritam Hopfield mrežnog treninga, dobivamo drugi rezultat. Uzorak koji proizvodi mreža podsjeća na memoriju X_3, lažnu memoriju. Ovaj primjer pokazuje jedan od problema nerazdvojivosti na Hopfieldovim mrežama. Drugi problem je kapacitet memorije ili najveći broj temeljnih uspomena koje se mogu pohraniti i dohvatiti ispravno. Hopfield je eksperimentalno pokazao da je najveći broj osnovnih sjećanja M_{max} koji se može pohraniti u n-neuon rekurentne ograničene mreže.

$$M_{max} = 0.15n$$

Također se može definirati kapacitet od Hopfieldove mreže na osnovi toga da se većina temeljnih sjećanja preuzmu savršeno:

$$M_{max} = \frac{n}{2\ln n}$$

5.4. Preuzimanje temeljnih uspomena

Može se pokazati da za savršeno preuzimanje svih temeljnih uspomena, njihov broj mora biti prepolovljen:

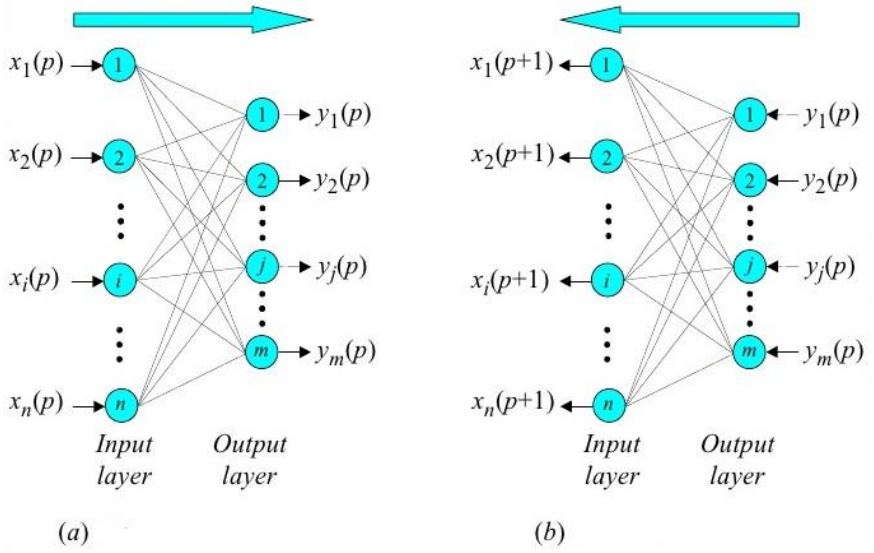
$$M_{max} = \frac{n}{4 \ln n}$$

Kao što možemo vidjeti, kapacitet skladišta Hopfieldove mreže mora biti prilično mali da bi temeljne uspomene bilo moguće dohvatiti. To je jedan od glavnih ograničenja Hopfieldove mreže. Strogo govoreći, Hopfieldova mreža predstavlja auto-asocijativni tip memorije. Drugim riječima, Hopfield mreža može dohvatiti oštećene ili nepotpune memorije, ali ne može ga povezati s drugim različitim memorijama. Nasuprot tome, ljudsko pamćenje je asocijativno. Jedna stvar može nas podsjetiti na drugu. Mi koristimo lanac mentalnih asocijacija za oporavak izgubljene memorije. Ako, na primjer, zaboravimo gdje smo ostavili kišobran, trebamo se sjetiti gdje smo ga zadnji put imali, što smo radili ili s kime smo razgovarali. Stoga pokušavamo uspostaviti lanac asocijacija, i time vratiti izgubljenu memoriju.

Zašto Hopfield mreža ne može raditi ovaj posao? Hopfield mreža je jednoslojna mreža, a time se pojavljuje izlazni uzorak na istom skupu neurona na koji je primijenjen ulazni uzorak. Kako bi povezali jednu memoriju s drugom, potrebna nam je rekurentna neuronska mreža sposobna za prihvaćanje ulaznog uzorka na jednom skupu neurona i proizvodnju povezanog, ali različitog izlaznog uzorka o nekom drugom skupu neurona. U stvari, trebamo dvoslojnu povratnu mrežu, dvosmjernu asocijativnu memoriju.

6. Dvosmjerna asocijativna memorija (DAM)

Dvosmjerno asocijativne memorije, koje je prvipredložio Bart Kosko, su heteroasociativne mreže. One povezuju uzorke iz jednog seta, set A, na obrasce iz drugog seta, set B i obrnuto. Kao i Hopfieldova mreža, dvosmjerno asociativna memorija, može generalizirati i proizvoditi korektne izlaze usprkos oštećenim ili nepotpunim ulazima. Osnovna arhitektura dvosmjerno asocijativne memorija je prikazana na slici 6.1. Sastoji se od dva potpuno spojena sloja: ulazni sloj i izlazni sloj.



Slika 6.1. DAM operacije (a) smjer unaprijed (b) smjer unatrag.

6.1. Rad dvosmjerno asocijativne memorije

Ulazni vektor X(p) se primjenjuje na transponiranu težinu matrice W^T da se dobije izlazni vektor Y(p), kao što je prikazano na slici 6.1.(a). Tada se izlazni vektor Y(p) nanosi na težinu matrice W kako bi proizveo novi ulazni vektor X(p + 1), kao što je na slici 6.1.(b). Ovaj proces se ponavlja sve dok ulazni i izlazni vektori ne postanu nepromijenjeni, ili drugim riječima, dvosmjerna asocijativna memorija postigne stabilno stanje.

Kako bi razvili dvosmjernu asocijativnu memoriju, moramo stvoriti korelacijske matrice za svaki par uzoraka koji želimo pohraniti. Korelacijka matrica je matrični produkt ulaznog vektora X i transponiranog izlaznog vektora Y^T. Težina dvosmjerne asocijativne memorije je zbroj svih korelacija matrice, koja je,

$$W = \sum_{m=1}^{M} X_m Y_m^T$$

gdje M predstavlja broj parova uzorka koji se pohranjuju u dvosmjernu asocijativnu memoriju. Kao i Hopfieldova mreža, dvosmjerno asocijativna memorija obično koristi McCulloch-ove i Pitts-ove neurone s aktivacijom znaka funkcije. Algoritam treninga dvosmjerne asocijativne memorije može se prikazati kako slijedi.

Korak 1: Skladištenje

Dvosmjerna asocijativna memorija je potrebna za pohranu M parova uzoraka. Na primjer, možda želite pohraniti četiri para:

$$\text{Set A: } X_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} X_2 = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} X_3 = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \end{bmatrix} X_4 = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

$$\text{Set B: } Y_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} Y_2 = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} Y_3 = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} Y_4 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

U tom slučaju, ulazni sloj dvosmjerno asocijativne memorije mora imati šest neurona, a izlazni sloj tri neurona. Težina matrice se određuje:

$$W = \sum_{m=1}^{4} X_m Y_m^T$$

Ili

$$W = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} [1 \quad 1 \quad 1] + \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} [-1 \quad -1 \quad -1] + \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \end{bmatrix} [1 \quad -1 \quad 1] + \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} [-1 \quad 1 \quad -1]$$

$$= \begin{bmatrix} 4 & 0 & 4 \\ 4 & 0 & 4 \\ 0 & 4 & 0 \\ 0 & 4 & 4 \\ 4 & 0 & 4 \\ 4 & 0 & 4 \end{bmatrix}$$

Korak 2: Testiranje

Dvosmjerna asocijativna memorija bi trebala biti u stanju preuzeti povezane vektore iz skupine A i primiti bilo vektor iz skupine B, teprimiti bilo koji vektor iz skupine B i preuzeti pridruženi vektor iz skupine A. Dakle prvo moramo potvrditi da je dvosmjerno asocijativna memorija u stanju prisjetiti se Y_m kada je prikazan sa X_m. To je,

$$Y_m = sign(W^T X_m), \qquad m = 1,2, \dots, M$$

Na primjer,

$$Y_1 = sign\,(W^T X_1) = sign \left\{ \begin{bmatrix} 4 & 4 & 0 & 0 & 4 & 4 \\ 0 & 0 & 4 & 4 & 0 & 0 \\ 4 & 4 & 0 & 0 & 4 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right\} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Zatim smo potvrdili da dvosmjerno asocijativna memorija podsjeća X_m kad je prikazana s Y_m. To je,

$$X_m = sign(WY_m), \qquad m = 1,2,\dots,M$$

Na primjer,

$$X_3 = sign(WY_3) = sign\left\{\begin{bmatrix} 4 & 0 & 4 \\ 4 & 0 & 4 \\ 0 & 4 & 0 \\ 0 & 4 & 0 \\ 4 & 0 & 4 \\ 4 & 0 & 4 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}\right\} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

U našem primjeru, sva četiri para se dobro pozivaju i možemo nastaviti do sljedećeg koraka.

Korak 3: Vađenje

Predstavimo nepoznati vektor (sonda) X dvosmjernoj asocijativnoj memoriji i dohvatimo pohranjenu povezanost. Sonda može predstavljati oštećenu ili nepotpunu verziju uzoraka iz skupa A (ili B) sa seta koji je pohranjen u dvosmjernoj asocijativnoj memoriji. To je,

$$X \neq X_m, \qquad m = 1,2,\dots,M$$

(a) Inicijalizacija algoritma dvosmjerno asocijativne memorije postavljanjem

$$X(0) = X, \qquad p = 0$$

i izračunati izlaz dvosmjerno asocijativne memorije ponavljanjem p

$$Y(p) = sign[W^T X(p)]$$

(b) Ažurirati ulazni vektor X (p):

$$X(p+1) = sign[WY(p)]$$

i ponoviti ponavljanje dok se ne postigne ravnoteža, kada ulazni i izlazni vektori ostaju nepromijenjeni s daljnjim ponavljanjima. Ulazni i izlazni obrasci će tada predstavljati povezani par.Dvosmjerna asocijativna memorija je bezuvjetno stabilna. To znači da se svaki skup asocijacija može naučiti bez rizika od nestabilnosti. Ova važna kvaliteta proizlazi iz dvosmjerne asocijativne memorije pomoću odnosa transpozicije između težinskih matrica u smijerovima naprijed i nazad.

Ako je težina matrice dvosmjerne asociajtivne memorije kvadratna i simetrična, onda je $W=W^T$. U tom slučaju, ulazni i izlazni slojevi su iste veličine, a dvosmjerna asocijativna memorija može se svesti na autoasociativnu Hopfieldovu mrežu.Tako se Hopfieldovamreža može smatrati posebnim slučajem asocijativne memorije.Ograničenja nametnuta kapacitetu pohrane Hopfield mreže mogu se proširiti i na dvosmjerno asocijativnu memoriju. Općenito, najveći broj asocijacija koje se pohranjuju u dvosmjernu asocijativnu memoriju, ne bi trebao biti veći od broja neurona u manjem sloju.

Drugi, još veći problem je netočna konvergencija. Dvosmjerna asocijativna memorija i dalje ostaje predmet intenzivnog istraživanja. Međutim, unatoč svim svojim aktualnim problemima i ograničenjima, dvosmjerna asocijativna memorija mogla bi postati jedna od najkorisnijih umjetnih neuronskih mreža.

6.2. Učenje neuronske mreže bez „učitelja”

Glavno svojstvo neuronske mreže je sposobnost da uči od svoje okoline, te poboljša svoju učinkovitost kroz učenje. Do sada smo razmotrili nadzirano ili aktivno učenje - učenje s vanjskim „učiteljem” ili supervisiorom koji predstavlja skup treninga za mrežu.

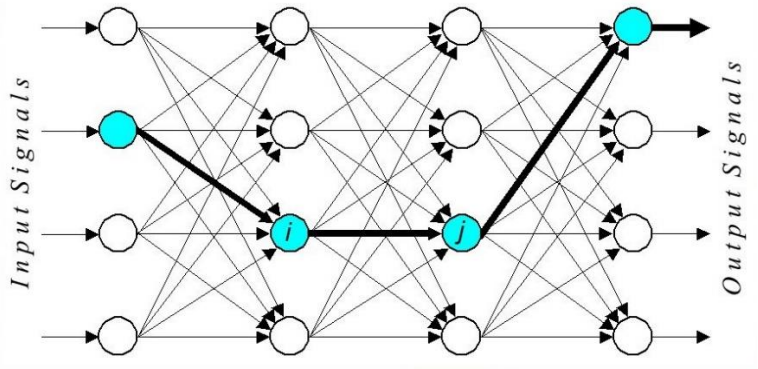
No, postoji još jedna vrsta učenja, a to je bez nadzora „učitelja“. Za razliku od nadziranog učenja, učenje bez nadzora ili učenje u samostalnoj organizaciji ne zahtijeva vanjskog učitelja. Tijekom treninga, neuronska mreža prima niz različitih ulaznih podataka u odgovarajuće kategorije. Učenje bez nadzora teži da slijedi neuro-biološke organizacije mozga. Algoritam za učenje bez nadzora ima za cilj brzo učenje. U stvari, samoorganizirajuća neuronska mreža uči mnogo brže od mreža povratnog širenja i na taj način može se koristiti u stvarnom vremenu.

7. Samoorganizacijske neuronske mreže

Samoorganizacijske neuronske mreže su učinkovite u radu sa neočekivanim i promijenjivim uvijetima. Sada ćemo razmotriti Hebbian i konkurentno učenje, koji se temelje na samoorganizirajućim mrežama.

7.1. Hebbian učenje

Godine 1949. neuropsiholog Donald Hebb predložio jednu od ključnih ideja u biološkom učenju, poznatu kao Hebbov zakon. Hebbov zakon kaže da ako je neuron *i* dovoljno blizu uzbuđivanja neurona *j* i sudjeluje u njegovoj aktivaciji, sinaptička veza između ta dva neurona ojačana je i neuron j postaje osjetljiv na podražaje od neurona i.



Slika 7.1. Hebbianovo učenje u neuronskim mrežama.

Možemo predstaviti Hebbov zakon u obliku dva pravila:

1. Ako su dva neurona s obje strane veze aktivirana sinkronizirano, onda je težina te veze povećana.
2. Ako su dva neurona s obje strane veze aktivirana nesinkronizirano, onda je težina te veze smanjena.

Hebbov zakon pruža osnovu za učenje bez učitelja. Učenje je ovdje lokalni fenomen koji se pojavljuje bez povratnih informacija iz okoline. Slika 7.1. pokazuje Hebbianovo učenje u neuronskim mrežama.

Promjena težine sinaptičke veze između para neurona povezana je s proizvodom dolaznih i odlaznih signala.Hebbian učenje podrazumijeva da se težina može samo povećati. Drugim riječima, Hebbov zakon omogućava snagu veze povećanja, ali to ne osigurava sredstva za smanjenje snage. Dakle, ponavljana primjena ulaznog signala može odvodititežinu u zasićenje. Da bismo riješili taj problem, možemo nametnuti ograničenja na rast sinaptičke težine. To se može učiniti uvođenjem nelinearnih faktora zaborava u Hebbov zakon kako slijedi.:

$$\Delta w_{ij}(p) = \alpha y_j(p) x_i(p) - \phi y_j(p) w_{ij}(p),$$

gdje je φ faktor zaboravljava.

7.2. Faktor zaborava

Faktor zaborava određuje propadanje težine u jednom ciklusu učenja. To obično pada u intervalu između 0 i 1. Ako je faktor zaborava 0, neuronska mreža je sposobna samo za jačanje sinaptičke težine, a kao rezultat toga, težine rastu prema beskonačnosti. S druge strane, ako je faktor zaborava blizu 1, mreža pamti vrlo malo onoga što uči. Stoga, dopuštaju samo malo „zaboravljanja“, dok ograničavaju rast mase. Jednadžba generaliziranog pravila aktivnosti proizvoda:

$$\Delta w_{ij}(p) = \phi y_j(p)\left[\lambda x_i(p) - w_{ij}(p)\right], \text{ gdje je } \lambda=\alpha/\varphi.$$

Generalizirano pravilo aktivnosti proizvoda podrazumijeva da, ako je presinaptička aktivnost (ulaz neurona i) na iteraciji p, $x_i(p)$, manji od $W_{ij}(p) / l$, a zatim su modificirane sinaptičke težine kod ponavljanja (p + 1), $W_{ij}(p + 1)$, će se smanjiti za iznos proporcionalno postsinaptičkim aktivnostima (Izlaz neurona j) kod iteracije p,$x_j(p)$. S druge strane, ako je $x_i(p)$ veći od $W_{ij}(p) / l$, tada će modificirana sinaptička težina na iteraciji (p + 1), $W_{ij}(p + 1)$ također povećat u odnosu na izlaz iz neurona j, $y_j(p)$. Drugim riječima, može se odrediti aktivnost točke ravnoteže za modificirane sinaptičke težine kao varijable jednake $W_{ij}(p) / l$. Ovaj pristup rješava problem beskonačnog povećanja sinaptičkih težina. Sada ćemo izvući generalizirani Hebbianov algoritam učenja.

Korak 1: Inicijalizacija

Postavljanje početne sinaptičke težine i pragova za male slučajne vrijednosti, u intervalu [0,1], te dodijeljivanje male pozitivne vrijednosti stope učenja α i faktora zaborava φ.

Korak 2: Aktivacija

Izračunati izlaz neurona na iteraciji p

$$y_j(p) = \sum_{i=1}^{n} x_i(p) w_{ij}(p) - \theta_j,$$

gdje je n broj neuronskih ulaza i θ_j je granična vrijednost od neurona j.

Korak 3: Učenje

Ažuriranje težina u mreži:

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p),$$

gdje je $\Delta w_{ij}(p)$ korekcija težine kod ponavljanja p. Korekcija težina određuje općeg pravila aktivnost proizvoda:

$$\Delta w_{ij}(p) = \phi y_j(p)\left[\lambda x_i(p) - w_{ij}(p)\right]$$

Korak 4: Ponavljane

Povećanje iteracije p za jedan, vraćanje na korak 2 i nastaviti sve dok sinaptičke težine ne dosegnu svoje stalne vrijednosti. Neuronska mreža stvarno može naučiti povezati podražaje koji su često predstavljeni zajedno, i najvažnije, mreža može naučiti bez „učitelja”.

7.3. Konkurentno učenje

Druga popularna vrsta nekontroliranog učenja je konkurentno učenje. U natjecateljskom učenju, neuroni se natječu da bi se aktivirali. Dok se u Hebbianovom učenju, nekoliko izlaznih neurona mogu aktivirati istovremeno, u natjecateljskom učenju samo je jedan izlazni neuron aktivan u bilo kojem trenutku. Izlaz neurona koji „osvoji natjecanje" se zove „pobjednik uzima sve" neuron. Osnova ideja natjecateljskog učenja uvedena je u ranim 1970-ih. Međutim, natjecateljsko učenje nije privuklo mnogo interesa do kasnih 1980-ih, kada je Teuvo Kohonen uveo posebnu klasu umjetnih neuronskih mreža koja se zove karta samoorganizirajućih značajki. Ove karte se baziraju na natjecateljskom učenju.

7.4. Karta samoorganizirajućih značajki?

Naš mozak dominira moždane kore, vrlo kompleksne strukture milijardi neurona i stotine milijardi sinapsi. Kora nje uniformna ni homogena. To uključuje područja, koja se razlikuju po debljini njihovih slojeva i vrstama neurona unutar njih, koji su odgovorni za različite ljudske aktivnosti i na taj način povezani s različitim osjetilnim ulazima. Može se reći da je svaki senzorni ulaz mapiran u odgovarajuće područje moždane kore; drugim riječima, kora je računalna katra samoorganizacije u ljudskom mozgu.

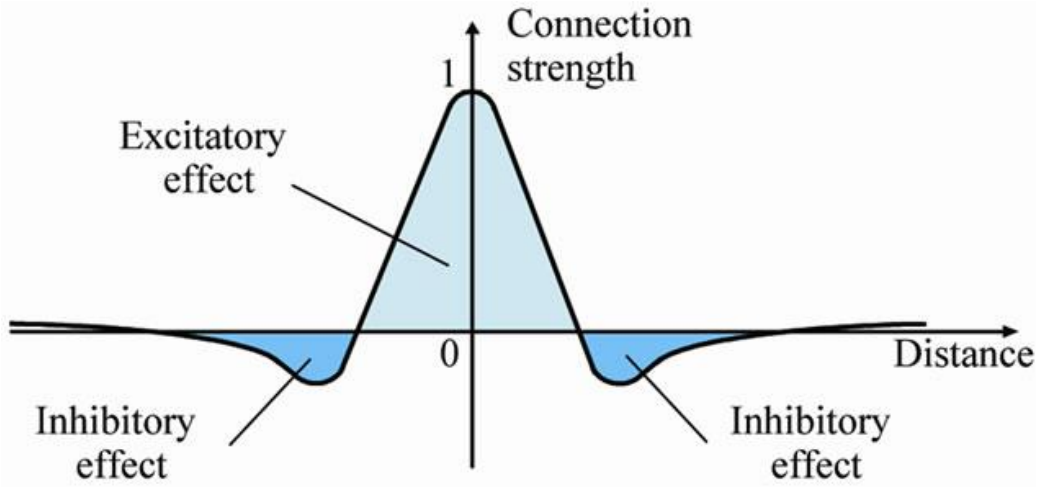
7.5. Modeliranje karte samoorganizacije

Kohonen je formulirao princip formiranja topografskih karata. To načelo navodi da prostorni položaj izlaznog neurona u topografskim kartama odgovara određenoj značajki ulaznog uzorka. Kohonen je također predložio model mapiranja značajki. On snima glavne značajke karte samoorganizacije u mozgu, a ipak se može lako prezentirati u računalu. Kohonenov model pruža topološko mapiranje, stavljanje fiksnih brojeva ulaznih uzoraka iz ulaznog sloja u više dimenzionalni izlaz ili Kohonen sloj.

Koliko je blizu neposredna fizička blizina, određuje mrežni dizajner. Pobjednici susjedstva mogu uključivati neurone s jednom, dvije ili čak tri pozicije na obje strane. Trening na Khonenovoj mreži počinje s pobjedničkim susjedstvom prilično velike veličine.Kohonen mreža se sastoji od jednoslojng računskog neurona, ali ima dvije različite vrste veza. Postoje prosljeđene veze od neurona u ulaznom sloju do neurona u izlaznom sloju, kao i bočne veze između neurona u izlaznom sloju. Bočne veze koriste se za stvaranje konkurencije između neurona. Neuron s najvećom razinom aktivacije među svim neuronima izlaznog sloja, postaje pobjednik. Taj neuron je jedini neuron kojiproizvodi izlazni signal. Aktivnost svih drugih neurona je potisnuta u natjecanju.Kada je taj ulazni uzorak prikazan na mreži, svaki neuron u Kohonenovu sloju prima punu kopiju ulaznog uzorka, promijenjenu njezinim putem iako su težine sinaptičkih veza između ulaznog sloja i Kohonenovog sloja. Bočne povratne veze proizvode ekscitatorne ili inhibicijske učinke, ovisno o udaljenosti od pobjedničkog neurona. To se postiže korištenjem funkcije „meksičkog šešira" koja opisuje sinaptičke težine između neurona u Kohonenovu sloju.

7.6. Funkcija „meksičkog šešira"?

Funkcija meksičkog šešira prikazana je na slici 7.2., koja predstavlja odnos između udaljenosti od „pobjednik uzima sve" neurona i jačine povezanosti unutar Kohonenovg sloja. Prema toj funkciji, u susjedstvu ima jak ekscitacijski učinak, u udaljinom susjedstvu ima blagi inhibitorni učinak i u vrlo udaljenom susjedstvu ima slab ekscksci tatorski učinak, koji se obično zanemaruje. U Kohonenovim mrežama, neuron uči prebacivanjem svojih težina susjedstvo mogu od neaktivnih veza do aktivnijih. Ako neuron ne reagira na određenom ulaznom uzorku, učenje se ne može dogoditi u tom neuronu.



Slika 7.2. Funkcija meksičkog šešira

7.7. Euklidova udaljenost?

Euklidova udaljenost između para n-po-1 vektora X i W_j definirana je:

$$d = \|X - W_j\| = \left[\sum_{i=1}^{n}(x_i - w_{ij})^2\right]^{1/2},$$

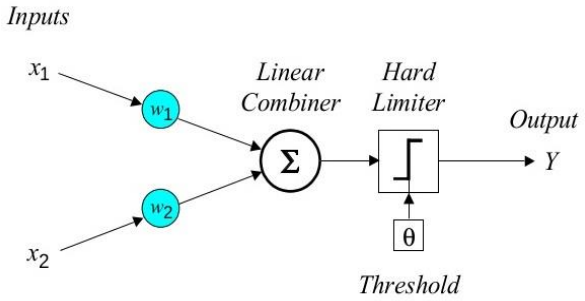
gdje su w_{ij} i x_i i-ti elementi vektora X i W_j. Sličnost između vektora X i Wj određena je kao recipročna vrijednost euklidske udaljenosti d. Štoje manja euklidsku udaljenost, veća će biti sličnost između vektora X i Wj. Kako prepoznati dobitni neuron, j_X, koji najbolje odgovara ulaznom vektor X, možemo primijeniti sljedeći uvjet:

$$j_X = \min_j \|X - W_j\|, \qquad j = 1,2,\dots,m$$

gdje je *m* broj neurona u Kohonenovu sloju.

8. Primjer treniranja Perceptrona

Perceptron je po strukturi najjednostavnija umjetna neuronska mreža koja se sastoji od dva ulaza i jednog neurona. Struktura perceptrona prikazana je na slici 8.1. a sam perceptron je detaljno opisan u poglavlju 3. ovoga rada.



Slika 8.1. Strukturni prikaz perceptrona.

Umjetna neuronska mreža iste strukture može se istrenirati za obavljanje različitih funkcija. U ovom primjeru perceptron će biti istreniran za obavljanje logičkih operacija AND i OR.

8.1. Treniranje perceptrona – algoritam

Treniranja perceptrona provodi se iterativnim postupkom do dobivanja željenog završnog stanja. Algoritam za provođenje iteracija sastoji se od sljedećih elemenata:

Ukupni ulaz u element za računanje: $\boldsymbol{X = x_1 * w_1 + x_2 * w_2}$

Izlazni signal iz elementa za računanje: $\boldsymbol{Y_{dobiveno} = step[X - \Theta]}$

Uočena greška na izlazu: $\boldsymbol{e = Y_{željeno} - Y_{dobiveno}}$

Formula za promijenu vrijednosti težinskih faktora: $\boldsymbol{w_i(p+1) = w_i(p) + \alpha * x_i(p) * e(p)}$

pri čemu je: $\boldsymbol{w_i(p+1)}$ - nova vrijednost težinskog faktora w_i

$\boldsymbol{w_i(p)}$ - trenutna vrijednost težinskog faktora w_i

α - faktor brzine učenja (0 < α < 1)

$\boldsymbol{x_i(p)}$ - trenutna vrijednost ulaznog signala x_i

e (p) - uočena greška na izlazu

$\boldsymbol{\Theta}$ - vrijednost praga za aktivaciju (0 < Θ < 1)

Prije početka provođenja iteracija potrebno je postaviti početne vrijednosti težinskih faktora w_1 i w_2, faktora brzine učenja α i praga za aktivaciju Θ. Za provođenje treniranja perceptrona za obavljanje logičkih funkcija AND i OR postavljene su sljedeće početne vrijednosti navedenih parametara:

$$w_1 = 0{,}3 \qquad \Theta = 0{,}2$$
$$w_2 = -0{,}1 \qquad \alpha = 0{,}1$$

S obzirom da postoje četiri kombinacije ulaznih parametara, jedan prolaz kroz sve četiri moguće kombinacije naziva se epohom. Svaka epoha u sebi sadrži četiri iteracije navedenog algoritma. Iterativni je postupak završen kada uz dobivene vrijednosti težinskih faktora w_1 i w_2 trenirani perceptron za svaku od ulaznih kombinacija daje točnu izlaznu vrijednost.

8.2. Treniranje perceptrona – logička funkcija AND

U slučaju treniranja mreže za obavljanje logičke funkcije AND, istrenirana mreža treba za svaku od ulaznih kombinacija x_1 i x_2 na izlazu dati željenu vrijednost $Y_{željeno}$ izlazne veličine u skladu sa sljedećom tablicom:

Logička funkcija AND		
X_1	X_2	$Y_{željeno}$
0	0	0
0	1	0
1	0	0
1	1	1

Provođenjem iterativnog postupka dobivaju se rezultati prikazani u sljedećoj tablici.

Iteracija	X_1	X_2	$Y_{željeno}$	$Y_{dobiveno}$	**e**	w_1	w_2
Epoha 1							
1.	0	0	0	0	0	0,3	– 0,1
2.	0	1	0	0	0	0,3	– 0,1
3.	1	0	0	1	– 1	0,2	– 0,1
4.	1	1	1	0	1	0,3	0,0
Epoha 2							
5.	0	0	0	0	0	0,3	0,0
6.	0	1	0	0	0	0,3	0,0
7.	1	0	0	1	– 1	0,2	0,0
8.	1	1	1	0	1	0,3	0,1
Epoha 3							
9.	0	0	0	0	0	0,3	0,1
10.	0	1	0	0	0	0,3	0,1
11.	1	0	0	1	– 1	0,2	0,1
12.	1	1	1	1	0	0,2	0,1
Epoha 4							
13.	0	0	0	0	0	0,2	0,1
14.	0	1	0	0	0	0,2	0,1
15.	1	0	0	0	0	0,2	0,1
16.	1	1	1	1	0	0,2	0,1

Tablica 8.1. Prikaz rezultata dobivenih provedenim iteracijama za logičku funkciju AND.

Kao što se iz rezultata prikazanih u Tablici 8.1. može vidjeti, točne vrijednosti težinskih faktora w_1 i w_2 dobivene su nakon 11. iteracije, s tim da je bilo nužno provesti još pet dodatnih iteracija da bi se provjerilo da dobivene vrijednosti odgovaraju za sve moguće ulazne kombinacije (cijelu epohu).

8.3. Treniranje perceptrona – logička funkcija OR

U slučaju treniranja mreže za obavljanje logičke funkcije OR, istrenirana mreža treba za svaku od ulaznih kombinacija x_1 i x_2 na izlazu dati željenu vrijednost $Y_{željeno}$ izlazne veličine u skladu sa sljedećom tablicom:

Logička funkcija OR		
X_1	X_2	$Y_{željeno}$
0	0	0
0	1	1
1	0	1
1	1	1

Provođenjem iterativnog postupka dobivaju se rezultati prikazani u sljedećoj tablici.

Iteracija	**X_1**	**X_2**	**$Y_{željeno}$**	**$Y_{dobiveno}$**	**e**	**w_1**	**w_2**
Epoha 1							
1.	0	0	0	0	0	0,3	– 0,1
2.	0	1	1	0	1	0,3	0,0
3.	1	0	1	1	0	0,3	0,0
4.	1	1	1	1	0	0,3	0,0
Epoha 2							
5.	0	0	0	0	0	0,3	0,0
6.	0	1	1	0	1	0,3	0,1
7.	1	0	1	1	0	0,3	0,1
8.	1	1	1	1	0	0,3	0,1
Epoha 3							
9.	0	0	0	0	0	0,3	0,1
10.	0	1	1	0	1	0,3	0,2
11.	1	0	1	1	0	0,3	0,2
12.	1	1	1	1	0	0,3	0,2
Epoha 4							
13.	0	0	0	0	0	0,3	0,2
14.	0	1	1	1	0	0,3	0,2
15.	1	0	1	1	0	0,3	0,2
16.	1	1	1	1	0	0,3	0,2

Tablica 8.2. Prikaz rezultata dobivenih provedenim iteracijama za logičku funkciju OR.

Kao što se iz rezultata prikazanih u Tablici 8.2. može vidjeti, točne vrijednosti težinskih faktora w_1 i w_2 dobivene su nakon 10. iteracije, s tim da je bilo nužno provesti još šest dodatnih iteracija da bi se provjerilo da dobivene vrijednosti odgovaraju za sve moguće ulazne kombinacije (cijelu epohu).

8.4. Treniranje perceptrona – usporedba dobivenih rezultata

Temeljem rezultata dobivenih iterativnim postupkom perceptron je moguće istrenirati za obavljanje logičkih operacija AND i OR, s tim da su vrijednosti faktora w_1 i w_2 dobivene za svaku od logičkih funkcija različite. Tako su za obavljanje logičke funkcije AND dobivene završne vrijednosti težinskih faktora perceptrona $w_1 = 0,2$ i $w_2 = 0,1$. Za obavljanje logičke funkcije OR dobivene završne vrijednosti težinskih faktora perceptrona su $w_1 = 0,3$ i $w_2 = 0,2$.

Osim u različitim vrijednosti težinskih faktora w_1 i w_2 postupci se razlikuju i u broj iteracija koje su uz identične vrijednosti polaznih parametara bile potrebne za dobivanje završnih rezultata. Tako su vrijednosti težinskih faktora za obavljanje logičke funkcije AND dobivene nakon 11. iteracije dok su vrijednosti za obavljanje logičke funkcije OR dobivene nakon provedenih 10. iteracija.

Iz provedenih je primjera vidljivo da se ista neuronska mreža drugačije ponaša kada ju se trenira za dvije različite funkcije (u našem slučaju, 2 različite logičke operacije) ali i da je moguće neuronsku mrežu unaprijed određene strukture istrenirati za obavljanje različitih funkcija.

9. Zaključak

Strojno učenje uključuje adaptivne mehanizme koji omogućuju računalima naučiti iz iskustva, uče primjerom i po analogiji. Mogućnosti učenja mogu poboljšati performanse inteligentnog sustava tijekom vremena. Jedan od najpopularnijih pristupa strojnog učenja su upravo umjetne neuronske mreže.

Umjetna neuronska mreža sastoji se od nekoliko vrlo jednostavnih i međusobno povezanih procesora, pod nazivom neuroni, koji su zasnovani na modeliranju bioloških neurona u mozgu. Neuroni su povezani izračunatim vezama kojima prolaze signali s jednog neurona na drugi. Svaka veza ima numeričku težinu povezanu s njom. Težine su osnova sredstva dugoročnog pamćenja u umjetnim neuronskim mrežama. Oni izražavaju snagu ili važnost za svaki ulaz neurona. Umjetna neuronska mreža „uči" kroz ponovljene prilagodbe tih težina.

Frank Rosenblatt predložio je najjednostavniji oblik neuronske mreže koju je nazvao perceptron. Sastoji se od jednog neurona s podesivim sinaptičkim težinama. Perceptron uči svoju zadaću čineći male prilagodbe u težinama kako bi se smanjila razlika između stvarnih i željenih rezultata. Učenje u višeslojnim neuronskih mrežama odvija se na isti način kao i u perceptronu.

John Hopfield je formulirao princip pohrane podataka u dinamički stabilnim mrežama, a također predložio jednoslojne mreže pomoću funkcije znaka aktivacije McCullochovih i Pittsovih neurona. Algoritam obuke Hopfieldovih mreža ima dvije osnovne faze: pohranu i dohvat. U prvoj fazi, mreža je potrebna za pohranu skupa stanja ili temeljnih sjećanja, određenih trenutnim izlazima svih neurona. Nakon što su težine izračunate, ostaju fiksne. U drugoj fazi, nepoznata oštećena ili nepotpuna verzija temeljne memorije prikazana je u mreže. Mrežni izlaz se obračunava i vraća prilagoditi ulaz. Ovaj proces se ponavlja sve dok izlaz ne postane konstanta.

Hebbov zakon kaže da ako je neuron *i* dovoljno blizu da uzbudi neuron *j* i više puta sudjeluje u aktivaciji, sinaptičke veze između ta dva neurona jačaju i neuron *j* postaje osjetljiv na podražaje iz neurona *i*. Ovaj je zakon u skladu s principima funkcioniranja bioloških neuronskih mreža u kojma veze između neurona jačaju što se češće koriste (aktiviraju).

Kao što je u radu na primjeru treniranja jednostavne umjetne neuronske mreže (perceptrona) i prikazano, umjetne neuronske mreže iste strukture mogu se istrenirati za obavljanje različitih funkcija. Ovime se umjetne neuronske mreže približavaju principima rada bioloških neuronskih mreža koje su u svojim osnovnim elementima identične te se prilagođavaju potrebama u skladu s vanjskim podražajima temeljem kojih se formiraju i uče.

Literatura

1. Negnevitsky, Miichael – Artificial Inteligence, A Guide to Inteligent Systems, 2002..
2. Fu, LiMin – Neural networks in computers inteligence, 1994.
3. Haykin, Simon – Neural networks, A comprehensive fundation, 1994.

- Internet stranice:

1. https://hr.wikipedia.org/wiki/Umjetna_neuronska_mre%C5%Bea
2. http://www.mathos.unios.hr/~mdjumic/uploads/diplomski/DUM05.pdf
3. http://eris.foi.hr/11neuronske/nn-predavanje4.html
4. https://en.wikipedia.org/wiki/Artificial_neural_network

Popis slika: